HTTP 协议详解

林超旗　整理

2010.06.22

目　录

HTTP 协议详解

引言

HTTP 是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。它于 1990 年提出，经过几年的使用与发展，得到不断地完善和扩展。目前在 WWW 中使用的是 HTTP/1.0 的第六版，HTTP/1.1 的规范化工作正在进行之中，而且 HTTP-NG(Next Generation of HTTP)的建议已经提出。

HTTP 协议的主要特点可概括如下：

1.支持客户/服务器模式。

2.简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有 GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于 HTTP 协议简单，使得 HTTP 服务器的程序规模小，因而通信速度很快。

3.灵活：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。

4.无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。

5.无状态：HTTP 协议是无状态协议。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。

一、HTTP 协议详解之 URL 篇

http（超文本传输协议）是一个基于请求与响应模式的、无状态的、应用层的协议，常基于 TCP 的连接方式，HTTP1.1 版本中给出一种持续连接的机制，绝大多数的 Web 开发，都是构建在 HTTP 协议之上的 Web 应用。

HTTP URL (URL 是一种特殊类型的 URI，包含了用于查找某个资源的足够的信息)的格式如下：

http://host[":"port][abs_path]

http 表示要通过 HTTP 协议来定位网络资源；host 表示合法的 Internet 主机域名或者 IP 地址；port 指定一个端口号，为空则使用缺省端口 80；abs_path 指定请求资源的 URI；如果 URL 中没有给出 abs_path，那么当它作为请求 URI 时，必须以“/”的形式给出，通常这个工作浏览器自动帮我们完成。

eg:

1、输入：www.guet.edu.cn

浏览器自动转换成：http://www.guet.edu.cn/

2、http:192.168.0.116:8080/index.jsp

二、HTTP 协议详解之请求篇

http 请求由三部分组成，分别是：请求行、消息报头、请求正文

1、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，各个方法的解释如下：

GET　　请求获取 Request-URI 所标识的资源

POST　　在 Request-URI 所标识的资源后附加新的数据

HEAD　　请求获取由 Request-URI 所标识的资源的响应消息报头

PUT　　请求服务器存储一个资源，并用 Request-URI 作为其标识

DELETE　请求服务器删除 Request-URI 所标识的资源

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 保留将来使用

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

应用举例：

GET 方法：在浏览器的地址栏中输入网址的方式访问网页时，浏览器采用 GET 方法向服务器获取资源，eg:GET /form.html HTTP/1.1 (CRLF)

POST 方法要求被请求服务器接受附在请求后面的数据，常用于提交表单。

```
eg：POST /reg.jsp HTTP/ (CRLF)
Accept:image/gif,image/x-xbit,... (CRLF)
...
HOST:www.guet.edu.cn (CRLF)
Content-Length:22 (CRLF)
Connection:Keep-Alive (CRLF)
Cache-Control:no-cache (CRLF)
(CRLF)          //该 CRLF 表示消息报头已经结束，在此之前为消息报头
user=jeffrey&pwd=1234  //此行以下为提交的数据
```

HEAD 方法与 GET 方法几乎是一样的，对于 HEAD 请求的回应部分来说，它的 HTTP 头部中包含的信息与通过 GET 请求所得到的信息是相同的。利用这个方法，不必传输整个资源内容，就可以得到 Request-URI 所标识的资源的信息。该方法常用于测试超链接的有效性，是否可以访问，以及最近是否更新。

2、请求报头后述

3、请求正文(略)

三、HTTP 协议详解之响应篇

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

1、状态行格式如下：

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。

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、状态描述、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，eg：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

eg：HTTP/1.1 200 OK （CRLF）

2、响应报头后述

3、响应正文就是服务器返回的资源的内容

四、HTTP 协议详解之消息报头篇

HTTP 消息由客户端到服务器的请求和服务器到客户端的响应组成。请求消息和响应消息都是由开始行（对于请求消息，开始行就是请求行，对于响应消息，开始行就是状态行），消息报头（可选），空行（只有 CRLF 的行），消息正文（可选）组成。

HTTP 消息报头包括普通报头、请求报头、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。

1、普通报头

在普通报头中，有少数报头域用于所有的请求和响应消息，但并不用于被传输的实体，只用于传输的消息。

eg：

Cache-Control　用于指定缓存指令，缓存指令是单向的（响应中出现的缓存指令在请求中未必会出现），且是独立的（一个消息的缓存指令不会影响另一个消息处理的缓存机制），HTTP1.0 使用的类似的报头域为 Pragma。

请求时的缓存指令包括：no-cache（用于指示请求或响应消息不能缓存）、no-store、max-age、max-stale、min-fresh、only-if-cached;

响应时的缓存指令包括：public、private、no-cache、no-store、no-transform、must-revalidate、proxy-revalidate、max-age、s-maxage.

eg：为了指示 IE 浏览器（客户端）不要缓存页面，服务器端的 JSP 程序可以编写如下：response.sehHeader("Cache-Control","no-cache");

//response.setHeader("Pragma","no-cache");作用相当于上述代码，通常两者//合用

这句代码将在发送的响应消息中设置普通报头域：Cache-Control:no-cache

Date 普通报头域表示消息产生的日期和时间

Connection 普通报头域允许发送指定连接的选项。例如指定连接是连续，或者指定“close”选项，通知服务器，在响应完成后，关闭连接

2、请求报头

请求报头允许客户端向服务器端传递请求的附加信息以及客户端自身的信息。

常用的请求报头

Accept

Accept 请求报头域用于指定客户端接受哪些类型的信息。eg：Accept：image/gif，表明客户端希望接受 GIF 图象格式的资源；Accept：text/html，表明客户端希望接受 html 文本。

Accept-Charset

Accept-Charset 请求报头域用于指定客户端接受的字符集。eg：Accept-Charset:iso-8859-1,gb2312.如果在请求消息中没有设置这个域，缺省是任何字符集都可以接受。

Accept-Encoding

Accept-Encoding 请求报头域类似于 Accept，但是它是用于指定可接受的内容编码。eg：Accept-Encoding:gzip.deflate.如果请求消息中没有设置这个域服务器假定客户端对各种内容编码都可以接受。

Accept-Language

Accept-Language 请求报头域类似于 Accept，但是它是用于指定一种自然语言。eg：Accept-Language:zh-cn.如果请求消息中没有设置这个报头域，服务器假定客户端对各种语言都可以接受。

Authorization

Authorization 请求报头域主要用于证明客户端有权查看某个资源。当浏览器访问一个页面时，如果收到服务器的响应代码为 401（未授权），可以发送一个包含 Authorization 请求报头域的请求，要求服务器对其进行验证。

Host（发送请求时，该报头域是必需的）

Host 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，eg：

我们在浏览器中输入：http://www.guet.edu.cn/index.html

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.guet.edu.cn

此处使用缺省端口号 80，若指定了端口号，则变成：Host：www.guet.edu.cn:指定端口号

User-Agent

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。不过，这个报头域不是必需的，如果我们自己编写一个浏览器，不使用 User-Agent 请求报头域，那么服务器端就无法得知我们的信息了。

请求报头举例：

GET /form.html HTTP/1.1 (CRLF)

Accept:image/gif,image/x-xbitmap,image/jpeg,application/x-shockwave-flash,application/vnd.ms-excel,application/vnd.ms-powerpoint,application/msword, */* (CRLF)

Accept-Language:zh-cn (CRLF)

Accept-Encoding:gzip,deflate (CRLF)

If-Modified-Since:Wed,05 Jan 2007 11:21:25 GMT (CRLF)

If-None-Match:W/"80b1a4c018f3c41:8317" (CRLF)

User-Agent:Mozilla/4.0(compatible;MSIE6.0;Windows NT 5.0) (CRLF)

Host:www.guet.edu.cn (CRLF)

Connection:Keep-Alive (CRLF)

(CRLF)

3、响应报头

响应报头允许服务器传递不能放在状态行中的附加响应信息，以及关于服务器的信息和对 Request-URI 所标识的资源进行下一步访问的信息。

常用的响应报头

Location

Location 响应报头域用于重定向接受者到一个新的位置。Location 响应报头域常用在更换域名的时候。

Server

Server 响应报头域包含了服务器用来处理请求的软件信息。与 User-Agent 请求报头域是相对应的。下面是

Server 响应报头域的一个例子：

Server：Apache-Coyote/1.1

WWW-Authenticate

WWW-Authenticate 响应报头域必须被包含在 401（未授权的）响应消息中，客户端收到 401 响应消息时候，并发送 Authorization 报头域请求服务器对其进行验证时，服务端响应报头就包含该报头域。

eg：WWW-Authenticate:Basic realm="Basic Auth Test!" //可以看出服务器对请求资源采用的是基本验证机制。

4、实体报头

请求和响应消息都可以传送一个实体。一个实体由实体报头域和实体正文组成，但并不是说实体报头域和实体正文要在一起发送，可以只发送实体报头域。实体报头定义了关于实体正文（eg：有无实体正文）

和请求所标识的资源的元信息。

常用的实体报头

Content-Encoding

Content-Encoding 实体报头域被用作媒体类型的修饰符，它的值指示了已经被应用到实体正文的附加内容的编码，因而要获得 Content-Type 报头域中所引用的媒体类型，必须采用相应的解码机制。Content-Encoding 这样用于记录文档的压缩方法，eg：Content-Encoding：gzip

Content-Language

Content-Language 实体报头域描述了资源所用的自然语言。没有设置该域则认为实体内容将提供给所有的语言阅读

者。eg：Content-Language:da

Content-Length

Content-Length 实体报头域用于指明实体正文的长度，以字节方式存储的十进制数字来表示。

Content-Type

Content-Type 实体报头域用语指明发送给接收者的实体正文的媒体类型。eg：

Content-Type:text/html;charset=ISO-8859-1

Content-Type:text/html;charset=GB2312

Last-Modified

Last-Modified 实体报头域用于指示资源的最后修改日期和时间。

Expires

Expires 实体报头域给出响应过期的日期和时间。为了让代理服务器或浏览器在一段时间以后更新缓存中(再次访问曾访问过的页面时，直接从缓存中加载，缩短响应时间和降低服务器负载)的页面，我们可以使用 Expires 实体报头域指定页面过期的时间。eg：Expires：Thu，15 Sep 2006 16:23:12 GMT

HTTP1.1 的客户端和缓存必须将其他非法的日期格式（包括 0）看作已经过期。eg：为了让浏览器不要缓存页面，我们也可以利用 Expires 实体报头域，设置为 0，jsp 中程序如下：response.setDateHeader("Expires","0");

五、利用 telnet 观察 http 协议的通讯过程

实验目的及原理：

利用 MS 的 telnet 工具，通过手动输入 http 请求信息的方式，向服务器发出请求，服务器接收、解释和接受请求后，会返回一个响应，该响应会在 telnet 窗口上显示出来，从而从感性上加深对 http 协议的通讯过程的认识。

实验步骤：

1、打开 telnet

1.1 打开 telnet

运行-->cmd-->telnet

1.2 打开 telnet 回显功能

set localecho

2、连接服务器并发送请求

2.1 open www.guet.edu.cn 80 //注意端口号不能省略

```
HEAD /index.asp HTTP/1.0
Host:www.guet.edu.cn

/*我们可以变换请求方法,请求桂林电子主页内容,输入消息如下*/
open www.guet.edu.cn 80

GET /index.asp HTTP/1.0  //请求资源的内容
Host:www.guet.edu.cn
```

2.2 open www.sina.com.cn 80 //在命令提示符号下直接输入 telnet www.sina.com.cn 80

```
HEAD /index.asp HTTP/1.0
Host:www.sina.com.cn
```

3、实验结果：

3.1 请求信息 2.1 得到的响应是:

```
HTTP/1.1 200 OK                                    //请求成功
Server: Microsoft-IIS/5.0                          //web 服务器
Date: Thu,08 Mar 200707:17:51 GMT
Connection: Keep-Alive
Content-Length: 23330
Content-Type: text/html
Expries: Thu,08 Mar 2007 07:16:51 GMT
Set-Cookie:ASPSESSIONIDQAQBQQQB=BEJCDGKADEDJKLKKAJEOIMMH; path=/
Cache-control: private

//资源内容省略
```

3.2 请求信息 2.2 得到的响应是:

```
HTTP/1.0 404 Not Found      //请求失败
```

```
Date: Thu, 08 Mar 2007 07:50:50 GMT
Server: Apache/2.0.54 <Unix>
Last-Modified: Thu, 30 Nov 2006 11:35:41 GMT
ETag: "6277a-415-e7c76980"
Accept-Ranges: bytes
X-Powered-By: mod_xlayout_jh/0.0.1vhs.markII.remix
Vary: Accept-Encoding
Content-Type: text/html
X-Cache: MISS from zjm152-78.sina.com.cn
Via: 1.0 zjm152-78.sina.com.cn:80<squid/2.6.STABLES-20061207>
X-Cache: MISS from th-143.sina.com.cn
Connection: close
```

失去了跟主机的连接

按任意键继续...

4、注意事项

- 出现输入错误，则请求不会成功。
- 报头域不分大小写。
- 更深一步了解 HTTP 协议，可以查看 RFC2616，在 http://www.letf.org/rfc 上找到该文件。
- 开发后台程序必须掌握 http 协议

六、HTTP 协议相关技术补充

1、基础

高层协议有：文件传输协议 FTP、电子邮件传输协议 SMTP、域名系统服务 DNS、网络新闻传输协议 NNTP 和 HTTP 协议等

中介由三种：代理(Proxy)、网关(Gateway)和通道(Tunnel)，一个代理根据 URI 的绝对格式来接受请求，重写全部或部分消息，通过 URI 的标识把已格式化过的请求发送到服务器。网关是一个接收代理，作为一些其它服务器的上层，并且如果必须的话，可以把请求翻译给下层的服务器协议。一 个通道作为不改变消息的两个连接之间的中继点。当通讯需要通过一个中介(例如：防火墙等)或者是中介不能识别消息的内容时，通道经常被使用。

代理(Proxy)：一个中间程序，它可以充当一个服务器，也可以充当一个客户机，为其它客户机建立

请求。请求是通过可能的翻译在内部或经过传递到其它的 服务器中。一个代理在发送请求信息之前，必须解释并且如果可能重写它。代理经常作为通过防火墙的客户机端的门户，代理还可以作为一个帮助应用来通过协议处 理没有被用户代理完成的请求。

网关(Gateway)：一个作为其它服务器中间媒介的服务器。与代理不同的是，网关接受请求就好象对被请求的资源来说它就是源服务器；发出请求的客户机并没有意识到它在同网关打交道。

网关经常作为通过防火墙的服务器端的门户，网关还可以作为一个协议翻译器以便存取那些存储在非 HTTP 系统中的资源。

通道(Tunnel)：是作为两个连接中继的中介程序。一旦激活，通道便被认为不属于 HTTP 通讯，尽管通道可能是被一个 HTTP 请求初始化的。当被中继 的连接两端关闭时，通道便消失。当一个门户(Portal)必须存在或中介(Intermediary)不能解释中继的通讯时通道被经常使用。

2、协议分析的优势–HTTP 分析器检测网络攻击

以模块化的方式对高层协议进行分析处理，将是未来入侵检测的方向。

HTTP 及其代理的常用端口 80、3128 和 8080 在 network 部分用 port 标签进行了规定

3、HTTP 协议 Content Lenth 限制漏洞导致拒绝服务攻击

使用 POST 方法时，可以设置 ContentLenth 来定义需要传送的数据长度，例如 ContentLenth:999999999，在传送完成前，内 存不会释放，攻击者可以利用这个缺陷，连续向 WEB 服务器发送垃圾数据直至 WEB 服务器内存耗尽。这种攻击方法基本不会留下痕迹。

http://www.cnpaf.net/Class/HTTP/0532918532667330.html

4、利用 HTTP 协议的特性进行拒绝服务攻击的一些构思

服务器端忙于处理攻击者伪造的 TCP 连接请求而无暇理睬客户的正常请求(毕竟客户端的正常请求比率非常之小)，此时从正常客户的角度看来，服务器失去响应，这种情况我们称作：服务器端受到了 SYNFlood 攻击（SYN 洪水攻击)。

而 Smurf、TearDrop 等是利用 ICMP 报文来 Flood 和 IP 碎片攻击的。本文用“正常连接”的方法来产生拒绝服务攻击。

19 端口在早期已经有人用来做 Chargen 攻击了，即 Chargen_Denial_of_Service，但是！他们用的方法是在两台 Chargen 服务器之间产生 UDP 连接，让服务器处理过多信息而 DOWN 掉，那么，干掉一台 WEB 服务器的条件就必须有 2 个：1.有 Chargen 服务 2.有 HTTP 服务

方法：攻击者伪造源 IP 给 N 台 Chargen 发送连接请求（Connect)，Chargen 接收到连接后就会返回每秒 72 字节的字符流（实际上根据网络实际情况，这个速度更快）给服务器。

5、Http 指纹识别技术

Http 指纹识别的原理大致上也是相同的：记录不同服务器对 Http 协议执行中的微小差别进行识别.Http 指纹识别比 TCP/IP 堆栈指纹识别复杂许 多,理由是定制 Http 服务器的配置文件、增加插件或组件使得更改 Http 的响应信息变的很容易,这样使得识别变的困难；然而定制 TCP/IP 堆栈的行为 需要对核心层进行修改,所以就容易识别.

要让服务器返回不同的 Banner 信息的设置是很简单的,象 Apache 这样的开放源代码的 Http 服务器,用户可以在源代码里修改 Banner 信息,然 后重起 Http 服务就生效了;对于没有公开源代码的 Http 服务器比如微软的 IIS 或者是 Netscape,可以在存放 Banner 信息的 Dll 文件中修 改,相关的文章有讨论的,这里不再赘述,当然这样的修改的效果还是不错的.另外一种模糊 Banner 信息的方法是使用插件。

常用测试请求:

1:HEAD/Http/1.0 发送基本的 Http 请求

2:DELETE/Http/1.0 发送那些不被允许的请求,比如 Delete 请求

3:GET/Http/3.0 发送一个非法版本的 Http 协议请求

4:GET/JUNK/1.0 发送一个不正确规格的 Http 协议请求

Http 指纹识别工具 Httprint,它通过运用统计学原理,组合模糊的逻辑学技术,能很有效的确定 Http 服务器的类型.它可以被用来收集和分析不同 Http 服务器产生的签名。

6、其他

为了提高用户使用浏览器时的性能，现代浏览器还支持并发的访问方式，浏览一个网页时同时建立多个连接，以迅速获得一个网页上的多个图标，这样能更快速完成整个网页的传输。

HTTP1.1 中提供了这种持续连接的方式，而下一代 HTTP 协议：HTTP-NG 更增加了有关会话控制、丰富的内容协商等方式的支持，来提供

更高效率的连接。

本文来自 CSDN 博客，转载请标明出处：
http://blog.csdn.net/gueter/archive/2007/03/08/1524447.aspx