# Tcpdump命令的基本使用

## 基本介绍

1. tcpdump采用命令行方式对接口的数据包进行筛选抓取，其丰富特性表现在灵活的表达式上。
2. 不带任何选项的tcpdump，默认会抓取第一个网络接口，且只有将tcpdump进程终止才会停止抓包。

## tcpdump选项

* 1. 命令格式

tcpdump [ -DenNqvX ] [ -c count ] [ -F file ] [ -i interface ] [ -r file ]

[ -s snaplen ] [ -w file ] [ expression ]

* 1. 抓包选项

1. -c：指定要抓取的包数量。注意，是最终要获取这么多个包。例如，指定"-c 10"将获取10个包，但可能已经处理了100个包，只不过只有10个包是满足条件的包。
2. **-i interface：指定tcpdump需要监听的接口。**若未指定该选项，将从系统接口列表中搜寻编号最小的已配置好的接口(不包括loopback接口，要抓取loopback接口使用tcpdump -i lo)，：一旦找到第一个符合条件的接口，搜寻马上结束。**可以使用'any'关键字表示所有网络接口。**
3. -n：对地址以数字方式显式，否则显式为主机名，也就是说**-n选项不做主机名解析**。
4. -nn：除了-n的作用外，还把端口显示为数值，否则显示端口服务名。
5. -N：不打印出host的域名部分。例如tcpdump将会打印'nic'而不是'nic.ddn.mil'。
6. -P：**指定要抓取的包是流入还是流出的包**。可以给定的值为"in"、"out"和"inout"，默认为"inout"。
7. -s len：**设置tcpdump的数据包抓取长度为len，如果不设置默认将会是65535字节**。对于要抓取的数据包较大时，长度设置不够可能会产生包截断，若出现包截断，

：输出行中会出现"[|proto]"的标志(proto实际会显示为协议名)。但是抓取len越长，包的处理时间越长，并且会减少tcpdump可缓存的数据包的数量，

：从而会导致数据包的丢失，所以在能抓取我们想要的包的前提下，抓取长度越小越好。

* 1. 输出选项

1. -e：输出的每行中都将包括数据链路层头部信息，例如源MAC和目标MAC。
2. -q：**快速打印输出。即打印很少的协议相关信息，从而输出行都比较简短**。
3. -X：**输出包的头部数据**，会以16进制和ASCII两种方式同时输出。
4. -XX：输出包的头部数据，会以16进制和ASCII两种方式同时输出，更详细。
5. -v：**当分析和打印的时候，产生详细的输出。**
6. -vv：产生比-v更详细的输出。
7. -vvv：产生比-vv更详细的输出。
8. -A：以ASCII格式打印出报文，不包括十六进制
   1. 其他功能选项
9. **-D：列出可用于抓包的接口。将会列出接口的数值编号和接口名，它们都可以用于"-i"后。**
10. -F：从文件中读取抓包的表达式。若使用该选项，则命令行中给定的其他表达式都将失效。
11. -w：将抓包数据输出到文件中而不是标准输出。可以同时配合"-G time"选项使得输出文件每time秒就自动切换到另一个文件。可通过"-r"选项载入这些文件以进行分析和打印
12. -r：从给定的数据包文件中读取数据。使用"-"表示从标准输入中读取。

## tcpdump表达式

* 1. 作用

1. 表达式用于筛选输出哪些类型的数据包，如果没有给定表达式，所有的数据包都将输出，否则只输出表达式为true的包。在表达式中出现的shell元字符建议使用单引号包围。
   1. 修饰符

tcpdump的表达式由一个或多个"单元"组成，每个单元一般包含ID的修饰符和一个ID(数字或名称)。有三种修饰符：

1. type：指定ID的类型。

以给定的值有host/net/port/portrange。例如"host foo"，"net 128.3"，"port 20"，"portrange 6000-6008"。默认的type为host。

1. dir：指定ID的方向。

可以给定的值包括src/dst/src or dst/src and dst，默认为src or dst。

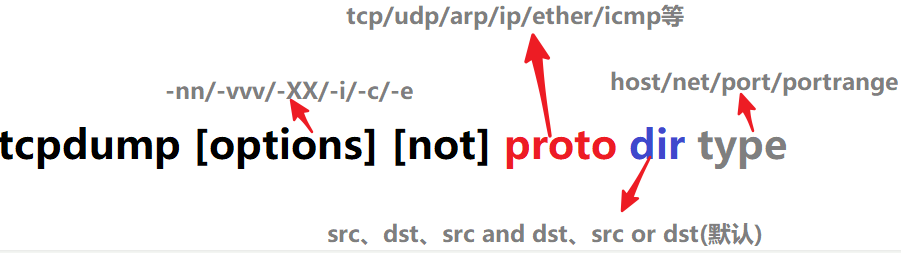
例如，"src foo"表示源主机为foo的数据包，"dst net 128.3"表示目标网络为128.3的数据包，"src or dst port 22"表示源或目的端口为22的数据包。

1. proto：通过给定协议限定匹配的数据包类型。

常用的协议有tcp/udp/arp/ip/ether/icmp等，若未给定协议类型，则匹配所有可能的类型。例如"tcp port 21"，"udp portrange 7000-7009"。

* 1. 使用说明

1. 一个基本的表达式单元格式为"proto dir type ID"



1. 除了使用修饰符和ID组成的表达式单元，还有关键字表达式单元：gateway，broadcast，less，greater以及算术表达式。
2. 表达式单元之间可以使用操作符" and / && / or / || / not / ! "进行连接，从而组成复杂的条件表达式。

如"host foo and not port ftp and not port ftp-data"，这表示筛选的数据包要满足"主机为foo且端口不是ftp(端口21)和ftp-data(端口20)的包"，常用端口和名字的对应关系可在linux系统中的/etc/service文件中找到。

1. 使用括号"()"可以改变表达式的优先级，但需要注意的是括号会被shell解释，所以应该使用反斜线"\"转义为"\(\)"，在需要的时候，还需要包围在引号中。

## 报头过滤工具

4.1 实例报文

tcpdump -A -n host 192.168.50.19 and 10.120.123.120 and port 8170 and tcp[20:2]=0x4854 or tcp[20:2]=0x4745 or tcp[20:2]=0x504f

1. 格式：

tcp（表示协议，如ip/udp）[20:2]（20表示跳过的字节数，2表示过滤的字节数）

1. 作用：

tcp[20:2]作用就是过滤http报文的头两位字节，因为tcp报文一般是20个字节，跳过20个字节就是http报文内容。

（IP，TCP,HTTP报文格式在第19页）

## tcpdump示例

注意，tcpdump只能抓取流经本机的数据包。

(1).默认启动

tcpdump

默认情况下，直接启动tcpdump将监视第一个网络接口(非lo口)上所有流通的数据包。这样抓取的结果会非常多，滚动非常快。

(2).监视指定网络接口的数据包

tcpdump -i eth1

如果不指定网卡，默认tcpdump只会监视第一个网络接口，如eth0。

(3).监视指定主机的数据包，例如所有进入或离开longshuai的数据包

tcpdump host longshuai

(4).打印helios<-->hot或helios<-->ace之间通信的数据包

tcpdump host helios and \( hot or ace \)

(5).打印ace与任何其他主机之间通信的IP数据包,但不包括与helios之间的数据包

tcpdump ip host ace and not helios

(6).截获主机hostname发送的所有数据

tcpdump src host hostname

(7).监视所有发送到主机hostname的数据包

tcpdump dst host hostname

(8).监视指定主机和端口的数据包

tcpdump tcp port 22 and host hostname

(9).对本机的udp 123端口进行监视(123为ntp的服务端口)

tcpdump udp port 123

(10)监视指定网络的数据包，如本机与192.168网段通信的数据包，"-c 10"表示只抓取10个包

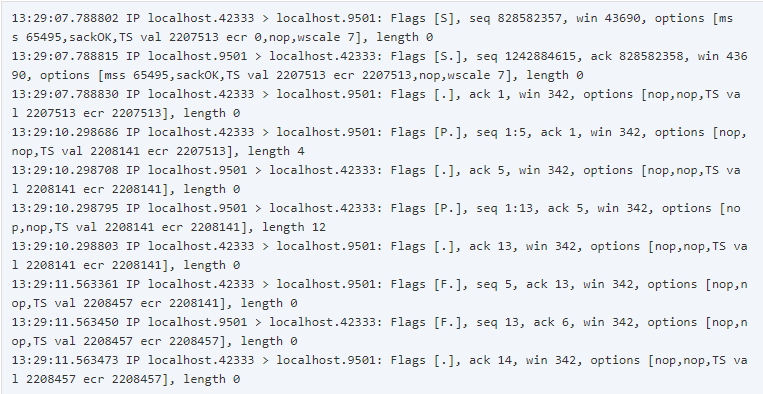
tcpdump -c 10 net 192.168

(11) 只抓取HTTP请求的报文

tcpdump -A -n host 192.168.50.19 and 10.120.123.120 and port 8170 and tcp[20:2]=0x4854 or tcp[20:2]=0x4745 or tcp[20:2]=0x504f

# 命令结果分析

## TCP/IP三次握手和四次挥手



1. 13:29:11.563473 时间带有精确到微妙
2. localhost.42333 > localhost.9501 表示通信的流向，42333是客户端，9501是服务器端

1. [S] 表示这是一个SYN请求
2. [.] 表示这是一个ACK确认包，(client)SYN->(server)SYN->(client)ACK 就是3次握手过程
3. [P] 表示这个是一个数据推送，可以是从服务器端向客户端推送，也可以从客户端向服务器端推
4. [F] 表示这是一个FIN包，是关闭连接操作，client/server都有可能发起
5. [R] 表示这是一个RST包，与F包作用相同，但RST表示连接关闭时，仍然有数据未被处理。可以理解为是强制切断连接
6. win 342是指滑动窗口大小
7. length 12指数据包的大小

## HTTP状态码

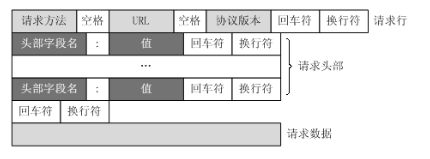
2.1 1xx系列

1XX系列：指定客户端应相应的某些动作，代表请求已被接受，需要继续处理。由于 HTTP/1.0 协议中没有定义任何 1xx 状态码，所以除非在某些试验条件下，服务器禁止向此类客户端发送 1xx 响应。

* 1. 2xx系列

1. 2XX系列：**代表请求已成功被服务器接收、理解、并接受**。这系列中最常见的有200、201状态码。
2. **200状态码：表示请求已成功，请求所希望的响应头或数据体将随此响应返回（还有一个响应是200 from disk cache 浏览器缓存）**
3. 201状态码：表示请求成功并且服务器创建了新的资源，且其 URI 已经随Location 头信息返回。假如需要的资源无法及时建立的话，应当返回 '202 Accepted'
4. 202状态码：服务器已接受请求，但尚未处理
   1. 3xx系列
5. 3XX系列：代表需要客户端采取进一步的操作才能完成请求，这些状态码用来重定向，后续的请求地址（重定向目标）在本次响应的 Location 域中指明。这系列中最常见的有301、302状态码。
6. 301状态码：被请求的资源已永久移动到新位置。服务器返回此响应（对 GET 或 HEAD 请求的响应）时，会自动将请求者转到新位置。
7. 302状态码：请求的资源临时从不同的URI响应请求，但请求者应继续使用原有位置来进行以后的请求
8. **304自从上次请求后，请求的网页未修改过。服务器返回此响应时，不会返回网页内容。 如果网页自请求者上次请求后再也没有更改过，您应将服务器配置为返回此响应(称为 If-Modified-Since HTTP 标头)。**
   1. 4xx系列
9. 4XX系列：表示请求错误。**代表了客户端看起来可能发生了错误，妨碍了服务器的处理**。常见有：401、404状态码。
10. 401状态码：请求要求身份验证。 对于需要登录的网页，服务器可能返回此响应。
11. 403状态码：服务器已经理解请求，但是拒绝执行它。与401响应不同的是，身份验证并不能提供任何帮助，而且这个请求也不应该被重复提交。
12. 404状态码：**请求失败，请求所希望得到的资源未被在服务器上发现**。没有信息能够告诉用户这个状况到底是暂时的还是永久的。假如服务器知道情况的话，应当使用410状态码来告知旧资源因为某些内部的配置机制问题，已经永久的不可用，而且没有任何可以跳转的地址。404这个状态码被广泛应用于当服务器不想揭示到底为何请求被拒绝或者没有其他适合的响应可用的情况下。
    1. 5xx系列
13. 5xx系列：**代表了服务器在处理请求的过程中有错误或者异常状态发生**，也有可能是服务器意识到以当前的软硬件资源无法完成对请求的处理。常见有500、503状态码。
14. 500状态码：服务器遇到了一个未曾预料的状况，导致了它无法完成对请求的处理。一般来说，这个问题都会在服务器的程序码出错时出现。
15. 503状态码：由于临时的服务器维护或者过载，服务器当前无法处理请求。通常，这个是暂时状态，一段时间会恢复

## HTTP请求报文格式





3.1 请求头字段详解

|  |  |  |
| --- | --- | --- |
| 字段名称 | 例子 | 作用 |
| Accept | Accept: text/plain | 表明浏览器接受的内容类型 |
| Accept-Charset | Accept-Charset: utf-8 | 表明浏览器接受的字符编码 |
| Accept-Encoding | Accept-Encoding: gzip, deflate | 表明浏览器接受的编码格式 |
| Accept-Datetime | Accept-Datetime:  Thu,31 May 2007 20:35:00 GMT | 表明浏览器接受的版本时间 |
| Accept-Language | Accept-Language: en-US | 表明浏览器接受的语言 |
| Authorization | Authorization:Basic  QWxhZGRpbjpvcGVuIHNlc2FtZQ== | 设置HTTP身份验证的凭证 |
| Cache-Control | Cache-Control: no-cache | 用于指定缓存机制的指令，  所有的请求和响应必须遵循。 |
| Connection | Connection: keep-alive  Connection: Upgrade | 设置当前连接和hop-by-hop  协议请求字段列表的控制选项 |
| Content-Length | Content-Length: 348 | 设置请求体的字节长度 |
| Content-MD5 | Content-MD5:  Q2hlY2sgSW50ZWdyaXR5IQ== | 设置基于MD5算法对请求体  内容进行Base64二进制编码 |
| Content-Type | Content-Type:application  /x-www-form-urlencoded | 设置请求体MIME类型  （适用POST和PUT请求） |
| Cookie | Cookie: $Version=1; Skin=new; | 发送本地cookie值到服务器，  本地cookie一般是服务器通过  set-cookie字段设置的。 |
| Date | Date:  Tue, 15 Nov 1994 08:12:31 GMT | 设置消息发送的日期和时间 |
| Expect | Expect: 100-continue | 标识客户端需要的特殊浏览  器行为 |
| Forwarded | Forwarded: for=192.0.2.60;  proto=http;by=203.0.113.43 | 披露客户端通过http代理  连接web服务的源信息 |
| From | From: user@example.com | 设置发送请求的用户的  email地址 |
| Host | Host: en.wikipedia.org:8080  Host: en.wikipedia.org | 设置服务器域名和TCP端口号  ，如果使用的是服务请求标准端  口号，端口号可以省略 |
| Origin | Origin:  http://www.baidu.com | 标识跨域资源请求（请求服务  端设置Access-Control-  Allow-Origin响应字段） |
| User-Agent | Mozilla/5.0 (X11; Linux  x86\_64;rv:12.0)Gecko/20100101  Firefox/21.0 | 帮助用户发送请求的代理软  件信息 |

## HTTP响应报文格式



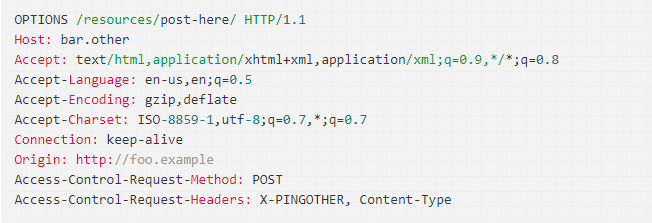
响应报文和请求报文的不同之处在于，响应行由http版本和状态码，原语组成，请求行由请求方法和请求地址，协议版本组成。

* 1. 跨域Cors字段

1. 预检请求

预检请求（option）:在 CORS 中，可以使用 OPTIONS 方法发起一个预检请求(一般都是浏览检测到请求跨域时，会自动发起)，以检测实际请求是否可以被服务器所接受。服务器基于从预检请求获得的信息来判断，是否接受接下来的实际请求。

1. 请求字段

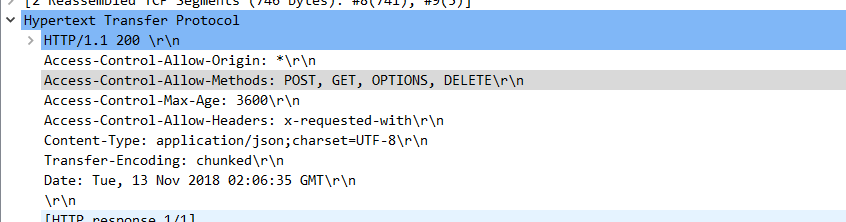


Origin字段表示，要进行跨域请求的网址。

Access-Control-Request-Method 首部字段告知服务器实际请求所使用的 HTTP 方法；

Access-Control-Request-Headers 首部字段告知服务器实际请求所携带的自定义首部字段。

1. 响应字段



Acess-Control-Allow-Origin表示服务器同意跨域的请求网址

Acess-Control-Allow-Headers表示允许的请求头

Acess-Control-Allow-Methods表示允许的请求方法

Acess-Control-Max-Age：3600表示预检请求有效的时间，在这段时间内不需要再进行预检。

* 1. 响应头字段

|  |  |  |
| --- | --- | --- |
| 字段名称 | 例子 | 作用 |
| Access-Control-  Allow-Origin | Access-Control-Allow-Origin: \* | 允许进行跨域的地址 |
| Accept-Patch | Accept-Patch:  text/example;charset=utf-8 | 指定服务器支持的补丁文档格式，适用于http的patch方法 |
| Accept-Ranges | Accept-Ranges: bytes | 服务器通过byte serving支持  的部分内容范围类型 |
| Age | Age: 12 | 对象在代理缓存中暂存的秒数 |
| Allow | Allow: GET, HEAD | 设置特定资源的有效行为，适用  方法不被允许的http 405错误 |
| Cache-Control | Cache-Control: max-age=3600 | 告诉服务端到客户端所有的缓  存机制是否可以缓存这个对象  ，单位是秒 |
| Connection | Connection: close | 设置当前连接和hop-by-hop协  议请求字段列表的控制选项 |
| ContentDisposition | Content-Disposition: attachment;  filename="fname.ext" | 告诉客户端弹出一个文件下载  框，并且可以指定下载文件名 |
| Content-Encoding | Content-Encoding: gzip | 设置数据使用的编码类型 |
| Content-Language | Content-Language: en | 为封闭内容设置自然语言或者目标用户语言 |
| Content-Length | Content-Length: 348 | 响应体的字节长度 |
| Content-Location | Content-Location: /index.htm | 设置返回数据的另一个位置 |
| Content-MD5 | Content-MD5: Q2hlY2sgSW50ZWdyaXR5IQ== | 设置基于MD5算法对响应体内容进行Base64二进制编码 |
| Content-Range | Content-Range: bytes 21010-47021/47022 | 标识响应体内容属于完整消息体中的那一部分 |
| Content-Type | Content-Type:  text/html; charset=utf-8 | 设置响应体的MIME类型 |
| Date | Date:  Tue, 15 Nov 1994 08:12:31 GMT | 设置消息发送的日期和时间 |
| Server | Server: Apache/2.4.1 (Unix) | 服务器名称 |
| Set-Cookie | Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1 | 设置HTTP Cookie |
| Status | Status: 200 OK | 设置HTTP响应状态 |
|  |  |  |
|  |  |  |
|  |  |  |

# HTTP缓存-200（from cache）和304

## 缓存相关字段

* 1. 通用字段

|  |  |
| --- | --- |
| 字段名称 | 释义 |
| Cache-Control | 控制缓存具体的行为 |
| Pragma | HTTP1.0时的遗留字段，当值为”no-cache”时强制验证缓存 |
| Date | 创建报文的日期时间(启发式缓存阶段所用) |

* 1. request字段

|  |  |
| --- | --- |
| 字段名称 | 释义 |
| If-Match | 条件请求，携带上一次请求中资源的ETag，服务器根据这个字段判断文件是否有新的修改 |
| If-None-Match | 和If-Match作用相反，服务器根据这个字段判断文件是否有新的修改 |
| If-Modified-Since | 比较资源前后两次访问最后的修改时间是否一致 |
| If-Unmodified-Since | 比较资源前后两次访问最后的修改时间是否一致 |

* 1. respose字段

|  |  |
| --- | --- |
| 字段名称 | 释义 |
| ETag | 服务器生成资源的唯一标识 |
| Vary | 代理服务器缓存的管理信息 |
| Age | 资源在缓存代理中存贮的时长(取决于max-age和s-maxage的大小) |

* 1. 实体字段

|  |  |
| --- | --- |
| 字段名称 | 释义 |
| Expires | 告知客户端资源缓存**失效的绝对时间** |
| Last-Modified | 资源最后一次修改的时间 |

## 协商缓存（304）

* 1. 根据修改时间

1. 原理：

浏览器在发送请求的时候服务器会检查请求头request header里面的If-modified-Since，如果最后修改时间相同则返回304，否则给返回头(response header)添加last-Modified并且返回数据(response body)。

1. 请求头携带两个字段

if-modified-since:Wed, 31 May 2017 03:21:09 GMT 表示最后修改时间

last-modified 表示资源最后修改时间

1. 服务器检查请求头的两个字段。

当最后修改时间一样，服务器返回304.

当最后修改时间不一样的时候，服务器会在响应头添加last-modified字段，并添加返回内容。

* 1. 根据hash编号

1. 原理：

浏览器在发送请求的时候服务器会检查请求头(request header)里面的if-none-match的值与当前文件的内容通过hash算法（例如 nodejs: cryto.createHash('sha1')）生成的内容摘要字符对比，相同则直接返回304，否则给返回头(response header)添加etag属性为当前的内容摘要字符，并且返回内容。

1. 请求头可携带字段

if-none-match:"42DD5684635105372FE7720E3B39B96A" 表示当前资源的ID

1. 服务器检查请求头字段

当资源ID编号一样的时候，服务器遍返回304。

当资源ID编号不一样的时候，服务器会在响应头添加etag字段，并添加返回内容。

## 强缓存（200 from disk cache）

* 1. 简介

强缓存是本地缓存，强缓存是利用http头中的Expires和Cache-Control两个字段来控制的，用来表示资源的缓存时间。**强缓存中，普通刷新会忽略它，但不会清除它，需要强制刷新。**浏览器强制刷新，请求会带上Cache-Control:no-cache和Pragma:no-cache。

* 1. Expires字段

1. Expires是http1.0的规范，它的值是一个绝对时间的GMT格式的时间字符串。
2. expires:Fri, 14 Apr 2017 10:47:02 GMT。这个时间代表这这个资源的失效时间，只要**发送请求时间是在Expires之前，那么本地缓存始终有效，则在缓存中读取数据**。
3. 这种方式有一个明显的缺点，**由于失效的时间是一个绝对时间，所以当服务器与客户端时间偏差较大时，就会导致缓存混乱**。
4. 如果同时出现Cache-Control:max-age和Expires，那么max-age优先级更高。

如：cache-control:max-age=691200

expires:Fri, 14 Apr 2017 10:47:02 GMT

表示资源可以被缓存的最长时间为691200秒，会优先考虑max-age。

* 1. Cache-Control字段

1. Cache-Control是在http1.1中出现的，主要是利用该字段的max-age值来进行判断，它是一个相对时间，例如Cache-Control:max-age=3600，代表着资源的有效期是3600秒。
2. 其他值选项

no-cache：不使用本地缓存。需要使用缓存协商，先与服务器确认返回的响应是否被更改，如果之前的响应中存在ETag，那么请求的时候会与服务端验证，如果资源未被更改，则可以避免重新下载。

no-store：直接禁止浏览器缓存数据，每次用户请求该资源，都会向服务器发送一个请求，每次都会下载完整的资源。

public：可以被所有的用户缓存，包括终端用户和CDN等中间代理服务器。

private：只能被终端用户的浏览器缓存，不允许CDN等中继缓存服务器对其缓存。

Cache-Control与Expires可以在服务端配置同时启用，同时启用的时候Cache-Control优先级高。

1. cache-control中 max-age和no-cache的区别

no-cache,不经过本地缓存版本的校验，直接发送请求到浏览器。

max-age>0时，先校验本地缓存版本。max-age<=0时，发送请求到浏览器。

# HTTP请求举例

1. 200 状态码，表示成功



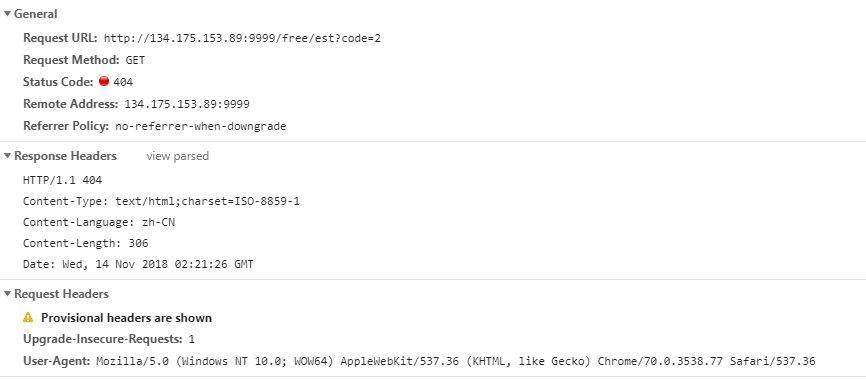
1. 200 from disk cache 强缓存，表示从本地缓存中获取



1. 304 协商缓存，与服务器确认



1. 404 客户端访问的资源不存在



1. 500 服务端有错误



# TCP/IP报文格式

## IP包头格式(IPv4)



1. 版本：指IP协议的版本，通信双方使用的IP协议版本必须一致。一般的值为0100（IPv4），0110（IPv6）。
2. 首部长度：长度4比特。这个字段的作用是为了描述IP包头的长度，因为在IP包头中有变长的可选部分。该部分占4个bit位，单位为32bit（4个字节），即本区域值= IP头部长度（单位为bit）/(8\*4)，因此，一个IP包头的长度最长为“1111”，即15\*4＝60个字节。IP包头最小长度为20字节。
3. 优先级与服务类型：长度8比特，定义了数据包传输的紧急程度以及时延、可靠性、传输成本等。
4. 总长度：16比特，以字节为单位描述IP包的总长度（包括头部和数据两部分），最大值为65535。第二行中标识符、标志和段偏移量通常联合使用，用于数据拆分时的分组和重组。
5. 标识符：对于上层发来的较大的数据包，往往需要拆分。路由器将一个大包进行拆分后，拆出来的所有部分被标上相同的值，该值即为标识符，用于告诉目的端哪些包属于同一个大包。
6. 标志：长度3比特。该字段第一位不使用。第二位是DF（Don't Fragment）位，DF位设为1时表明路由器不能对该上层数据包分段。如果一个上层数据包无法在不分段的情况下进行转发，则路由器会丢弃该上层数据包并返回一个错误信息。第三位是MF（More Fragments）位，当路由器对一个上层数据包分段，则路由器会在除了最后一个分段的IP包的包头中将MF位设为1。
7. 段偏移量：长度13比特，表示一个数据包在原先被拆分前的大包中的位置。接收端据此来还原和组装IP包。
8. TTL：表示IP包的生存时间，长度8比特。长度8比特。当IP包进行传送时，先会对该字段赋予某个特定的值。当IP包经过每一个沿途的路由器的时候，每个沿途的路由器会将IP包的TTL值减少1。如果TTL减少为0，则该IP包会被丢弃。这个字段可以防止由于路由环路而导致IP包在网络中不停被转发。
9. 协议号：长度8比特，标识上一层即传输层在本次数据传输中所使用的协议。比如6代表TCP，17代表UDP等
10. 首部校验和：长度16位。用来做IP头部的正确性检测，但不包含数据部分。 因为每个路由器要改变TTL的值,所以路由器会为每个通过的数据包重新计算这个值。
11. 源地址：长度32比特，标识IP包的起源地址。
12. 目标地址：长度32比特，表示IP包的目的地址。
13. 可选项：可变长字段，主要用于测试，由起源设备跟据需要改写。
14. 填充：因为IP包头长度（Header Length）部分的单位为32bit，所以IP包头的长度必须为32bit的整数倍。因此，在可选项后面，IP协议会填充若干个0，以达到32bit的整数倍。

## TCP报文格式



1. 16位源端口号和16位目的端口号。
2. 32位序号：一次TCP通信过程中某一个传输方向上的字节流的每个字节的编号，通过这个来确认发送的数据有序，比如现在序列号为1000，发送了1000，下一个序列号就是2000。
3. 32位确认号：用来响应TCP报文段，给收到的TCP报文段的序号加1，三握时还要携带自己的序号。
4. 4位头部长度：标识该TCP头部有多少个4字节，共表示最长15\*4=60字节。同IP头部。
5. 6位保留。6位标志。URG（紧急指针是否有效）ACK（表示确认号是否有效）PSH（提示接收端应用程序应该立即从TCP接收缓冲区读走数据）RST（表示要求对方重新建立连接）SYN（表示请求建立一个连接）FIN（表示通知对方本端要关闭连接）
6. 16位窗口大小：TCP流量控制的一个手段，用来告诉对端TCP缓冲区还能容纳多少字节。
7. 16位校验和：由发送端填充，接收端对报文段执行CRC算法以检验TCP报文段在传输中是否损坏。
8. 16位紧急指针：一个正的偏移量，它和序号段的值相加表示最后一个紧急数据的下一字节的序号。