万维网和HTTP协议

1.万维网

万维网WWW（World Wide Web）是一个大规模的、联机式的信息储藏所。是无数网络站点的集合。

万维网以客户服务器方式工作。客户程序向服务器发出请求，服务器程序向客户程序送回客户所要的万维网文档。在一个客户程序主窗口上显示出的万维网文档称为**页面**（page）。

要实现上面的过程，必须解决以下问题：
(1) 怎么标志分布在整个互联网上的万维网文档——**统一资源定位符URL。**

(2) 用什么样的协议实现万维网上的各种连接——**超文本传送协议HTTP。**

(3) 怎样使不同风格的万维网文档都能在互联网上的主机上显示——**超文本标记语言HTML。**

2.URL与URI

2.1 URL

统一资源定位符URL（Uniform Resources Locator）用来**表示资源在互联网上的位置和访问这些资源的方法**，并使每一个资源在整个互联网范围内具有唯一的标识符URL。即URL**实际上就是在互联网上资源的地址。**

URL的一般形式

> **<协议>://<主机>:<端口>/<路径>**

(1) 第一部分是最左边的<协议>。这里的<协议>是指用什么协议来获取该万维网文档。现在最常用的就是http（超文本传送协议HTTP），其次是ftp（文件传送协议FTP）。

(2) 第二部分是<主机>，它指出这个万维网的文档在哪台主机上。<主机>就是指该主机在互连网上的域名，如www.baidu.com。

(3) 后面的<端口>和<路径>有时可以省略。HTTP的默认端口号是80，通常可省略。路径有时也可以省略，表示访问互联网上的某个主页。如访问www.baidu.com和访问www.baidu.com/idnex.html的效果是一样的，都是访问百度的首页。

2.2 URI

统一资源标识符URI（Uniform Recourses Identifier）就是由某个协议方案表示的资源定位标识符。协议方案是指访问资源所使用的协议类型名称。

> 乍一看，URI和URL似乎并没有什么区别。但是URI强调的是给资源标记命名，URL强调的是给资源定位（资源在互联网上的具体位置）。URI只知道某个资源的存在，但是不知道如何找到该资源；而URL不仅知道资源的存在，还知道如何去找到这个资源，即它知道资源的位置。

打个比喻，如果仅仅告诉你一个人的名字叫tom，你仅仅知道有这个叫tom的人，但是你却无法找到tom这个人，这个名字tom就相当于URI。现在如果给了你tom这个人的名片（URL），上面有tom的住址，通过这个名片不仅仅知道tom这个人，还能定位到tom所在的位置。

> 显然，URL所包含的信息比URI更多，URL是URI的子集。

3.HTTP协议

超文本传送协议HTTP（HyperText Transfer Protocol）定义了浏览器（即万维网客户进程）向服务器请求Web页面的方式，以及服务器向浏览器传送页面的方式。

3.1 HTTP的特点

(1) **HTTP使用了面向连接的TCP作为传输层协议**，保证了数据的可靠传输。HTTP客户首先发起一个与服务器的TCP连接，一旦连接建立。该浏览器和服务器进程就可以通过套接字接口访问TCP。

(2) **HTTP协议本身是无连接的。**这就是说，虽然HTTP使用了TCP连接，但通信的双方在交换HTTP报文之间不需要先建立HTTP连接。

(3) **HTTP协议是无状态的。**无状态协议对于事务没有记忆能力，HTTP服务器并不保存关于客户的任何信息。同一个客户在短时间内两次请求同一个对象，服务器会对这两次请求作出两次响应，因为服务器并不记得曾经这个客户访问过，也不记得为这个客户服务器过多少次。

HTTP协议设计成无状态的原因：

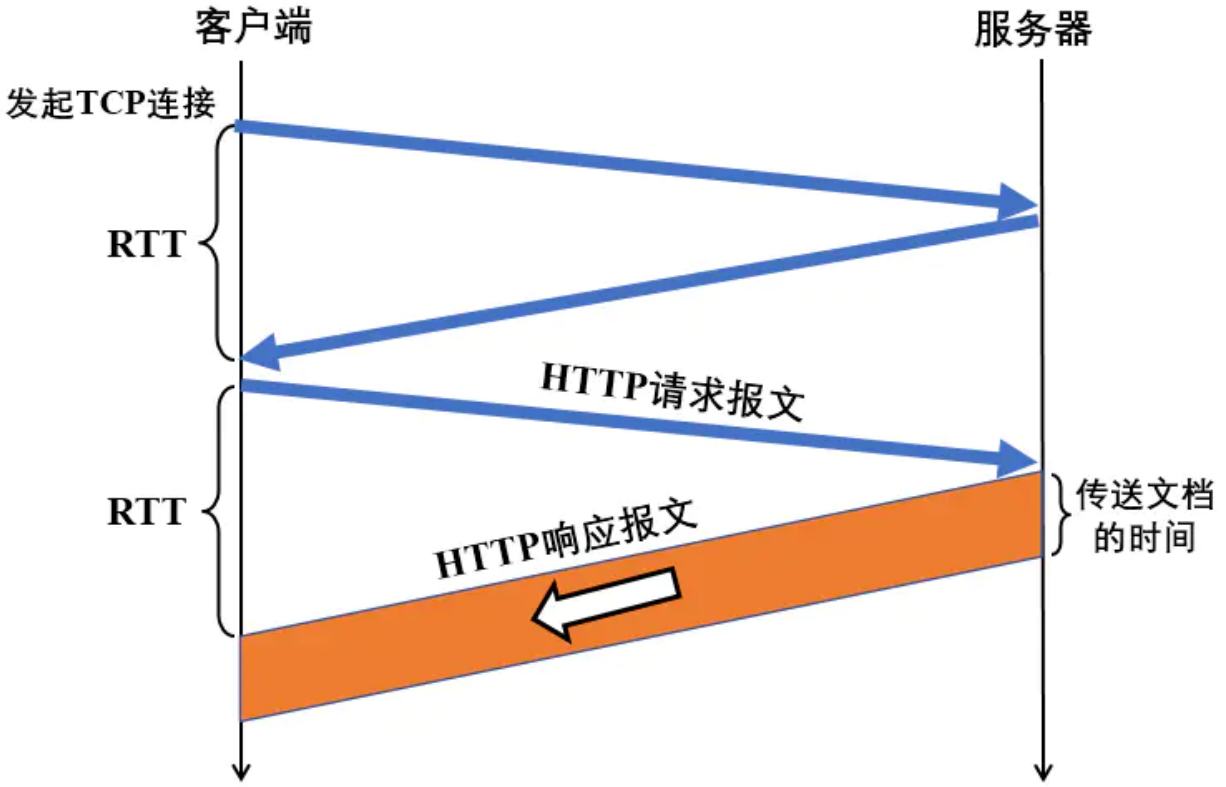
(1) 简化了服务器的设计，无状态协议简单，使服务器更容易支持大量并发的HTTP请求。

(2) 有状态的协议更加复杂，需要维护客户端的状态，并且如果客户或服务器失效，会产生状态不一致，解决这种不一致的代价高。

(3) 不需保存客户的状态信息可以减少服务器的CPU及内存的消耗。

3.2 持续连接和非持续连接

在介绍这两种连接方式之前，先看以下一个请求响应报文需要传输的时间。

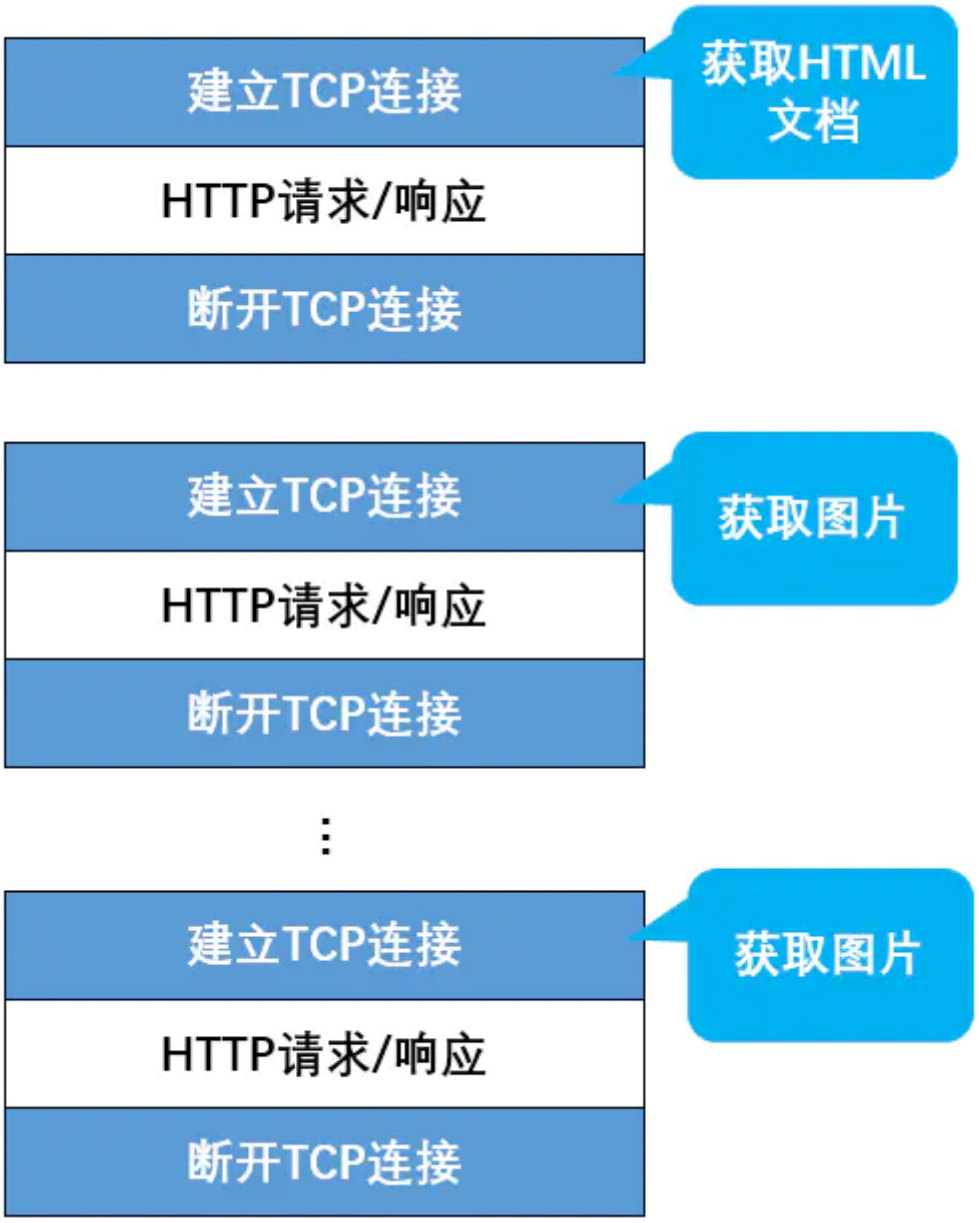


上图表示从浏览器请求一个文档所需的时间，HTTP协议首先和服务器建立TCP连接。这需要三次握手，当建立TCP连接的三次报文握手的前两部分后（即经过了一个RTT时间后），客户将把HTTP请求报文作为建立TCP连接的第三次握手的第三个报文的数据（前面介绍过三次握手中第三次握手客户端可以携带数据），发送给服务器。服务器收到HTTP请求报文后，就把所请求的文档作为响应报文返回给客户。

从上面的过程可以看出，**从服务器请求一个文档的所需的时间就是该文档传输时间加上两倍的往返时间RTT。**

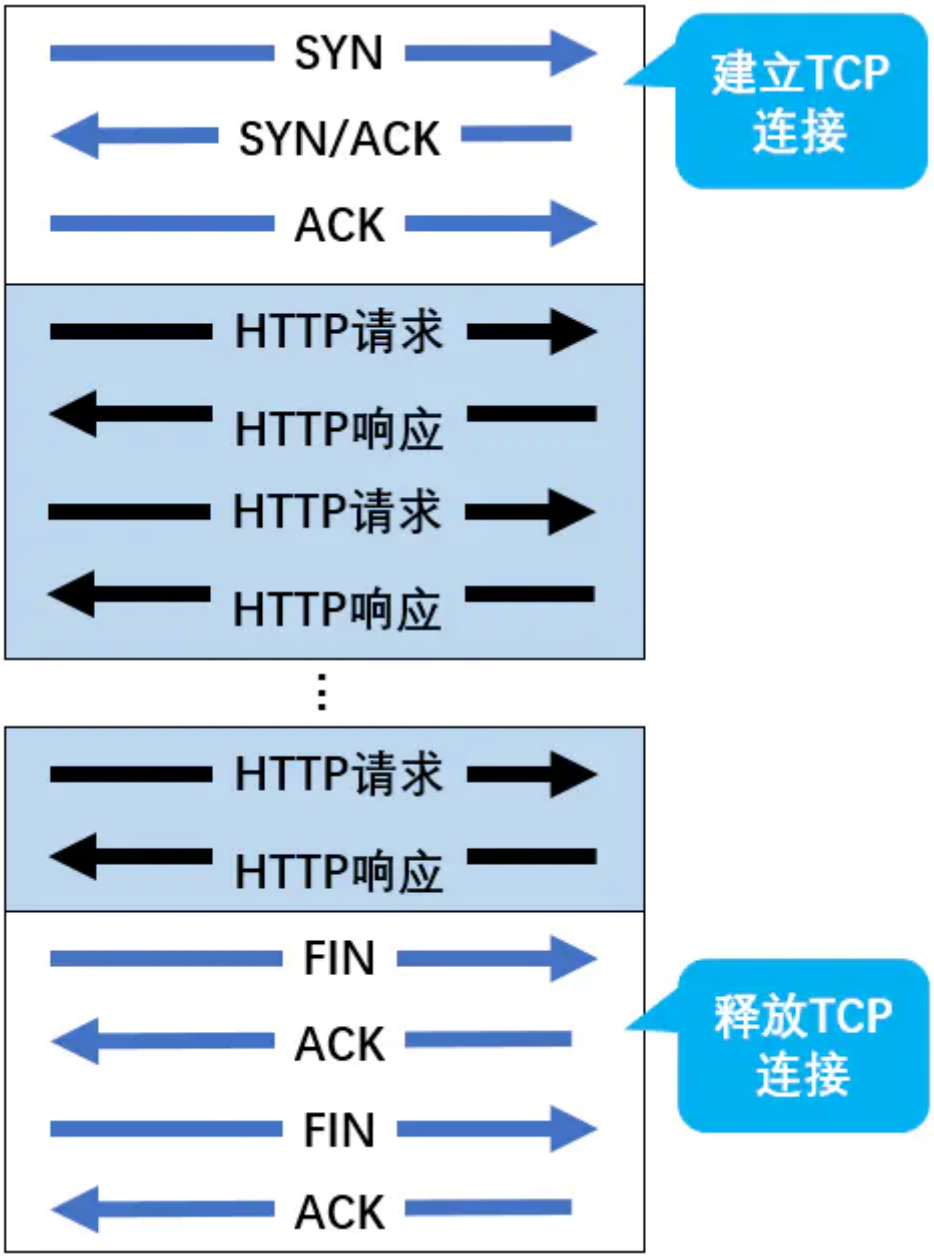
非连续连接是指当一系列请求到达服务器时，每个请求/响应应对是经过一个单独的TCP进行的。如果使用浏览器浏览一个页面，这个页面包含多个链接对象（如图片）那么就需要进行依次链接，每一次链接都导致2 × RTT的开销。另一个开销就是服务器和客户每一次建立新的TCP连接都要分配缓存和变量。特别万维网服务器往往要同时服务器大量客户请求，所以非持续连接会使万维网服务器负担很重。

下图表示非连续连接的工作方式，对于每一个请求都建立一个TCP连接。



为了解决非连续连接的问题，HTTP 1.1使用持续连接（默认），所谓的持续连接是指服务器在发送响应后仍然一段时间内保持这条连接，使同一个客户（浏览器）和该服务器可以继续在这条链接上传送后续的HTTP请求报文和响应报文。

下图表示持续连接下的多次请求和响应的交互。

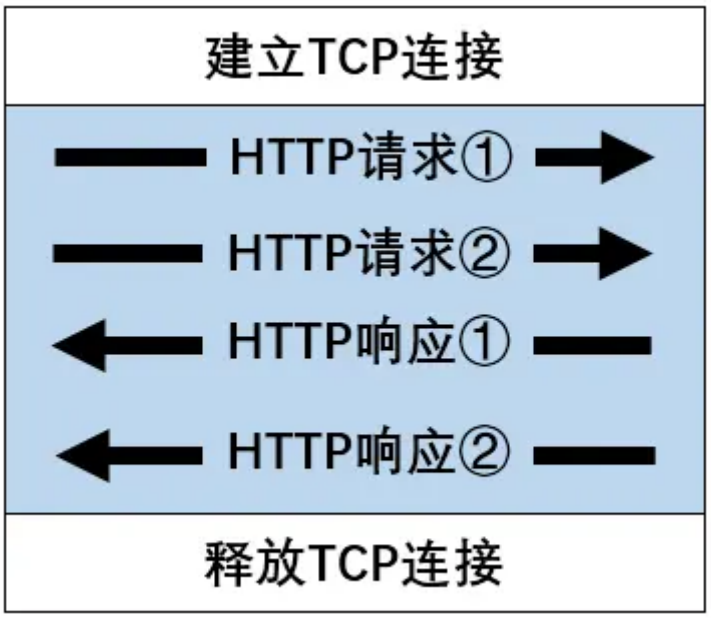


持久连接的好处在于减少了TCP连接的重复建立和断开所造成的额外开销，减轻了服务器的负载。另外，减少开销的那部分时间，使HTTP请求和响应能更早的结束，这样Web页面的显示速度也就提高了。

HTTP/1.1协议的持续连接有两种工作方式，即**非流水线方式**（without pipelining）和**流水线方式**（with pipelining）。

非流水线方式的工作特点：客户在收到前一个响应后才能发出下一个请求。因此，在TCP连接已建立后，客户每访问一次对象都要用去一个往返时间RTT。这比非持续连接要用去两倍RTT的开销，节省了建立TCP所需的一个RTT时间。上图的持续连接方式就是非流水线方式，这种方式的缺点是服务器在发送完一个对象后，其TCP连接就处于空闲状态，浪费了服务器资源。

流水线方式的工作特点：客户在收到HTTP响应报文之前就能够接着发送新的请求报文。于是一个接着一个的请求报文到达服务器后，服务器就可以连续发回响应报文。因此，使用流水线方式时，客户访问所有的对象只需花费一个RTT时间。流水线工作方式使TCP连接中的空闲时间减少，提高了文档下载效率。



3.3 Cookie(了解即可)

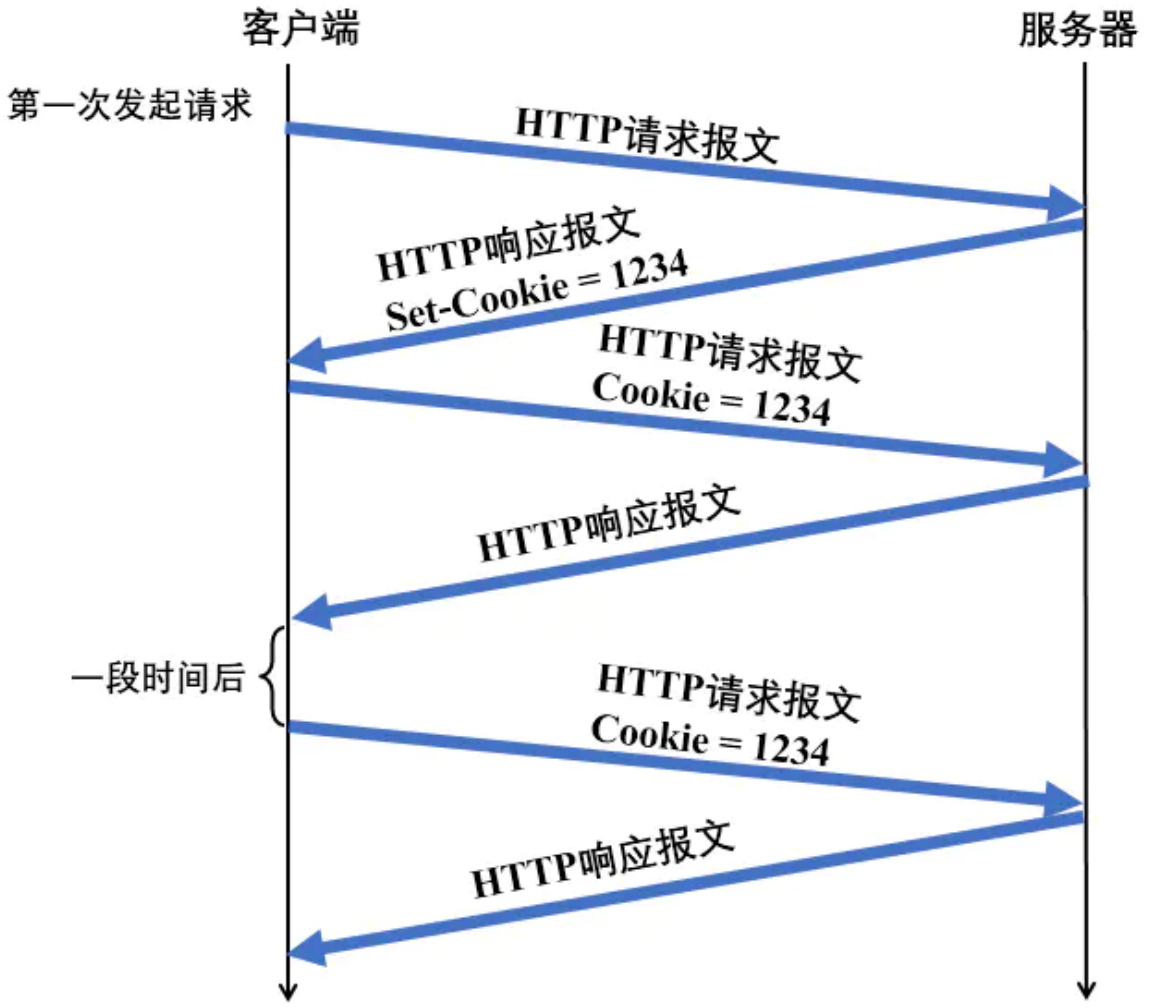
前面介绍，HTTP 是无状态协议，即HTTP 协议自身不对请求和响应之间的通信状态进行保存。但是在狠多情况下需要保存用户状态，例如用户登录到一家购物网站，即使他跳转到该站的其他页面后，也需要能继续保持登录状态。针对这个实例，网站为了能够掌握是谁送出的请求，需要保存用户的状态。

为了解决上面的问题，目前大多站点使用Cookie技术。**Cookie是网站为了辨别用户身份，进行会话跟踪而存储在客户端上的数据。**

Cookie技术的4个组件：

> (1) 在HTTP响应报文中的一个cookie首部行。
> (2) 在HTTP请求报文中的一个cookie首部行。
> (3) 在用户端系统中保留一个cookie文件，由用户的浏览器进行管理。
> (4) 位于Web站点的一个后端数据库。

Cookie技术通过在请求和响应报文中写入Cookie信息来控制客户端的状态。其具体工作流程如下所示



> (1) 当用户访问某个使用Cookie的网站时，该网站的服务器就为用户产生一个唯一的识别码，并以此作为索引在服务器的后端服务器数据库中产生一个表项。

> (2) 服务器在给用户的HTTP响应报文中添加一个叫做Set-Cookie的首部行。如Set-Cookie : 1234

> (3) 当用户收到这个响应时，其浏览器就在它管理的特定的Cookie文件中添加一行，其中包括这个服务器主机名和Set-Cookie后面的给出的识别码。当用户继续浏览这个网站时，每发送一个HTTP请求报文，其浏览器就会从Cookie文件中取出这个网站的识别码，并放到HTTP请求报文的Cookie首部行中，即Coolkie : 1234。

于是，该网站就可以跟踪用户在该网站的活动。服务器并不知道用户的真实姓名以及其他信息，但是它知道用户在什么时候访问了哪些页面，以及访问这些页面的顺序。例如，如果用户在网上购物，那么服务器可以为该用户维护一个所购物品的列表，使的用户在购物结束后一起支付。

> (4) 如果过段时间用户再次访问该网站时，那么浏览器会在其请求报文中继续使用首部行Cookie:1234，如果用户在该网站上保存了姓名、电子邮件等信息，服务器可以使用Cookie来验证该用户，因此以后用户在网站购物时就不必在重新输入姓名等信息了，对用户显然是很方便的。

下图表示第一次访问该网站的响应报文的响应头和请求报文的请求头，只需要关注响应头中Set-Cookie字段。

Response Headers

Cache-Control: private
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Tue, 27 Aug 2019 05:54:51 GMT
Set-Cookie: __jsluid_h=**7da89a257cc0ee8187698a5ddf105528**;
max-age = 31536000; path=/; HttpOnly
....

Request Headers

Accept:text/html,application/xhtml+xml,application/xml;q=0.9,
image/webp,image/apng,*/*;q=0.8,application/signed-
exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: keep-alive
....

下图表示第二次访问某网站的响应报文的响应头和请求报文的请求头，在请求头中可以看出首部字段中使用了Cookie字段。

Response Headers

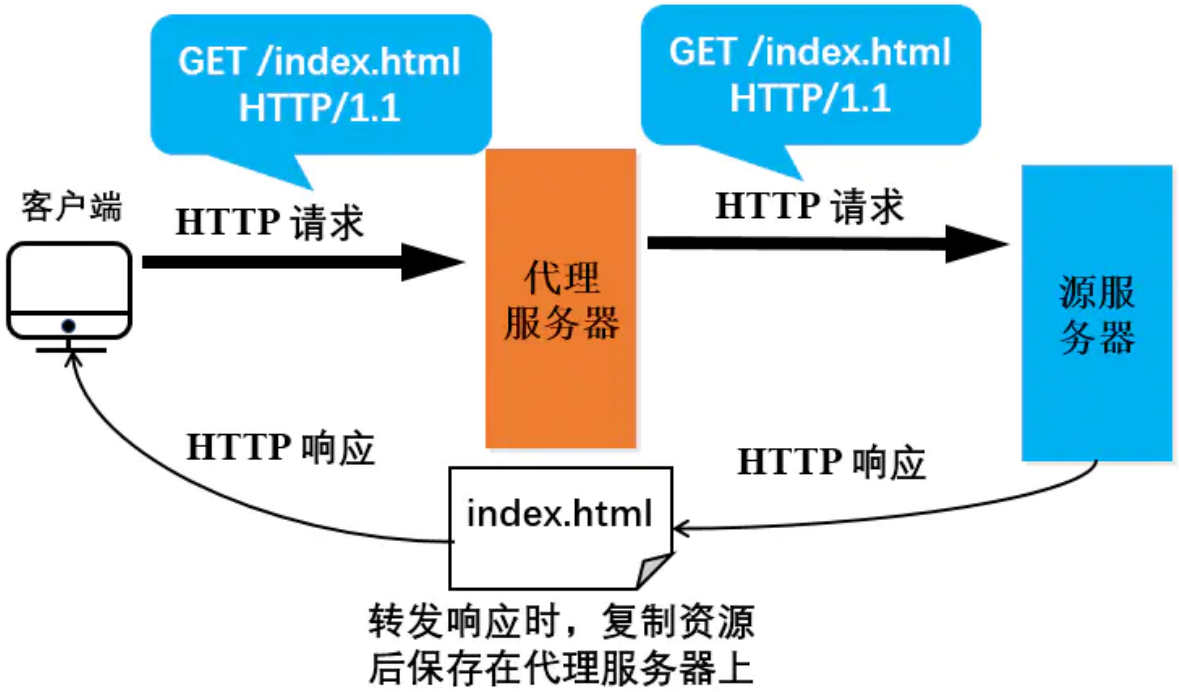
Cache-Control: private
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Tue, 27 Aug 2019 05:57:51 GMT
Transfer-Encoding: chunked
Vary: Accept-Encoding
…

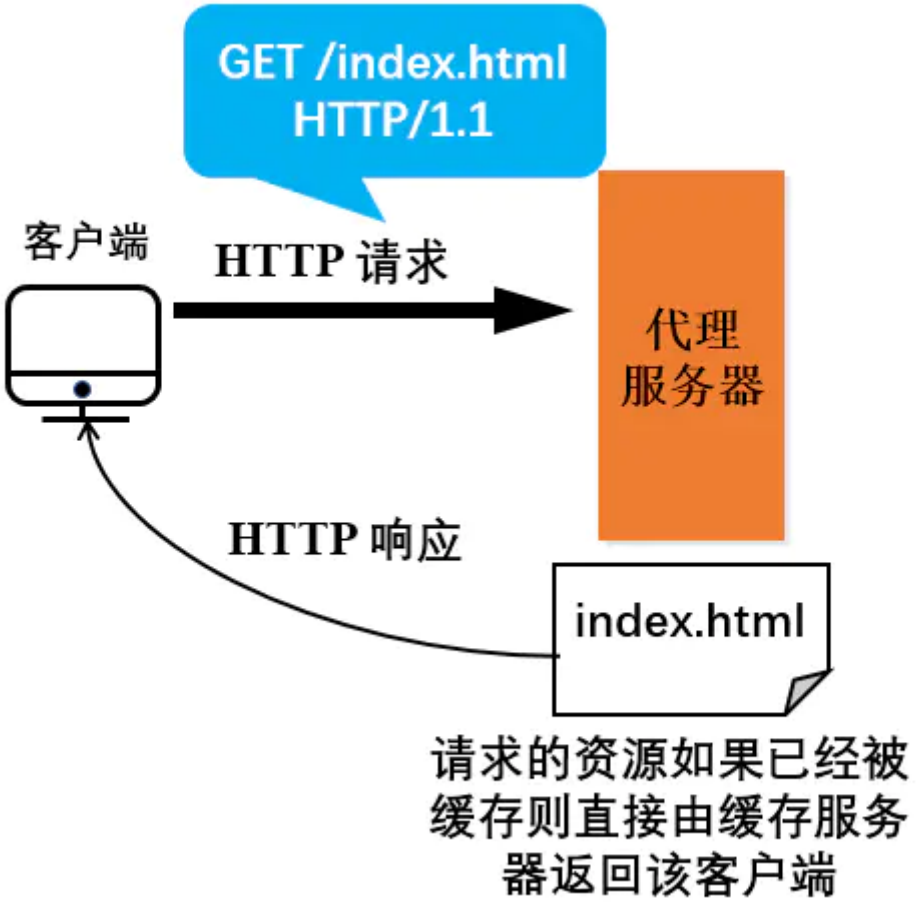
Request Headers

Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cache-Control: max-age=0
Connection: keep-alive
Cookie:__jsluid_h=**7da89a257cc0ee8187698a5ddf105528**
....

4.缓存

Web缓存器又称**代理服务器**（proxy server），它能够将最近的一些请求和响应暂存在本地磁盘中。当新请求到达时，如果代理服务器中没有这个请求，那么就会从互联网中访问该资源，将其返回给代理服务器，代理服务器会保存一份资源的副本，之后将资源返回给客户端。若代理服务器发现这个请求与暂时存放的请求相同，就返回暂存响应，而不需按URL的地址再次去互联网访问该资源。





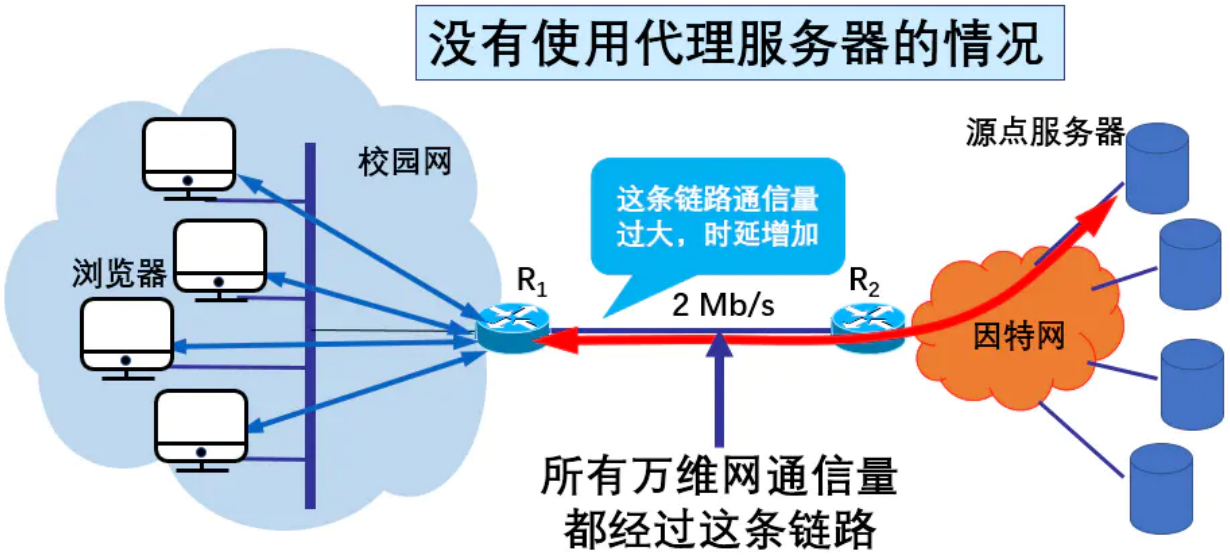
> 上面第一幅图表示代理服务器中没有这个请求的资源，第二幅图表示客户端再次请求相同的资源的情况。

缓存的好处：

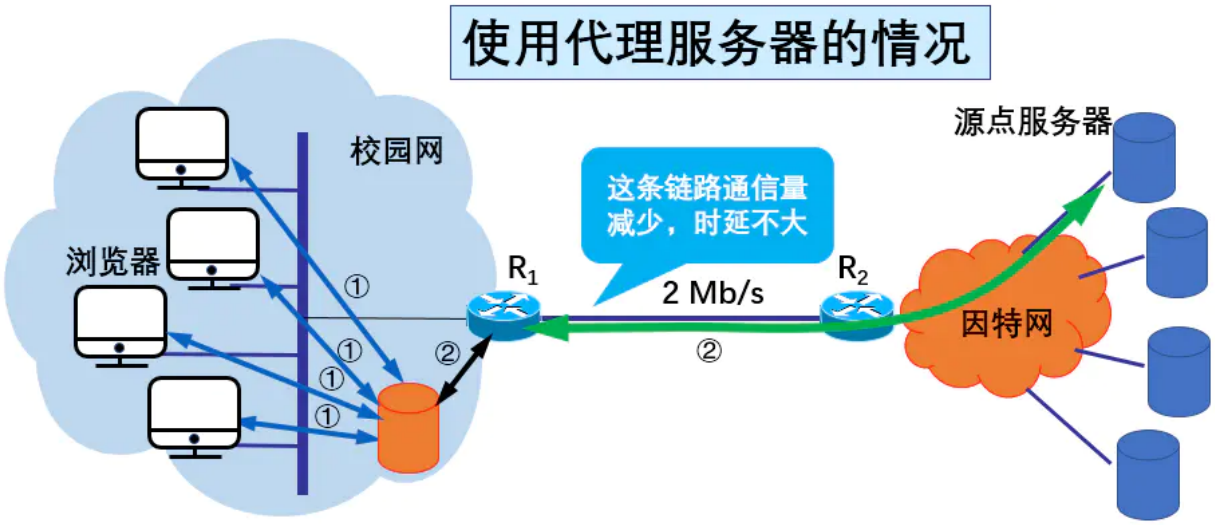
(1) **大大减少对客户请求的响应的时间。**

(2) **可以避免多次从源服务器转发资源，大大减少了一个机构接入链路到因特网的通信量，从而降低了费用。**

下图表示没有使用代理服务器的情况，所有的计算机都通过这条2Mbit/s的链路与互联网上的源点服务器建立TCP连接。因此校园网各计算机访问互联网的通信量往往会使这条链路过载，使得时延大大增加。



> 下图表示使用了代理服务器的情况，校园网中的计算机先和代理服务器建立TCP连接，如果代理服务器存放了请求的资源，则直接将请求的资源返回给计算机的浏览器（图中的①）。如果没有请求的对象，那么代理服务器将与互联网上的源点服务器建立TCP连接（图中的②），并发送HTTP请求报文，源点服务器将请求的资源返回给代理服务器，代理服务器先将资源复制到自己的本地服务器中，然后在将该资源返回给请求的计算机。



所以，在使用代理服务器的情况下，由于有相当大一部分通信局限在校园网内部，因此链路上的通信量大大减少，因而减少了互联网的时延。

5.条件GET方法

尽管缓存能够减少响应时间，但是也引入了一个新的问题——缓存的有效性问题。

> 当遇上服务器上的资源更新时，如果还是使用原来的缓存，那么就会出现缓存不一致的问题。

HTTP协议有一种机制，允许缓存证实它的对象是最新的。**这种机制就是条件GET方法。**

如果请求报文使用的是GET方法，并且请求报文中含一个“If-Modified-Since”首部行。那么，这个HTTP请求就是一个条件GET请求报文。下面用一个例子说明一下条件GET方法。

首先，一个代理服务器代表一个请求浏览器，向某个Web服务器发送一个请求报文：

```
GET  /fruit.kiwi.gif  HTTP/1.1
Host:www.exotiquecuisine.com
```

其次，该Web服务器向代理服务器发送具有被请求对象的响应报文：

```
HTTP/1.1  200  OK
Date: Sat, 8, Oct, 2011 15:35:12
Server: Apache/1.3.0
Last-Modified: Wed, 7 Sep 2011 09:23:24
Content-Type: image/gif

{data,data,data……..}
```

代理服务器将该对象保存在本地磁盘中。注意，响应报文中的红色部分，表示该对象的最后修改日期。

一段时间后，另一个用户经过缓存请求同一个对象，该对象仍在代理服务器中，由于过了一段时间了，Web服务器上的该对象可能已经被修改过了，所以代理服务器通过发送一个条件GET执行最新检查。具体来说，该代理服务器发送：

```
GET  /fruit.kiwi.gif  HTTP/1.1
Host:www.exotiquecuisine.com
If-Modified: Wed, 7 Sep 2011 09:23:24
```

这里，请求报文中的If-Modified首部行的值和之前响应报文中Last-Modified首部行的值一样。这个条件GET报文告诉服务器，如果在指定日期后该对象被修改过，才发送该对象。通俗来说，就是代理服务器告诉服务器如果在2011年9月7日 09:23:24之后对该对象有修改过，那就把修改过的对象发送给我。

假设该对象没有被修改过，那么Web服务器向代理服务器发送一个响应报文：

```
HTTP/1.1 304 Not Modified
Date: Sat, 15, Oct, 2011 15:35:12
Server: Apache/1.3.0

{empty entity body}
```

304 响应码表示请求的对象没有被修改。所以，响应体的内容是空没有包含对象，由于对象没有被修改，即代理服务器中的对象和Web服务器中的该对象是一致的，所有Web服务器没必要再传送一次该对象，包含对象只会浪费带宽，并增加用户的响应时间。

6.总结