[Java网络编程总结](http://blog.csdn.net/tt_zhang/article/details/7662860)

标签： [网络](http://www.csdn.net/tag/%e7%bd%91%e7%bb%9c)[编程](http://www.csdn.net/tag/%e7%bc%96%e7%a8%8b)[java](http://www.csdn.net/tag/java)[socket](http://www.csdn.net/tag/socket)[url](http://www.csdn.net/tag/url)[tcp](http://www.csdn.net/tag/tcp)

2012-06-14 14:07 4484人阅读 [评论](http://blog.csdn.net/tt_zhang/article/details/7662860#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/tt_zhang/article/details/7662860#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

java（22） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

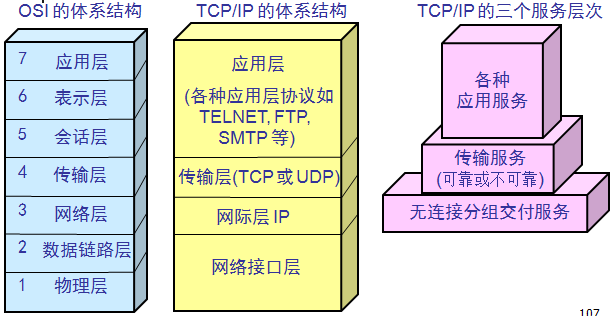
目录[(?)[-]](http://blog.csdn.net/tt_zhang/article/details/7662860)

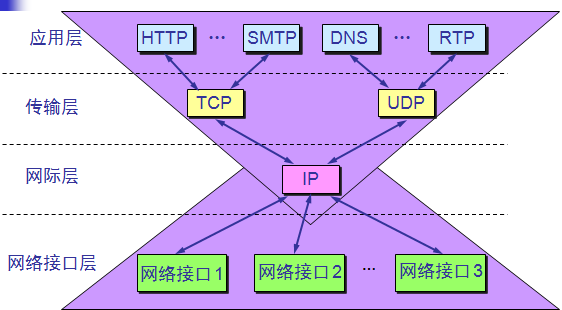
1. [Java网络编程总结](http://blog.csdn.net/tt_zhang/article/details/7662860#t0)
   1. [一网络编程基本概念](http://blog.csdn.net/tt_zhang/article/details/7662860#t1)
      1. [OSI与TCPIP体系模型](http://blog.csdn.net/tt_zhang/article/details/7662860#t2)
      2. [IP和端口](http://blog.csdn.net/tt_zhang/article/details/7662860#t3)
      4. [TCP和UDP协议](http://blog.csdn.net/tt_zhang/article/details/7662860#t5)
      5. [Socket](http://blog.csdn.net/tt_zhang/article/details/7662860#t6)
      6. [常用应用层协议](http://blog.csdn.net/tt_zhang/article/details/7662860#t7)
   2. [二Java网络编程常用类](http://blog.csdn.net/tt_zhang/article/details/7662860#t8)
      1. [InteAddress类](http://blog.csdn.net/tt_zhang/article/details/7662860#t9)
      2. [URL和URLConnection类](http://blog.csdn.net/tt_zhang/article/details/7662860#t10)
      3. [URLDecoder和URLEncoder](http://blog.csdn.net/tt_zhang/article/details/7662860#t11)
      4. [Socket和ServerSocket类](http://blog.csdn.net/tt_zhang/article/details/7662860#t12)
      5. [DatagramSocket类](http://blog.csdn.net/tt_zhang/article/details/7662860#t13)
   3. [三总结](http://blog.csdn.net/tt_zhang/article/details/7662860#t14)

**[Java网络编程总结](http://www.cnblogs.com/oubo/archive/2012/01/16/2394641.html)**

**一、网络编程基本概念**

**1.OSI与TCP/IP体系模型**





**2.IP和端口**

解决了文章最开始提到的定位的问题。

IP在互联网中能唯一标识一台计算机，是每一台计算机的唯一标识（身份证）；网络编程是和远程计算机的通信，所以必须先能定位到远程计算机；IP帮助解决此问题；一台计算机中可能有很多进程，具体和哪一个进程进行通信，这就得靠端口来识别；

**IP和端口能唯一定位到需要通信的进程。这里的IP表示地址，区别于IP协议。在OSI体系还是TCP/IP体系中，IP协议位于网际层，来封装IP地址到报文中。**

**3.TCP和UDP协议**

**TCP**是**Tranfer Control Protocol**的简称，是一种面向连接的保证可靠传输的协议。通过TCP协议传输，得到的是一个顺序的无差错的数据流。发送方和接收方的成对的两个socket之间必须建立连接，以便在TCP协议的基础上进行通信，当一个socket（通常都是server socket）等待建立连接时，另一个socket可以要求进行连接，一旦这两个socket连接起来，它们就可以进行双向数据传输，双方都可以进行发送或接收操作。

**UDP**是**User Datagram Protocol**的简称，是一种无连接的协议，每个数据报都是一个独立的信息，包括完整的源地址或目的地址，它在网络上以任何可能的路径传往目的地，因此能否到达目的地，到达目的地的时间以及内容的正确性都是不能被保证的。

**比较：**

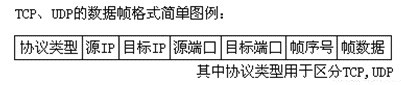
UDP：

1. 每个数据报中都给出了完整的地址信息，因此无需要建立发送方和接收方的连接。
2. UDP传输数据时是有大小限制的，每个被传输的数据报必须限定在64KB之内。
3. UDP是一个不可靠的协议，发送方所发送的数据报并不一定以相同的次序到达接收方

TCP：

1. 面向连接的协议，在socket之间进行数据传输之前必然要建立连接，所以在TCP中需要连接时间。
2. TCP传输数据大小限制，一旦连接建立起来，双方的socket就可以按统一的格式传输大的数据。
3. TCP是一个可靠的协议，它确保接收方完全正确地获取发送方所发送的全部数据。

**数据桢：**



**应用：**

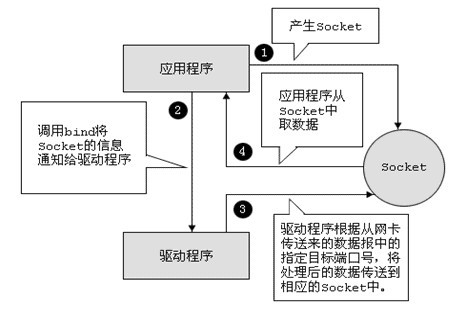
* **TCP**在网络通信上有极强的生命力，例如远程连接（Telnet）和文件传输（FTP）都需要不定长度的数据被可靠地传输。但是可靠的传输是要付出代价的，对数据内容正确性的检验必然占用计算机的处理时间和网络的带宽，因此TCP传输的效率不如UDP高。
* **UDP**操作简单，而且仅需要较少的监护，因此通常用于局域网高可靠性的分散系统中client/server应用程序。例如视频会议系统，并不要求音频视频数据绝对的正确，只要保证连贯性就可以了，这种情况下显然使用UDP会更合理一些。

**4.Socket**

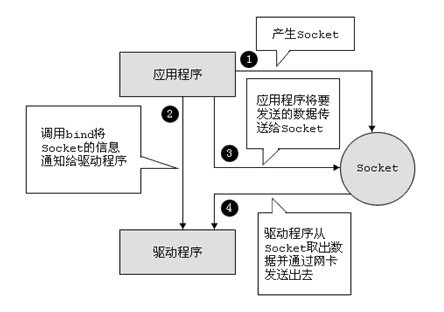
Socket是网络驱动层提供给应用程序编程接口和一种机制。我们可以把 Socket 比喻成是一个港口码头。应用程序只要把货物放到港口码头上，就算完成了货物的运送。对于接收方应用程序也要创建一个港口码头，只需要等待货物到达码头后将货物取走。

Socket 是在应用程序中创建的，它是通过一种绑定机制与驱动程序建立关系，告诉自己所对应的 IP 和 Port。在网络上传输的每一个数据帧，必须包含发送者的 IP 地址和端口号。创建完 Socket 以后，应用程序写入到 Socket 的数据，由 Socket 交给驱动程序向网络上发送数据，计算机从网络上收到与某个 Socket 绑定的 IP 和 Port 相关的数据后，由驱动程序再交给 Socket ，应用程序就可以从这个 Socket 中读取接收到的数据。网络应用程序就是这样通过 Socket 发送和接收的。

**Socket数据发送过程：**



**Socket数据接收过程：**



**5.常用应用层协议**

应用层协议是为了解决某一类应用问题，而问题的解决又往往是通过位于不同主机中的多个应用进程之间的通信和协同工作来完成的。应用层的具体内容就是规定应用进程在通信时所遵循的协议。

**二、Java网络编程常用类**

**1.InteAddress类**

[**Java**](http://lib.csdn.net/base/17)中的InetAddress是一个代表IP地址的封装。IP地址可以由字节数组和字符串来分别表示，InetAddress将IP地址以对象的形式进行封装，可以更方便的操作和获取其属性。InetAddress没有构造方法，可以通过两个静态方法获得它的对象。

[复制代码](http://blog.csdn.net/tt_zhang/article/details/7662860)

//根据主机名来获取对应的InetAddress实例

InetAddress ip = InetAddress.getByName("www.baidu.com");

//判断是否可达

System.out.println("baidu是否可达：" + ip.isReachable(2000));

//获取该InetAddress实例的IP字符串

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

//根据原始IP地址(字节数组形式)来获取对应的InetAddress实例

InetAddress local = InetAddress.getByAddress(new byte[]{127,0,0,1});

System.out.println("本机是否可达：" + local.isReachable(5000));

//获取该InetAddress实例对应的全限定域名

System.out.println(local.getCanonicalHostName());

[复制代码](http://blog.csdn.net/tt_zhang/article/details/7662860)

**2.URL和URLConnection类**

网络中的URL（Uniform Resource Locator）是统一资源定位符的简称。它表示Internet上某一资源的地址。通过URL我们可以访问Internet上的各种网络资源，比如最常见的WWW，FTP站点。 URL可以被认为是指向互联网资源的“指针”，通过URL可以获得互联网资源相关信息，包括获得URL的InputStream对象获取资源的信息，以及一个到URL所引用远程对象的连接URLConnection。 URLConnection对象可以向所代表的URL发送请求和读取URL的资源。通常，创建一个和URL的连接，需要如下几个步骤：

1. 创建URL对象，并通过调用openConnection方法获得URLConnection对象；
2. 设置URLConnection参数和普通请求属性；
3. 向远程资源发送请求；
4. 远程资源变为可用，程序可以访问远程资源的头字段和通过输入流来读取远程资源返回的信息。

这里需要重点讨论一下第三步：如果只是发送GET方式请求，使用connect方法建立和远程资源的连接即可；如果是需要发送POST方式的请求，则需要获取URLConnection对象所对应的输出流来发送请求。这里需要注意的是，由于GET方法的参数传递方式是将参数显式追加在地址后面，那么在构造URL对象时的参数就应当是包含了参数的完整URL地址，而在获得了URLConnection对象之后，就直接调用connect方法即可发送请求。而POST方法传递参数时仅仅需要页面URL，而参数通过需要通过输出流来传递。另外还需要设置头字段。以下是两种方式的代码：

[复制代码](http://blog.csdn.net/tt_zhang/article/details/7662860)

//1. 向指定URL发送GET方法的请求

String urlName = url + "?" + param;

URL realUrl = new URL(urlName);

//打开和URL之间的连接

URLConnection conn = realUrl.openConnection();

//设置通用的请求属性

conn.setRequestProperty("accept", "\*/\*");

conn.setRequestProperty("connection", "Keep-Alive");

conn.setRequestProperty("user-agent","Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)");

//建立实际的连接

conn.connect();

//2. 向指定URL发送POST方法的请求

URL realUrl = new URL(url);

//打开和URL之间的连接

URLConnection conn = realUrl.openConnection();

//设置通用的请求属性

conn.setRequestProperty("accept", "\*/\*");

conn.setRequestProperty("connection", "Keep-Alive");

conn.setRequestProperty("user-agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)");

//发送POST请求必须设置如下两行

conn.setDoOutput(true);

conn.setDoInput(true);

//获取URLConnection对象对应的输出流

out = new PrintWriter(conn.getOutputStream());

//发送请求参数

out.print(param);

[复制代码](http://blog.csdn.net/tt_zhang/article/details/7662860)

**3.URLDecoder和URLEncoder**

这两个类可以别用于将application/x-www-form-urlencoded MIME类型的字符串转换为普通字符串，将普通字符串转换为这类特殊型的字符串。使用URLDecoder类的静态方法decode()用于解码，URLEncoder类的静态方法encode()用于编码。具体使用方法如下：

//将application/x-www-form-urlencoded字符串转换成普通字符串

String keyWord = URLDecoder.decode("%E6%9D%8E%E5%88%9A+j2ee", "UTF-8");

System.out.println(keyWord);

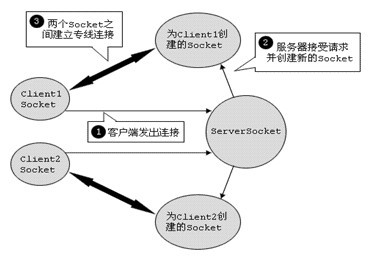
//将普通字符串转换成 application/x-www-form-urlencoded字符串

String urlStr = URLEncoder.encode( "ROR敏捷开发最佳指南" , "GBK");

System.out.println(urlStr);

**4.Socket和ServerSocket类**

网络上的两个程序通过一个双向的通讯连接实现数据的交换，这个双向链路的一端称为一个Socket。Socket通常用来实现客户方和服务方的连接。Socket是TCP/IP协议的一个十分流行的编程界面，一个Socket由一个IP地址和一个端口号唯一确定。 但是，Socket所支持的协议种类也不光TCP/IP一种，因此两者之间是没有必然联系的。在Java环境下，Socket编程主要是指基于TCP/IP协议的网络编程。 Server端Listen(监听)某个端口是否有连接请求，Client端向Server端发出Connect(连接)请求，Server端向Client端发回Accept（接受）消息。一个连接就建立起来了。Server端和Client端都可以通过Send，Write等方法与对方通信。 TCP Socket的通信过程如下图：



**5.DatagramSocket类**

UDP协议是一种不可靠的网络协议，它在通讯实例的两端个建立一个Socket，但这两个Socket之间并没有虚拟链路，这两个Socket只是发送和接受数据报的对象。 包**java.net**中提供了两个类**DatagramSocket**和**DatagramPacket**用来支持数据报通信，DatagramSocket用于在程序之间建立传送数据报的通信连接， DatagramPacket则用来表示一个数据报。 **DatagramSocket的构造方法：**

DatagramSocket()；

DatagramSocket(int prot);

DatagramSocket(int port, InetAddress laddr);

其中，port指明socket所使用的端口号，如果未指明端口号，则把socket连接到本地主机上一个可用的端口。laddr指明一个可用的本地地址。给出端口号时要保证不发生端口冲突，否则会生成SocketException类例外。注意：上述的两个构造方法都声明抛弃非运行时例外SocketException，程序中必须进行处理，或者捕获、或者声明抛弃。 用数据报方式编写client/server程序时，无论在客户方还是服务方，首先都要建立一个DatagramSocket对象，用来接收或发送数据报，然后使用DatagramPacket类对象作为传输数据的载体。

**三、总结**

* ***网络编程的核心是IP、端口、协议三大元素***
* ***网络编程的本质是进程间通信***
* ***网络编程的2个主要问题：1是定位主机，2是数据传输***

参考资料：

<http://blog.csdn.net/driverking/article/details/6573992> <http://blog.sina.com.cn/s/blog_6213b4e50100je4g.html><http://52android.blog.51cto.com/2554429/482954> <http://hi.baidu.com/refresh/blog/item/1ab994dd4f6957315982ddaa.html>