# 域名解析成IP

每个主机在网络中的位置都是以IP标识的，IP才是主机在网络中的位置，域名只是为了方便用户记忆而已，这就要求浏览器能够识别域名并且将其转化为对应的IP地址。

所以浏览器会有一个DNS缓存，其中记录了一些域名与IP的对应关系，供浏览器快速查找需要的IP。但是这个DNS缓存不可能存下所有的域名-IP地址，何况IP地址有时候还会变化，因此当在DNS缓存中没有找到的时候，就要先向DNS服务器请求域名解，DNS域名解析时用的是UDP协议。

①客户端从浏览器中输入[www.baidu.com](http://www.baidu.com)网站网址后回车，首先浏览器会查询浏览器本身的DNS缓存，一般只有几分钟的缓存，找到了就返回域名对应的IP；如果找不到，系统就会查询本地hosts文件和本地DNS缓存信息，如果找到了，就返回域名对应IP；

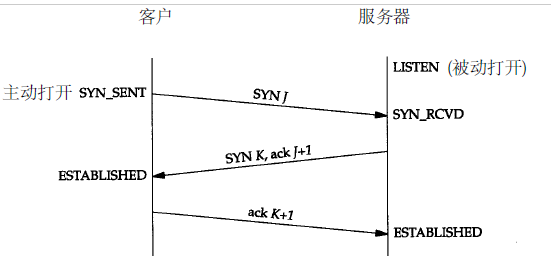
②如果没有找到对应的域名解析记录，那么那么系统会把浏览器的解析请求，交给客户端本地设置的DNS服务器地址解析（称为Local DNS，LDNS），如果LDNS服务的本地缓存有对应的解析记录，就会直接返回IP地址；

③如果没有LDNS会负责继续请求其他DNS服务器；此时就是外网的DNS服务器了，根域名服务器，根据浏览器得到的域名，根域名服务器看到.com，会返回.com的顶级域名服务器的ip给LDNS；然后LDNS通过顶级域名服务器的ip，找到了顶级域名服务器，.com顶级域名服务器看到了是找baidu.com一级域名服务器，就将其服务器的ip返回给LDNS；然后一层一层往下找，直到找到了www.baidu.com的DNS记录，并得到对应的IP地址，这时候LDNS会把找到的[www.baidu.com的ip](http://www.baidu.com的ip)发送给客户端浏览器，并记录在缓存中，以便未来再次访问。

④客户端浏览器收到ip之后，就可以通过ip地址找到对应的web服务器了，即服务端主机；接下来就是三次握手建立连接了

# 与目的主机进行TCP连接（三次握手）

得到域名对应的ip地址后，也就表示可以将数据送达到目的主机了，即可以向服务器发送http请求了，但是http是应用层写，tcp是传输层，所以发送http请求之前，开始我们常说的三次握手。Http请求是使用TCP进行传输的，可以保证可靠传输，并且有序，需要建立连接，才能进行数据传输。



①向目的主机发送TCP连接请求报文

②该TCP报文中SYN标志位为1，产生一个随机数j，表示连接请求

③该TCP报文通过获取的ip（DNS）找到服务器主机，然后获得MAC地址（ARP），通过网关，最终到达目的主机。

④目的主机收到数据帧后，通过ip协议传输帧，再到TCP协议，封装成请求应答报文；

⑤该报文中SYN标志为1，产生一个随机数k，ack标志位j+1，表示连接请求应答

⑥该请求应答报文通过接收到的源ip-》Mac（arp）-》网关，发送到我的主机；

⑦我的主机收到数据帧，通过ip协议传输帧，再到TCP协议，封装成请求确认报文

⑧该请求确认报文通过目标ip-》Mac（arp）-》网关，发送到目的主机

⑨请求确认报文的ack为k +1，表示请求确认；

⑩目的主机接收到数据帧，连接建立完成

# 发送和收取数据

建立连接之后，就可以发送数据了，即发送http请求

以get方法为例：

①浏览器向服务器发送get方法报文

②该get方法报文通过tcp-》ip-》mac-》网关-》目的主机

③目的主机收到数据帧，通过ip-tcp-http，http协议单元回应http协议格式封装好的HTML形式数据；

④该HTML数据通过tcp-》ip-》mac-》网关-》我的主机

⑤我的主机收到数据帧，通过ip-tcp-http-浏览器，以网页形式显式HTML

## http请求头详解

Post请求是可以有请求体的，而get请求不能有请求体

1. 格式
   1. 请求首行
   2. 请求头信息
   3. 空行
   4. 请求体

**GET请求**

|  |
| --- |
| GET /hello/index.jsp HTTP/1.1  **Host**: localhost  **User-Agent**: Mozilla/5.0 (Windows NT 5.1; rv:5.0) Gecko/20100101 Firefox/5.0  **Accept**: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  **Accept-Language**: zh-cn,zh;q=0.5  **Accept-Encoding:** gzip, deflate  **Accept-Charset**: GB2312,utf-8;q=0.7,\*;q=0.7  **Connection**: keep-alive  **Cookie**: JSESSIONID=369766FDF6220F7803433C0B2DE36D98  -------------此处是空行------------，get没有请求体 |

|  |  |
| --- | --- |
| GET。。。 | GET请求，请求服务器路径为/hello/index.jsp,协议为http1.1 |
| Host:l。。。 | 请求的主机名为localhost |
| User-agent:。。。 | 显示用户系统版本和浏览器版本信息 |
| Accpet：text。。。 | 告诉服务器，当前客户端可以接受的类型是文档类型，这里\*/\*表示什么都可以接受 |
| Accept-language | 当前客户端支持的语言，可以在浏览器的工具选项中找到语言信息 |
| Accept-encodeing | 支持的压缩格式，数据在网络上传递时，可能服务器会把数据压缩后发送 |
| Accept-charset | 客户端支持的编码 |
| Connection | 客户端支持的链接方式，保持一段时间链接，默认3000ms |
| Cookie | 会在请求中把上一次服务器响应中发送过来的cookie在请求中一并发送，Cookies最典型的应用是判定注册用户是否已经登录网站，用户可能会得到提示，是否在下一次进入此网站时保留用户信息以便简化登录手续，这些都是Cookies的功用。另一个重要应用场合是“购物车”之类处理。 |

**Post请求**

|  |
| --- |
| POST /hello/index.jsp HTTP/1.1  **Accept**: image/gif, image/jpeg, image/pjpeg, image/pjpeg, application/msword, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/x-ms-application, application/x-ms-xbap, application/vnd.ms-xpsdocument, application/xaml+xml, \*/\*  **Referer**: http://localhost:8080/hello/index.jsp  **Accept-Language**: zh-cn,en-US;q=0.5  **User-Agent**: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; InfoPath.2; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)  **Content-Type**: application/x-www-form-urlencoded  **Accept-Encoding**: gzip, deflate  **Host**: localhost:8080  **Content-Length**: 13  **Connection**: Keep-Alive  **Cache-Control**: no-cache  **Cookie**: JSESSIONID=E365D980343B9307023A1D271CC48E7D  ---------------------------此处为空行-----------------------  Keyword=hello |

|  |  |
| --- | --- |
| Referer | 请求来自哪个页面，例如你在百度上点击链接到本页面，那么referer就是百度网址，如果你在浏览器中直接输入网址，那么就没有referer请求头了 |
| Content-type | 表单的数据类型，说明会使用了url个编码数据；url编码的数据都是以%为前缀的，后面跟随两位16进制，如传智两个字使用utf-8变为**%E4%BC%A0%E6%99%BA** |
| Content-length | 请求体的长度，这里表示13个字节 |
| Keyword=hello | 请求体内容，hello是表单输入的数据，keyword是表单字段的名字 |

## http相应头详解

① 格式

响应首行；

响应头信息；

空行；

响应体。

|  |
| --- |
| HTTP/1.1 200 OK  Server: Apache-Coyote/1.1  Content-Type: text/html;charset=UTF-8  Content-Length: 724  Set-Cookie: JSESSIONID=C97E2B4C55553EAB46079A4F263435A4; Path=/hello  Date: Wed, 25 Sep 2012 04:15:03 GMT |

|  |  |
| --- | --- |
| http/1.1 | 响应协议为HTTP1.1.状态码200，表示请求成功，ok对状态码的解释 |
| Server：apache | 服务器的版本信息 |
| Content-type | 响应体使用的是utf-8编码 |
| Content-length | 响应体为724字节 |
| Set-cookie | 响应给客户端的cookie |
| Date | 响应的时候，这可能会有8小时时差 |

### 响应码

1XX Informational（信息性状态码）接收的请求正在处理

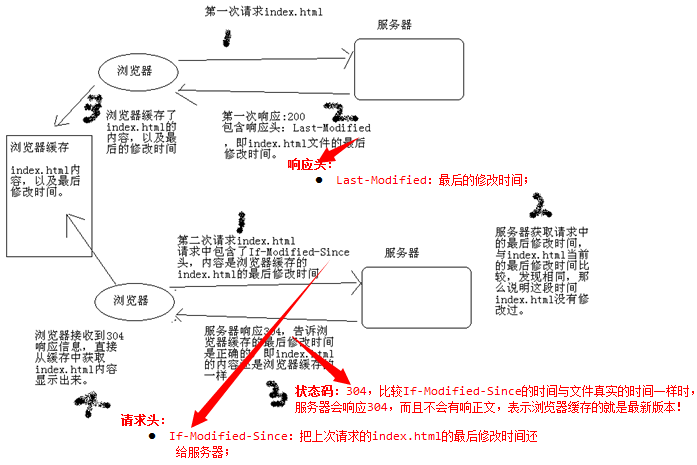
2XX Success（成功状态码） 请求正常处理完毕

3XX Redirection（重定向状态码） 需要进行附加操作以完成请求

4XX Client Error（客户端错误状态码）服务器无法处理请求

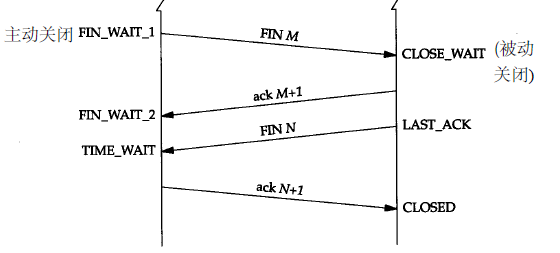
5XX Server Error（服务器错误状态码）服务器处理请求出错

|  |  |
| --- | --- |
| 200 | 请求成功，浏览器会把响应体内容显示在浏览器中 |
| 404 | 请求的资源没有找到，说明**客户端**错误地请求了不存在的资源 |
| 500 | 请求资源找到了，但是**服务器**内部出现了故障 |
| 302 | 重定向，当响应码为302是，表示服务器要求浏览器重新再发一个请求，服务器会发送一个响应头location，它指定了新请求的url地址 |
| 304 | 当用户第一次请求index.html时，服务器会添加一个名为last-Modified响应头，这个头说明了index.html的最后修改时间，浏览器会把index.html内容，以及最后的响应时间缓存下来。当用户第二次请求index.html时，在请求中包含一个if-modified-since请求头，它的值就是第一次请求时服务器通过last-modified响应头发送给浏览器的值，即index.html最后的修改时间，if-modified-since请求头就是在告诉服务器，我这里浏览器缓存的index.html最后的修改时间是这个，您看看现在的index.html最后修改的时间是不是这个，如果是，那就不用响应这个请求，我直接把缓存中的内容显示出来即可。服务器获取if-modified-since值，与index.html的当前最后修改时间比对，如果相同，服务器发送响应码304，表示index.html与上次浏览器缓存的相同，无需再次发送，浏览器显示自己的缓存页面，如果不同，那么说明index.html已经进行了修改，服务器响应200 |



# 与目的主机断开TCP连接（四次挥手）

数据传输结束之后需要断开连接，与建立连接不同，断开连接需要多一次手，四次挥手



①浏览器向目的主机发出连接结束报文，此时进入FIN WAIT状态；

②连接结束报文标志位FIN=1，并且产生随机数M

③TCP连接结束请求报文通过ip-》Mac（arp）-》网关-》目的主机

④目的主机接收到数据帧，通过ip-》tcp，通过tcp协议单元回应结束应答报文

⑤结束应答报文中ack = M + 1，表示收到结束请求，当前只是进行回应，因为目的主机可能还有数据要传，并不急着断开连接。

⑥等到浏览器发送完所有数据后，目的主机向我的主机发出tcp连接结束请求报文；

⑦该报文FIN标志位1，并且产生随机数N，表示结束请求

⑧tcp结束请求报文通过ip-》Mac（arp）-》网关-》我的主机

⑨我的主机收到数据帧，通过ip-》tcp，tcp协议单元回应结束应答报文，此时进入TIME WAIT状态，因为不相信网络是可靠的，**如果目的主机没收到，还能够重发结束应答报文**

⑩该回应结束应答报文中的FIN标志为1，ack=N+1；表示结束应答，该tcp报文通过ip-》Mac（arp）-》网关-》目的主机；目的主机关闭连接，如果TIME WAIT等待结束后，没有收到回复，说明目的主机连接正常关闭了，我的主机也关闭连接

FIN WAIT状态是主动发起请求时，等待确认信息，而TIME WAIT状态是收到结束请求后发送确认信息后等待是否需要重发；

为什么断开连接需要四次挥手？

因为建立连接时，目的主机可以直接发送SYN+ACK应答报文，而当目的主机收到FIN后，可能还有数据要发，并不一定直接断开，所以先发送一次应答，告知我的主机收到了连接结束请求。等确认所有数据都发完了，在发送FIN，同时等待我的主机应答，这里的FIN和ACK不能一起发送，因为可能还有数据要传输，所以需要四次