网络协议笔记4-应用层（HTTP）

应用层常见协议

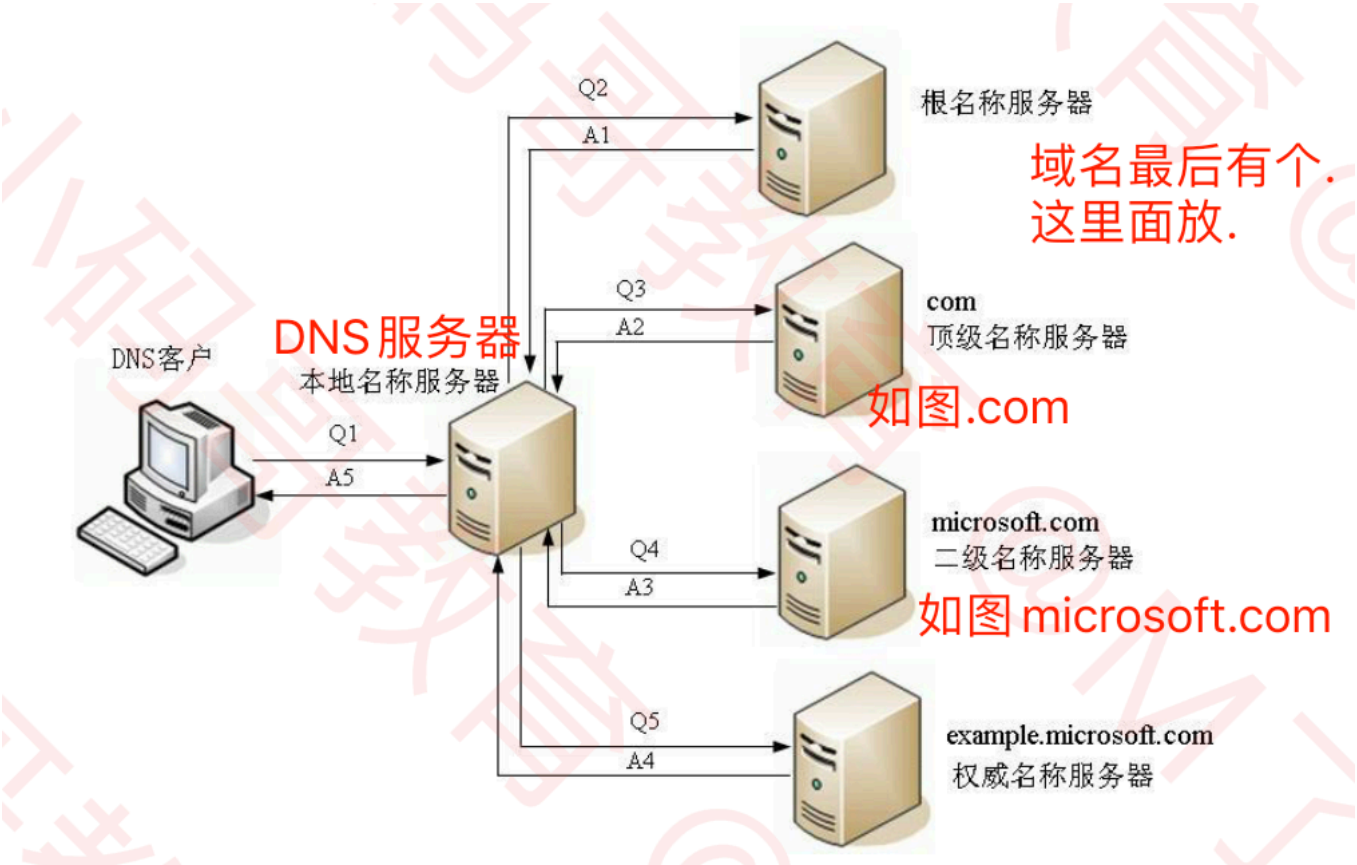
- 超文本传输协议：HTTP，HTTPS
- 文件传输：FTP
- 电子邮件：SMTP、POP3、IMAP
- 动态主机配置：DHCP
- 域名系统：DNS

域名

- 包含顶级、二级等等域名，
  - 顶级是.com .org .edu等
  - 二级是.com等等前面一级
  - 以此类推

DNS

- 全称是Domain Name System 域名系统
  - 利用DNS协议，可以将域名解析成对应的ip地址
  - DNS可以基于UDP，也可以基于TCP协议，占用端口53
  - DNS服务器解析步骤图示：



IP地址分配

- 分为静态IP地址和动态IP地址
  - 静态IP地址
    - 手动设置
    - 使用场景：不怎么挪动的台式机，服务器等
  - 动态IP地址
    - 从DHCP服务器自动获取IP地址
      - DHCP:(Dynamic Host Protocol)，动态主机配置协议，基于UDP协议，客户端是68端口，服务器是67端口
      - DHCP服务器会在IP地址池中选择一个IP地址出租给客户端，过一段时间会回收
    - 使用场景：移动设备，无线设备
- DHCP分配IP的四个阶段
  - DISCOVER：发现服务器。发广播包（源ip：0.0.0.0 目标IP：255.255.255.255，目标MAC：FF：FF：FF：FF：FF：FF）
  - OFFER：提供租约：服务器返回可以租用的IP，租用期限，子网掩码，网关，DNS等
  - REQUEST：选择IP地址：客户端选择一个offer，发送广播包回应
  - ACKNOWLEDGE：确认：被选中的服务器发送ACK数据包给客户端，ip地址分配完毕

DHCP细节

- DHCP可以跨网段分配IP地址，借助中继代理实现
- 客户端在租期不足的时候，自动向DHCP服务器发送REQUEST信息申请续约

HTTP协议

- 超文本传输协议（英语：HyperText Transfer Protocol，缩写：HTTP）
- 设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法
- 通过HTTP或者HTTPS协议请求的资源由统一资源标识符（Uniform Resource Identifiers，URI）来标识。

版本

- http/0.9 只支持get
- http/1.0 支持post head等，支持请求头，响应头和更多数据类型
- http/1.1 （经典，广泛使用），支持put，delete等。默认采用持续连接（Connection: keep-alive），能很好地配合代理服务器工作。还支持以管道方式在同时发送多个请求，以便降低线路负载，提高传输速度。
- http/2.0
- http/3.0

报文格式

- ABNF核心规则图示：

规则	形式定义	意
ALPHA	%x41-5A / %x61-7A	大写和小写ASCII
DIGIT	%x30-39	数字
HEXDIG	DIGIT / "A" / "B" / "C" / "D" / "E" / "F"	十六进制数字（0
DQUOTE	%x22	双
SP	%x20	空
HTAB	%x09	横向
WSP	SP / HTAB	空格或横
LWSP	*(WSP / CRLF WSP)	直线空白
VCHAR	%x21-7E	可见（打
CHAR	%x01-7F	任何7-位US-ASCII字
OCTET	%x00-FF	8位
CTL	%x00-1F / %x7F	控制
CR	%x0D	回
LF	%x0A	换
CRLF	CR LF	互联网标准换行
BIT	"0" / "1"	二进

HTTP-message = start-line **start-line = request-line / status**

***(header-field CRLF)**

CRLF

[message-body]

/	任选一个
*	0个或多个。2*表示至少2个，3*6表示3到6个
( )	组成一个整体
[ ]	可选（可有可无）

```
request-line = method SP request-target SP HTTP-version CRLF
HTTP-version = HTTP-name "/" DIGIT "." DIGIT
HTTP-name = %x48.54.54.50 ; HTTP

GET /hello/ HTTP/1.1

status-line = HTTP-version SP status-code SP reason-phrase CRLF
status-code = 3DIGIT
reason-phrase = *( HTAB / SP / VCHAR / obs-text )

HTTP/1.1 200
HTTP/1.1 200 OK


header-field = field-name ":" OWS field-value OWS

field-name = token

field-value = *( field-content / obs-fold )

OWS = *( SP / HTAB )


message-body = *OCTET
```

URL编码

- URL中出现的特殊字符（比如中文，空格）需要encode

请求方法

- HTTP支持如下9种请求方法
  1. GET
     - 最常见的用于读取的操作，请求参数拼接在URL后面
  2. POST
     - 常用语添加修改删除操作，请求参数放到请求体中
  3. HEAD
     - 得到和GET请求一样的响应，但没有响应体

    - 可在下载前获取响应头中的文件大小，以决定是否继续下载，节省资源
4. PUT
    - 对已存在的资源进行整体覆盖
5. DELETE
    - 请求服务器删除Request-URI所标识的资源。
6. CONNECT
    - HTTP/1.1协议中预留给能够将连接改为隧道方式的代理服务器。通常用于SSL加密服务器的链接（经由非加密的HTTP代理服务器）。方法名称是区分大小写的。当某个请求所针对的资源不支持对应的请求方法的时候，服务器应当返回状态码405（Method Not Allowed），当服务器不认识或者不支持对应的请求方法的时候，应当返回状态码501（Not Implemented）。
7. OPTIONS
    - 这个方法可使服务器传回该资源所支持的所有HTTP请求方法。用'*'来代替资源名称，向Web服务器发送OPTIONS请求，可以测试服务器功能是否正常运作。
    - 例如发送 OPTIONS * HTTP/1.1 HOST:localhost:8080 。 服务器返回实现了的请求方法 GET和HEAD以及POST
8. TRACE
    - 回显服务器收到的请求，主要用于测试或诊断。
9. PATCH
    - 用于将局部修改应用到资源。

头部字段

- 请求头字段
    - User-Agent : 浏览器的浏览器身份标识字符串. 例如：User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/21.0，常设
    - Host : 服务器的域名(用于虚拟主机)，以及服务器所监听的传输控制协议端口号。如果所请求的端口是对应的服务的标准端口，则端口号可被省略。自超文件传输协议版本1.1（HTTP/1.1）开始便是必需字段。 例如：Host: en.wikipedia.org。 常设
    - Date： 发送该消息的日期和时间(按照 RFC 7231 中定义的"超文本传输协议日期"格式来发送) 。 如：Date: Tue, 15 Nov 1994 08:12:31 GMT，常设。
    - Referer： 表示浏览器所访问的前一个页面，正是那个页面上的某个链接将浏览器带到了当前所请求的这个页面。 如：从localhost:8080/index.html跳转到百度，该字段就是localhost:8080/index.html 。 常设。
    - Content-Type： 请求体的 多媒体类型 （用于POST和PUT请求中）。例如：Content-Type: application/x-www-form-urlencoded 。 常设。
    - Content-Length：以 八位字节数组 （8位的字节）表示的请求体的长度。如：Content-Length: 348 。常设
    - Accept： 能够接受的回应内容类型（Content-Types）。例如： Accept: text/plain 。 常设
    - Accept-Charset： 能够接受的字符集 。 Accept-Charset: utf-8 常设
    - Accept-Encoding 能够接受的编码方式列表。Accept-Encoding: gzip, deflate 常设
    - Accept-Language 能够接受的回应内容的自然语言列表。Accept-Language: en-US 常设
    - Range 仅请求某个实体的一部分。字节偏移以0开始。Range: bytes=500-999 常设。 **常用来断点续传，从500开始接受数据**
    - Origin： 发起一个针对 跨来源资源共享 的请求（要求服务器在回应中加入一个'访问控制-允许来源'（'Access-Control-Allow-Origin'）字段）。Origin: http://www.example-social-network.com。常设: 标准
    - Cookie：与响应头的set-cookie对应，之前由服务器通过 Set- Cookie （下文详述）发送的一个 超文本传输协议Cookie。 Cookie: $Version=1; Skin=new; 常设: 标准
    - Connection：该浏览器想要优先使用的连接类型 。 Connection: keep-alive 。 常设
    - Cache-Control： 用来指定在这次的请求/响应链中的所有缓存机制 都必须 遵守的指令。Cache-Control: no-cache。常设
- 响应头字段
    - Date：此条消息被发送时的日期和时间 Date: Tue, 15 Nov 1994 08:12:31 GMT 常设
    - Last-Modified：所请求的对象的最后修改日期 。 Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT 常设
    - Server： 服务器的名字 。Server: Apache/2.4.1 (Unix) 。 常设
    - Expires： 指定一个日期/时间，超过该时间则认为此回应已经过期。Expires: Thu, 01 Dec 1994 16:00:00 GMT 常设: 标准
    - Content-Type： 当前内容的MIME类型 。Content-Type: text/html; charset=utf-8 常设
    - Content-Encoding： 在数据上使用的编码类型。Content-Encoding: gzip 常设
    - Content-Length：回应消息体的长度，以 字节 （8位为一字节）为单位。Content-Length: 348 常设
    - Content-Disposition： 一个可以让客户端下载文件并建议文件名的头部。文件名需要用双引号包裹。 Content-Disposition: attachment; filename="fname.ext" 常设。

- Accept-Ranges：这个服务器支持哪些种类的部分内容范围。Accept-Ranges: bytes 常设
- Content-Range：这条部分消息是属于某条完整消息的哪个部分。Content-Range: bytes 21010-47021/47022。常设
- Access-Control-Allow-Origin：指定哪些网站可参与到跨来源资源共享过程中。Access-Control-Allow-Origin: * *临时*
- Location：用来 进行重定向，或者在创建了某个新资源时使用。Location: http://www.w3.org/pub/WWW/People.html。 常设
- Set-Cookie：返回一个cookie让客户端存储。Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1 常设: 标准
- Connection：针对该连接所预期的选项 。 Connection: close。常设
- Cache-Control： 向从服务器直到客户端在内的所有缓存机制告知，它们是否可以缓存这个对象。其单位为秒 Cache-Control: max-age=3600 常设

Cookie

指某些网站为了辨别用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密） - 分类： Cookie 保存在客户端中，按在客户端中的存储位置，可分为内存 Cookie 和硬盘 Cookie。 内存 Cookie 由浏览器维护，保存在内存中，浏览器关闭即消失，存在时间短暂。硬盘 Cookie 保存在硬盘里，有过期时间，除非用户手动清理或到了过期时间，硬盘 Cookie 不会清除，存在时间较长。所以，按存在时间，可分为非持久 Cookie 和持久 Cookie。 - 用途 因为 HTTP 协议是无状态的，即服务器不知道用户上一次做了什么，这严重阻碍了交互式 Web 应用程序的实现。在典型的网上购物场景中，用户浏览了几个页面，买了一盒饼干和两瓶饮料。最后结帐时，由于 HTTP 的无状态性，不通过额外的手段，服务器并不知道用户到底买了什么，所以 Cookie 就是用来绕开 HTTP 的无状态性的“额外手段”之一。服务器可以设置或读取 Cookies 中包含的信息，借此维护用户跟服务器会话中的状态。

在刚才的购物场景中，当用户选购了第一项商品，服务器在向用户发送网页的同时，还发送了一段 Cookie，记录着那项商品的信息。当用户访问另一个页面，浏览器会把 Cookie 发送给服务器，于是服务器知道他之前选购了什么。用户继续选购饮料，服务器就在原来那段 Cookie 里追加新的商品信息。结帐时，服务器读取发送来的 Cookie 即可。

Cookie 另一个典型的应用是当登录一个网站时，网站往往会请求用户输入用户名和密码，并且用户可以勾选“下次自动登录”。如果勾选了，那么下次访问同一网站时，用户会发现没输入用户名和密码就已经登录了。这正是因为前一次登录时，服务器发送了包含登录凭据（用户名加密码的某种加密形式）的 Cookie 到用户的硬盘上。第二次登录时，如果该 Cookie 尚未到期，浏览器会发送该 Cookie，服务器验证凭据，于是不必输入用户名和密码就让用户登录了。

- 缺陷：
  - Cookie 会被附加在每个 HTTP 请求中，所以无形中增加了流量。
  - 由于 HTTP 请求中的 Cookie 是明文传递的，所以安全性成问题，除非使用超文本传输安全协定。
  - Cookie 的大小限制在 4 KB 左右，对于复杂的存储需求来说是不够用的。[3]

状态码

分为以下五类

- 1XX信息响应
  - 100 Continue: 服务器已经接收到请求头，并且客户端应继续发送请求主体（在需要发送身体的请求的情况下：例如，POST请求），或者如果请求已经完成，忽略这个响应。服务器必须在请求完成后向客户端发送一个最终响应。要使服务器检查请求的头部，客户端必须在其初始请求中发送Expect: 100-continue作为头部，并在发送正文之前接收100 Continue状态代码。响应代码417期望失败表示请求不应继续。
- 2XX成功响应
  - 200 OK: 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。实际的响应将取决于所使用的请求方法。在GET请求中，响应将包含与请求的资源相对应的实体。在POST请求中，响应将包含描述或操作结果的实体。
- 3XX重定向
  - 302 Found: 要求客户端执行临时重定向（原始描述短语为“Moved Temporarily”）。由于这样的重定向是临时的，客户端应当继续向原有地址发送以后的请求。只有在Cache-Control或Expires中进行了指定的情况下，这个响应才是可缓存的。 新的临时性的URI应当在响应的Location域中返回。除非这是一个HEAD请求，否则响应的实体中应当包含指向新的URI的超链接及简短说明。 如果这不是一个GET或者HEAD请求，那么浏览器禁止自动进行重定向，除非得到用户的确认，因为请求的条件可能因此发生变化。 注意：虽然RFC 1945和RFC 2068规范不允许客户端在重定向时改变请求的方法，但是很多现存的浏览器将302响应视作为303响应，并且使用GET方式访问在Location中规定的URI，而无视原先请求的方法。因此状态码303和307被添加了进来，用以明确服务器期待客户端进行何种反应。
  - 304 Not Modified: 表示资源在由请求头中的If-Modified-Since或If-None-Match参数指定的这一版本之后，未曾被修改。在这种情况下，由于客户端仍然具有以前下载的副本，因此不需要重新传输资源。
- 4XX客户端错误
  - 401 Unauthorized: 类似于403 Forbidden，401语义即“未认证”，即用户没有必要的凭据。
  - 403 Forbidden: 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。如果这不是一个HEAD请求，而且服务器希望能够讲清楚为何请求不能被执行，那么就应该在实体内描述拒绝的原因。当然服务器也可以返回一个404响应，假如它不希望让客户端获得任何信息。
- 5XX服务端错误
  - 501 Not Implemented: 服务器不支持当前请求所需要的某个功能。当服务器无法识别请求的方法，并且无法支持其对任何资源的请求。服

务器一定支持的方法是GET和HEAD

- 502 Bad Gateway: 作为网关或者代理工作的服务器尝试执行请求时，从上游服务器接收到无效的响应。