（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：

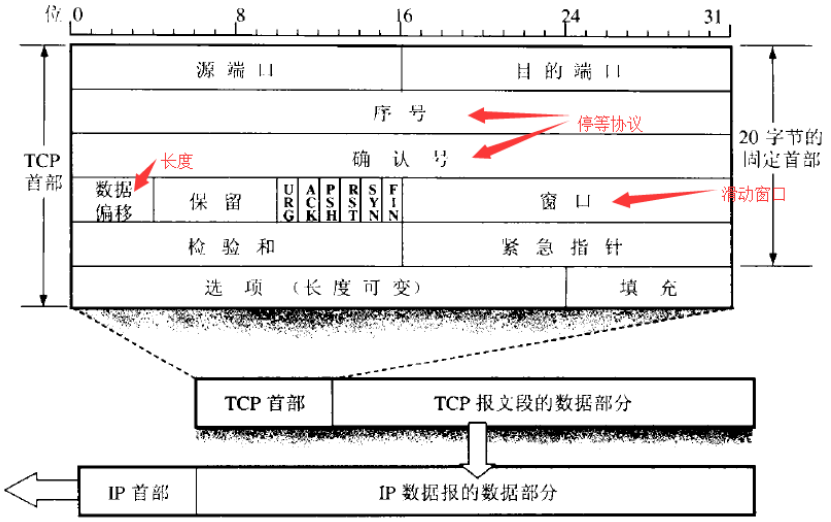
a)udp 是无连接的
b)udp 是尽最大努力交付，不保证可靠交付
c)udp 是面向报文的：对应用层交下来的报文，在添加首部后就向下交付给 IP 层。既不可并，也不拆分。udp 一次交付一个完整的报文，必须选择合适大小的报文。
d)udp 没有拥塞控制：无时延（低）、网络拥塞会丢报文
e)udp 支持一对一、一对多和多对多的通信

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

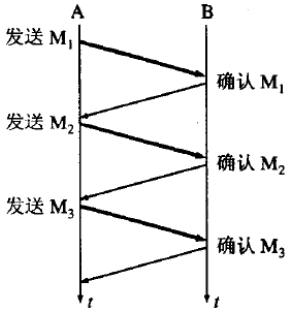
d）全双工通信

e）面向字节流

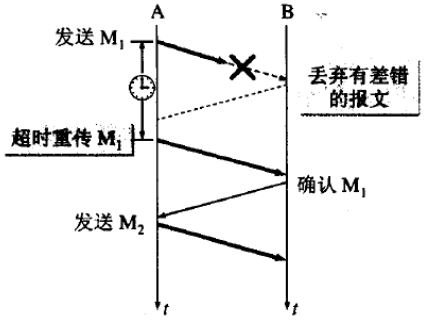
2.2.1 可靠传输的工作原理

a)停等协议（自动重传请求 ARQ）

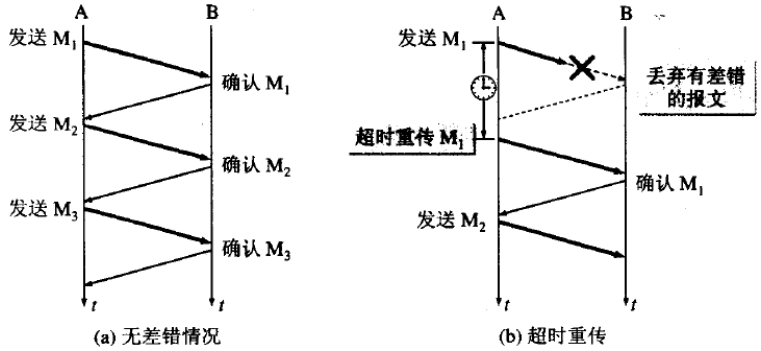
无差错：



出现差错：超时重传



确认丢失和确认迟到：



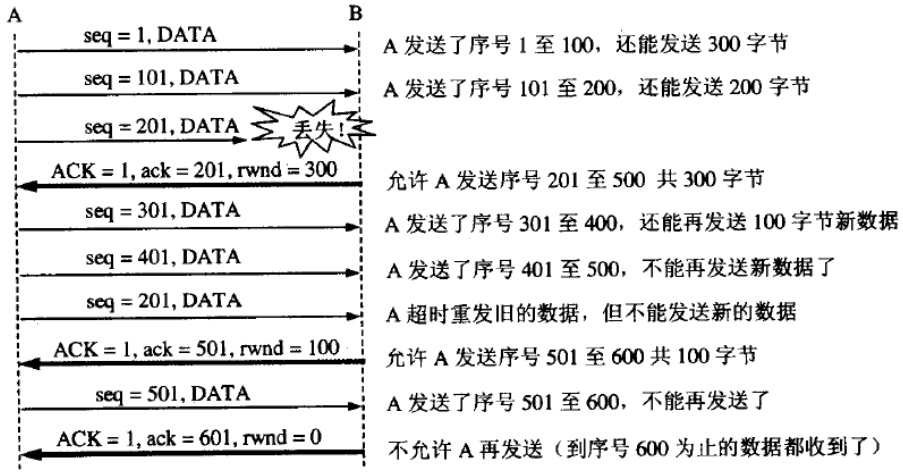
b)连续 ARQ 协议

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。

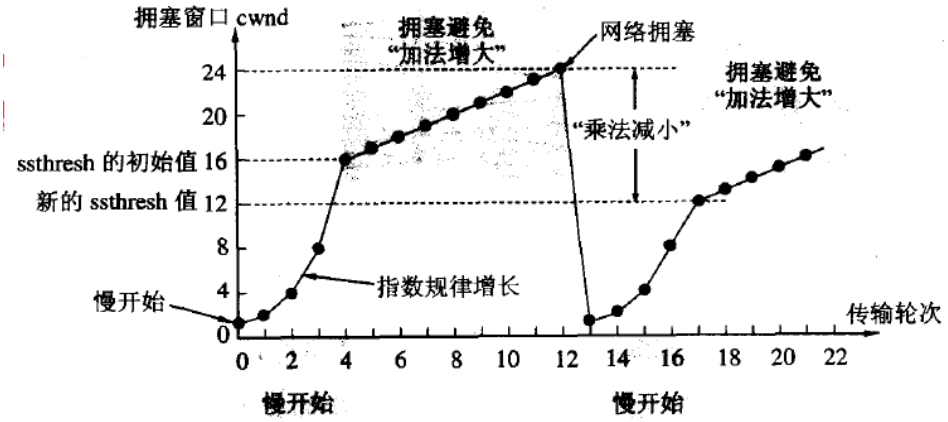
接收方**积累确认**：接收方不必对收到的分组逐个发送确认，对按序到达的最后一个分组发送确认。

2.2.2 流量控制

利用滑动窗口实现流量控制



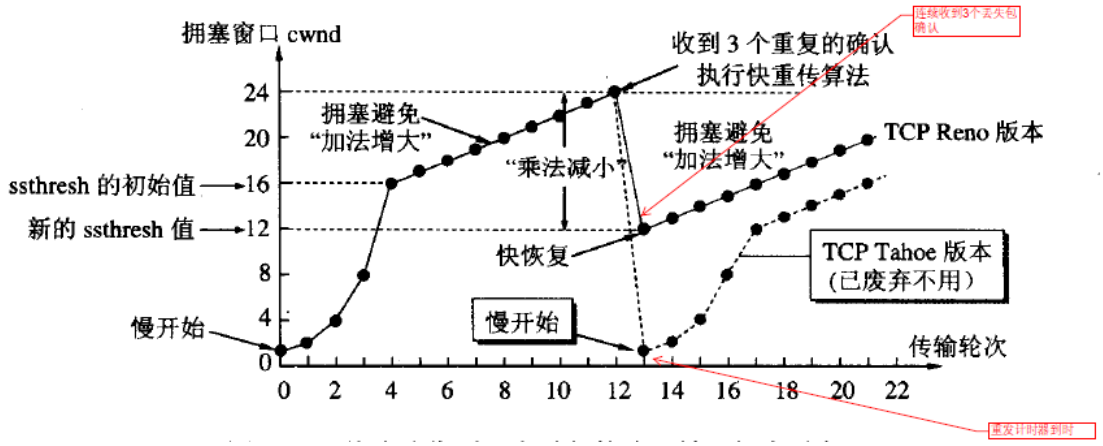
2.2.3 拥塞控制（慢开始、拥塞避免、快重传、快恢复）



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

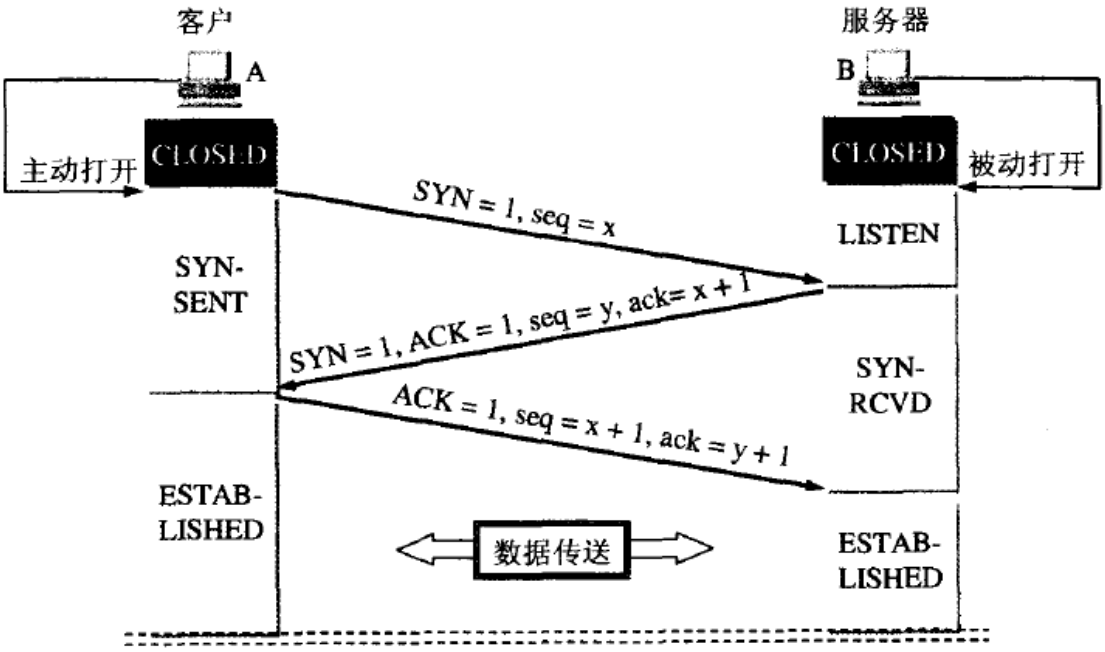


note：

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：

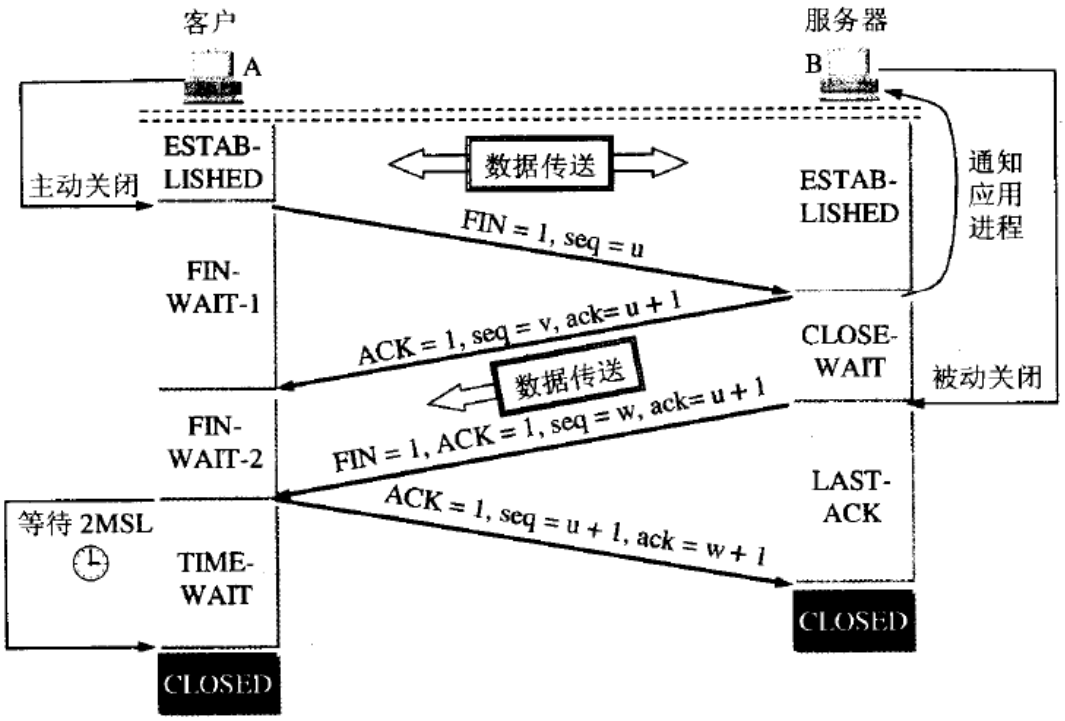
为什么 A 还要发送一次确认呢？这主要是为了防止已失效的连接请求报文段突然又传送到了 B，因而产生错误。

所谓“已失效的连接请求报文段”是这样产生的。考虑一种正常情况。A 发出连接请求，但因连接请求报文丢失而未收到确认。于是 A 再重传一次连接请求。后来收到了确认，建立了连接。数据传输完毕后，就释放了连接。A 共发送了两个连接请求报文段，其中第一个丢失，第二个到达了 B。没有“已失效的连接请求报文段”。

现假定出现一种异常情况，即 A 发出的第一个连接请求报文段并没有丢失，而是在某些网络结点长时间滞留了，以致延误到连接释放以后的某个时间才到达 B。本来这是一个早已失效的报文段。但 B 收到此失效的连接请求报文段后，就误认为是 A 又发出一次新的连接请求。于是就向 A 发出确认报文段，同意建立连接。假定不采用三次握手，那么只要 B 发出确认，新的连接就建立了。

由于现在 A 并没有发出建立连接的请求，因此不会理睬 B 的确认，也不会向 B 发送数据。但 B 却以为新的运输连接已经建立了，并一直等待 A 发来数据。B 的许多资源就这样白白浪费了。

b）连接释放（四次挥手）



过程解释：

数据传输结束后，通信的双方都可释放连接。现在 A 和 B 都处于 ESTABLISHED 状态（图 5-32）。A 的应用进程先向其 TCP 发出连接释放报文段，并停止再发送数据，主动关闭 TCP 连接。A 把连接释放报文段首部的 FIN 置 1，其序号 seq = u，它等于前面已传送过的数据的最后一个字节的序号加 1。这时 A 进入 FIN-WAIT-1（终止等待 1）状态，等待 B 的确认。请注意，TCP 规定，FIN 报文段即使不携带数据，它也消耗掉一个序号。

B 收到连接释放报文段后即发出确认，确认号是 ack = u + 1，而这个报文段自己的序号是 v，等于 B 前面已传送过的数据的最后一个字节的序号加 1。然后 B 就进入 CLOSE-WAIT（关闭等待）状态。TCP 服务器进程这时应通知高层应用进程，因而从 A 到 B 这个方向的连接就释放了，这时的 TCP 连接处于**半关闭**(half-close)状态，即 A 已经没有数据要发送了，但 B 若发送数据，A 仍要接收。也就是说，从 B 到 A 这个方向的连接并未关闭。这个状态可能会持续一些时间。

A 收到来自 B 的确认后，就进入 FIN-WAIT-2（终止等待 2）状态，等待 B 发出的连接释放报文段。

若 B 已经没有要向 A 发送的数据，其应用进程就通知 TCP 释放连接。这时 B 发出的连接释放报文段必须使 FIN = 1。现假定 B 的序号为 w（在半关闭状态 B 可能又发送了一些数据）。B 还必须重复上次已发送过的确认号 ack = u + 1。这时 B 就进入 LAST-ACK（最后确认）状态，等待 A 的确认。

A 在收到 B 的连接释放报文段后，必须对此发出确认。在确认报文段中把 ACK 置 1，确认号 ack = w + 1，而自己的序号是 seq = u + 1（根据 TCP 标准，前面发送过的 FIN 报文段要消耗一个序号）。然后进入到 TIME-WAIT（时间等待）状态。请注意，现在 TCP 连接还没有释放掉。必须经过**时间等待计时器**(TIME-WAIT timer)设置的时间 2MSL 后，A 才进入到 CLOSED 状态。时间 MSL 叫做**最长报文段寿命**(Maximum Segment Lifetime)，RFC 793 建议设为 2 分钟。但这完全是从工程上来考虑，对于现在的网络，MSL = 2 分钟可能太长了一些。因此 TCP 允许不同的实现可根据具体情况使用更小的 MSL 值。因此，从 A 进入到 TIME-WAIT 状态后，要经过 4 分钟才能进入到 CLOSED 状态，才能开始建立下一个新的连接。当 A 撤销相应的传输控制块 TCB 后，就结束了这次的 TCP 连接。

note：

为什么 A 在 TIME-WAIT 状态必须等待 2MSL 的时间呢？这有两个理由。

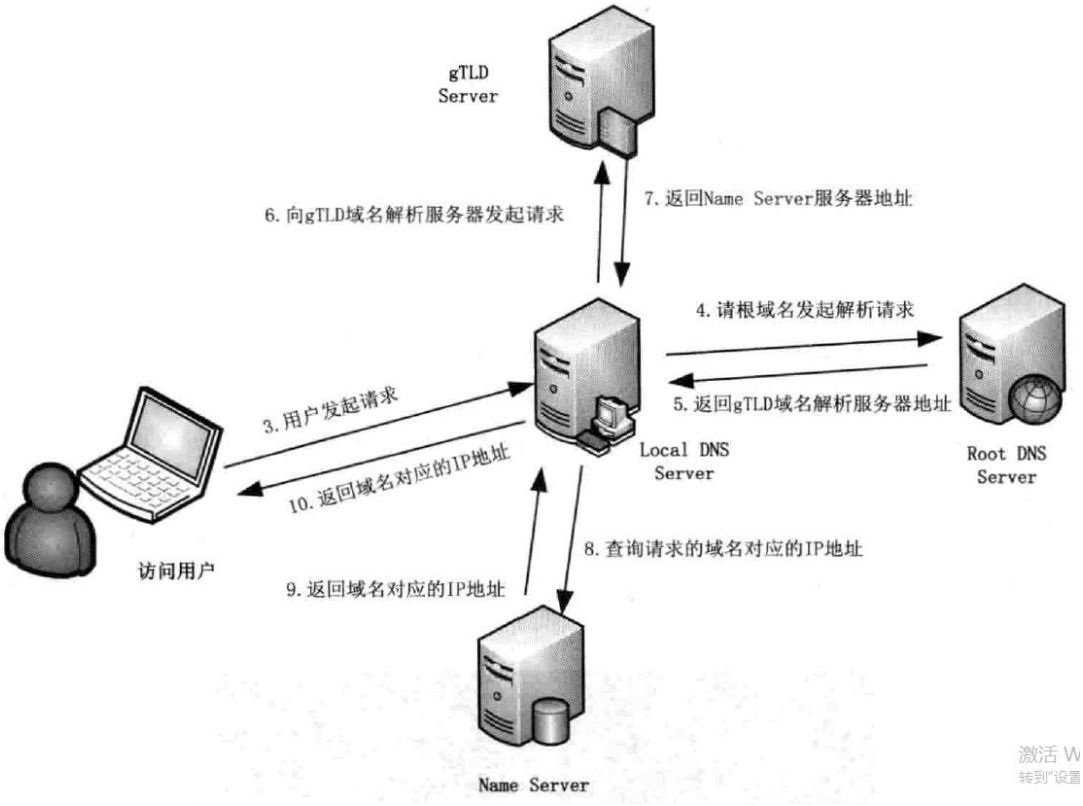
第一，为了保证 A 发送的最后一个 ACK 报文段能够到达 B。这个 ACK 报文段有可能丢失，因而使处在 LAST-ACK 状态的 B 收不到对已发送的 FIN + ACK 报文段的确认。B 会超时重传这个 FIN + ACK 报文段，而 A 就能在 2MSL 时间内收到这个重传的 FIN + ACK 报文段。接着 A 重传一次确认，重新启动 2MSL 计时器。最后，A 和 B 都正常进入到 CLOSED 状态。如果 A 在 TIME-WAIT 状态不等待一段时间，而是在发送完 ACK 报文段后立即释放连接，那么就无法收到 B 重传的 FIN + ACK 报文段，因而也不会再发送一次确认报文段。这样，B 就无法按照正常步骤进入 CLOSED 状态。

第二，防止上一节提到的“已失效的连接请求报文段”出现在本连接中。A 在发送完最后一个 ACK 报文段后，再经过时间 2MSL，就可以使本连接持续的时间内所产生的所有报文段都从网络中消失。这样就可以使下一个新的连接中不会出现这种旧的连接请求报文段。

B 只要收到了 A 发出的确认，就进入 CLOSED 状态。同样，B 在撤销相应的传输控制块 TCB 后，就结束了这次的 TCP 连接。我们注意到，B 结束 TCP 连接的时间要比 A 早一些。

（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：

(1) 浏览器分析链接指向页面的 URL。
(2) 浏览器向 DNS 请求解析 www.tsinghua.edu.cn 的 IP 地址。
(3) 域名系统 DNS 解析出清华大学服务器的 IP 地址为 166.111.4.100。
(4) 浏览器与服务器建立 TCP 连接（在服务器端 IP 地址是 166.111.4.100，端口是 80）
(5) 浏览器发出取文件命令：GET /chn/yxsz/index.htm。
(6) 服务器 www.tsinghua.edu.cn 给出响应，把文件 index.htm 发送给浏览器。
(7) 释放 TCP 连接。
(8) 浏览器显示“清华大学院系设置”文件 index.htm 中的所有文本。

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：

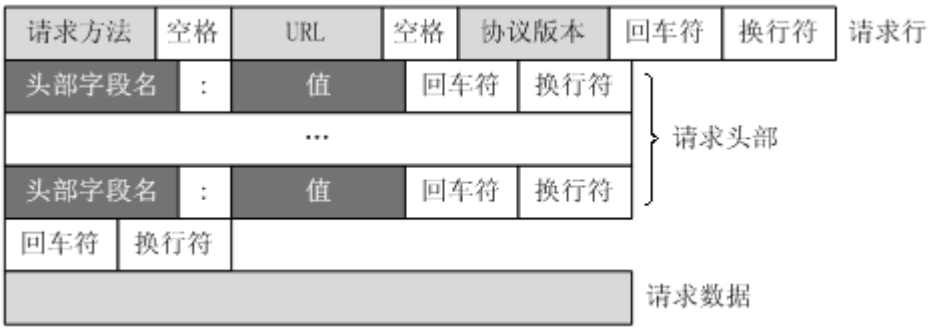
首先，任何格式的内容都可以发送。

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）：



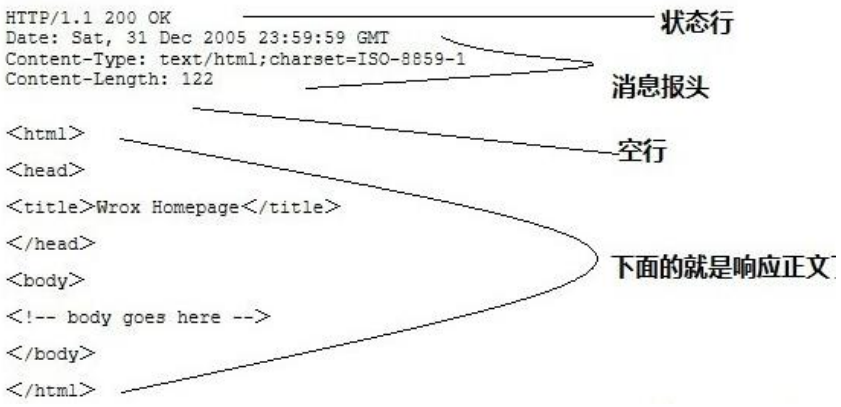
GET / HTTP/1.0

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。

序号	方法	描述
1	GET	请求指定的页面信息，并返回实体主体。
2	HEAD	类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
3	POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
4	PUT	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	请求服务器删除指定的页面。
6	CONNECT	HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
7	OPTIONS	允许客户端查看服务器的性能。
8	TRACE	回显服务器收到的请求，主要用于测试或诊断。

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：

分类	分类描述
1**	信息，服务器收到请求，需要请求者继续执行操作
2**	成功，操作被成功接收并处理
3**	重定向，需要进一步的操作以完成请求
4**	客户端错误，请求包含语法错误或无法完成请求
5**	服务器错误，服务器在处理请求的过程中发生了错误

列表：

状态码	状态码英文名称	中文描述
100	Continue	继续。客户端应继续其请求
101	Switching Protocols	切换协议。服务器根据客户端的请求切换协议。只能切换到更高级的协议，例如，切换到HTTP的新版本协议
200	OK	请求成功。一般用于GET与POST请求
201	Created	已创建。成功请求并创建了新的资源
202	Accepted	已接受。已经接受请求，但未处理完成
203	Non-Authoritative Information	非授权信息。请求成功。但返回的meta信息不在原始的服务器，而是一个副本
204	No Content	无内容。服务器成功处理，但未返回内容。在未更新网页的情况下，可确保浏览器继续显示当前文档
205	Reset Content	重置内容。服务器处理成功，用户终端（例如：浏览器）应重置文档视图。可通过此返回码清除浏览器的表单域
206	Partial Content	部分内容。服务器成功处理了部分GET请求
300	Multiple Choices	多种选择。请求的资源可包括多个位置，相应可返回一个资源特征与地址的列表用于用户终端（例如：浏览器）选择
301	Moved Permanently	永久移动。请求的资源已被永久的移动到新URI，返回信息会包括新的URI，浏览器会自动定向到新URI。今后任何新的请求都应使用新的URI代替
302	Found	临时移动。与301类似。但资源只是临时被移动。客户端应继续使用原有URI
303	See Other	查看其它地址。与301类似。使用GET和POST请求查看
304	Not Modified	未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。客户端通常会缓存访问过的资源，通过提供一个头信息指出客户端希望只返回在指定日期之后修改的资源

305	Use Proxy	使用代理。所请求的资源必须通过代理访问
306	Unused	已经被废弃的HTTP状态码
307	Temporary Redirect	临时重定向。与302类似。使用GET请求重定向
400	Bad Request	客户端请求的语法错误，服务器无法理解
401	Unauthorized	请求要求用户的身份认证
402	Payment Required	保留，将来使用
403	Forbidden	服务器理解请求客户端的请求，但是拒绝执行此请求
404	Not Found	服务器无法根据客户端的请求找到资源（网页）。通过此代码，网站设计人员可设置"您所请求的资源无法找到"的个性页面
405	Method Not Allowed	客户端请求中的方法被禁止
406	Not Acceptable	服务器无法根据客户端请求的内容特性完成请求
407	Proxy Authentication Required	请求要求代理的身份认证，与401类似，但请求者应当使用代理进行授权
408	Request Time-out	服务器等待客户端发送的请求时间过长，超时
409	Conflict	服务器完成客户端的PUT请求是可能返回此代码，服务器处理请求时发生了冲突
410	Gone	客户端请求的资源已经不存在。410不同于404，如果资源以前有现在被永久删除了可使用410代码，网站设计人员可通过301代码指定资源的新位置
411	Length Required	服务器无法处理客户端发送的不带Content-Length的请求信息

412	Precondition Failed	客户端请求信息的先决条件错误
413	Request Entity Too Large	由于请求的实体过大，服务器无法处理，因此拒绝请求。为防止客户端的连续请求，服务器可能会关闭连接。如果只是服务器暂时无法处理，则会包含一个Retry-After的响应信息
414	Request-URI Too Large	请求的URI过长（URI通常为网址），服务器无法处理
415	Unsupported Media Type	服务器无法处理请求附带的媒体格式
416	Requested range not satisfiable	客户端请求的范围无效
417	Expectation Failed	服务器无法满足Expect的请求头信息
500	Internal Server Error	服务器内部错误，无法完成请求
501	Not Implemented	服务器不支持请求的功能，无法完成请求
502	Bad Gateway	充当网关或代理的服务器，从远端服务器接收到了一个无效的请求
503	Service Unavailable	由于超载或系统维护，服务器暂时的无法处理客户端的请求。延时的长度可包含在服务器的Retry-After头信息中
504	Gateway Time-out	充当网关或代理的服务器，未及时从远端服务器获取请求
505	HTTP Version not supported	服务器不支持请求的HTTP协议的版本，无法完成处理