Java基础学习总结—网络编程

网络基础概念

首先理清一个概念：网络编程 != 网站编程， 网络编程现在一般称为TCP/IP编程。

计算机网络：

把分布在不同地理区域的计算机与专门的外部设备用通信线路互连成一个规模大，功能强的网络系统，从而使众多的计算机可以方便的互相传递信息，共享硬件，软件，数据信息等资源。

计算机网络的主要功能：

资源共享

信息传输与集中处理

均衡负载与分布处理

综合信息服务

网络通信协议及接口

网络通信协议：

计算机昂罗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。

网络通信接口：

为了使两个结点之间进行对话，必须在他们之间建立通信工具（即接口），使彼此之间能进行信息交换。接口包括两部分：

* 硬件装置：实现节点之间的信息传送。
* 软件装置：规定双方进行通信的约定协议。

通信协议分层思想

为什么要分层？

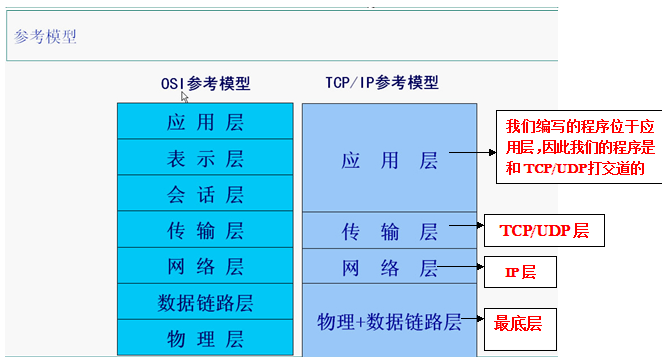
由于结点之间联系很复杂，在制定协议时，把复杂成分分解成一些简单的成分，再将他们复合起来。最常用的复合方式使使层次方式，即同层间可以通信，上一层可以调用下一层，而与下一层不发生关系。各层不影响，利于系统的开发和扩展。

通信协议的分层规定

把用户应用程序徐作为最高层，把物理通信线路作为最底层，将其间的协议处理分为若干曾，规定每层处理的任务，也规定每层的接口标准。



参考模型



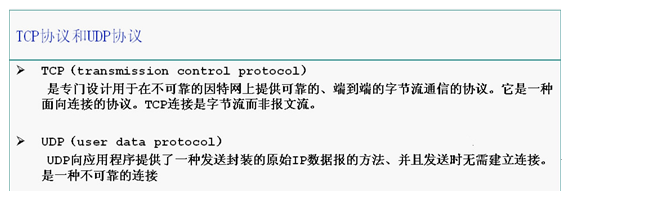
IP协议



每个人的电脑都有一个独一无二的IP地址，这样互相通信时就不会传错信息了。

IP地址是用一个点来分成四段的，再计算机内部IP地址是用四个字节来表示的，一个字节代表一段，每一个字节戴白哦的数最大只能到达255.

TCP协议和UDP协议



TCP和UDP位于同一层，都是建立再IP层，都是建立在IP层的基础之上。由于两台电脑之间有不同的IP地址，因此两台电脑就可以区分开来，也就可以互相通话了。通话一般有两种通话方式：第一种就是TCP，第二种UDP。TCCP是可靠的链接，TCP就像打电话，需要先打通对方电话，等待对方有回应之后才会跟对方继续说话，也就是一定要确认可以发信息以后才会把信息发出去。TCP上传任何东西都是可靠的，只要两台机器上建立起了连接，在本机上发送的数据就一定能传到对方的机器上，UDP就好比发电报，发出去就完事了，对方有没有接受到它都不管，多以UDP是不可靠的。TCP传送数据虽然可靠，但传送得比较慢，UDP传送数据不可靠，但是速度比较快。

Socket编程

一般的网络编程都称为socket编程，socket的英文意思为 插座。

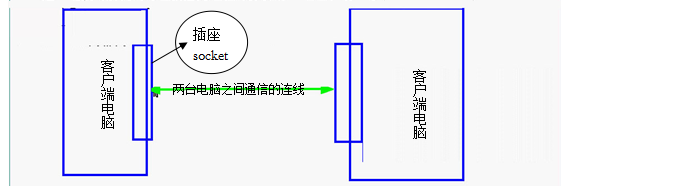
Socket

两个java应用程序可通过一个双向的网络通信连接时间数据交换，这个双向链路的一端称为一个socket。

Socket通常用来实现client-server连接。

Java.net包中定义的两个类socket serverSocket,分别用来实现双向连接的client和server端

建立连接时所需的寻址信息为远程计算机的IP地址和端口号（port number）



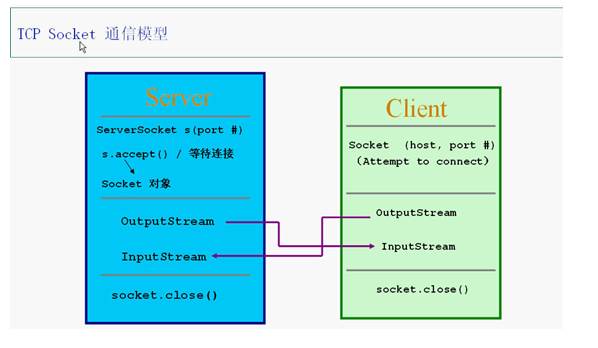
两台电脑都安装上一个插座，然后使用一根线的两端插到两台电脑的插座上，这样两台电脑就建立好了连接。这个插座就是socket。

因为互相之间都能互相通信，我说你时我的server只是从逻辑的意义上来讲，我应该把东西先发到你那里去，然后由你来处理，转发。所以你叫server。但从技术意义上来讲，只有TCP才会分server和client。对于UDP来说，从严格意义上来讲，并没有所谓的server和client。TCP的server的插座就叫serverSocket，Client的插座就叫socket。

两台计算机互相连接，那么首先必须得知道他们的IP地址，但是只提供IP地址是不够的，还必须要有连接的端口号，也就是要连接到哪个应用程序上。

端口号是用来区分一台机器上不同的应用程序的。端口号在计算机内部是占2个字节。一台机器上最多有65536个端口号。一个应用程序可以占用多个端口号如果被一个一个用程序占用了，那么其他的应用程序就无法使用这个端口号了。记住一点，我们编写的程序要占用端口号的话占用1024以上的端口号，1024一下的端口号不要去占用，因为系统有可能会随时征用。端口号本身又分为TCP端口和UDP端口，TCP的8888端口和UCP的8888端口号是完全不同的两个端口号。TCP端口号和UDP端口都有65536个。

TCP socket通信模型



Socket使用范例

服务器端ServerSocket代码如下：

import java.net.\*;

import java.io.\*;

public class TestServerSocket{

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

ServerSocket ss = new ServerSocket(6666);

/\*创建一个ServerSocket对象时往往会给它指定一个端口号

指定端口号的意思是这个new出来的ServerSocket对象要使用的

是哪一个端口号，通过哪一个端口号来监听客户端的连接

因此指定一个端口号的意义就是为了告诉计算机ServerSocket对象

在哪个地方监听客户端的连接\*/

/\*服务器端接收客户端连接的请求是不间断地接收的，所以服务器端的

编程一般都是死循环，永不休止地运行着。\*/

while(true){

Socket s = ss.accept();

/\*在服务器端调用accept()方法接受客户端的连接对象,accept()方法是

一个阻塞式方法，一直在傻傻地等待着是否有客户端申请连接上来

然后服务器端的Socket插座就和客户端的Socket插座建立了连接了\*/

/\*客户端能否连接上服务器端，取决于服务器端是否接受客户端的连接请求

如果接受了客户端的连接请求，那么在服务器端就安装上一个Socket插座

通过这个插座与连接上的客户端就可以建立连接，互相通信了\*/

System.out.println("A Client Connected!");

/\*使用InputStream流接收从客户端发送过来的信息，使用DataInputStream数据流处理接收到的信息\*/

DataInputStream dis = new DataInputStream(s.getInputStream());

/\*使用readUTF(方法将接收到的信息全部读取出来，存储到变量str里面

readUTF()方法也是一个阻塞式方法，会傻傻地等待客户端发送信息过来，然后将接收到的信息读取出来

如果客户端不写东西过来，它就一直在服务器端傻傻地等待着，直到客户端写东西过来为止

堵塞式的方法效率往往是不高的,比如说一个客户端连接上来了，但是它迟迟不发送信息，

那么服务器端的程序就阻塞住了，这样另外一个客户端就连接不上来了，因为另外一个客户端要想连接

上服务器端，就必须得在服务器端调用accept()方法，可accept()方法必须得在下一次循环时才能够被

调用，现在服务器端的程序运行到调用readUTF()这个方法时就阻塞住了，它要等待着已经连接上来的

那个客户端发送信息过来后将信息读取出来，如果客户端一直不发信息到服务器端，那么readUTF()方法

就一直无法读取到信息，那么服务器端的程序会阻塞在这里，无法进行下次循环，这样其他的客户端就

无法连接到服务器端了\*/

String str = dis.readUTF();

System.out.println(str);

}

}

}

客户端Socket代码如下：

import java.net.\*;

import java.io.\*;

public class TestClientSocket{

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

Socket s = new Socket("127.0.0.1",6666);

/\*Client申请连接到Server端上\*/

/\*连接上服务器端以后，就可以向服务器端输出信息和接收从服务器端返回的信息

输出信息和接收返回信息都要使用流式的输入输出原理进行信息的处理\*/

/\*这里是使用输出流OutputStream向服务器端输出信息\*/

OutputStream os = s.getOutputStream();

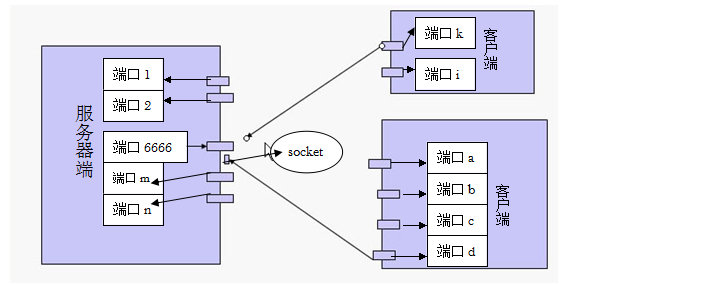
DataOutputStream dos = new DataOutputStream(os);

Thread.sleep(30000);/\*客户端睡眠30秒后再向服务器端发送信息\*/

dos.writeUTF("Hello Server!");

}

}



客户端通过端口号6666想服务器端请求连接，服务器端接受客户端的连接请求以后，就在服务器端上安装一个socket，然后让这个socket与客户端的socket连接，这样服务器端口就可以与客户端互相通信了，当有另外一个客户端申请连接时，服务器端接受了以后，又会安装另外一个socket与这个客户端的socket进行连接。