**（1）体系结构**

**1.1 OSI七层体系结构**

应用层

表示层

会话层

运输层

网络层

数据链路层

物理层

**1.2 TCP/IP五层体系结构**

应用层（telnet、ftp、smtp等）

运输层（tcp、udp）

网际层IP

网络结构层

**（2）运输层**

**2.1 UDP**（复用、分用、差错检验）

**特点**：

a)udp是无连接的

b)udp是尽最大努力交付，不保证可靠交付

c)udp是面向报文的：对应用层交下来的报文，在添加首部后就向下交付给IP层。既不可并，也不拆分。udp一次交付一个完整的报文，必须选择合适大小的报文。

d)udp没有拥塞控制：无时延（低）、网络拥塞会丢报文

e)udp支持一对一、一对多和多对多的通信

**note**：

复用：应用层所有的应用进程都可以通过运输层在传送到IP层

分用：运输层从IP层收到数据后必须交付给指明的应用进程

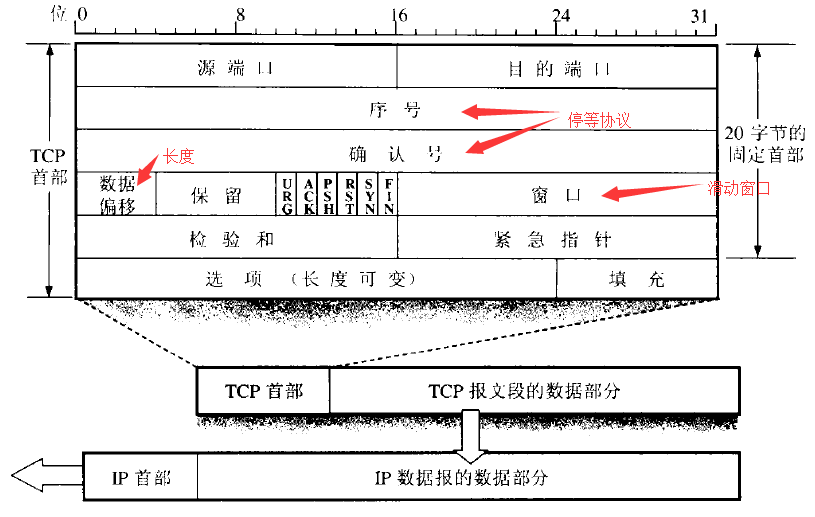
差错检验：

对象：首部和数据部分

检验步骤：a）将二进制反码求和；b）将和求反码即为检验和

**2.2 TCP**（可靠传输、流量控制、拥塞控制）

**首部格式**：



**特点**：

a）面向连接的运输层协议

b）每一条tcp连接只能有两个端点（点对点的）

c）可靠交付

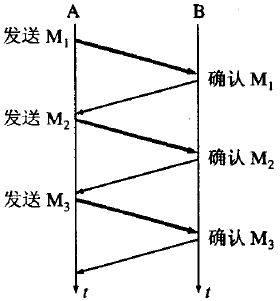
d）全双工通信

e）面向字节流

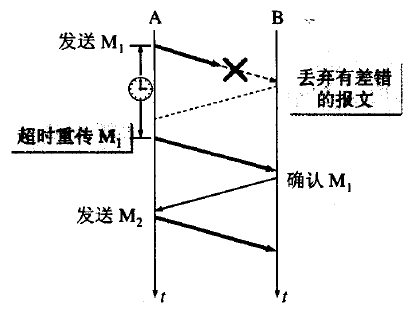
**2.2.1 可靠传输的工作原理**

**a)停等协议（自动重传请求ARQ）**

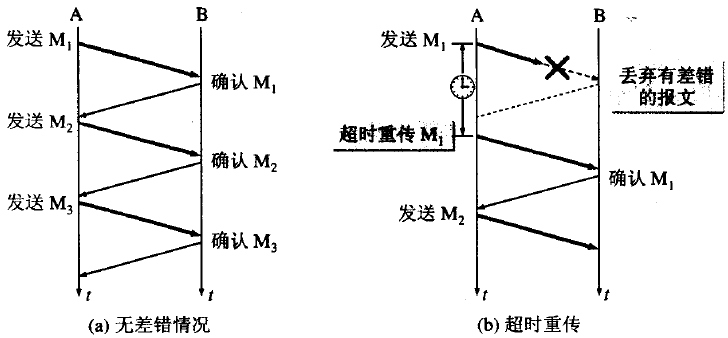
无差错：



出现差错：超时重传



确认丢失和确认迟到：



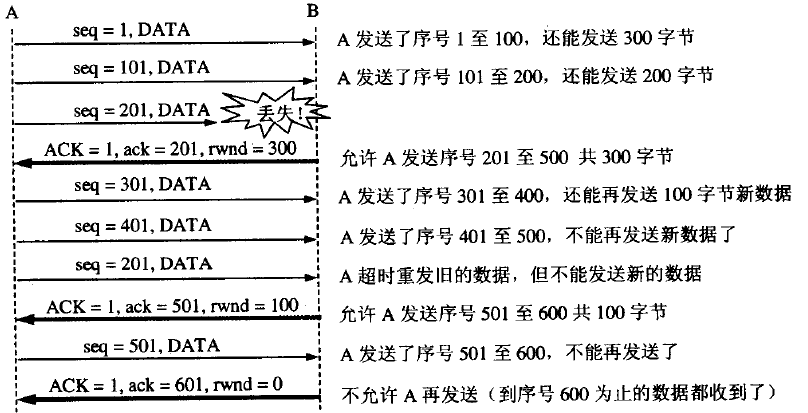
**b)连续ARQ协议**

发送发维持**发送窗口**：位于发送窗口内的n个分组都可以连续的发送出去，不需要等待对方的确认。

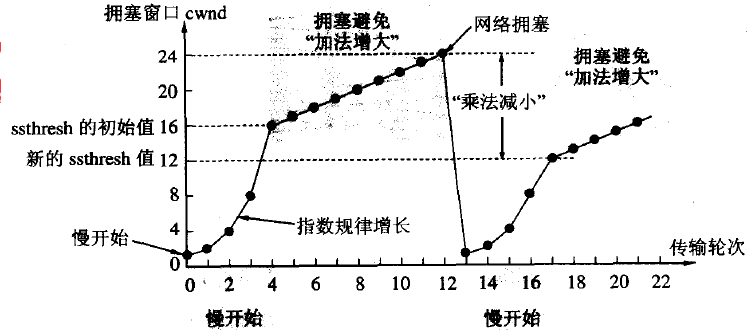
接收方**积累确认**：接收方不必对收到的分组逐个发送确认，对按序到达的最后一个分组发送确认。

**2.2.2流量控制**

利用滑动窗口实现流量控制



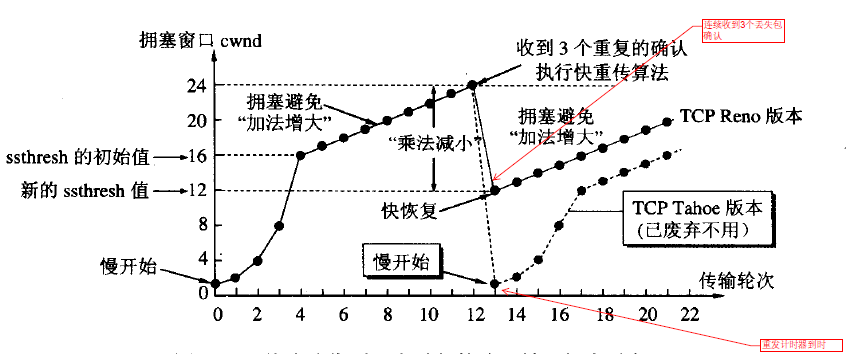
**2.2.3拥塞控制**（慢开始、拥塞避免、快重传、快恢复）



改进：

快重传：当发送方连续收到三个重复确认，就执行“乘法减小”，ssthresh /=2

快恢复：ssthresh减半，cwnd=ssthresh，加法增大

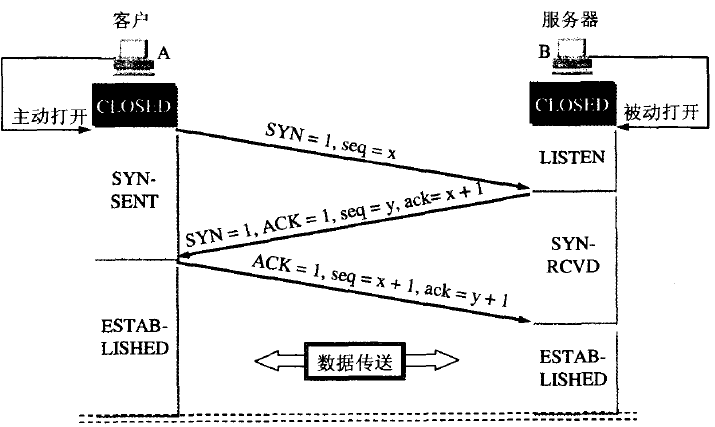


**note**：

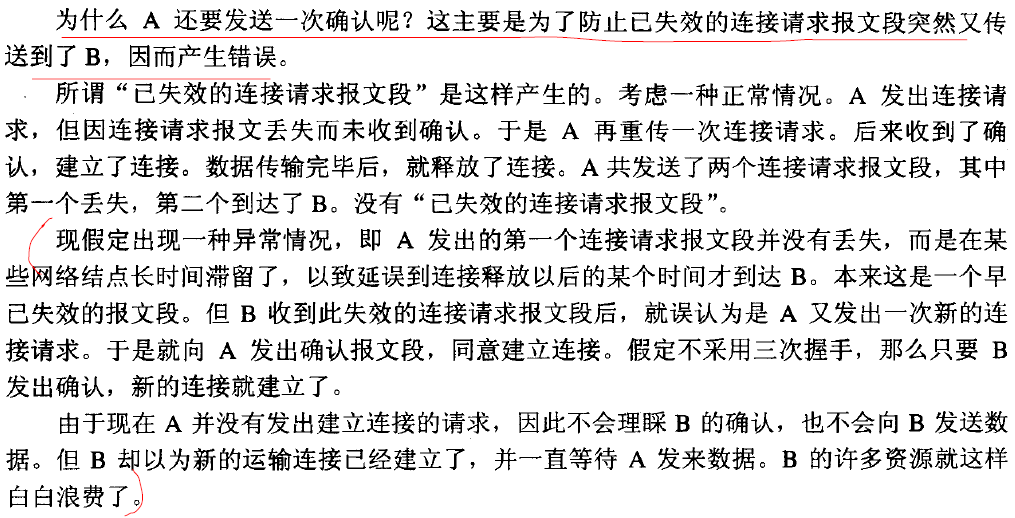
发送窗口=Min｛拥塞窗口cwnd、（对方）接受窗口｝

**2.2.4连接管理**

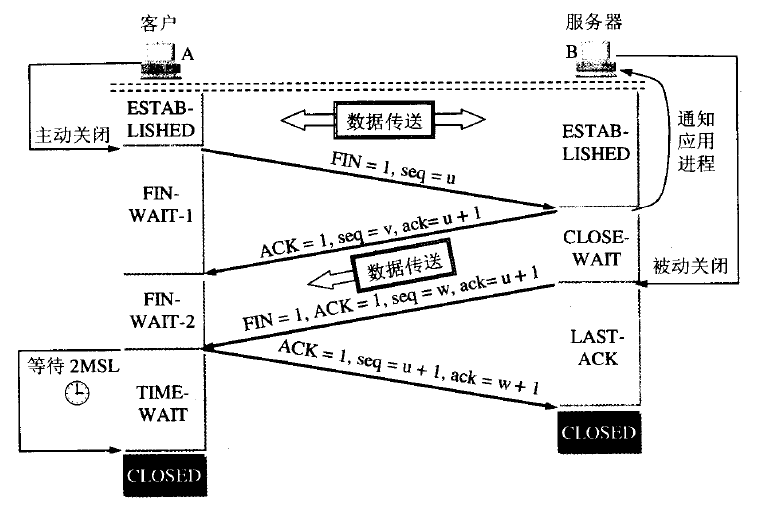
**a）建立连接（三次握手）**



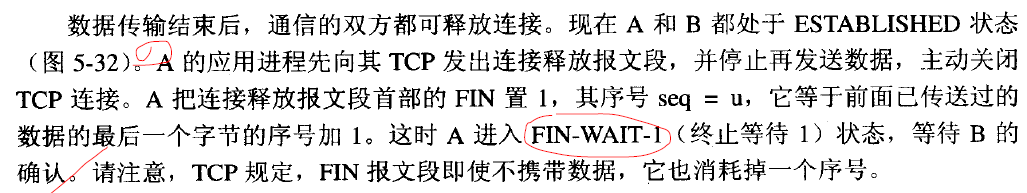
**note**：

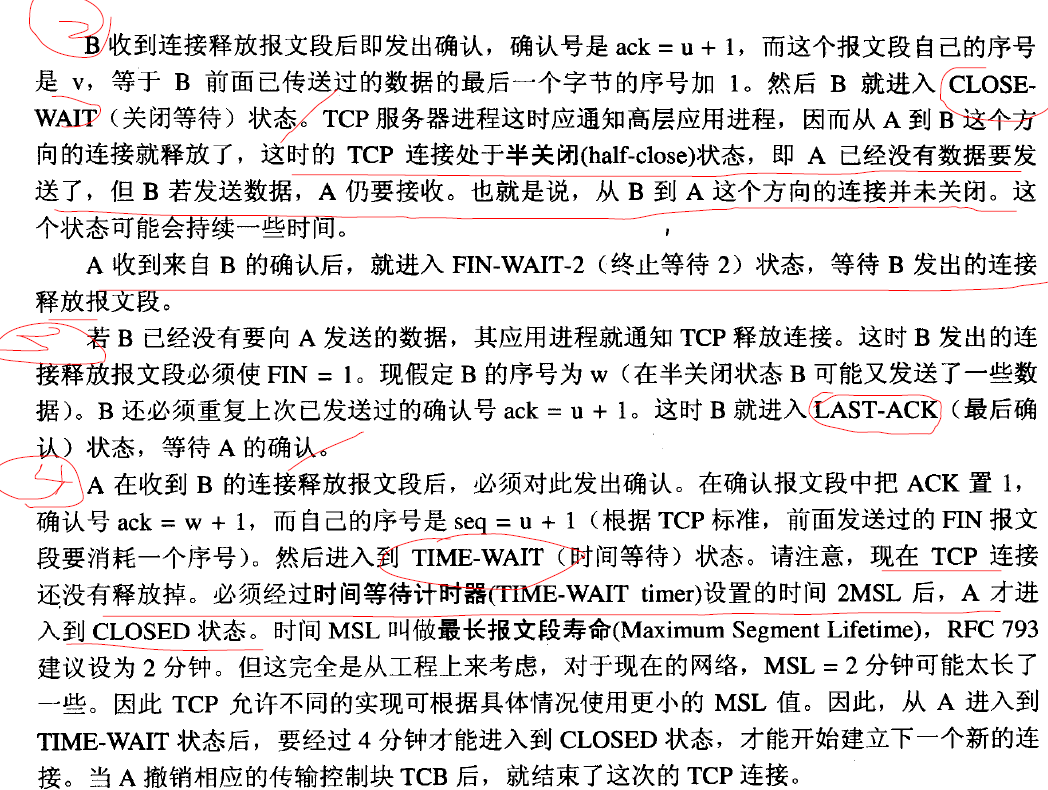


**b）连接释放（四次挥手）**

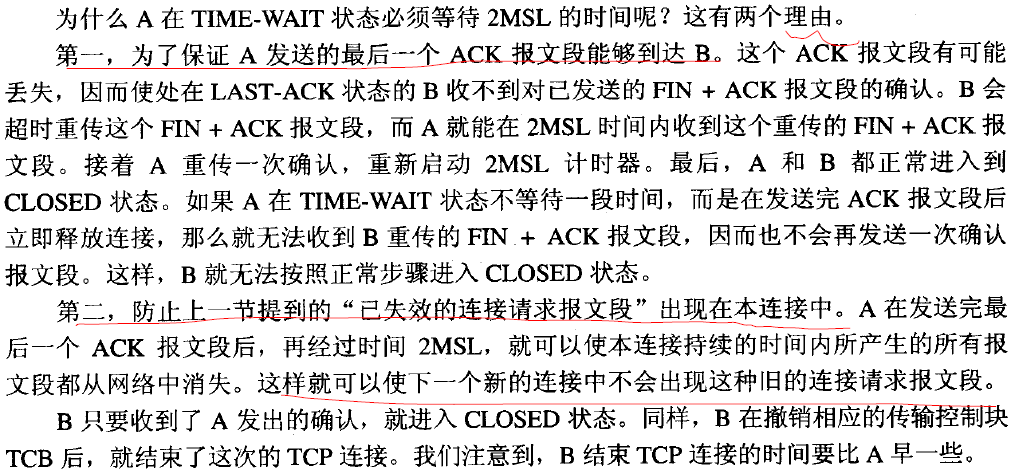


**过程解释**：



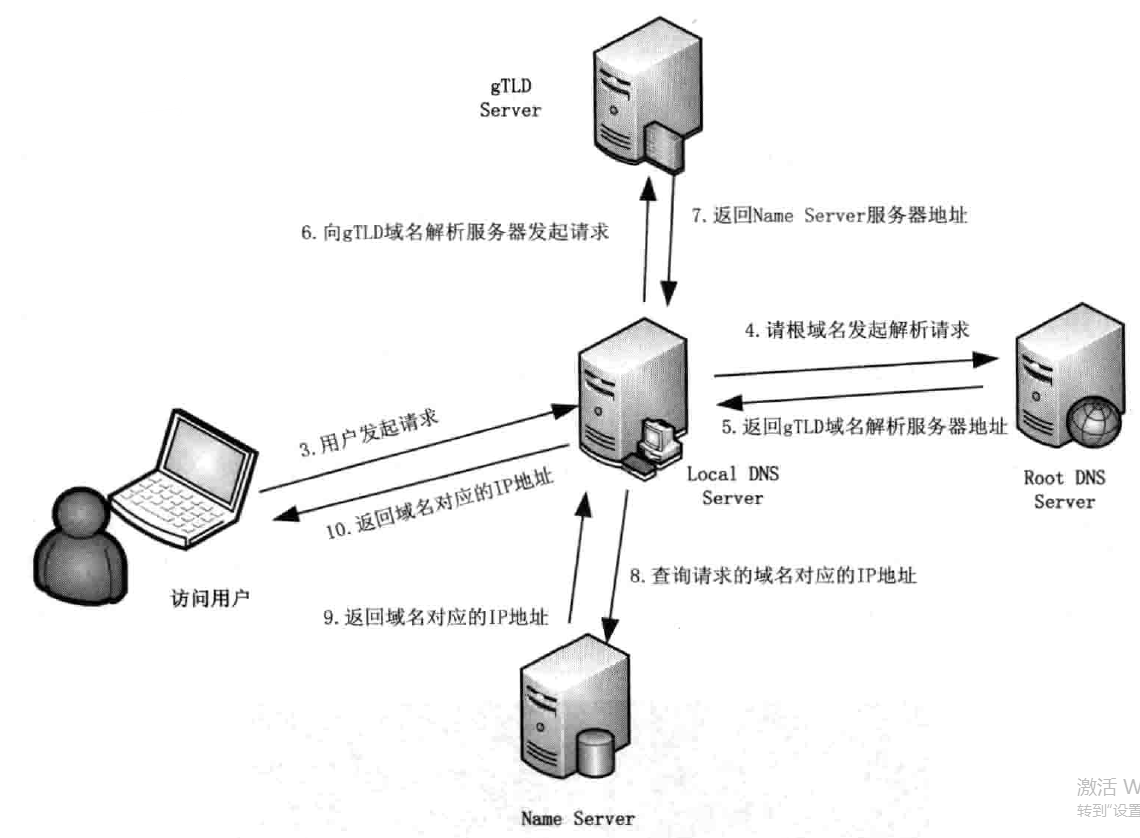


**note**：



**（3）应用层**

**3.1 DNS**



a）浏览器检查缓存中是否存在已经这个域名已经解析过的IP地址

b）浏览器查询操作系统缓存中是否存在

c）去本地域名服务器LDNS（存在缓存）请求解析

d）到Root Server域名服务器请求解析

e）根域名服务器返回给本地域名服务器一个所查询域的主域名服务器（gTLD）

f）本地域名服务器再向上异步返回的gTLD服务器发送请求

g）接受请求的gTLD服务器查找并返回此域名对应的Name Server域名服务器的地址

h）Name Server域名服务器会查询存储的域名和IP的映射关系表（返回IP和TTL）

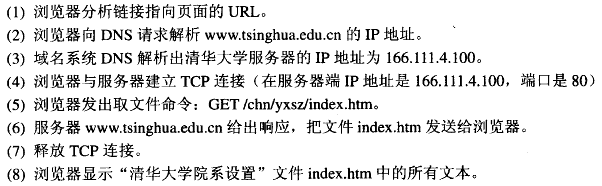
i）Local DNS Server会缓存这个域名和IP的对应关系，缓存时间由TTL值控制

j）把解析的结果返回给用户，用户根据TTL值还存在本地系统缓存中

**3.2 HTTP**

概念：HTTP 是基于 TCP/IP 协议的应用层协议。它不涉及数据包（packet）传输，主要规定了客户端和服务器之间的通信格式，默认使用80端口。

eg : 访问<http://www.tsinghua.edu.cn/chn/yxsz/index.html>的过程：



**3.2.1 http发展过程**

**HTTP/0.9**

**通信格式：**

客户端（request）：只有一个命令GET

GET /index.html （命令表示，TCP 连接（connection）建立后，客户端向服务器请求（request）网页index.html）

服务器（response）：协议规定，服务器只能回应HTML格式的字符串，不能回应别的格式，服务器发送完毕，就关闭TCP连接。

<html>

<body>Hello World</body>

</html>

**HTTP/1.0**

**简介**：

首先，任何格式的内容都可以发送。

其次，除了GET命令，还引入了POST命令和HEAD命令。

再次，HTTP请求和回应的格式也变了。除了数据部分，每次通信都必须包括头信息（HTTP header），用来描述一些元数据。

其他的新增功能还包括状态码（status code）、多字符集支持、多部分发送（multi-part type）、权限（authorization）、缓存（cache）、内容编码（content encoding）等。

**客户端-请求格式（request）**：



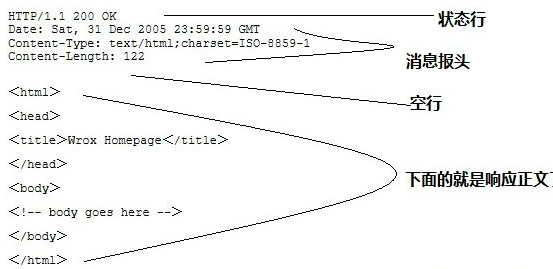
GET / HTTP/1.0

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_5)

Accept: \*/\*

解释：第一行是请求命令，必须在尾部添加协议版本（HTTP/1.0）。后面就是多行头信息，描述客户端的情况。

**服务器-响应格式（response）**：



HTTP/1.0 200 OK

Content-Type: text/plain

Content-Length: 137582

Expires: Thu, 05 Dec 1997 16:00:00 GMT

Last-Modified: Wed, 5 August 1996 15:55:28 GMT

Server: Apache 0.84

<html>

<body>Hello World</body>

</html>

解释：回应的格式是"头信息 + 一个空行（\r\n） + 数据"。其中，第一行是"协议版本 + 状态码（status code） + 状态描述"。

**请求头/响应头-字段解释**：

**Content-Type**：关于字符的编码，1.0版规定，头信息必须是 ASCII 码，后面的数据可以是任何格式。因此，服务器回应的时候，必须告诉客户端，数据是什么格式。

**Accept**：客户端请求的时候，可以使用Accept字段声明自己可以接受哪些数据格式

**Content-Encoding**：由于发送的数据可以是任何格式，因此可以把数据压缩后再发送。Content-Encoding字段说明数据的压缩方法

**Accept-Encoding**：客户端在请求时，用该字段说明自己可以接受哪些压缩方法

**缺点**：

每个TCP连接只能发送一个请求。发送数据完毕，连接就关闭，如果还要请求其他资源，就必须再新建一个连接。

TCP连接的新建成本很高，因为需要客户端和服务器三次握手，并且开始时发送速率较慢（slow start）。所以，HTTP 1.0版本的性能比较差。随着网页加载的外部资源越来越多，这个问题就愈发突出了。

**Solution**：

为了解决这个问题，有些浏览器在请求时，用了一个非标准的Connection字段。

Connection: keep-alive

字段要求服务器不要关闭TCP连接，以便其他请求复用。服务器同样回应这个字段。

Connection: keep-alive

一个可以复用的TCP连接就建立了，直到客户端或服务器主动关闭连接。但是，这不是标准字段，不同实现的行为可能不一致，因此不是根本的解决办法。

**HTTP/1.1**

**持久连接：**

TCP连接默认不关闭，可以被多个请求复用，不用声明Connection: keep-alive

**管道机制：**

在同一个TCP连接里面，客户端可以同时发送多个请求。这样就进一步改进了HTTP协议的效率

**Content-Length 字段：**

一个TCP连接现在可以传送多个回应，势必就要有一种机制，区分数据包是属于哪一个回应的。这就是Content-length字段的作用，声明本次回应的数据长度。

**其他功能：**

新增了许多动词方法：PUT、PATCH、HEAD、 OPTIONS、DELETE。

户端请求的头信息新增了Host字段，用来指定服务器的域名。有了Host字段，就可以将请求发往同一台服务器上的不同网站，为虚拟主机的兴起打下了基础。

**缺点：**

虽然1.1版允许复用TCP连接，但是同一个TCP连接里面，所有的数据通信是按次序进行的。服务器只有处理完一个回应，才会进行下一个回应。要是前面的回应特别慢，后面就会有许多请求排队等着。这称为"队头堵塞"（Head-of-line blocking）。

为了避免这个问题，只有两种方法：一是减少请求数，二是同时多开持久连接。这导致了很多的网页优化技巧，比如合并脚本和样式表、将图片嵌入CSS代码、域名分片（domain sharding）等等。如果HTTP协议设计得更好一些，这些额外的工作是可以避免的。

**HTTP请求方法**：

HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法。

HTTP1.1新增了五种请求方法： OPTIONS, PUT, DELETE, TRACE 和 CONNECT方法。



**GET和POST方法区别**：

概念：Get是向服务器发索取数据的一种请求，而Post是向服务器提交数据的一种请求。

|  |  |  |
| --- | --- | --- |
|  | GET | POST |
| 语义 | 用于信息获取 | 表示可能修改变服务器上的资源的请求 |
| 幂等性 | Yes | No |
| URL长度限制 | Yes | No |
| 参数提交方式 | URL | Request Body（表单） |
| 浏览器缓存 | Yes | No |

安全性：POST的安全性要比GET的安全性高。比如：通过GET提交数据，用户名和密码将明文出现在URL上，因为(1)登录页面有可能被浏览器缓存，(2)其他人查看浏览器的历史纪录，那么别人就可以拿到你的账号和密码了。

本质：GET和POST本质上就是TCP链接，并无差别。但是由于HTTP的规定和浏览器/服务器的限制，导致他们在应用过程中体现出一些不同。

**HTTP请求头信息**：

Accept：告诉服务器，客户端支持的数据类型。

Accept-Charset：告诉服务器，客户端采用的编码。

Accept-Encoding：告诉服务器，客户机支持的数据压缩格式。

Accept-Language：告诉服务器，客户机的语言环境。

Host：客户机通过这个头告诉服务器，想访问的主机名。

If-Modified-Since:客户机通过这个头告诉服务器，资源的缓存时间。

Referer:客户机通过这个头告诉服务器，它是从哪个资源来访问服务器的。（一般用于防盗链）

User-Agent:客户机通过这个头告诉服务器，客户机的软件环境。

Cookie：客户机通过这个头告诉服务器，可以向服务器带数据。

Connection：客户机通过这个头告诉服务器，请求完后是关闭还是保持链接。

Date：客户机通过这个头告诉服务器，客户机当前请求时间。

**HTTP响应头信息**：

Location:这个头配合302状态码使用，告诉用户端找谁。

Server:服务器通过这个头，告诉浏览器服务器的类型。

Content-Encoding:服务器通过这个头，告诉浏览器数据采用的压缩格式。

Content-Length:服务器通过这个头，告诉浏览器回送数据的长度。

Content-Language：服务器通过这个头，告诉服务器的语言环境。

Content-Type:服务器通过这个头，回送数据的类型

Last-Modified:服务器通过这个头，告诉浏览器当前资源的缓存时间。

Refresh:服务器通过这个头，告诉浏览器隔多长时间刷新一次。

Content-Disposition:服务器通过这个头，告诉浏览器以下载的方式打开数据。

Transfer-Encoding:服务器通过这个头，告诉浏览器数据的传送格式。

ETag:与缓存相关的头。

Expires:服务器通过这个头，告诉浏览器把回送的数据缓存多长时间。-1或0不缓存。

Cache-Control和Pragma：服务器通过这个头，也可以控制浏览器不缓存数据。

Connection:服务器通过这个头，响应完是保持链接还是关闭链接。

Date:告诉客户机，返回响应的时间。

**HTTP状态码**：

分类：



列表：

