传智播客 www.itcast.cn —高级软件人才实作培训专家！

Java 预热班讲义

讲师：毕向东

网络编程

- 网络模型

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

  TCP/IP参考模型

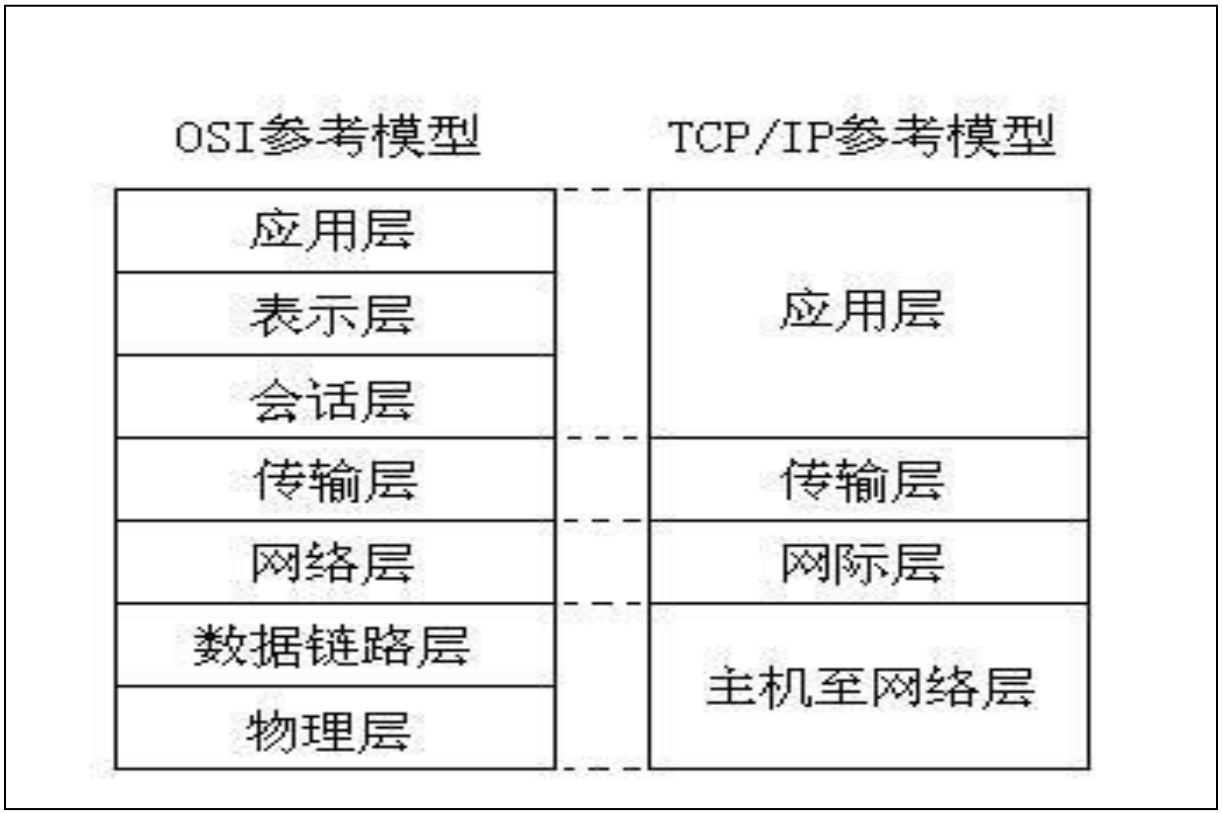
- 网络通讯要素

  IP地址

  端口号

  传输协议

网络参考模型


OSI参考模型
TCP/IP参考模型
应用层
表示层
会话层
传输层
网络层
数据链路层
物理层
应用层
传输层
网际层
主机至网络层

七层简述

1.层物理层：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,到达目的地后在转化为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.层数据链路层：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.层网络层：主要将从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.层传输层：定义了一些传输数据的协议和端口号（WWW端口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。常常把这一层数据叫做段。

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受会话请求（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能识别的东西（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）。

网络通讯要素

- 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
- 端口号
  - 用于标识进程的逻辑地址，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。
- 传输协议
  - 通讯的规则
  - 常见协议：TCP，UDP

TCP和UDP

- 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

- 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

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。

UDP传输

- DatagramSocket与DatagramPacket
- 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。

发送端

- 在发送端，要在数据包对象中明确目的地IP及端口。

```
DatagramSocket ds = new DatagramSocket();
byte[] by = “hello,udp”.getBytes();
DatagramPacket dp = new DatagramPacket(by,0,by.length,
                    InetAddress.getByName(“127.0.0.1”),10000);
ds.send(dp);
ds.close();
```

接收端

- 在接收端，要指定监听的端口。

```
DatagramSocket ds = new DatagramSocket(10000);
byte[] by = new byte[1024];
DatagramPacket dp = new DatagramPacket(by,by.length);
ds.receive(dp);
String str = new String(dp.getData(),0,dp.getLength());
System.out.println(str+"--"+dp.getAddress());
ds.close();
```

UDP聊天程序

- 通过键盘录入获取要发送的信息。
- 将发送和接收分别封装到两个线程中。

TCP传输

- 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。

基本思路（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。

基本思路（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。

客户端

- 通过Socket建立对象并指定要连接的服务端主机以及端口。

```
Socket s = new Socket("192.168.1.1",9999);
OutputStream out = s.getOutputStream();
out.write("hello".getBytes());
s.close();
```

服务端

- 建立服务端需要监听一个端口

```
ServerSocket ss = new ServerSocket(9999);
Socket s = ss.accept ();
InputStream in = s.getInputStream();
byte[] buf = new byte[1024];
int num = in.read(buf);
String str = new String(buf,0,num);
System.out.println(s.getInetAddress().toString()+":"+str);
s.close();
ss.close();
```

思考

- 对于Web服务器而言，当有多个客户端同时访问服务器时，服务端又如何提供服务呢？

Tcp传输最容易出现的问题

- 客户端连接上服务端，两端都在等待，没有任何数据传输。
- 通过例程分析：

  因为read方法或者readLine方法是阻塞式。
- 解决办法：

  自定义结束标记

  使用shutdownInput，shutdownOutput方法。

练习

- 建立一个群聊服务端。
- 客户端向服务端上传一个图片。
- 客户端向服务端发送用户名请求登陆，服务端通过验证，返回“欢迎光临”，未通过“用户不存在”