Hillstone Networks Inc.

HTTP 协议介绍

v1.0

目录

1 HTTP 协议基础

1.1 HTTP 协议介绍

HTTP（HyperText Transfer Protocol），中文解释为超文本传输协议。它是一种基于应用层的请求/响应形式的协议。

它的作用就是在 www 系统（可以理解成为浏览者与 Internet 之间交互信息的一种服务）上存取数据。通俗来讲，HTTP 协议使客户端通过固定统一的协议标准来发送和接收数据包以使用服务器端提供的 web 服务。

在 1990 年，HTTP 就成为 WWW 的支撑协议。当时由其创始人 WWW 之父蒂姆 •贝纳斯 •李（TimBemers—Lee）提出，随后 WWW 联盟（WWW Consortium）成立，组织了 IETF（Internet Engineering Task Force）小组进一步完善和发布 HTTP 协议。

HTTP 是应用层协议，同其他应用层协议一样，是为了实现某一类具体应用的协议，并由某一运行在用户空间的应用程序来实现其功能。HTTP 是一种协议规范，这种规范记录在文档上，为真正通过 HTTP 协议进行通信的 HTTP 的实现程序。

HTTP 协议是基于 C/S（即 client：客户端 server：服务器端）架构进行通信的，而 HTTP 协议的服务器端实现程序有 httpd、nginx 等，其客户端的实现程序主要是 Web 浏览器，例如 Firefox、InternetExplorer、Google chrome、Safari、Opera 等，此外，客户端的命令行工具还有 elink、crul 等。

Web 服务是基于 TCP 的，因此为了能够随时响应客户端的请求，Web 服务器需要监听处于 tcp/ip 模型下的传输层 80/TCP 端口。这样客户端浏览器和 Web 服务器之间就可以通过 HTTP 协议传输数数据进行通信了。

1.1.1HTTP 版本

1）HTTP/0.9

0.9 协议是适用于各种数据信息的简洁快速协议，但是远不能满足日益发展的各种应用的需要。0.9 协议就是一个交换信息的无序协议，仅仅限于文字。由于无法进行内容的协商，在双发的握手和协议中，并有规定双发的内容是什么，

也就是图片是无法显示和处理的。

最早版本是 1991 年发布的 0.9 版。该版本极其简单，只有一个命令 GET：GET /index.html

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

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：

Hello World

服务器发送完毕，就关闭 TCP 连接。

2）HTTP/1.0

HTTP/1.0 是最重要的面向事务的应用层协议。该协议对每一次请求/响应建立并拆除一次连接。其特点是简单、易于管理，所以它符合了大家的需要，得到了广泛的应用。

首先，任何格式的内容都可以发送。这使得互联网不仅可以传输文字，还能传输图像、视频、二进制文件。这为互联网的大发展奠定了基础。

其次，除了 GET 命令，还引入了 POST 命令和 HEAD 命令，丰富了浏览器与服务器的互动手段。

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。

3）HTTP/1.1

在 1.0 协议中，双方规定了连接方式和连接类型，这已经极大扩展了 HTTP 的领域，但对于互联网最重要的速度和效率，并没有太多的考虑。毕竟，作为协议的制定者，当时也没有想到 HTTP 协议会有那么快的普及速度关于 HTTP1.1 协议的具体内容可以参考 RFC 2616。

持久连接

1.1 版的最大变化，就是引入了持久连接（persistent connection），即 TCP 连接默认不关闭，可以被多个请求复用，不用声明 Connection: keep-alive。

客户端和服务器发现对方一段时间没有活动，就可以主动关闭连接。

不过，规范的做法是，客户端在最后一个请求时，发送 Connection: close，

明确要求服务器关闭 TCP 连接：

Connection: close

目前，对于同一个域名，大多数浏览器允许同时建立 6 个持久连接。

管道机制

1.1 版还引入了管道机制（pipelining），即在同一个 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。

举例来说，客户端需要请求两个资源。以前的做法是，在同一个 TCP 连接里面，先发送 A 请求，然后等待服务器做出回应，收到后再发出 B 请求。管道机制则是允许浏览器同时发出 A 请求和 B 请求，但是服务器还是按照顺序，先回应 A 请求，完成后再回应 B 请求。

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：Content-Length: 3495。上面代码告诉浏览器，本次回应的长度是 3495 个字节，后面的字节就属于下一个回应了。

在 1.0 版中，Content-Length 字段不是必需的，因为浏览器发现服务器关闭了 TCP 连接，就表明收到的数据包已经全了。

分块传输编码

使用 Content-Length 字段的前提条件是，服务器发送回应之前，必须知道回应的数据长度。对于一些很耗时的动态操作来说，这意味着，服务器要等到所有操作完成，才能发送数据，显然这样的效率不高。更好的处理方法是，产生一块数据，就发送一块，采用"流模式"（stream）取代"缓存模式"（buffer）。

因此，1.1 版规定可以不使用 Content-Length 字段，而使用"分块传输编码"（chunked transfer encoding）。

只要请求或回应的头信息有 Transfer-Encoding 字段，就表明回应将由数量未定的数据块组成：

Transfer-Encoding: chunked

每个非空的数据块之前，会有一个 16 进制的数值，表示这个块的长度。最后是一个大小为 0 的块，就表示本次回应的数据发送完了。

下面是一个例子：

HTTP/1.1 200 OK

Content-Type: text/plain

Transfer-Encoding: chunked

25

This is the data in the first chunk

1C

and this is the second one

3

con

8

sequence

0

其他功能

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

另外，客户端请求的头信息新增了 Host 字段，用来指定服务器的域名：

Host: example.com

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。

4）2.0

HTTP2.0 的前世是 HTTP1.0 和 HTTP1.1。虽然之前仅仅只有两个版本，但这两个版本所包含的协议规范之庞大，足以让任何一个有经验的工程师为之头疼。网络协议新版本并不会马上取代旧版本。实际上，1.0 和 1.1 在之后很长的一段时间内一直并存，这是由于网络基础设施更新缓慢所决定的。关于 HTTP2.0 协

议的具体内容可以参考 RFC 7540。

二进制协议

HTTP/1.1 版的头信息肯定是文本（ASCII 编码），数据体可以是文本，也可以是二进制。HTTP/2 则是一个彻底的二进制协议，头信息和数据体都是二进制，并且统称为"帧"（frame）：头信息帧和数据帧。

二进制协议的一个好处是，可以定义额外的帧。HTTP/2 定义了近十种帧，为将来的高级应用打好了基础。如果使用文本实现这种功能，解析数据将会变得非常麻烦，二进制解析则方便得多。

多工

HTTP/2 复用 TCP 连接，在一个连接里，客户端和浏览器都可以同时发送多个请求或回应，而且不用按照顺序一一对应，这样就避免了"队头堵塞"。

举例来说，在一个 TCP 连接里面，服务器同时收到了 A 请求和 B 请求，于是先回应 A 请求，结果发现处理过程非常耗时，于是就发送 A 请求已经处理好的部分， 接着回应 B 请求，完成后，再发送 A 请求剩下的部分。这样双向的、实时的通信，就叫做多工（Multiplexing）。

数据流

因为 HTTP/2 的数据包是不按顺序发送的，同一个连接里面连续的数据包，可能属于不同的回应。因此，必须要对数据包做标记，指出它属于哪个回应。

HTTP/2 将每个请求或回应的所有数据包，称为一个数据流（stream）。每个数据流都有一个独一无二的编号。数据包发送的时候，都必须标记数据流 ID，用来区分它属于哪个数据流。另外还规定，客户端发出的数据流，ID 一律为奇数，服务器发出的，ID 为偶数。

数据流发送到一半的时候，客户端和服务器都可以发送信号（RST_STREAM 帧），取消这个数据流。1.1 版取消数据流的唯一方法，就是关闭 TCP 连接。这就是说，HTTP/2 可以取消某一次请求，同时保证 TCP 连接还打开着，可以被其他请求使用。客户端还可以指定数据流的优先级。优先级越高，服务器就会越早回应。

头信息压缩

HTTP 协议不带有状态，每次请求都必须附上所有信息。所以，请求的很多字段都是重复的，比如 Cookie 和 User Agent，一模一样的内容，每次请求都必须附带，这会浪费很多带宽，也影响速度。

HTTP/2 对这一点做了优化，引入了头信息压缩机制（header compression）。一方面，头信息使用 gzip 或 compress 压缩后再发送；另一方面，客户端和服务器同时维护一张头信息表，所有字段都会存入这个表，生成一个索引号，以后就不发送同样字段了，只发送索引号，这样就提高速度了。

服务器推送

HTTP/2 允许服务器未经请求，主动向客户端发送资源，这叫做服务器推送（server push）。常见场景是客户端请求一个网页，这个网页里面包含很多静态资源。正常情况下，客户端必须收到网页后，解析 HTML 源码，发现有静态资源，再发出静态资源请求。其实，服务器可以预期到客户端请求网页后，很可能会再请求静态资源，所以就主动把这些静态资源随着网页一起发给客户端了。

1.1.2 HTTP 请求报文格式：

HTTP 请求报文由请求行、请求头部、空行 和 请求包体 4 个部分组成，如下图所示：

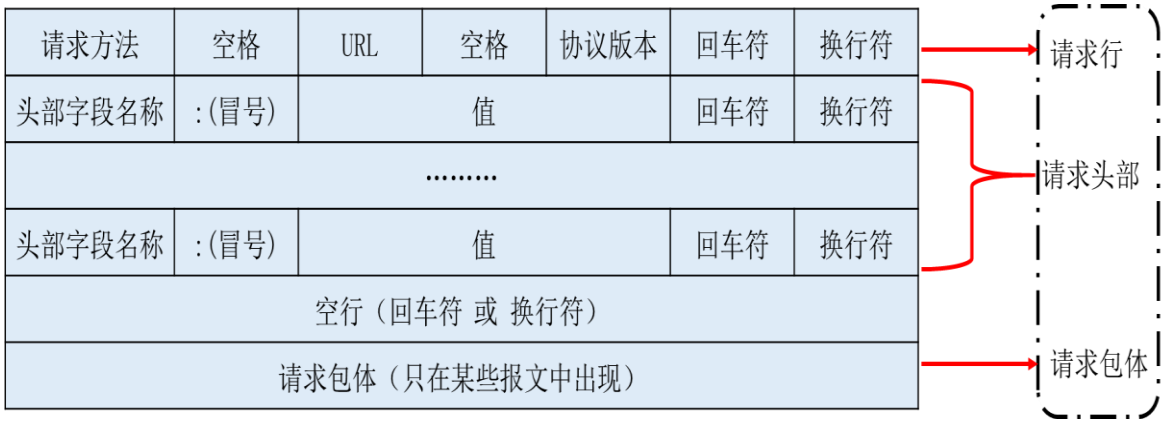


图 1.1 HTTP 请求报文格式

请求行

由请求方法字段、URL 字段、协议版本字段构成。常用的请求方法有如下几种：

1）GET：当客户端要从服务器中读取文档时使用。

当点击网页上的链接或者通过在浏览器的地址栏输入网址来浏览网页的，使用的都是 GET 方式。GET 方法要求服务器将 URL 定位的资源放在响应报文的数据部分，回送给客户端。使用 GET 方法时，请求参数和对应的值附加在 URL 后面，利用一个问号（“?”）代表 URL 的结尾与请求参数的开始，传递参数长度受限制。例如，/search?hl=zh-CN&source=hp&q=domety&aq=f&oq=。同时由于需要请求的参数直接会被显示在地址栏，这种方式不适合传送私密数据。具体 GET 请求报文格式如图 1.2 所示。

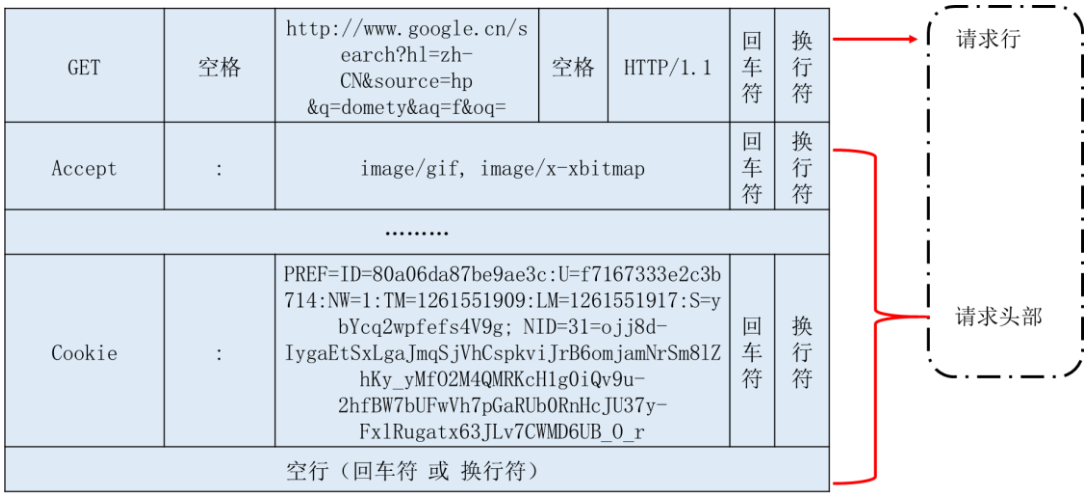


图 1.2 GET 请求报文格式示例

2）POST：对于上面提到的不适合使用 GET 方式的情况，可以考虑使用 POST 方式，因为使用 POST 方法可以允许客户端给服务器提供信息较多。POST 方法将请求参数封装在 HTTP 请求包体中，以名称/值的形式出现，可以传输大量数据，这样 POST 方式对传送的数据大小没有限制，而且也不会显示在 URL 中。报文示例如图 1.3：

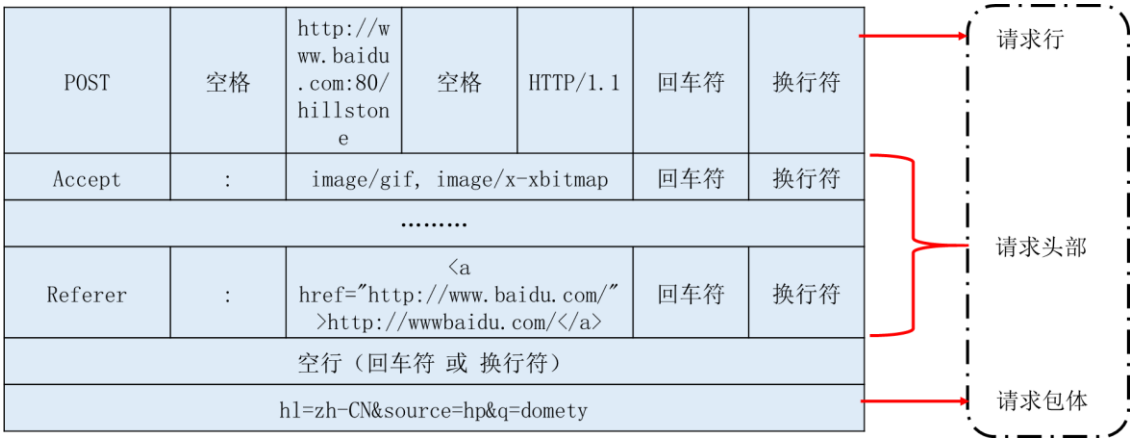


图 1.3 POST 请求报文格式示例

3）HEAD：客户想要得到关于文档的某些信息，但并不是要这个文档时使用。服务端接受到 HEAD 请求后只返回响应头，而不会发送响应内容。当我们只需要查看某个页面的状态的时候，使用 HEAD 是非常高效的，因为在传输的过程中省去了页面内容。

4）PUT：当客户将新的或者更新的文档存储在服务器上时使用。

5）COPY：当需要将文件复制到另一个位置时使用。

6）MOVE：当需要将文件移动到另一个位置时使用。

7）DELETE：当需要将服务器上的文档移走时使用。

8）LINK：当需要创建从一个文档到另一个位置的链接时使用。

9）UNLINK：当需要删除由 LINK 方法创建的链接时使用。

10）OPTION：当客户向服务器询问到一些可用的选项时使用。

URL（统一资源定位符）：方法、 端口、路径　格式为：方法://主机:端口/路径　例如：http://www.baidu.com:80/hillstone

1）方法：用来读取文档的协议，如：http1.0 或 http1.1 等。

2）主机：放置信息的计算机。此处可以是准确的 IP 地址，也可以是域名。

3）端口：服务器的端口号，可选。默认为知名端口 80。

4）路径：放置文件的路径名。

请求头部

请求头部由关键字/值对组成，每行一对，关键字和值用英文冒号“:”分隔。请求头部通知服务器有关于客户端请求的信息

常见的请求头字段含义：

Accept： 浏览器可接受的 MIME 类型。

Accept-Charset：浏览器可接受的字符集。

Accept-Encoding：浏览器能够进行解码的数据编码方式，比如 gzip。Servlet 能够向支持 gzip 的浏览器返回经 gzip 编码的 HTML 页面。许多情形下这可以减少 5 到 10 倍的下载时间。

Accept-Language：浏览器所希望的语言种类，当服务器能够提供一种以上的语言版本时要用到。

Authorization：授权信息，通常出现在对服务器发送的 WWW-Authenticate 头的应答中。

Content-Length：表示请求消息正文的长度。

Host： 客户机通过这个头告诉服务器，想访问的主机名。Host 头域指定请求资源的 Intenet 主机和端口号，必须表示请求 url 的原始服务器或网关的位置。HTTP/1.1 请求必须包含主机头域，否则系统会以 400 状态码返回。

If-Modified-Since：客户机通过这个头告诉服务器，资源的缓存时间。只有当所请求的内容在指定的时间后又经过修改才返回它，否则返回 304“Not Modified”应答。

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(防盗链)。包含一个 URL，用户从该 URL 代表的页面出发访问当前请求的页面。

User-Agent：User-Agent 头域的内容包含发出请求的用户信息。浏览器类型，如果 Servlet 返回的内容与浏览器类型有关则该值非常有用。

Cookie：客户机通过这个头可以向服务器带数据，这是最重要的请求头信息之一。

Pragma：指定“no-cache”值表示服务器必须返回一个刷新后的文档，即使它是代理服务器而且已经有了页面的本地拷贝。

From：请求发送者的 email 地址，由一些特殊的 Web 客户程序使用，浏览器不会用到它。

Connection：处理完这次请求后是否断开连接还是继续保持连接。如果 Servlet 看到这里的值为“Keep- Alive”，或者看到请求使用的是 HTTP 1.1(HTTP 1.1 默认进行持久连接)，它就可以利用持久连接的优点，当页面包含多个元素时(例如 Applet，图片)，显著地减少下载所需要的时间。要实现这一点，Servlet 需要在应答中发送一个 Content-Length 头，最简单的实现方法是：先把内容写入 ByteArrayOutputStream，然后在正式写出内容之前计算它的大小。

Range：Range 头域可以请求实体的一个或者多个子范围。例如，

表示头 500 个字节：bytes=0-499

表示第二个 500 字节：bytes=500-999

表示最后 500 个字节：bytes=-500

表示 500 字节以后的范围：bytes=500-

第一个和最后一个字节：bytes=0-0,-1

同时指定几个范围：bytes=500-600,601-999

如果传输数据过程突然出现网络中断的情况，在网络恢复后续传就不用重新完整的传一遍，HTTP 客户端可以通过请求曾获取失败的实体的一个范围（或者说一部分），来恢复下载该实体。

但是服务器可以忽略此请求头，如果无条件 GET 包含 Range 请求头，响应会以状态码 206(PartialContent)返回而不是以 200 (OK)。

UA-Pixels，UA-Color，UA-OS，UA-CPU：由某些版本的 IE 浏览器所发送的非标准的请求头，表示屏幕大小、颜色深度、操作系统和 CPU 类型。

请求包体

若方法字段是 GET，则此项为空，没有数据。若方法字段是 POST,则通常来说此处放置的就是要提交的数据（可对比图 1.2 和图 1.3），比如要使用 POST 方法提交一个表单，其中有 user 字段中数据为“admin”, password 字段为 123456，那么这里的请求数据就是 user=admin&password=123456，使用&来连接各个字段。

1.1.3 HTTP 响应报文格式

HTTP 响应报文由状态行、响应头、空行和响应内容 4 个部分构成。如图 1.4 所示：

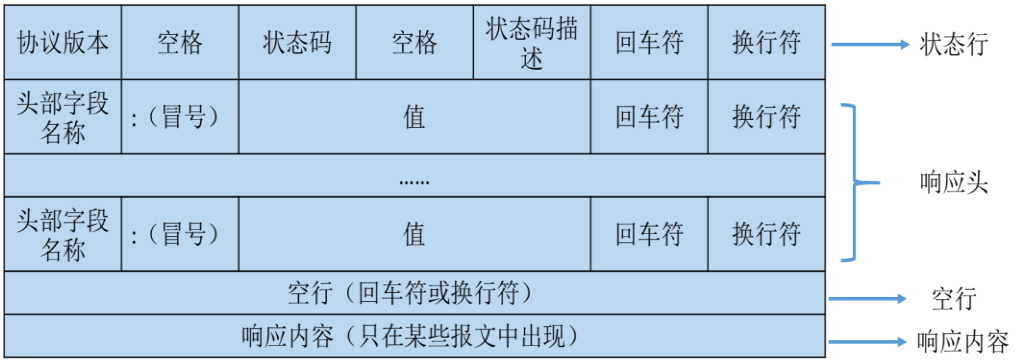


图 1.4 HTTP 响应报文格式

状态行

由 HTTP 协议版本、状态码、状态码描述三部分构成，它们之间由空格隔开。

1）协议版本后面会详细阐述。

2）状态码和状态码描述：由 3 位数字组成，第一位标识响应的类型，详见附录 1。

1xx：表示服务器已接收了客户端的请求，客户端可以继续发送请求。

2xx：表示服务器已成功接收到请求并进行处理。

3xx：表示服务器要求客户端重定向。

4xx：表示客户端的请求有非法内容。

5xx：标识服务器未能正常处理客户端的请求而出现意外错误。

响应头

一般情况下，响应头会包含以下，甚至更多的信息。

Location：服务器返回给客户端，用于重定向到新的位置

Server： 包含服务器用来处理请求的软件信息及版本信息

Vary：标识不可缓存的请求头列表

Connection: 连接方式。对于请求端来讲：close 是告诉服务端，断开连接，不用等待后续的请求了。keeplive 则是告诉服务端，在完成本次请求的响应后，保持连接，等待本次连接后的后续请求。对于响应端来讲：close 表示连接已经关闭。keeplive 则表示连接保持中，可以继续处理后续请求。Keep-Alive 表示如

果请求端保持连接，则该请求头部信息表明期望服务端保持连接多长时间（秒），例如 300 秒表示为 Keep-Alive: 300

响应内容

响应体就是响应的消息体（即真正传输的有用数据），如果是纯数据就是返回纯数据，如果请求的是 HTML 页面，那么返回的就是 HTML 代码，如果是 JS 就是 JS 代码，如此之类。

一个成功响应返回的 HTTP 响应报文如图 1.5.

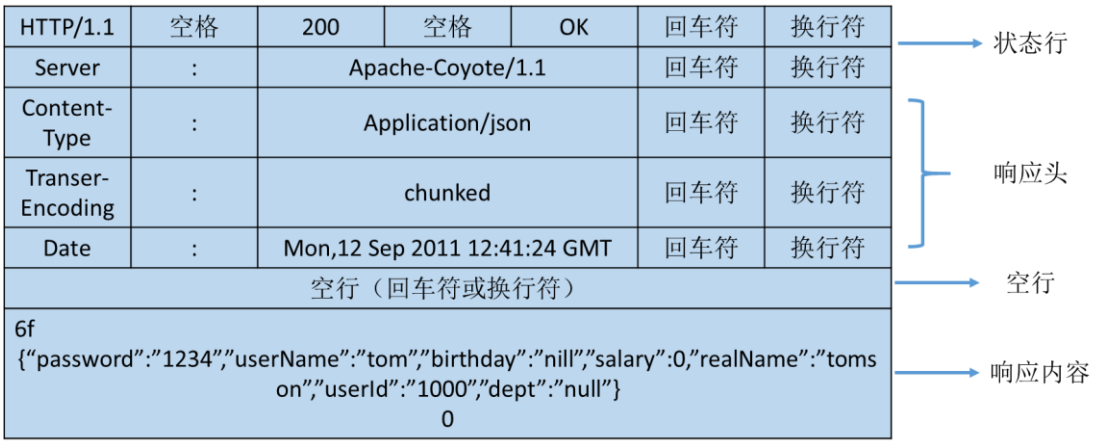


图 1.5 HTTP 响应报文示例

2 HTTP 工作原理

当一个客户机与服务器建立连接后，发送一个请求给服务器，请求的格式是：统一资源标识符（URI）、协议版本号，后面是类似 MIME （通用因特网邮件扩充协议，Multipurpose Internet Mail Extensions）的信息，包括请求修饰符、客户机信息和可能的内容。

服务器接到请求后，给予相应的响应信息，其格式是：一个状态行包括信息的协议版本号、一个成功或错误的代码，后面也是类似 MIME 的信息，包括服务器信息、实体信息和可能的内容。

客户端连接到 web 服务器：HTTP 客户端与 web 服务器建立一个 TCP 连接。

客户端向服务器发起 HTTP 请求：通过已建立的 TCP 连接，客户端向服务器发送一个请求报文。

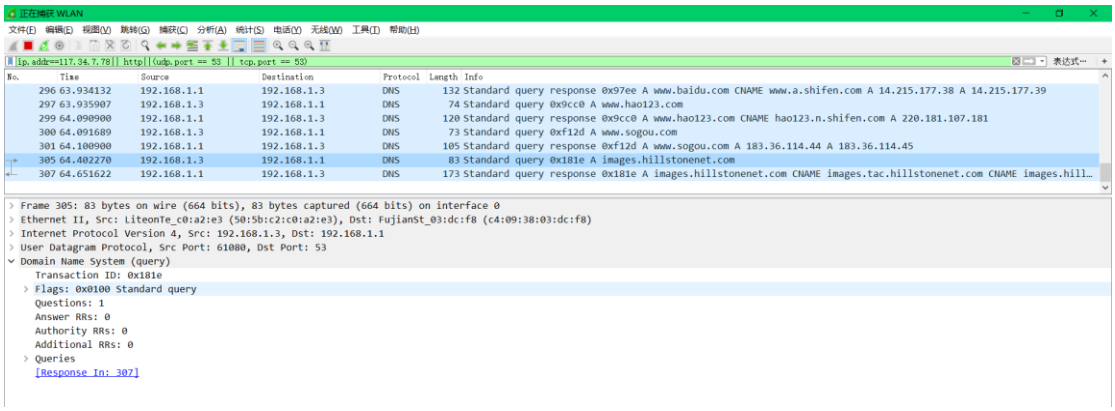
服务器接收 HTTP 请求并返回 HTTP 响应：服务器解析请求，定位请求资源，服务器将资源副本写到 TCP 连接，由客户端读取。

释放 TCP 连接：若 connection 模式为 close，则服务器主动关闭 TCP 连接，客户端被动关闭连接，释放 TCP 连接;若 connection 模式为 keepalive，则该连接会保持一段时间，在该时间内可以继续接收请求。

客户端浏览器解析 HTML 内容：客户端将服务器响应的 html 文本解析并显示。

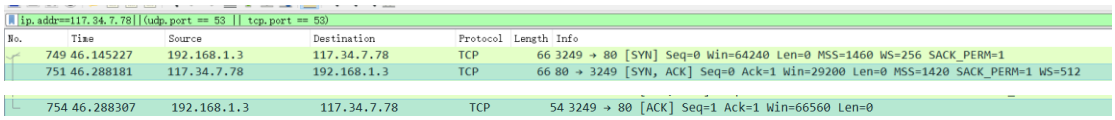
例如：在浏览器地址栏键入 URL，按下回车之后会经历以下流程：

1、浏览器向 DNS 服务器请求解析该 URL 中的域名所对应的 IP 地址。

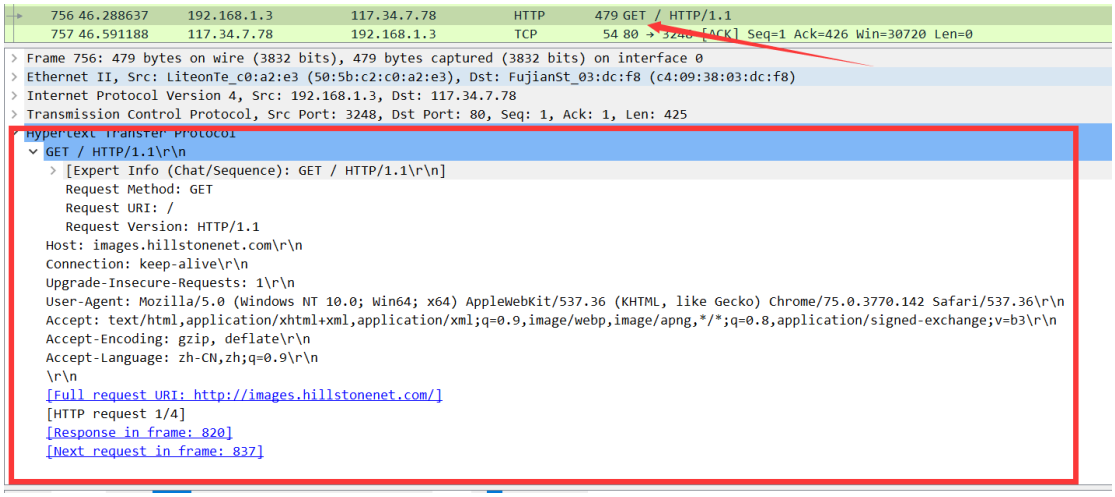


2、解析出 IP 地址后，根据该 IP 地址和默认端口 80，和服务器建立 TCP

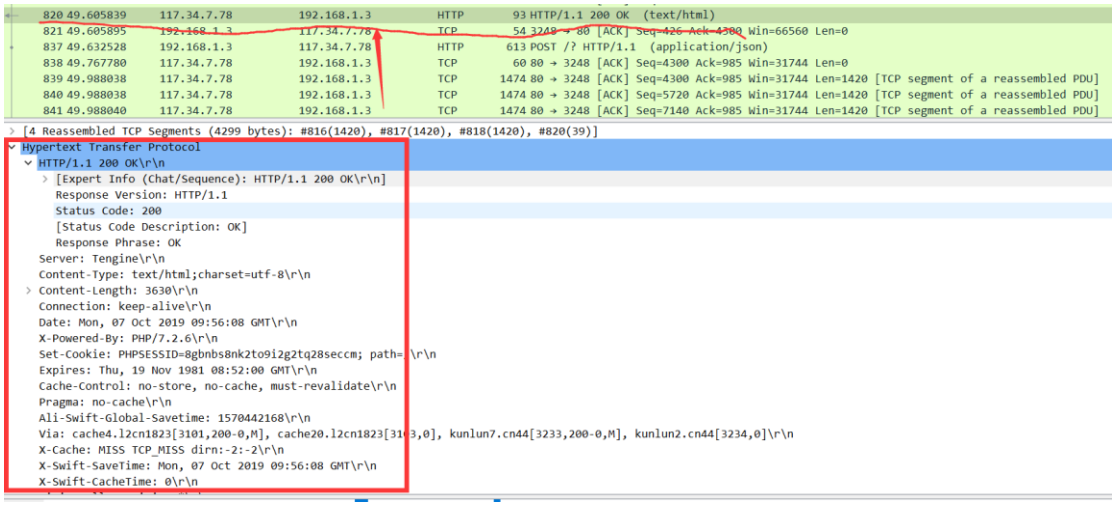
连接。



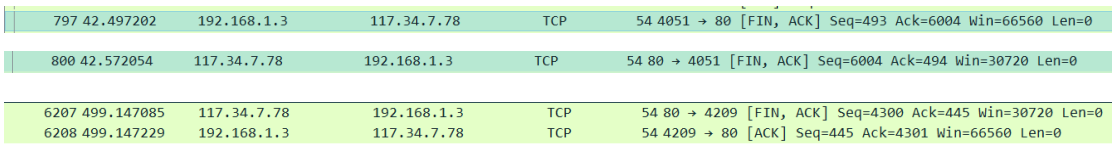
3、浏览器发出读取文件(URL 中域名后面部分对应的文件)的 HTTP 请求。



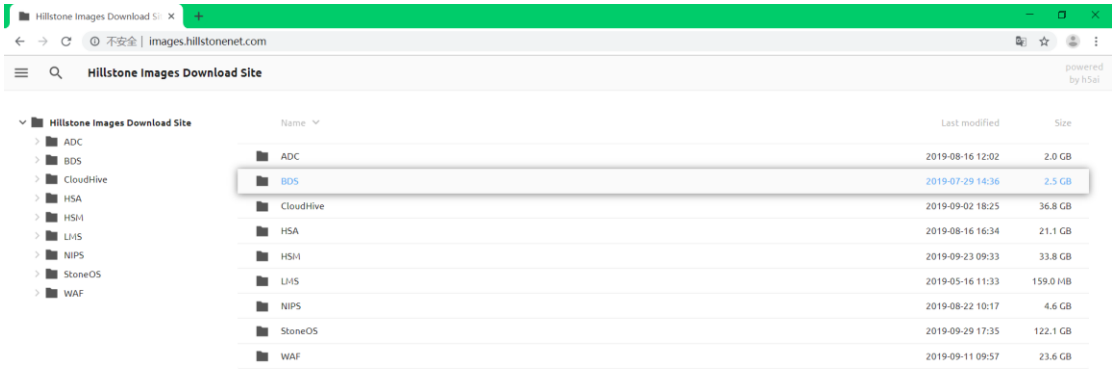
4、服务器对浏览器请求作出响应，并把对应的 html 文本发送给浏览器。



5、释放 TCP 连接。



6、浏览器将该 html 文本并显示内容。


Hillstone Images Download Site
不安全 | images.hillstonenet.com
Hillstone Images Download Site
powered by h5ai
Hillstone Images Download Site
ADC
BDS
CloudHive
HSA
HSM
LMS
NIPS
StoneOS
WAF
Name
Last modified
Size
ADC 2019-08-16 12:02 2.0 GB
BDS 2019-07-29 14:36 2.5 GB
CloudHive 2019-09-02 18:25 36.8 GB
HSA 2019-08-16 16:34 21.1 GB
HSM 2019-09-23 09:33 33.8 GB
LMS 2019-05-16 11:33 159.0 MB
NIPS 2019-08-22 10:17 4.6 GB
StoneOS 2019-09-29 17:35 122.1 GB
WAF 2019-09-11 09:57 23.6 GB

附录 HTTP 响应状态表

1xx:信息

消息	描述
100 Continue	服务器仅接收到部分请求，但是一旦服务器并没有拒绝该请求，客户端应该继续发送其余的请求。
101 Switching Protocols	服务器转换协议：服务器将遵从客户的请求转换到另外一种协议。

2xx:成功

消息	描述
200 OK	请求成功（其后是对GET和POST请求的应答文档。）
201 Created	请求被创建完成，同时新的资源被创建。
202 Accepted	供处理的请求已被接受，但是处理未完成。
203 Non-authoritative Information	文档已经正常地返回，但一些应答头可能不正确，因为使用的是文档的拷贝。
204 No Content	没有新文档。浏览器应该继续显示原来的文档。如果用户定期地刷新页面，而Servlet可以确定用户文档足够新，这个状态代码是很有用的。
205 Reset Content	没有新文档。但浏览器应该重置它所显示的内容。用来强制浏览器清除表单输入内容。
206 Partial Content	客户发送了一个带有Range头的GET请求，服务器完成了它。

3xx:重定向

消息	描述
300 Multiple Choices	多重选择。链接列表。用户可以选择某链接到达目的地。最多允许五个地址。
301 Moved Permanently	所请求的页面已经转移至新的url。
302 Found	所请求的页面已经临时转移至新的url。
303 See Other	所请求的页面可在别的url下被找到。
304 Not Modified	未按预期修改文档。客户端有缓冲的文档并发出了一个条件性的请求（一般是提供If-Modified-Since头表示客户只想比指定日期更新的文档）。服务器告诉客户，原来缓冲的文档还可以继续使用。
305 Use Proxy	客户请求的文档应该通过Location头所指明的代理服务器提取。
306 *Unused*	此代码被用于前一版本。目前已不再使用，但是代码依然被保留。
307 Temporary Redirect	被请求的页面已经临时移至新的url。

4xx:客户端错误

消息	描述
400 Bad Request	服务器未能理解请求。
401 Unauthorized	被请求的页面需要用户名和密码。
401.1	登录失败。
401.2	服务器配置导致登录失败。
401.3	由于 ACL 对资源的限制而未获得授权。
401.4	筛选器授权失败。
401.5	ISAPI/CGI 应用程序授权失败。
401.7	访问被 Web 服务器上的 URL 授权策略拒绝。这个错误代码为 IIS 6.0 所专用。
402 Payment Required	此代码尚无法使用。
403 Forbidden	对被请求页面的访问被禁止。
403.1	执行访问被禁止。
403.2	读访问被禁止。
403.3	写访问被禁止。
403.4	要求 SSL。
403.5	要求 SSL 128。
403.6	IP 地址被拒绝。
403.7	要求客户端证书。
403.8	站点访问被拒绝。
403.9	用户数过多。
403.10	配置无效。
403.11	密码更改。
403.12	拒绝访问映射表。
403.13	客户端证书被吊销。
403.14	拒绝目录列表。
403.15	超出客户端访问许可。
403.16	客户端证书不受信任或无效。
403.17	客户端证书已过期或尚未生效。
403.18	在当前的应用程序池中不能执行所请求的 URL。这个错误代码为 IIS 6.0 所专用。
403.19	不能为这个应用程序池中的客户端执行 CGI。这个错误代码为 IIS 6.0 所专用。
403.20	Passport 登录失败。这个错误代码为 IIS 6.0 所专用。
404 Not Found	服务器无法找到被请求的页面。

404.0	（无）–没有找到文件或目录。
404.1	无法在所请求的端口上访问 Web 站点。
404.2	Web 服务扩展锁定策略阻止本请求。
404.3	MIME 映射策略阻止本请求。
405 Method Not Allowed	请求中指定的方法不被允许。
406 Not Acceptable	服务器生成的响应无法被客户端所接受。
407 Proxy Authentication Required	用户必须首先使用代理服务器进行验证，这样请求才会被处理。
408 Request Timeout	请求超出了服务器的等待时间。
409 Conflict	由于冲突，请求无法被完成。
410 Gone	被请求的页面不可用。
411 Length Required	"Content-Length" 未被定义。如果无此内容，服务器不会接受请求。
412 Precondition Failed	请求中的前提条件被服务器评估为失败。
413 Request Entity Too Large	由于所请求的实体的太大，服务器不会接受请求。
414 Request-url Too Long	由于 url 太长，服务器不会接受请求。当 post 请求被转换为带有很长的查询信息的 get 请求时，就会发生这种情况。
415 Unsupported Media Type	由于媒介类型不被支持，服务器不会接受请求。
416 Requested Range Not Satisfiable	服务器不能满足客户在请求中指定的 Range 头。
417 Expectation Failed	执行失败。
423	锁定的错误。

5xx:服务器错误

消息	描述
500 Internal Server Error	请求未完成。服务器遇到不可预知的情况。
500.12	应用程序正忙于在 Web 服务器上重新启动。
500.13	Web 服务器太忙。
500.15	不允许直接请求 Global.asa。
500.16	UNC 授权凭据不正确。这个错误代码为 IIS 6.0 所专用。
500.18	URL 授权存储不能打开。这个错误代码为 IIS 6.0 所专用。
500.100	内部 ASP 错误。
501 Not Implemented	请求未完成。服务器不支持所请求的功能。
502 Bad Gateway	请求未完成。服务器从上游服务器收到一个无效的响应。
502.1	CGI 应用程序超时。
502.2	CGI 应用程序出错。
503 Service Unavailable	请求未完成。服务器临时过载或当机。
504 Gateway Timeout	网关超时。
505 HTTP Version Not Supported	服务器不支持请求中指明的HTTP协议版本。