HTTP协议详解

1. 简介

HTTP是Hyper Text Transfer Protocol（超文本传输协议）的缩写，用于从WWW服务器传输超文本到本地浏览器的传送协议。

HTTP的第一版本HTTP/0.9是一种简单的用于网络间原始数据传输的协议；

HTTP/1.0由RFC 1945定义，在原HTTP/0.9的基础上，有了进一步的改进，允许消息以类MIME信息格式存在，包括请求/响应范式中的已传输数据和修饰符等方面的信息；

目前广泛使用的HTTP/1.1(RFC 2616)的要求更加严格，以确保服务的可靠性，增强了在HTTP/1.0没有充分考虑到分层代理服务器、高速缓冲存储器、持久连接需求或虚拟主机等方面的效能；

安全增强版的HTTP(即S-HTTP或HTTPS)，则是HTTP协议与安全套接口层(SSL)的结合，使HTTP的协议数据在传输过程中更加安全。默认HTTP的端口号为80，HTTPS的端口号为443。

2. 特点

1)基于请求响应模型，支持客户/服务器模式。

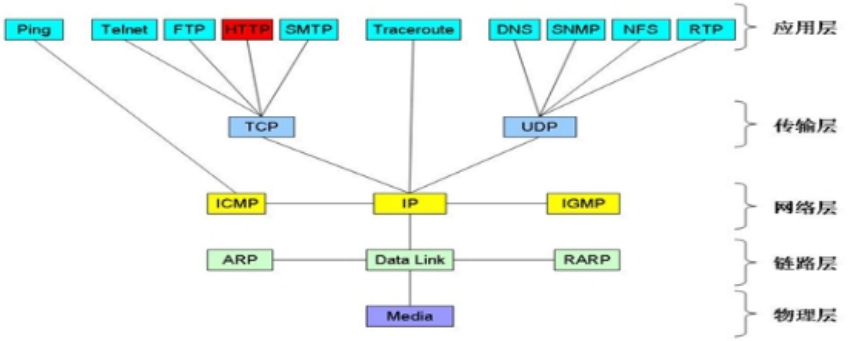
2)简单灵活：HTTP协议简单，客户向服务器请求服务时，只需传送请求方法和路径。HTTP允许传输任意类型的数据对象，正在传输的类型由Content-Type加以标记。

3)无状态：无状态是指协议对于事务处理没有记忆能力，缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大，另一方面，在服务器不需要先前信息时它的应答就较快。

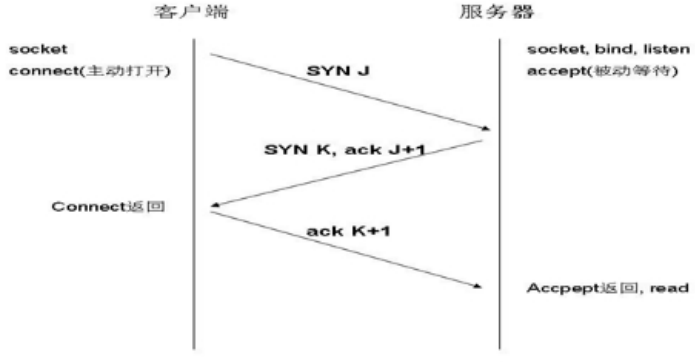
4)*面向连接*：HTTP/1.0默认使用短连接，即每次连接只处理一个请求，服务器处理完客户端请求，并收到应答后即断开连接；HTTP/1.1默认使用长连接，即完成一次请求后，客户端和服务器之间用于传输HTTP数据的TCP连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接。使用长连接的HTTP协议，会在响应头有加入以下代码：Connection: Keep-Alive。Keep-Alive不会永久保持连接，它有一个保持时间，可以在不同的服务器软件（如Apache）中设定这个时间。

3. 原理

HTTP属于应用层协议，在传输层使用TCP协议，在网络层使用IP协议。IP协议主要解决网络路由和寻址问题，TCP协议主要解决如何在IP层之上可靠的传递数据包，使在网络上的另一端收到发端发出的所有包，并且顺序与发出顺序一致。TCP有可靠，面向连接的特点。HTTP是一个无状态的面向连接的协议，无状态不代表HTTP不能保持TCP连接，更不能代表HTTP使用的是UDP协议（无连接）：

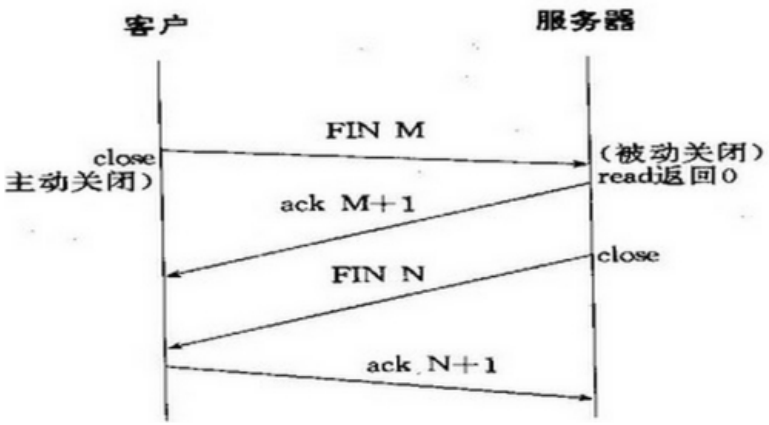


HTTP协议的长连接和短连接，实质上是TCP协议的长连接和短连接。TCP连接的建立是需要三次握手，过程如下：

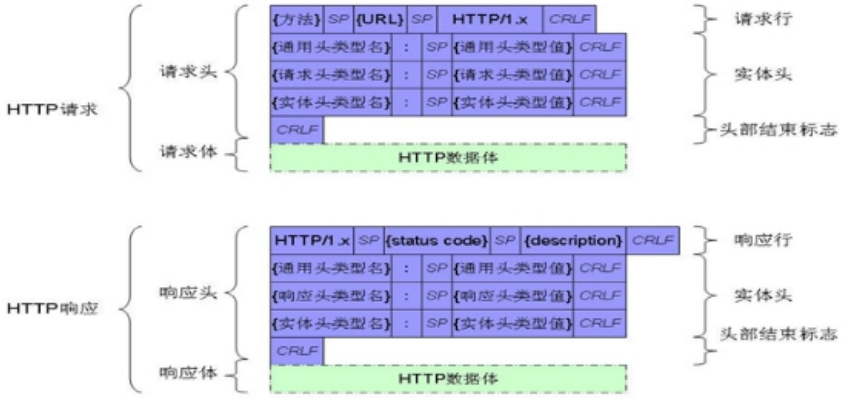


TCP连接是全双工的，每个方向都必须单独进行关闭，首先进行关闭的一方将执行主动关闭，而另一方执行被动关闭。当一方完成数

据发送后发送一个FIN来终止这个方向的连接，收到一个FIN只意味着这一方向上没有数据流动，一个TCP连接在收到一个FIN后仍能发送数据。TCP连接的释放则需要4次握手：



HTTP协议数据结构：



HTTP工作过程：

1）用户点击URL为http://www.xyz.com/index.html的链接，浏览器分析超链接中的URL

2）浏览器向DNS请求解析www.xyz.com的IP地址

3）DNS将解析出的IP地址202.2.16.21返回浏览器

4）浏览器与服务器建立TCP连接(80端口)

5）浏览器请求文档：GET /index.html

6）服务器给出响应，将文档 index.html发送给浏览器

7）释放TCP连接

8）浏览器显示index.html中的内容

4. 请求

HTTP请求由三部分组成，分别是：请求行、消息报头、请求正文。

4.1. 请求行

请求行以一个方法符号开头，以空格分开，后面跟着请求的URI和协议的版本，格式如下：Method Request-URI HTTP-Version CRLF

其中Method表示请求方法；Request-URI是一个统一资源标识符；HTTP-Version表示请求的HTTP协议版本；CRLF表示回车和换行（除了作为结尾的CRLF外，不允许出现单独的CR或LF字符）。

请求方法（所有方法全为大写）有多种：

方法名称	方法说明
GET	请求获取Request-URI所标识的资源，用于获取资源内容
POST	在Request-URI所标识的资源后附加新的数据，用于添加新的内容
HEAD	请求获取由Request-URI所标识的资源的响应消息报头，类似于GET, 但是不返回body信息，用于检查页面的状态或对象是否存在
PUT	请求服务器存储一个资源，并用Request-URI作为其标识，用于创建或修改某个内容
DELETE	请求服务器删除Request-URI所标识的资源，用于删除某个内容
TRACE	请求服务器回送收到的请求信息，主要用于测试或诊断
CONNECT	保留该方法名，用于代理进行传输，如使用SSL
OPTIONS	请求查询服务器的性能，或者查询与资源相关的选项和需求，询问可以执行哪些方法

注：GET、HEAD、PUT、DELETE、TRACE和OPTIONS都有等效(idempotence)属性,即发出上述方法的多次请求产生同样的效果，后一个请求把前一个请求覆盖掉了。

4.2. 消息报头

HTTP消息报头包括通用报头、请求报头、实体报头，每一个报头域都是由名字+“：”+空格+值组成，消息报头域的名字是大小写无关的。

通用报头(general-header)：

参数名称	参数说明
Cache-Control	客户端希望服务端如何缓存自己的请求数据，如"Cache-Control: no-cache"，"Cache-Control: max-age=0"
Connection	客户端是否希望与服务端之间保持长连接，如"Connection: close", "Connection: keep-alive"
Keep-Alive	如果浏览器请求保持长连接，则该参数表明WEB服务器保持连接多长时间（秒），例如：Keep-Alive：300
Date	只有当请求方法为POST或PUT方法时客户端才可能会有些字段
Pragma	包含了客户端一些特殊请求信息，如 "Pragma: no-cache" 客户端希望代理或应用服务器不应缓存与该请求相关的结果数据
Via	一般用在代理网关向应用服务器发送的请求头中，表明该来自客户端的请求经过了网关代理，格式为：“Via: 请求协议版本 网关标识 [其它信息] ”，如：" Via: 1.1 webcache_250_199.hexun.com:80 (squid)"

请求报头(request-header)：

参数名称	参数说明
Host	指定被请求资源的Internet主机和端口号，必需指定的参数，如果不指定端口号，则使用缺省端口号80
User-Agent	获取客户端操作系统及浏览器信息，主要用于统计的目的；仅使用该参数判断浏览器版本及类型容易出现偏差，如Opera V29.0标识如下：Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.90 Safari/537.36 OPR/29.0.1795.47，该字符串包含多种浏览器；可结合浏览器独有的属性来判断：如Firefox的getBoxObjectFor函数，Safari的openDatabase函数
Accept	指定客户端接受哪些类型的信息，如Accept：text/html，表明客户端希望接受html文本；Accept：image/gif，表明客户端希望接受gif格式的资源
Accept-Charset	客户端所能识别的字符集，格式：“Accept-Charset: 字符集1[:权重]，字符集2[:权重]”，如：“ Accept-Charset: iso-8859-5, unicode-1-1;q=0.8”
Accept-Encoding	客户端所能识别的编码压缩格式，如：“Accept-Encoding: gzip, deflate”
Accept-Language	客户端所能识别的语言，格式：“Accept-Language: 语言1[:权重]，语言2[:权重]”，如：” Accept-Language: zh, en;q=0.7”
Cookie	证明客户端有权查看某个资源，存储于客户端，向同一域名的服务端发送属于该域的cookie，如：“Cookie: MailUserName=will”
If- Modified-Since	该字段与客户端缓存相关，客户端所访问的URL自该指定日期以来在服务端是否被修改过，如果修改过则服务端返回新的修改后 的信息；如果未修改过则服务器返回304表明此请求所指URL未曾修改过，如：“If-Modified-Since: Fri, 2 Sep 2006 19:37:36 GMT”
If-None-Match	该字段与客户端缓存相关，客户端发送URL请求的同时发送该字段及标识，如果服务端的标识与客户端的标识一致，则返回304表明此URL未修改过，如果不一致则服务端返回完整的数据信息，如：“If-None-Match: 0f0a893aad8c61:253, 0f0a893aad8c61:252, 0f0a893aad8c61:251”

实体报头(entity-header) (此类头存在时要求有数据体)：

参数名称	参数说明
Content-Encoding	客户端所能识别的编码压缩格式，如：“Content-Encoding: gzip, deflate”
Content-Length	客户端以POST方法上传数据时数据体部分的内容长度，如：“ Content-Length: 24”
Content- Type	客户端发送的数据体的内容类型，如：“Content-Type: application/x-www-form-urlencoded”为普通的POST方法发送的数据；“Content-Type: multipart/form-data; boundary=--5169208281820”，则表明数据体由多部分组成，分隔符为 “--5169208281820”

4.3. 请求正文

请求头和请求正文之间是一个空行，它表示请求头已经结束，接下来的是请求正文。请求正文取决于请求方法，如客户端通过POST方法提交的数据信息即在请求正文中。

4.4. 请求报头实例

Host：rss.sina.com.cn

User-Agent：Mozilla/5、0 (Windows; U; Windows NT 5、1; zh-CN; rv:1、8、1、14) Gecko/20080404 Firefox/2、0、0、14

Accept：text/xml,application/xml,application/xhtml+xml,text/html;q=0、9,text/plain;q=0、8,image/png,*/*;q=0、5

Accept-Language：zh-cn,zh;q=0、5

Accept-Encoding：gzip,deflate

Accept-Charset：gb2312,utf-8;q=0、7,*;q=0、7

Keep-Alive：300

Connection：keep-alive

Cookie：userId=C5bYpXrimdmsiQmsBPnE1Vn8ZQmdWSm3WRlEB3vRwTnRtW <-- Cookie

If-Modified-Since：Sun, 01 Jun 2008 12:05:30 GMT

Cache-Control：max-age=0

5. 响应

在接收和解释请求消息后，服务器返回一个HTTP响应消息。HTTP响应也是由三个部分组成，分别是：状态行、消息报头、响应正文。

5.1. 状态行

状态行格式为HTTP-Version Status-Code Reason-Phrase CRLF

其中，HTTP-Version表示服务器HTTP协议的版本；Status-Code表示服务器发回的响应状态代码；Reason-Phrase表示状态代码的文本描述，例如HTTP/1.1 200 OK（CRLF）。

状态代码有三位数字组成，第一个数字定义了响应的类别，且有五种可能取值：

1xx：指示信息--表示请求已接收，继续处理

2xx：成功--表示请求已被成功接收、理解、接受

3xx：重定向--要完成请求必须进行更进一步的操作

4xx：客户端错误--请求有语法错误或请求无法实现

5xx：服务器端错误--服务器未能实现合法的请求

常见状态代码说明：

状态代码	状态描述	状态说明
200	OK	客户端请求成功
400	Bad Request	客户端请求有语法错误，不能被服务器所理解
401	Unauthorized	请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用
403	Forbidden	服务器收到请求，但是拒绝提供服务
404	Not Found	请求资源不存在，如：输入了错误的URL
500	Internal Server Error	服务器发生不可预期的错误
503	Server Unavailable	服务器当前不能处理客户端的请求，一段时间后可能恢复正常

5.2. 消息报头

HTTP消息报头包括通用报头、响应报头、实体报头，每一个报头域都是由名字+“：”+空格+值 组成，消息报头域的名字是大小写无关的。

通用报头(general-header)：

参数名称	参数说明
Cache-Control	服务端要求中间代理及客户端如何缓存自己响应的数据，如“Cache-Control: no-cache”，如：“Cache-Control: private” 不希望被缓存，“Cache-Control: public” 可以被缓存
Connection	服务端是否希望与客户端之间保持长连接，如“Connection: close”, “Connection: keep-alive”
Date	只有当请求方法为POST或PUT方法时客户端才可能会有些字段
Pragma	包含了服务端一些特殊响应信息，如 “Pragma: no-cache” 服务端希望代理或客户端不应缓存结果数据
Transfer-Encoding	服务端向客户端传输数据所采用的传输模式(仅在HTTP1.1中出现)，如：“Transfer-Encoding: chunked”，注：该字段的优先级要高于“Content-Length”字段的优先级
Via	表明该来自服务器的响应经过了网关代理，格式为："Via: 响应协议版本 网关标识 [其它信息] "，如：" Via: 1.0 36.D05.sina.com:80 (squid/2.6.STABLE13)"

响应报头(response-header)：

参数名称	参数说明
Accept-Ranges	表明服务端接收的数据单位，如：“Accept-Ranges: bytes”
Age	当代理服务器用自己缓存的实体去响应请求时，该参数表明该实体从产生到现在经过多长时间了
Location	服务端向客户端返回此信息以使客户端进行重定向，如：“Location: http://www.hexun.com”
Server	服务端返回的用于标识自己的一些信息，如：“ Server: Microsoft-IIS/6.0”
ETag	服务端返回响应数据的标识字段，客户端可根据此字段的值向服务器发送某URL是否更新的信息，ETag的作用跟Last-Modified 的作用差不多，主要供WEB服务器判断一个对象是否改变了
Last-Modified	WEB服务器认为对象的最后修改时间，比如文件的最后修改时间，动态页面的最后产生时间等等。例如：Last-Modified：Tue, 06 May 2008 02:42:43 GMT
Expired	WEB服务器表明该实体将在什么时候过期，对于过期了的对象，只有在跟WEB服务器验证了其有效性后，才能用来响应客户请求，是 HTTP/1.0的头部。例如：Expires：Sat, 23 May 2009 10:02:12 GMT

实体报头(entity-header) (此类头存在时要求有数据体)：

参数名称	参数说明
Content-Encoding	服务端所响应数据的编码格式，如：“Content-Encoding: gzip”
Content-Length	服务端所返回数据的数据体部分的内容长度，如：“ Content-Length: 24”
Content- Type	服务端所返回的数据体的内容类型，如：“Content-Type: text/html; charset=gb2312”
Set-Cookie	服务端返回给客户端的cookie数据，如：“Set-Cookie: JSessionId=icnh2ku2dqlmkciyobgvzl55; path=/”

5.3. 响应正文

响应头和响应正文之间也必须用空行分隔，响应正文就是服务器返回的资源的内容。响应正文既取决于请求方法，也取决于响应状态码。

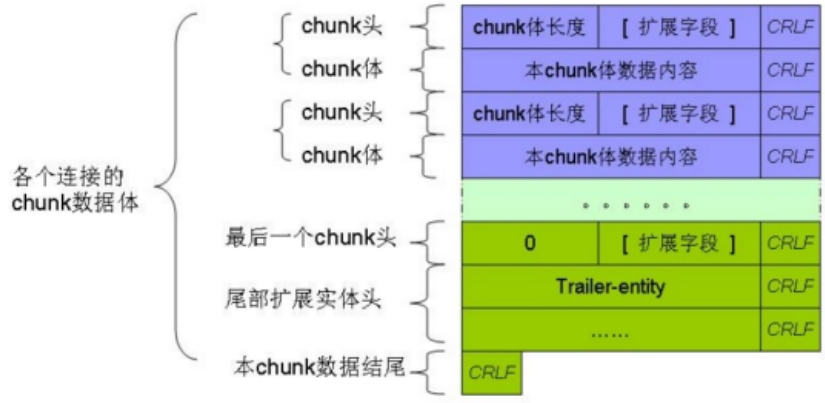
5.4. 响应报头实例

Status：OK - 200 <-- 响应状态码，表示 web 服务器处理的结果。
Date：Sun, 01 Jun 2008 12:35:47 GMT
Server：Apache/2、0、61 (Unix)
Last-Modified：Sun, 01 Jun 2008 12:35:30 GMT
Accept-Ranges：bytes
Content-Length：18616
Cache-Control：max-age=120
Expires：Sun, 01 Jun 2008 12:37:47 GMT
Content-Type：application/xml
Age：2
X-Cache：HIT from 236-41、D07071951、sina、com、cn <-- 反向代理服务器使用的 HTTP 头部
Via：1.0 236-41.D07071951.sina.com.cn:80 (squid/2.6.STABLE13)
Connection：close

6. chunked传输

当客户端向服务器请求一个静态页面或者一张图片时，服务器可以很清楚的知道内容大小，然后通过Content-length消息首部字段告诉客户端需要接收多少数据。但是如果是动态页面等时，服务器是不可能预先知道内容大小，这时就可以使用Transfer-Encoding：chunk模式来传输数据了。即如果要一边产生数据，一边发给客户端，服务器就需要使用"Transfer-Encoding: chunked"这样的方式来代替Content-Length。

编码使用若干个Chunk组成，由一个标明长度为0的chunk结束，每个Chunk有两部分组成，第一部分是该Chunk的长度(以十六进制表示)和长度单位（一般不写），第二部分就是指定长度的内容，每个部分用CRLF隔开。在最后一个长度为0的chunk中的内容是称为footer的内容，是一些没有写的头部内容。另外，在HTTP头里必须含有：” Transfer-Encoding: chunked”通用头字段。Chunk编码的格式如下：



7. restful风格

REST(Representational State Transfer)是网络服务接口的一种风格，并不是一个标准。通过 REST风格体系架构，请求和响应都是基于资源表示的传输来构建的。资源是通过全局ID 来标识的，这些ID一般使用一个统一资源标识符（URI）。客户端应用使用 HTTP 方法（如GET、POST、PUT 或 DELETE）来操作一个或多个资源。通常，GET是用于获取或列出一个或多个资源，POST用于创建，PUT用于更新或替换，而DELETE则用于删除资源。如上所述，PUT和POST的区别是前者是等效的，而后者则没有该属性。如果多次使用只产生了一个结果，即每次使用后一个请求总是把前一个请求覆盖掉了，应该使用PUT，否则应使用POST。

基于 REST 的架构风格，人们把它使用到了 Web 服务中。在目前主流的三种 Web 服务实现方案中，RESTful 的 Web 服务比基于 SOAP 和 XML-RPC 方式的 Web 服务更加简洁高效。它直接使用 HTTP 协议就可以实现 Web 服务，不需要额外的封装协议和远程进程的调用。资源的表现形式可以是 HTML，也可以是 XML，JSON 等其他数据形式，这取决于 Web 服务的消费者是人还是机器。

REST主要有以下特点：

· 从资源的角度来考察整个网络，每个资源有唯一标识
· 使用通用的连接器接口操作资源
· 对资源的操作不会改变资源标识
· 连接协议具有无状态性
· 能够使用 Cache 机制来增进性能

HTTP 请求在 RESTful Web 服务中的典型应用

资源	GET	PUT	POST	DELETE
一组资源http://www.abc.com/resources/	列出 URI 及该资源组中每个资源的详细信息	使用一组给定的资源替换当前整组资源	在本组资源中创建/追加一个新资源	删除整组资源
单个资源http://www.abc.com/resources/1	获取给定资源的详细信息	替换/创建指定的资源，并将其追加到相应的资源组	把指定的资源作为资源组，并在其下创建/追加一个新元素，使其隶属于当前资源	删除指定元素