一、五层网络协议

TCP/IP 层	网络设备
应用层	
传输层	四层交换机、也有工作在四层的路由器
网络层	路由器、三层交换机
数据链路层	网桥（现已很少使用）、以太网交换机（二层交换机）、网卡（其实网卡是一半工作在物理层、一半工作在数据链路层）
物理层	中继器、集线器、还有我们通常说的双绞线也工作在物理层

应用层

与其它计算机进行通讯的一个应用，它是对应应用程序的通信服务的。例如，一个没有通信功能的字处理程序就不能执行通信的代码，从事字处理工作的程序员也不关心OSI的第7层。但是，如果添加了一个传输文件的选项，那么字处理器的程序员就需要实现OSI的第7层。示例：TELNET，HTTP,FTP,NFS,SMTP等。

传输层

这层的功能包括是否选择差错恢复协议还是无差错恢复协议，及在同一主机上对不同应用的数据流的输入进行复用，还包括对收到的顺序不对的数据包的重新排序功能。示例：TCP，UDP，SPX。

网络层

这层对端到端的包传输进行定义，它定义了能够标识所有结点的逻辑地址，还定义了路由实现的方式和学习的方式。为了适应最大传输单元长度小于包长度的传输介质，网络层还定义了如何将一个包分解成更小的包的分段方法。示例：IP,IPX等。

数据链路层

它定义了在单个链路上如何传输数据。这些协议与被讨论的各种介质有关。示例：ATM，FDDI等。

物理层

OSI的物理层规范是有关传输介质的特性标准，这些规范通常也参考了其他组织制定的标准。连接头、帧、帧的使用、电流、编码及光调制等都属于各种物理层规范中的内容。物理层常用多个规范完成对所有细节的定义。示例：Rj45，802.3等。

邓哥传信

二、七层网络协议

OSI 层	功能	TCP/IP 协议
应用层(Application layer)	文件传输，电子邮件，文件服务，虚拟终端	TFTP，HTTP，SNMP，FTP，SMTP，DNS，Telnet
表示层(Presentation layer)	数据格式化，代码转换，数据加密	没有协议
会话层(Session layer)	解除或建立与其他接点的联系	没有协议
传输层(Transport layer)	提供端对端的接口	TCP，UDP
网络层(Network layer)	为数据包选择路由	IP，ICMP，RIP，OSPF，BGP，IGMP
数据链路层(Data link layer)	传输有地址的帧，错误检测功能	SLIP，CSLIP，PPP，ARP，RARP，MTU
物理层(Physical layer)	以二进制数据形式在物理媒体上传输数据	ISO2110，IEEE802，IEEE802.2

应用层

与其它计算机进行通讯的一个应用，它是对应应用程序的通信服务的。例如，一个没有通信功能的字处理程序就不能执行通信的代码，从事字处理工作的程序员也不关心OSI的第7层。但是，如果添加了一个传输文

件的选项，那么字处理器的程序员就需要实现OSI的第7层。示例：TELNET，HTTP,FTP,NFS,SMTP等。

表示层

主要功能是定义数据格式及加密。例如，FTP允许你选择以二进制或ASCII格式传输。如果选择二进制，那么发送方和接收方不改变文件的内容。如果选择ASCII格式，发送方将把文本从发送方的字符集转换成标准的ASCII后发送数据。在接收方将标准的ASCII转换成接收方计算机的字符集。示例：加密，ASCII等。

会话层

定义了如何开始、控制和结束一个会话，包括对多个双向消息的控制和管理，以便在只完成连续消息的一部分时可以通知应用，从而使表示层看到的数据是连续的，在某些情况下，如果表示层收到了所有的数据，则用数据代表表示层。示例：RPC，SQL等。

传输层

这层的功能包括是否选择差错恢复协议还是无差错恢复协议，及在同一主机上对不同应用的数据流的输入进行复用，还包括对收到的顺序不对的数据包的重新排序功能。示例：TCP，UDP，SPX。

网络层

这层对端到端的包传输进行定义，它定义了能够标识所有结点的逻辑地址，还定义了路由实现的方式和学习的方式。为了适应最大传输单元长度小于包长度的传输介质，网络层还定义了如何将一个包分解成更小的包的分段方法。示例：IP,IPX等。

数据链路层

它定义了在单个链路上如何传输数据。这些协议与被讨论的各种介质有关。示例：ATM，FDDI等。

物理层

OSI的物理层规范是有关传输介质的特性标准，这些规范通常也参考了其他组织制定的标准。连接头、帧、帧的使用、电流、编码及光调制等都属于各种物理层规范中的内容。物理层常用多个规范完成对所有细节的定义。示例：Rj45，802.3等。

三、HTTP协议

超文本传输协议（HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议。

图 15-4　HTTP 请求报文

Accept

作用： 浏览器端可以接受的媒体类型,

例如： Accept: text/html 代表浏览器可以接受服务器回发的类型为 text/html 也就是我们常说的html文档,

如果服务器无法返回text/html类型的数据,服务器应该返回一个406错误(non acceptable)

通配符 * 代表任意类型

例如 Accept: */* 代表浏览器可以处理所有类型,(一般浏览器发给服务器都是发这个)

Accept-Encoding：

作用： 浏览器申明自己接收的编码方法，通常指定压缩方法，是否支持压缩，支持什么压缩方法（gzip，deflate），（注意：这不是只字符编码）；

例如： Accept-Encoding: zh-CN,zh;q=0.8

Accept-Language

作用： 浏览器申明自己接收的语言。
语言跟字符集的区别：中文是语言，中文有多种字符集，比如big5，gb2312，gbk等等；
例如： Accept-Language: en-us

Connection

例如： Connection: keep-alive 当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的TCP连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接
例如： Connection: close 代表一个Request完成后，客户端和服务器之间用于传输HTTP数据的TCP连接会关闭， 当客户端再次发送Request，需要重新建立TCP连接。

Host（发送请求时，该报头域是必需的）

作用: 请求报头域主要用于指定被请求资源的Internet主机和端口号，它通常从HTTP URL中提取出来的
例如: 我们在浏览器中输入：http://www.hzau.edu.cn
浏览器发送的请求消息中，就会包含Host请求报头域，如下：
Host：www.hzau.edu.cn
此处使用缺省端口号80，若指定了端口号，则变成：Host：指定端口号

Referer

当浏览器向web服务器发送请求的时候，一般会带上Referer，告诉服务器我是从哪个页面链接过来的，服务器籍此可以获得一些信息用于处理。比如从我主页上链接到一个朋友那里，他的服务器就能够从HTTP Referer中统计出每天有多少用户点击我主页上的链接访问他的网站。

User-Agent

作用：告诉HTTP服务器， 客户端使用的操作系统和浏览器的名称和版本.
我们上网登陆论坛的时候，往往会看到一些欢迎信息，其中列出了你的操作系统的名称和版本，你所使用的浏览器的名称和版本，这往往让很多人感到

很神奇，实际上，服务器应用程序就是从User-Agent这个请求报头域中获取到这些信息User-Agent请求报头域允许客户端将它的操作系统、浏览器和其它属性告诉服务器。
例如： User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; CIBA; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET4.0C; InfoPath.2; .NET4.0E)

Cache-Control

可缓存性

public
表明响应可以被任何对象（包括：发送请求的客户端，代理服务器，等等）缓存。
private
表明响应只能被单个用户缓存，不能作为共享缓存（即代理服务器不能缓存它）。
no-cache
强制所有缓存了该响应的缓存用户，在使用已存储的缓存数据前，发送带验证器的请求到原始服务器
only-if-cached
表明如果缓存存在，只使用缓存，无论原始服务器数据是否有更新。

到期

max-age=<seconds>
设置缓存存储的最大周期，超过这个时间缓存被认为过期(单位秒)。与Expires相反，时间是相对于请求的时间。
s-maxage=<seconds>
覆盖max-age 或者 Expires 头，但是仅适用于共享缓存(比如各个代理)，并且私有缓存中它被忽略。
max-stale[=<seconds>]
表明客户端愿意接收一个已经过期的资源。 可选的设置一个时间(单位秒)，表示响应不能超过的过时时间。
min-fresh=<seconds>

表示客户端希望在指定的时间内获取最新的响应。

重新验证和重新加载

must-revalidate

缓存必须在使用之前验证旧资源的状态，并且不可使用过期资源。

proxy-revalidate

与must-revalidate作用相同，但它仅适用于共享缓存（例如代理），并被私有缓存忽略。

immutable

表示响应正文不会随时间而改变。资源（如果未过期）在服务器上不发生改变，因此客户端不应发送重新验证请求头（例如If-None-Match或If-Modified-Since）来检查更新，即使用户显式地刷新页面。在Firefox中，immutable只能被用在 https:// transactions.

其他

no-store

缓存不应存储有关客户端请求或服务器响应的任何内容。

no-transform

不得对资源进行转换或转变。Content-Encoding, Content-Range, Content-Type等HTTP头不能由代理修改。例如，非透明代理可以对图像格式进行转换，以便节省缓存空间或者减少缓慢链路上的流量。 no-transform指令不允许这样做。