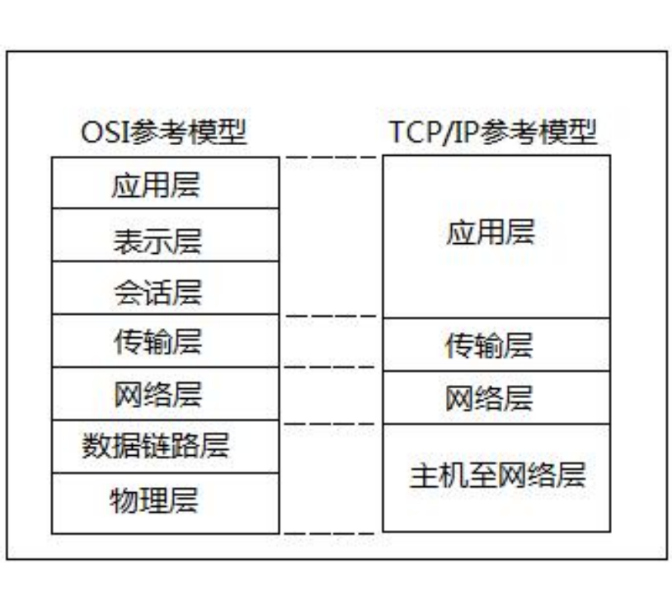
Chapter 9 网络编程

网络参考模型

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

TCP/IP 参考模型



七层描述

1. 物理层：主要定义物理设备标准，如网线的接口类型、光纤的接口类型、各种传输介质的传输速率等。它的主要作用是传输比特流（就是由1、0转化为电流强弱来进行传输，到达目的地后再转化为1、0，也就是我们常说的数模转换与模数转换）。这一层的数据叫做比特。

2. 数据链路层：主要将从物理层接收的数据进行MAC地址（网卡的地址）的封装与解封装。常把这一层的数据叫做帧。在这一层工作的设备是交换机，数据通过交换机来传输。

3. 网络层：主要将下层接收到的数据进行IP地址（例，192.168.0.1）的封装与解封装。在这一层工作的设备是路由器，常把这一层的数据叫做数据包。

4. 传输层：定义了一些传输数据的协议和端口号（WWW端口号80等），如：TCP（传输控制协议，传输效率低，可靠性强，用于传输可靠性要求高，数据量大的数据），UDP（用户数据报协议，与TCP特性恰恰相反，用于传输可靠性要求不高，数据量小的数据，如QQ聊天数据就是通过这种方式传输的）。主要是将从下层接收的数据进行分段和传输，到达目的地址后再进行重组。常常把这一层叫做段。

5. 会话层：通过传输层（端口号：传输端口与接收端口）建立数据传输的通路。主要在你的系统之间发起会话或者接收会话请求（设备之间需要互相认识可以是IP也可以是MAC或者是主机名）。

6. 表示层：主要是进行对接收的数据进行解释，加密与解密、压缩与解压缩等（也就是把计算机能够识别的东西转换成人能够识别的东西（如图片、声音等）。

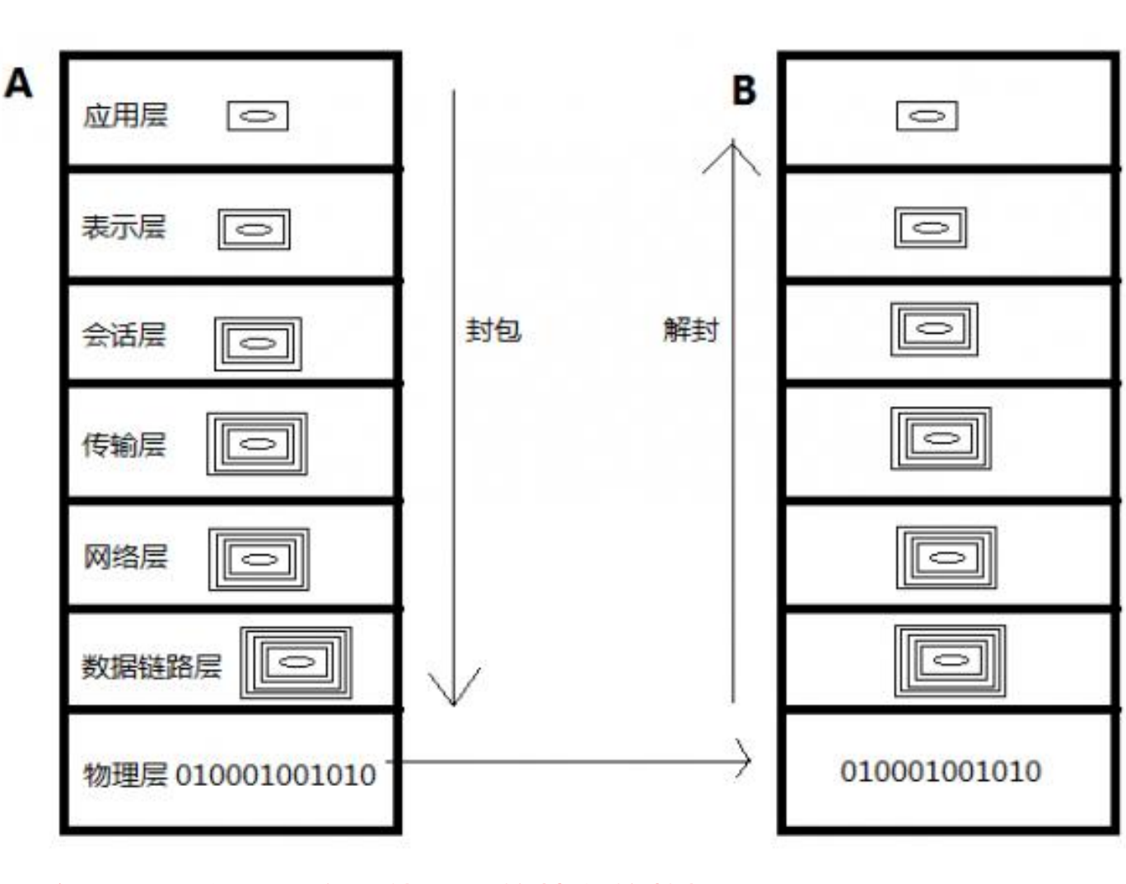
7. 应用层：主要是一些终端的应用，比如说FTP（各种文件下载）、WEB（IE浏览）、QQ之类的（可以把它理解成我们在电脑屏幕上可以看到的东西，就是终端应用）。

P.S.

1、每个网卡的MAC地址都是全球唯一的。

2、路由器实现将数据包发送到指定的地点。

3、应用软件之间通信的过程就是层与层之间封包、解封包的过程。



每一层都在识别自己能识别的特有的数据

4、OSI参考模型虽然设计精细，但过于麻烦，效率不高，因此才产生了简化版的TCP/IP参考模型。

网络编程主要在网际层和传输层

JavaWeb开发主要在应用层

网络通讯要素

每一层都有自己的协议,也就是通信规则

传输层最常用的协议是:udp和tcp

网际层最常用的协议的ip

应用层常用协议:http,ftp

1. IP地址：InetAddress

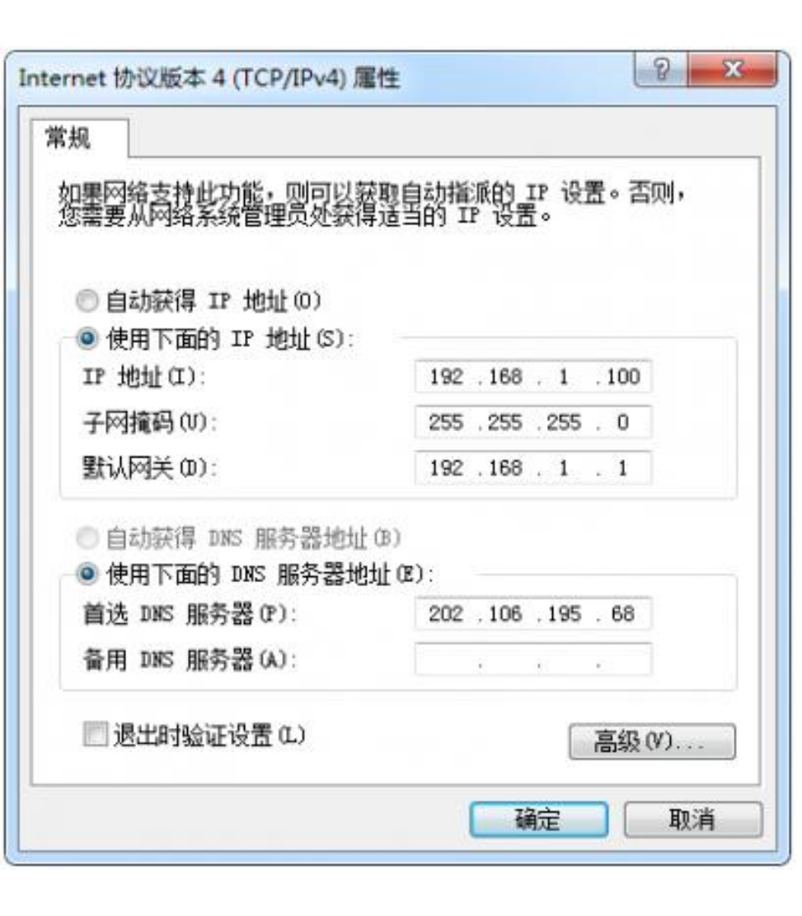
网络中设备的标识。

不易记忆，可用主机名。

本地回环地址：127.0.0.1 主机名：localhost。

P.S.

1、查看本机IP地址。


Internet 协议版本 4 (TCP/IPv4) 属性

常规

如果网络支持此功能，则可以获取自动指派的 IP 设置。否则，您需要从网络系统管理员处获得适当的 IP 设置。

○ 自动获得 IP 地址(O)

● 使用下面的 IP 地址(S):

IP 地址(I): 192 . 168 . 1 . 100

子网掩码(U): 255 . 255 . 255 . 0

默认网关(D): 192 . 168 . 1 . 1

○ 自动获得 DNS 服务器地址(B)

● 使用下面的 DNS 服务器地址(E):

首选 DNS 服务器(P): 202 . 106 . 195 . 68

备用 DNS 服务器(A): . . .

☐ 退出时验证设置(L)

2、IPV4数量已经不够分配，所以产生了IPV6。

3、在没有连接互联网的情况，为了让访问本机方便，所以分配了一个默认的IP地址，也就是本地回环地址。

4、通过ping 127.0.0.1可以测试网络是不是通，如果不通，可能是网卡出问题了。


```
C:\Users\Administrator>ping 127.0.0.1

正在 Ping 127.0.0.1 具有 32 字节的数据:
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=64
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=64
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=64
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=64

127.0.0.1 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 0ms, 平均 = 0ms
```

5、每台机器都有自己指定的计算机名。


控制面板 ▸ 系统和安全 ▸ 系统

笔和触摸: 没有可用于此显示器的笔或触控输入

计算机名称、域和工作组设置

计算机名: PC-20140621KFIX

计算机全名: PC-20140621KFIX

计算机描述:

工作组: WORKGROUP

Windows 激活

Windows 已激活

产品 ID: 00426-OEM-8992662-00006

2. 端口号

用于标识进程（应用程序）的逻辑地址，不同进程的标识。

有效端口：0~65535，其中0~1024系统使用或保留端口。

P.S.

1、当一台计算机A向另一台计算机B发送QQ信息时，首先路由器通过数据包中的IP地址定位该信息发送到哪一台机器。然后计算机B接收到数据包后，通过此数据包中的端口号定位到发送给本机的QQ应用程序。


当前有 28 个程序已连接或尝试连接网络，共建立 140 个连接，暂未发现可疑程序存在。

看网页、下载、聊天、玩网络游戏时会建立网络连接，用来完成与对方的联系和数据传送。您可以查看和管理连接，提高网络安全性。

进程	安全等级	协议	本地IP	本地端口	目标IP	目标端口	目标IP归属地	状态	管理
QQ.exe	安全	UDP	0.0.0.0	56435	0.0.0.0	0			
QQ.exe	安全	TCP	192.168.1.117	62744	27.115.124.27	80	上海市 联通	连接	
QQ.exe	安全	TCP	192.168.1.117	62750	27.115.124.27	80	上海市 联通	连接	
QQ.exe	安全	TCP	192.168.1.117	62813	27.115.124.27	80	上海市 联通	连接	
QQ.exe	安全	TCP	192.168.1.117	62819	27.115.124.27	80	上海市 联通	连接	

2、所谓防火墙，其功能就是将发送到某程序端口的数据屏蔽掉以及将从该程序端口发出的数据也屏蔽掉。

3. 传输协议

通讯的规则。

常见协议：UDP、TCP。

UDP

将数据及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内。

因无连接，是不可靠协议。

不需要建立连接，速度快。

应用案例：QQ、FeiQ聊天、在线视频用的都是UDP传输协议。

TCP

建立连接，形成传输数据的通道。

在连接中进行大数据量传输。

通过三次握手完成连接，是可靠协议。

必须建立连接，效率会稍低。

应用案例：FTP，File Transfer Protocol（文件传输协议）。

示例：

```
import java.net.InetAddress;
import java.net.UnknownHostException;

public class IPDemo
{
    public static void main(String[] args) throws UnknownHostException {
        //获取本地主机IP地址对象
        InetAddress ip = InetAddress.getLocalHost();
        //ip = InetAddress.getByName("PC-20140621KFIX");
        //ip = InetAddress.getByName("192.168.1.100");

        System.out.println(ip.getHostAddress());
        System.out.println(ip.getHostName());

        System.out.println("----------------");

        //获取其他主机的IP地址对象。
        ip = InetAddress.getByName("www.baidu.com");

        System.out.println(ip.getHostAddress());
        System.out.println(ip.getHostName());
    }
}
```

复制代码

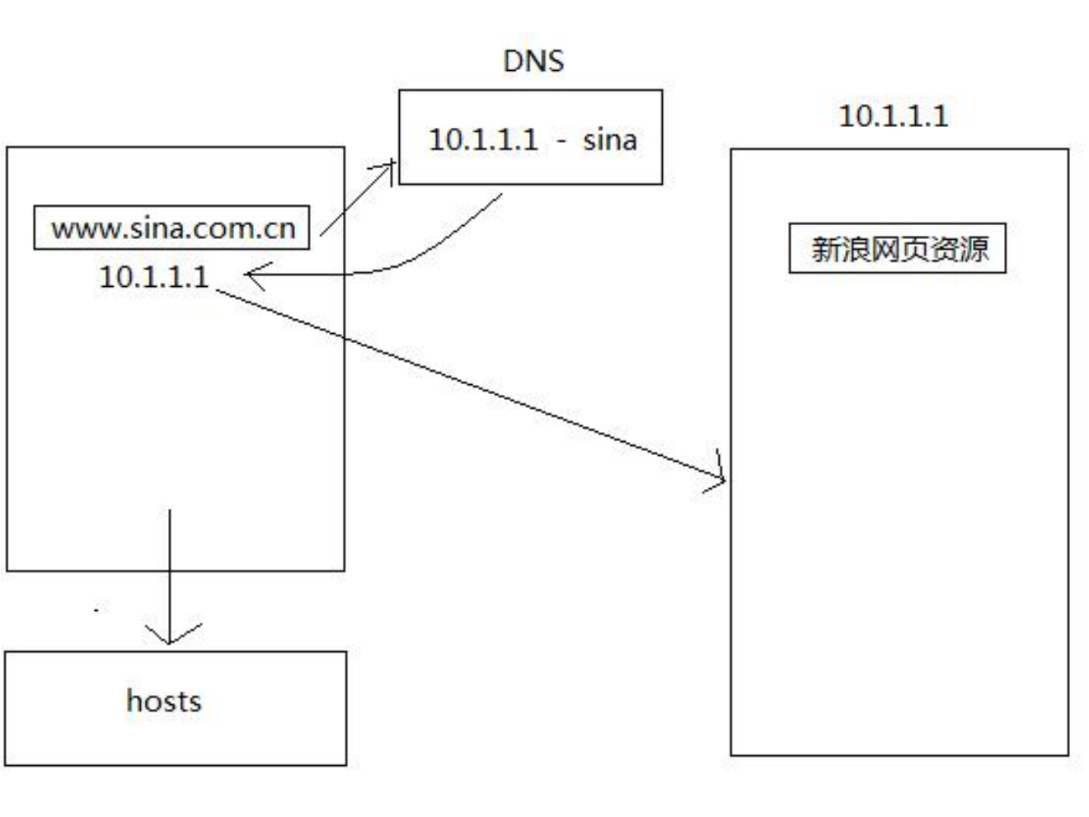
运行结果：


```
D:\code\day25>javac IPDemo.java

D:\code\day25>java IPDemo
192.168.1.100
PC-20140621KFIX
----------------
61.135.169.125
www.baidu.com
```

P.S.

InetAddress类中有一个静态方法：static InetAddress[] getAllByName(String host)，此方法是在给定主机名的情况下，根据系统上配置的名称服务返回其IP地址所组成的数据。这是由于有些主机名对应的IP地址不唯一，如新浪、百度，都是服务器集群。

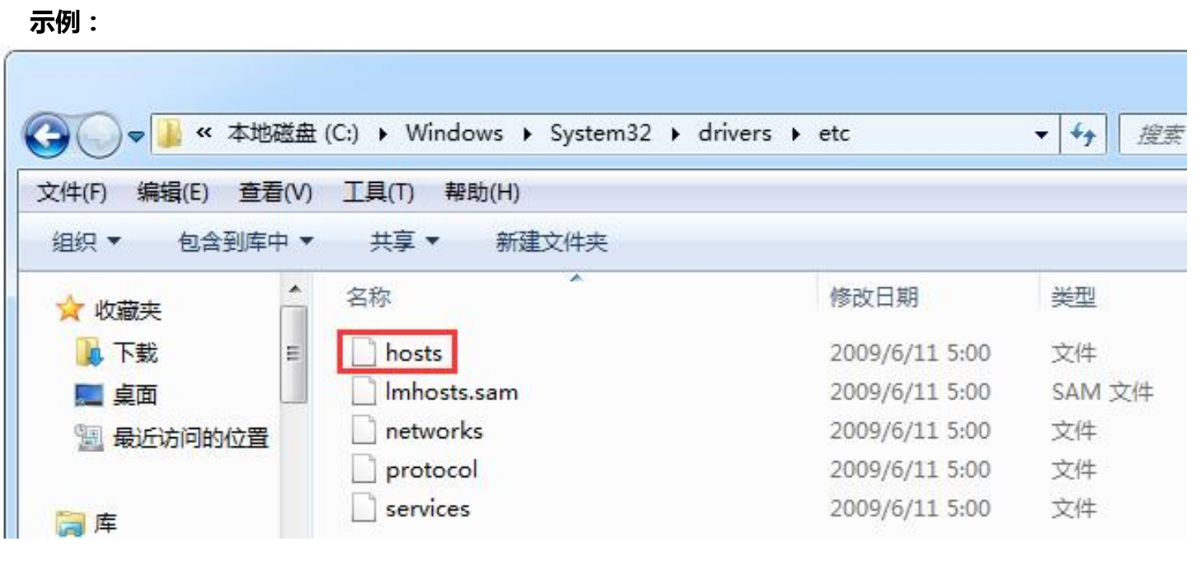
域名解析



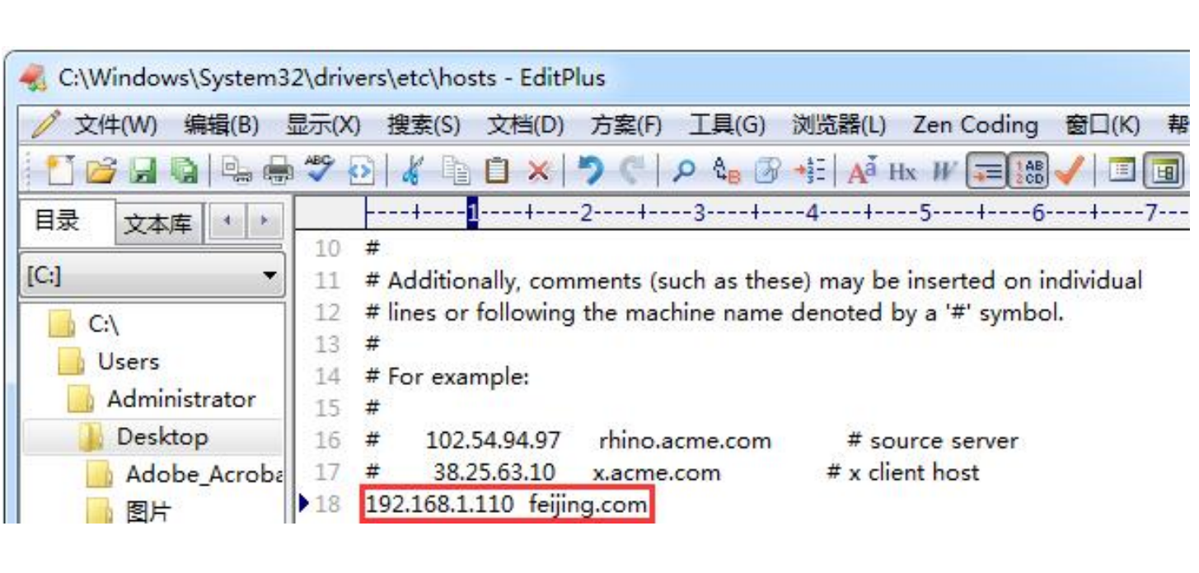
在浏览器中输入新浪的域名，DNS解析域名成IP，然后计算机再通过获取到的IP访问新浪服务器。

域名解析，最先走是本地的hosts（C:\WINDOWS\system32\drivers\etc\hosts）文件，解析失败了，才去访问DNS服务器解析、获取IP地址。

示例：


本地磁盘 (C:) ▸ Windows ▸ System32 ▸ drivers ▸ etc

名称	修改日期	类型
hosts	2009/6/11 5:00	文件
lmhosts.sam	2009/6/11 5:00	SAM 文件
networks	2009/6/11 5:00	文件
protocol	2009/6/11 5:00	文件
services	2009/6/11 5:00	文件


C:\Windows\System32\drivers\etc\hosts - EditPlus

```
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host
192.168.1.110  feijing.com
```

```
import java.net.InetAddress;
import java.net.UnknownHostException;

public class IPDemo
{
        public static void main(String[] args) throws UnknownHostException {

            InetAddress ip = InetAddress.getLocalHost();

            ip = InetAddress.getByName("192.168.1.110");

            System.out.println(ip.getHostAddress());
            System.out.println(ip.getHostName());
        }
}
```

复制代码

运行结果：


```
D:\code\day25>javac IPDemo.java

D:\code\day25>java IPDemo
192.168.1.110
feijing.com
```

应用：通过hosts文件可以屏蔽游戏网站内容弹出，例如：在hosts文件中添加，127.0.0.1 www.game18.com。

UDP协议-发送端&接收端

Socket

Socket就是为网络服务提供的一种机制。

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。

每一个应用程序都有一个"插座"

"插座"的概念就相当于计算机应用程序

UDP传输

DatagramSocket（用来发送和接收数据报包的套接字）与DatagramPacket（数据报包）。

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。

示例：

UDP发送端

```
import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.InetAddress;

public class UDPSendDemo
{
    public static void main(String[] args) throws Exception {

        System.out.println("发送端启动......");
        /*
        * 创建UDP传输的发送端。
        * 思路：
        * 1. 建立udp的socket服务。
        * 2. 将要发送的数据封装到数据包中。
        * 3. 通过udp的socket服务将数据包发送出去。
        * 4. 关闭socket服务。
        */

        //1. udpsocket服务。使用DatagramSocket对象。
        //如果发送端端口未指定，就会随机分配未被使用的端口。
        DatagramSocket ds = new DatagramSocket(8888);

        //2. 将要发送的数据封装到数据包中。
        String str = "udp传输演示，哥们来了！";

        //使用DatagramPacket将数据封装到该对象包中。
        byte[] buf = str.getBytes();

        DatagramPacket dp = new
DatagramPacket(buf,buf.length,InetAddress.getByName("192.168.1.100"),10000);

        //3. 通过udp的socket服务将数据包发送出去，使用send方法。
        ds.send(dp);

        //4. 关闭资源
        ds.close();
    }
}
```

复制代码

UDP接收端

```
import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.InetAddress;

public class UDPReceDemo
{
    public static void main(String[] args) throws Exception {

        System.out.println("接收端启动......");
        /*
        * 建立UDP接收端的思路。
        * 思路：
        * 1. 建立udp的socket服务，因为是要接收数据，必须要明确一个端口号。
        * 2. 创建数据包，用于存储接收到的数据，方便用数据包对象的方法解析这些数
据。
        * 3. 使用socket服务的receive方法将接收的数据存储到数据包中。
        * 4. 通过数据包的方法解析数据包中的数据。
        * 5. 关闭资源。
        */

        //1. 建立udpsocket服务。
        DatagramSocket ds = new DatagramSocket(10000);

        //2. 创建数据包。
        byte[] buf = new byte[1024];
        DatagramPacket dp = new DatagramPacket(buf,buf.length);

        //3. 使用接收方法将数据存储到数据包中。
        ds.receive(dp);//阻塞式的。

        //4. 通过数据包对象的方法，解析其中的数据，比如：地址，端口，数据内容。
        String ip = dp.getAddress().getHostAddress();
        //获取的端口号是发送端的端口号。
        int port = dp.getPort();
        String text = new String(dp.getData(),0,dp.getLength());

        System.out.println(ip + ":" + port + ":" + text);

        //5. 关闭资源
        ds.close();
    }
}
```

复制代码

运行结果：


```
D:\code\day25>javac UDPSendDemo.java

D:\code\day25>java UDPSendDemo
发送端启动......
```


```
D:\code\day25>javac UDPReceDemo.java

D:\code\day25>java UDPReceDemo
接收端启动......
192.168.1.100:53528:udp传输演示，哥们来了！
```

P.S.

由于UDP协议传输数据，只管发送数据，而不管接收端是否能够接收到数据。因此，应该首先启动接收端程序，再启动发送端程序。

聊天程序（双窗口模式）

UDP发送端

```
import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.InetAddress;

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class UDPSendDemo
{
    public static void main(String[] args) throws Exception {

        System.out.println("发送端启动......");

        DatagramSocket ds = new DatagramSocket(8888);

        BufferedReader bufr = new BufferedReader(new
InputStreamReader(System.in));

        String line = null;

        while((line = bufr.readLine()) != null){

            byte[] buf = line.getBytes();

```

```
                DatagramPacket dp = new DatagramPacket(buf,buf.length,InetAddress.getByName("192.168.1.100"),10000);

                ds.send(dp);

                if("886".equals(line))
                    break;
            }
            ds.close();
        }
}
```
复制代码

UDP接收端

```
import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.InetAddress;

public class UDPReceDemo
{
        public static void main(String[] args) throws Exception {

            System.out.println("接收端启动......");

            DatagramSocket ds = new DatagramSocket(10000);

            while(true){

                byte[] buf = new byte[1024];

                DatagramPacket dp = new DatagramPacket(buf,buf.length);
                ds.receive(dp);

                String ip = dp.getAddress().getHostAddress();
                int port = dp.getPort();
                String text = new String(dp.getData(),0,dp.getLength());

                System.out.println(ip + ":" + port + ":" + text);
            }
        }
}
```
复制代码

运行结果：

```
D:\code\day25>javac UDPSendDemo.java

D:\code\day25>java UDPSendDemo
发送端启动......
hehe
haha
houhou
heihei
886
```

```
D:\code\day25>javac UDPReceDemo.java

D:\code\day25>java UDPReceDemo
接收端启动......
192.168.1.100:8888:hehe
192.168.1.100:8888:haha
192.168.1.100:8888:houhou
192.168.1.100:8888:heihei
192.168.1.100:8888:886
```

聊天程序（单窗口模式-群聊）

UDP发送端

```
import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.InetAddress;

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Send implements Runnable
{
        private DatagramSocket ds;

        public Send(DatagramSocket ds){
            this.ds = ds;
        }

        public void run(){

            try{

                BufferedReader bufr = new BufferedReader(new InputStreamReader(System.in));

                String line = null;

                while((line = bufr.readLine()) != null){

                    byte[] buf = line.getBytes();

                    //255是广播地址，这样，所有192.168.1.*这个网段的所有人都会收到信息。
                    DatagramPacket dp = new DatagramPacket(buf,buf.length,InetAddress.getByName("192.168.1.255"),10001);

                    ds.send(dp);

                    if("886".equals(line))
                        break;
                }

                ds.close();

            }catch(Exception e){
                e.printStackTrace();
            }
        }
}
```
复制代码

UDP接收端

```
import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.InetAddress;

public class Rece implements Runnable
{
        private DatagramSocket ds;

        public Rece(DatagramSocket ds){
            this.ds = ds;
        }

        public void run(){

            try{

                while(true){

                    byte[] buf = new byte[1024];

                    DatagramPacket dp = new DatagramPacket(buf,buf.length);
                    ds.receive(dp);

                    String ip = dp.getAddress().getHostAddress();
                    int port = dp.getPort();
                    String text = new String(dp.getData(),0,dp.getLength());

                    System.out.println(ip + ":" + port + ":" + text);

                    if(text.equals("886")){
                        System.out.println(ip + "...退出聊天室");
                    }
                }

            }catch(Exception e){
                e.printStackTrace();
            }
        }
}
```
复制代码

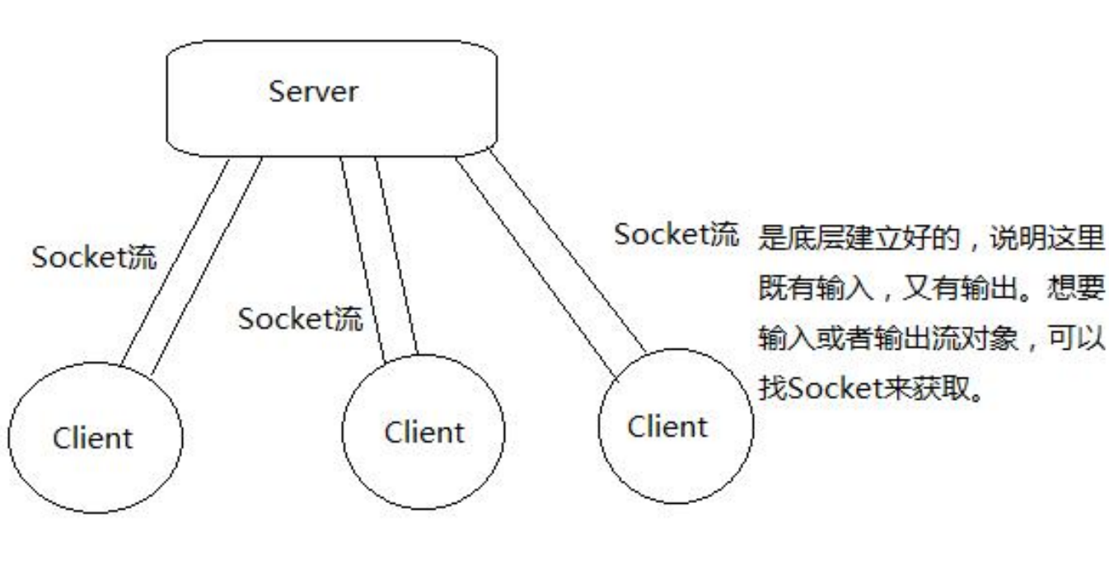
启动发送端、接收端线程程序

```
import java.io.IOException;
import java.net.DatagramSocket;

public class ChatDemo
{
        public static void main(String[] args) throws IOException {

            DatagramSocket send = new DatagramSocket();

            DatagramSocket rece = new DatagramSocket(10001);

            Send s = new Send(send);
            Rece r = new Rece(rece);

            new Thread(s).start();
            new Thread(r).start();

        }
}
```
复制代码

运行结果：

```
D:\code\day25>javac ChatDemo.java

D:\code\day25>java ChatDemo
你好啊
192.168.1.100:55147:你好啊
在哪儿呢？
192.168.1.100:55147:在哪儿呢？
哈哈
192.168.1.100:55147:哈哈
886
192.168.1.100:55147:886
192.168.1.100...退出聊天室
```

TCP协议-客户端&服务端



是底层建立好的，说明这里既有输入，又有输出。想要输入或者输出流对象，可以找Socket来获取。

客户端（Client）首先与服务端（Server）建立连接，形成通道（其实就是IO流），然后，数据就可以在通道之间进行传输，并且单个Server端可以同时与多个Client端建立连接。

Socket和ServerSocket，建立客户端和服务器端.
建立连接后，通过Socket中的IO流进行数据的传输。
关闭socket。
同样，客户端与服务器端是两个独立的应用程序。

TCP客户端

客户端需要明确服务器的ip地址以及端口，这样才可以去试着建立连接，如果连接失败，会出现异常。
连接成功，说明客户端与服务端建立了通道，那么通过IO流就可以进行数据的传输，而Socket对象已经提供了输入流和输出流对象，通过getInputStream(),getOutputStream()获取即可。
与服务端通讯结束后，关闭Socket。

TCP服务端

服务端需要明确它要处理的数据是从哪个端口进入的。
当有客户端访问时，要明确是哪个客户端，可通过accept()获取已连接的客户端对象，并通过该对象与客户端通过IO流进行数据传输。
当该客户端访问结束，关闭该客户端。

示例：

TCP客户端

```
import java.net.Socket;
import java.io.OutputStream;
import java.io.IOException;
import java.net.UnknownHostException;

public class ClientDemo
{
        public static void main(String[] args) throws UnknownHostException,IOException {

            //客户端发数据到服务端
            /*
            * TCP传输，客户端建立的过程。
            * 1. 创建TCP客户端Socket服务，使用的是Socket对象。
            *    建议该对象一创建就明确目的地，要连接的主机。
            * 2. 如果连接建立成功，说明数据传输通道已建立。
            *    该通道就是socket流，是底层建立好的。既然是流，说明这里既有输入，又有输出。
            *    想要输入或者输出流对象，可以找Socket来获取。
            *    可以通过getOutputStream()，和getInputStream()来获取两个字节流。
            * 3. 使用输出流，将数据写出。
            * 4. 关闭资源。
            */

            //创建客户端socket服务。
            Socket socket = new Socket("192.168.1.100",10002);

            //获取socket流中的输出流
            OutputStream out = socket.getOutputStream();

            //使用输出流将指定的数据写出去。
            out.write("tcp演示：哥们又来了！".getBytes());

            //断开链接，关闭资源，socket获取的输出流也被关闭，没有必要再写代码关闭。
            socket.close();
        }
}
```
复制代码

TCP服务端

```
import java.net.ServerSocket;
import java.net.Socket;
import java.io.InputStream;
import java.io.IOException;

public class ServerDemo{

        public static void main(String[] args) throws IOException {

                //服务端接收客户端发送过来的数据，并打印到控制台上。
               /*
                * 建立tcp服务端的思路：
                * 1. 创建服务端socket服务，通过ServerSocket对象。
                * 2. 服务端必须对外提供一个端口，否则客户端无法连接。
                * 3. 获取连接过来的客户端对象。
                * 4. 通过客户端对象获取socket流读取客户端发来的数据，
                *      并打印在控制台上。
                * 5. 关闭资源，关客户端，关服务端。
                */

                //1. 创建服务端对象
                ServerSocket ss = new ServerSocket(10002);

                //2. 获取连接过来的客户端对象。
                Socket s = ss.accept();//阻塞式
                String ip = s.getInetAddress().getHostAddress();

                //3. 通过socket对象获取输入流，要读取客户端发来的数据。
                InputStream in = s.getInputStream();

                byte[] buf = new byte[1024];

                int len = in.read(buf);
                String text = new String(buf,0,len);
                System.out.println(ip + ":" + text);

                s.close();
                ss.close();
        }
}
```
复制代码

运行结果：

```
D:\code\day25>javac ClientDemo.java

D:\code\day25>java ClientDemo
```

```
D:\code\day25>javac ServerDemo.java

D:\code\day25>java ServerDemo
192.168.1.100:tcp演示：哥们又来了！
```

P.S.

TCP协议传输数据必须先开服务端，再开客户端。否则，客户端根本连接不上服务端。

TCP协议-服务端和客户端交互

TCP客户端

```
import java.net.Socket;
import java.io.OutputStream;
import java.io.IOException;
import java.net.UnknownHostException;
import java.io.InputStream;

public class ClientDemo
{
        public static void main(String[] args) throws UnknownHostException,IOException {

            Socket socket = new Socket("192.168.1.100",10002);

            OutputStream out = socket.getOutputStream();

            out.write("tcp演示：哥们又来了！".getBytes());

            //读取客户端返回的数据，使用Socket读取流。
            InputStream in = socket.getInputStream();

            byte[] buf = new byte[1024];

            int len = in.read(buf);

            String text = new String(buf,0,len);

            System.out.println(text);

            socket.close();
        }
}
```
复制代码

TCP服务端

```
import java.net.ServerSocket;
import java.net.Socket;
import java.io.InputStream;
import java.io.IOException;
import java.io.OutputStream;

public class ServerDemo{

        public static void main(String[] args) throws IOException {

            ServerSocket ss = new ServerSocket(10002);

            Socket s = ss.accept();
            String ip = s.getInetAddress().getHostAddress();

            InputStream in = s.getInputStream();

            byte[] buf = new byte[1024];

            int len = in.read(buf);
            String text = new String(buf,0,len);
            System.out.println(ip + ":" + text);

            //使用客户端socket对象的输出流给客户端返回数据
            OutputStream out = s.getOutputStream();
            out.write("收到".getBytes());

            s.close();
            ss.close();
        }
}
```
复制代码

运行结果：

```
D:\code\day25>javac ServerDemo.java

D:\code\day25>java ServerDemo
192.168.1.100:tcp演示：哥们又来了！
```

```
D:\code\day25>javac ClientDemo.java

D:\code\day25>java ClientDemo
收到
```

练习：文本转换TCP客户端和服务端

TCP客户端

```
import java.net.Socket;
import java.io.OutputStream;
import java.io.IOException;
import java.net.UnknownHostException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class TransClient
{
        public static void main(String[] args) throws UnknownHostException,IOException {

            /*
            * 思路：
            * 客户端：
            * 1. 需要先有socket端点。
            * 2. 客户端的数据源：键盘。
            * 3. 客户端的目的：socket。
            * 4. 接收服务端的数据，源：socket。
            * 5. 将数据显示再打印出来。目的：控制台。
            * 6. 在这些流中操作的数据，都是文本数据。
             *
            * 转换客户端：
            * 1. 创建Socket客户端对象。
            * 2. 获取键盘录入。
            * 3. 将录入的信息发送给socket输出流。
```

```
            */

            //1. 创建socket客户端对象。
            Socket s = new Socket("192.168.1.100",10004);

            //2. 获取键盘录入
            BufferedReader bufr = new BufferedReader(new
    InputStreamReader(System.in));

            //3. socket输出流
            //new BufferedWriter(new OutputStreamWriter(s.getOutputStream()));
            PrintWriter out = new PrintWriter(s.getOutputStream(),true);

            //4. socket输入流，读取服务端返回的大写数据。
            BufferedReader bufIn = new BufferedReader(new
    InputStreamReader(s.getInputStream()));

            String line = null;

            while((line = bufr.readLine()) != null){
                if("over".equals(line))
                    break;

                out.println(line);

                //读取服务端发回的一行大写数据。
                String upperStr = bufIn.readLine();

                System.out.println(upperStr);
            }

            s.close();
        }
}
```

复制代码

TCP服务端

```
import java.net.ServerSocket;
import java.net.Socket;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.IOException;

public class TransServer
{
        public static void main(String[] args) throws IOException {
            /*
            *
            * 转换服务器。
            * 分析：
            * 1. serversocket服务。
            * 2. 获取socket对象。
            * 3. 源：socket，读取客户端发过来需要转换的数据。
            * 4. 目的：显示在控制台上。
            * 5. 将数据转换成大写发给客户端。
            */

            //1. 创建ServerSocket。
            ServerSocket ss = new ServerSocket(10004);

            //2. 获取socket对象。
            Socket s = ss.accept();

            //获取ip。
            String ip = s.getInetAddress().getHostAddress();
            System.out.println(ip + "......connected");

            //3. 获取socket读取流，并装饰。
            BufferedReader bufIn = new BufferedReader(new
    InputStreamReader(s.getInputStream()));

            //4. 获取socket的输出流，并装饰。
            PrintWriter out = new PrintWriter(s.getOutputStream(),true);

            String line = null;

            while((line = bufIn.readLine()) != null){
                        System.out.println(line);
                        out.println(line.toUpperCase());
            }

            s.close();
            ss.close();
        }
}
```

复制代码

运行结果：

```
D:\code\day25>javac TransServer.java

D:\code\day25>java TransServer
192.168.1.100......connected
abc
haha
xixi
heihei
```

```
D:\code\day25>javac TransClient.java

D:\code\day25>java TransClient
abc
ABC
haha
HAHA
xixi
XIXI
heihei
HEIHEI
over
```

常见问题：

1、上面练习中之所以客户端结束后，服务端也随之结束的原因在于：客户端的socket关闭后，服务端获取的客户端socket读取流也关闭了，因此读取不到数据，line = bufIn.readLine()为null，循环结束，ServerSocket的close方法也就执行关闭了。

2、上面练习中的客户端和服务端的PrintWriter对象out获取到数据后，一定要刷新，否则对方（服务端或客户端）就获取不到数据，程序便无法正常执行。刷新操作可以通过PrintWriter类的println()方法实现，也可以通过PrintWriter类的flush()方法实现。但是，由于获取数据的方法是BufferedReader对象bufIn的readLine()方法（阻塞式方法），此方法只有遇到“\r\n”标记时，才认为数据读取完毕，赋值给String对象line。所以，使用PrintWriter类的flush()方法刷新数据时一定要记得追加“\r\n”！

练习：TCP协议上传文本文件

TCP服务端

```
import java.net.ServerSocket;
import java.net.Socket;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.BufferedWriter;
import java.io.PrintWriter;
import java.io.IOException;
import java.io.FileWriter;

public class UploadServer
{
        public static void main(String[] args) throws IOException {

            ServerSocket ss = new ServerSocket(10005);

            Socket s = ss.accept();

            System.out.println(s.getInetAddress().getHostAddress() + "......connected");

            BufferedReader bufIn = new BufferedReader(new
    InputStreamReader(s.getInputStream()));

            BufferedWriter bufw = new BufferedWriter(new
    FileWriter("d:\\demo\\server.txt"));

            String line = null;

            while((line = bufIn.readLine()) != null){
                        //if("over".equals(line))
                        //      break;
                        bufw.write(line);
                        bufw.newLine();
            }

            PrintWriter out = new PrintWriter(s.getOutputStream(),true);
            out.println("上传成功");

            bufw.close();
            s.close();
            ss.close();
        }
}
```

复制代码

TCP客户端

```
import java.net.Socket;
import java.io.OutputStream;
import java.io.IOException;
import java.net.UnknownHostException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.FileReader;

public class UploadClient
{
        public static void main(String[] args) throws
UnknownHostException,IOException {

                Socket s = new Socket("192.168.1.100",10005);

                BufferedReader bufr = new BufferedReader(new
FileReader("d:\\demo\\client.txt"));

                PrintWriter out = new PrintWriter(s.getOutputStream(),true);

                String line = null;

                while((line = bufr.readLine()) != null){
                        out.println(line);
                }

                //告诉服务端，客户端写完了。
                s.shutdownOutput();
                //不要写over，上传的文本中可能也存在“over”
                //out.println("over");

                BufferedReader bufIn = new BufferedReader(new
InputStreamReader(s.getInputStream()));

                String str = bufIn.readLine();
                System.out.println(str);

                bufr.close();
                s.close();
        }
}
```

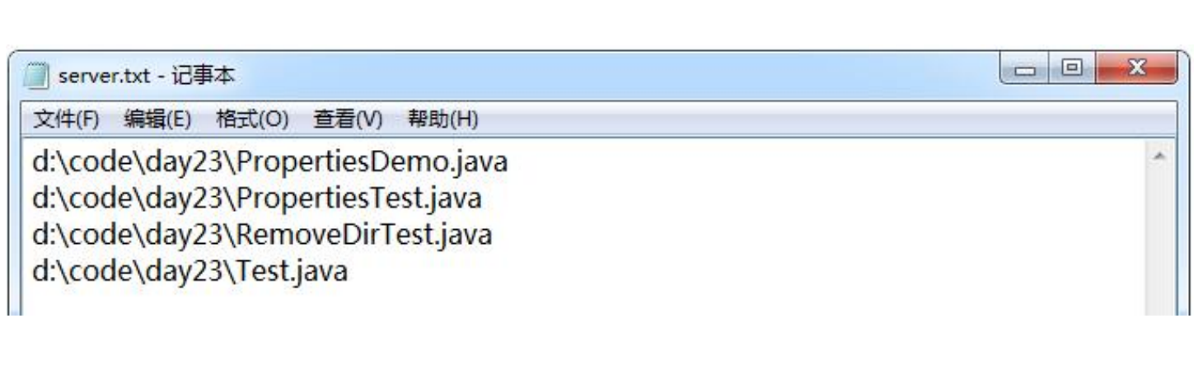
复制代码

运行结果：

```
D:\code\day25>javac UploadServer.java

D:\code\day25>java UploadServer
192.168.1.100......connected
```

```
D:\code\day25>javac UploadClient.java

D:\code\day25>java UploadClient
上传成功
```

名称	修改日期	类型
client.txt	2015/6/7 19:41	文本文档
server.txt	2015/6/9 21:06	文本文档



```
d:\code\day23\PropertiesDemo.java
d:\code\day23\PropertiesTest.java
d:\code\day23\RemoveDirTest.java
d:\code\day23\Test.java
```

~END~

* * *

~爱上海，爱黑马~