

## 29 | HTTP/1: HTTP性能优化

2019-10-10 李兵

《浏览器工作原理与实践》

课程介绍 >



讲述：李兵

时长 13:44 大小 12.58M



谈及浏览器中的网络，就避不开 HTTP。我们知道 HTTP 是浏览器中**最重要且使用最多**的协议，是**浏览器和服务**器之间的**通信语言**，也是互联网的基石。而随着浏览器的发展，HTTP 为了能适应新的形式也在持续进化，我认为学习 HTTP 的最佳途径就是了解其发展史，所以在接下来的三篇文章中，我会从浏览器发展的视角来和你聊聊 HTTP 演进。这三篇分别是**即将完成使命的 HTTP/1**、正在向我们走来的 HTTP/2，以及未来的 HTTP/3。

本文主要介绍的是 HTTP/1.1，我们先讲解 HTTP/1.1 的进化史，然后再介绍在进化过程中所遇到的各种瓶颈，以及对应的解决方法。

### 超文本传输协议 HTTP/0.9

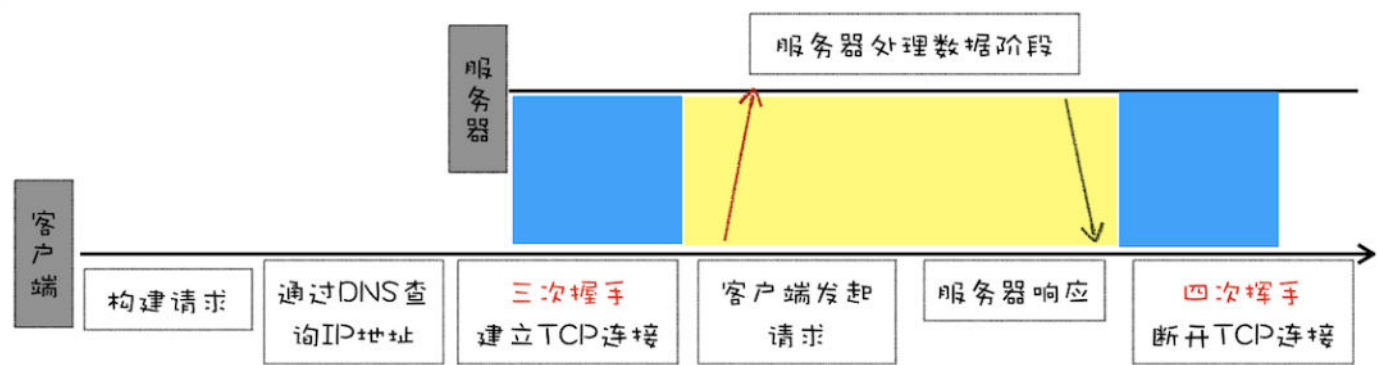
首先我们来看看诞生最早的 HTTP/0.9。HTTP/0.9 是于 1991 年提出的，主要用于学术交流，需求很简单——用来在网络之间传递 HTML 超文本的内容，所以被称为**超文本传输协**



议。整体来看，它的实现也很简单，采用了基于请求响应的模式，从客户端发出请求，服务器返回数据。

下面我们就来看看 HTTP/0.9 的一个完整的请求流程（可参考下图）。

- 因为 HTTP 都是基于 TCP 协议的，所以客户端先要根据 IP 地址、端口和服务器建立 TCP 连接，而建立连接的过程就是 TCP 协议三次握手的过程。
- 建立好连接之后，会发送一个 GET 请求行的信息，如GET /index.html用来获取 index.html。
- 服务器接收请求信息之后，读取对应的 HTML 文件，并将数据以 ASCII 字符流返回给客户端。
- HTML 文档传输完成后，断开连接。



HTTP/0.9 请求流程

总的来说，当时的需求很简单，就是用来传输体积很小的 HTML 文件，所以 HTTP/0.9 的实现有以下三个特点。

- 第一个是只有一个请求行，并没有 **HTTP 请求头和请求体**，因为只需要一个请求行就可以完整表达客户端的需求了。
- 第二个是服务器也没有返回头信息，这是因为服务器端并不需要告诉客户端太多信息，只需要返回数据就可以了。
- 第三个是返回的文件内容是以 ASCII 字符流来传输的，因为都是 HTML 格式的文件，所以使用 ASCII 字节码来传输是最合适的。

## 被浏览器推动的 HTTP/1.0

HTTP/0.9 虽然简单，但是已经可以满足当时的需求了。不过变化是这个世界永恒不变的主旋律，1994 年底出现了拨号上网服务，同年网景又推出一款浏览器，从此万维网就不局限于学术交流了，而是进入了高速的发展阶段。随之而来的是万维网联盟（W3C）和 HTTP 工作组（HTTP-WG）的创建，它们致力于 HTML 的发展和 HTTP 的改进。

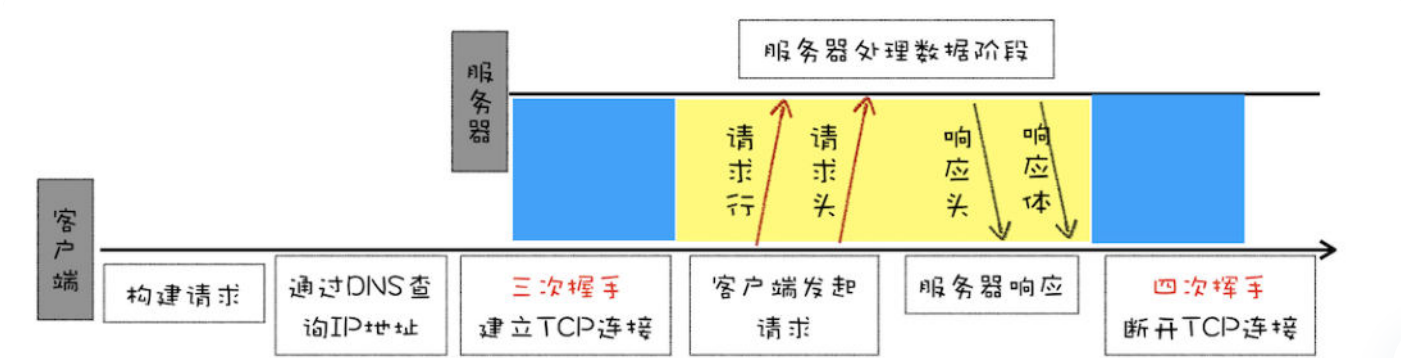
万维网的高速发展带来了很多新的需求，而 HTTP/0.9 已经不能适用新兴网络的发展，所以这时就需要一个新的协议来支撑新兴网络，这就是 HTTP/1.0 诞生的原因。不过在详细分析 HTTP/1.0 之前，我们先来分析下新兴网络都带来了哪些新需求。

首先在浏览器中展示的不单是 HTML 文件了，还包括了 JavaScript、CSS、图片、音频、视频等不同类型的文件。因此**支持多种类型的文件下载是 HTTP/1.0 的一个核心诉求**，而且文件格式不仅仅局限于 ASCII 编码，还有很多其他类型编码的文件。

那么该如何实现多种类型文件的下载呢？

文章开头我们说过，HTTP 是浏览器和服务器之间的通信语言，不过 HTTP/0.9 在建立好连接之后，只会发送类似GET /index.html的简单请求命令，并没有其他途径告诉服务器更多的信息，如文件编码、文件类型等。同样，服务器是直接返回数据给浏览器的，也没有其他途径告诉浏览器更多的关于服务器返回的文件信息。

这种简单的交流型形式无疑不能满足传输多种类型文件的需求，那为了让客户端和服务端能更深入地交流，HTTP/1.0 引入了请求头和响应头，它们都是以 Key-Value 形式保存的，在 HTTP 发送请求时，会带上请求头信息，服务器返回数据时，会先返回响应头信息。至于 HTTP/1.0 具体的请求流程，你可以参考下图。



HTTP/1.0 的请求流程


有了请求头和响应头，浏览器和服务器就能进行更加深入的交流了。

## 那 HTTP/1.0 是怎么通过请求头和响应头来支持多种不同类型的数据呢？

要支持多种类型的文件，我们就需要解决以下几个问题。

- 首先，浏览器需要知道服务器返回的数据是什么类型的，然后浏览器才能根据不同的数据类型做针对性的处理。
- 其次，由于万维网所支持的应用变得越来越广，所以单个文件的数据量也变得越来越大。为了减轻传输性能，服务器会对数据进行压缩后再传输，所以浏览器需要知道服务器压缩的方法。
- 再次，由于万维网是支持全球范围的，所以需要支持国际化的支持，服务器需要对不同的地区提供不同的语言版本，这就需要浏览器告诉服务器它想要什么语言版本的页面。
- 最后，由于增加了各种不同类型的文件，而每种文件的编码形式又可能不一样，为了能够准确地读取文件，浏览器需要知道文件的编码类型。

基于以上问题，HTTP/1.0 的方案是通过请求头和响应头来进行协商，在发起请求时候会通过 HTTP 请求头告诉服务器它期待服务器返回什么类型的文件、采取什么形式的压缩、提供什么语言的文件以及文件的具体编码。最终发送出来的请求头内容如下：

 复制代码

```
1 accept: text/html
2 accept-encoding: gzip, deflate, br
3 accept-Charset: ISO-8859-1,utf-8
4 accept-language: zh-CN,zh
```

其中第一行表示期望服务器返回 html 类型的文件，第二行表示期望服务器可以采用 gzip、deflate 或者 br 其中的一种压缩方式，第三行表示期望返回的文件编码是 UTF-8 或者 ISO-8859-1，第四行是表示期望页面的优先语言是中文。

服务器接收到浏览器发送过来的请求头信息之后，会根据请求头的信息来准备响应数据。不过有时候会有一些意外情况发生，比如浏览器请求的压缩类型是 gzip，但是服务器不支持 gzip，只支持 br 压缩，那么它会通过响应头中的 content-encoding 字段告诉浏览器最终的





压缩类型，也就是说最终浏览器需要根据响应头的信息来处理数据。下面是一段响应头的数据信息：

 复制代码

```
1 content-encoding: br
2 content-type: text/html; charset=UTF-8
```

其中第一行表示服务器采用了 br 的压缩方法，第二行表示服务器返回的是 html 文件，并且该文件的编码类型是 UTF-8。

有了响应头的信息，浏览器就会使用 br 方法来解压文件，再按照 UTF-8 的编码格式来处理原始文件，最后按照 HTML 的方式来解析该文件。这就是 HTTP/1.0 支持多文件的一个基本的处理流程。

HTTP/1.0 除了对多文件提供良好的支持外，还依据当时实际的需求引入了很多其他的特性，这些特性都是通过请求头和响应头来实现的。下面我们来看看新增的几个典型的特性：

- 有的请求服务器可能无法处理，或者处理出错，这时候就需要告诉浏览器服务器最终处理该请求的情况，这就引入了**状态码**。状态码是通过响应行的方式来通知浏览器的。
- 为了减轻服务器的压力，在 HTTP/1.0 中提供了 **Cache 机制**，用来缓存已经下载过的数据。
- 服务器需要统计客户端的基础信息，比如 Windows 和 macOS 的用户数量分别是多少，所以 HTTP/1.0 的请求头中还加入了**用户代理**的字段。

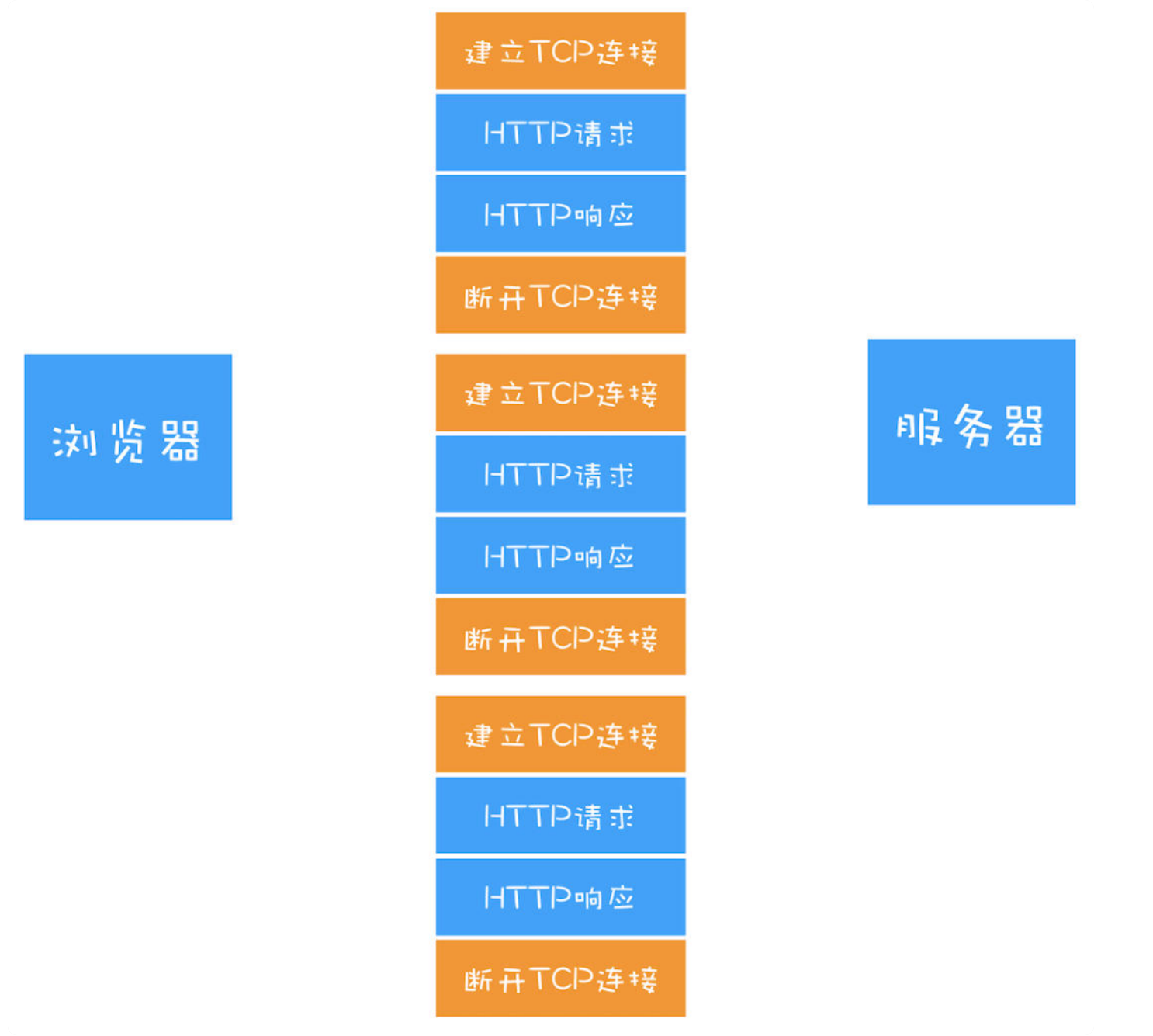
## 缝缝补补的 HTTP/1.1

不过随着技术的继续发展，需求也在不断迭代更新，很快 HTTP/1.0 也不能满足需求了，所以 HTTP/1.1 又在 HTTP/1.0 的基础之上做了大量的更新。接下来我们来看看 HTTP/1.0 遇到了哪些主要的问题，以及 HTTP/1.1 又是如何改进的。

### 1. 改进持久连接

HTTP/1.0 每进行一次 HTTP 通信，都需要经历建立 TCP 连接、传输 HTTP 数据和断开 TCP 连接三个阶段（如下图）。



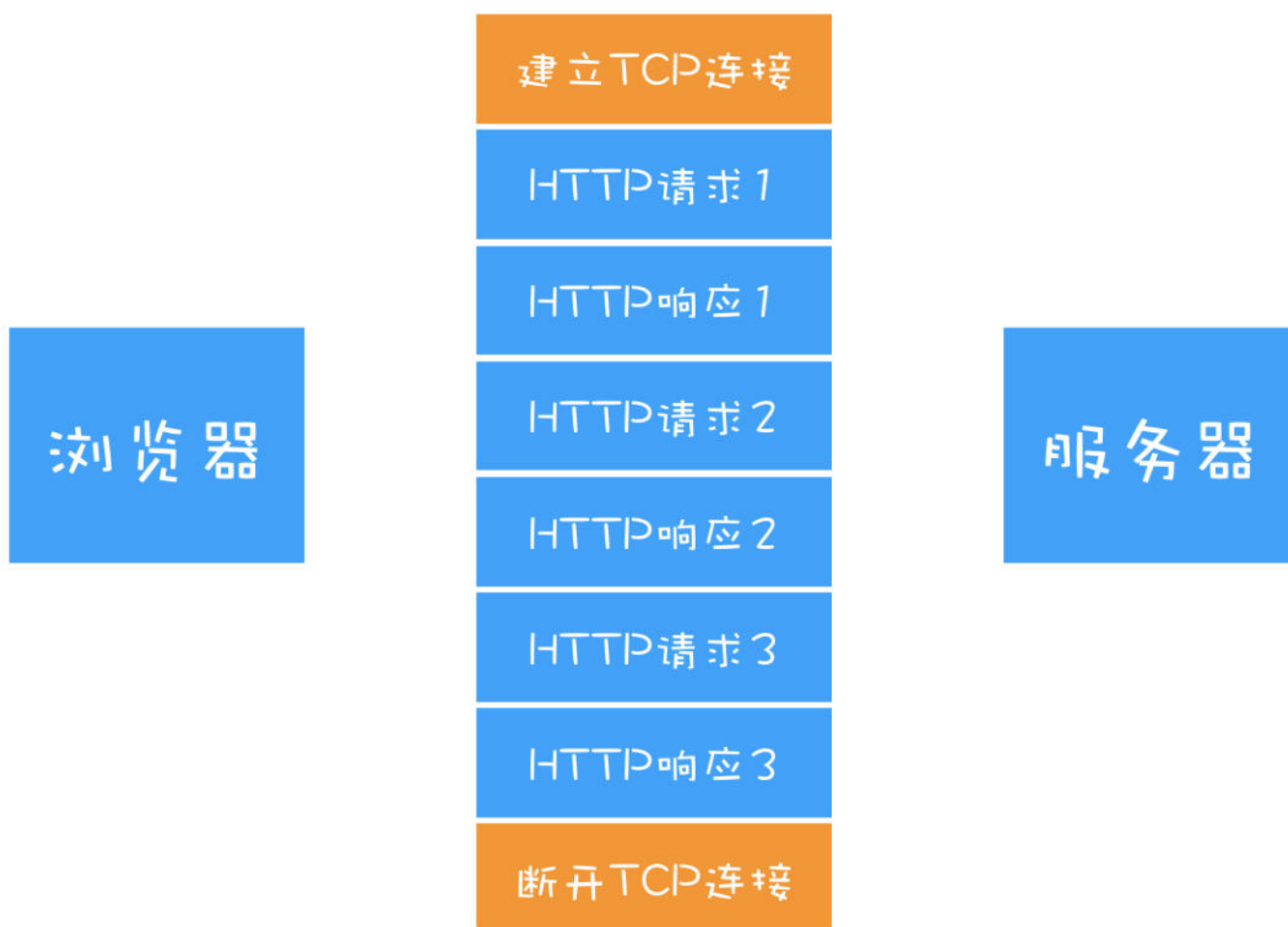


HTTP/1.0 的短连接

在当时，由于通信的文件比较小，而且每个页面的引用也不多，所以这种传输形式没什么大问题。但是随着浏览器普及，单个页面中的图片文件越来越多，有时候一个页面可能包含了几百个外部引用的资源文件，如果在下载每个文件的时候，都需要经历建立 TCP 连接、传输数据和断开连接这样的步骤，无疑会增加大量无谓的开销。

为了解决这个问题，HTTP/1.1 中增加了持久连接的方法，它的特点是在一个 TCP 连接上可以传输多个 HTTP 请求，只要浏览器或者服务器没有明确断开连接，那么该 TCP 连接会一直保持。





HTTP/1.0 的持久连接

从上图可以看出，HTTP 的持久连接可以有效减少 TCP 建立连接和断开连接的次数，这样的好处是减少了服务器额外的负担，并提升整体 HTTP 的请求时长。

持久连接在 HTTP/1.1 中是默认开启的，所以你不需要专门为了持久连接去 HTTP 请求头设置信息，如果你不想要采用持久连接，可以在 HTTP 请求头中加上 `Connection: close`。目前浏览器中对于同一个域名，默认允许同时建立 6 个 TCP 持久连接。

## 2. 不成熟的 HTTP 管线化

持久连接虽然能减少 TCP 的建立和断开次数，但是它需要等待前面的请求返回之后，才能进行下一次请求。如果 TCP 通道中的某个请求因为某些原因没有及时返回，那么就会阻塞后面的所有请求，这就是著名的**队头阻塞**的问题。

HTTP/1.1 中试图通过管线化的技术来解决**队头阻塞**的问题。HTTP/1.1 中的管线化是指将多个 HTTP 请求整批提交给服务器的技术，虽然可以整批发送请求，不过服务器依然需要根据请求顺序来回复浏览器的请求。



Firefox、Chrome 都做过管线化的试验，但是由于各种原因，它们最终都放弃了管线化技术。

### 3. 提供虚拟主机的支持

在 HTTP/1.0 中，每个域名绑定了一个唯一的 IP 地址，因此一个服务器只能支持一个域名。但是随着虚拟主机技术的发展，需要实现在一台物理主机上绑定多个虚拟主机，每个虚拟主机都有自己的单独的域名，这些单独的域名都公用同一个 IP 地址。

因此，HTTP/1.1 的请求头中增加了 **Host 字段**，用来表示当前的域名地址，这样服务器就可以根据不同的 Host 值做不同的处理。

### 4. 对动态生成的内容提供了完美支持

在设计 HTTP/1.0 时，需要在响应头中设置完整的数据大小，如 `Content-Length: 901`，这样浏览器就可以根据设置的数据大小来接收数据。不过随着服务器端的技术发展，很多页面的内容都是动态生成的，因此在传输数据之前并不知道最终的数据大小，这就导致了浏览器不知道何时会接收完所有的文件数据。

HTTP/1.1 通过引入 **Chunk transfer 机制**来解决这个问题，服务器会将数据分割成若干个任意大小的数据块，每个数据块发送时会附上上个数据块的长度，最后使用一个零长度的块作为发送数据完成的标志。这样就提供了对动态内容的支持。

### 5. 客户端 Cookie、安全机制

除此之外，HTTP/1.1 还引入了客户端 Cookie 机制和安全机制。其中，Cookie 机制我们在 [《03 | HTTP 请求流程：为什么很多站点第二次打开速度会很快？》](#) 这篇文章中介绍过了，而安全机制我们会在后面的安全模块中再做介绍，这里就不赘述了。

## 总结

好了，今天就介绍到这里，下面我来总结下本文的主要内容。

本文我们重点强调了 HTTP 是浏览器和服务器的通信语言，然后我们从需求演变的角度追溯了 HTTP 的发展史，在诞生之初的 HTTP/0.9 因为需求简单，所以和服务器之间的通信过程也相对简单。





由于万维网的快速崛起，带来了大量新的需求，其中最核心的一个就是需要支持多种类型的文件下载，为此 HTTP/1.0 中引入了请求头和响应头。在支持多种类型文件下载的基础之上，HTTP/1.0 还提供了 Cache 机制、用户代理、状态码等一些基础信息。

但随着技术和需求的发展，人们对文件传输的速度要求越来越高，故又基于 HTTP/1.0 推出了 HTTP/1.1，增加了持久连接方法来提升连接效率，同时还尝试使用管线化技术提升效率（不过由于各种原因，管线化技术最终被各大厂商放弃了）。除此之外，HTTP/1.1 还引入了 Cookie、虚拟主机的支持、对动态内容的支持等特性。

虽然 HTTP/1.1 在 HTTP/1.0 的基础之上做了大量的优化，但是由于一些效率问题始终很难解决，所以最终还是被 HTTP/2 所取代，这就是我们下一篇文章要介绍的内容了。

## 思考时间

今天留给你的思考题：你认为 HTTP/1.1 还有哪些不足？

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

分享给需要的人，Ta订阅超级会员，你将得 50 元

Ta单独购买本课程，你将得 20 元

 生成海报并分享

 赞 15  提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 28 | WebComponent：像搭积木一样构建Web应用

下一篇 30 | HTTP/2：如何提升网络速度？



# JVM + NIO + Spring

各大厂面试题及知识点详解

限时免费



## 精选留言 (30)

写留言



Geek\_07b5b0

2019-10-10

感谢李兵老师，我今天实习面试用到了很多从这个课程中学到的东西，人生第一次面试，给了我很大鼓励！

共 5 条评论 >

64



mfist

2019-10-11

1对头阻塞问题，没有解决  
2文本传输效率问题，而且不安全  
3header中每次都传输类似头，增加了传输成本

共 1 条评论 >

33



一步

2019-10-22

对于 队头阻塞问题，只要传输层是TCP，就不会得到根本上的解决，http/2 利用流的的机制很大程度的缓解了这个问题，http/3 传输层换成了 UDP 才彻底解决这个问题



27



大蓝

2020-03-16

学的时候不曾感觉，学完一个月，会发现这个课程带给我一个知识树这样子的知识结构，感谢！

我现在又回来继续看看知识哪里遗漏的



13



君自兰芳

2020-11-12

“目前浏览器中对于同一个域名，默认允许同时建立 6 个 TCP 持久连接”

我又有疑问了☹☹！，每个TCP发送请求的数量有上限吗？上限是多少？

什么时候需要开启一个新的TCP连接？什么情况下可以关闭一个TCP连接？

共 4 条评论 >

4



lee

2019-11-21

“目前浏览器中对于同一个域名，默认允许同时建立 6 个 TCP 持久连接”，那就是如果浏览器同时请求n个不同域名的后台服务器，那就是允许同时建立 $n * 6$ 个TCP持久连接吗？

作者回复: 是的

共 3 条评论 >

3



JAY

2019-10-11

上传文件给服务器的HTTP请求和其它请求有什么不一样？如何获取当前上传文件的文件名？谢谢！

共 4 条评论 >

3



太白北路

2020-08-25

安全问题，传输内容是不加密的



2



4!!

2020-02-05

使用哪个版本的http协议是由谁决定的？浏览器和服务端自动设置的？还是由服务端设置的？

共 2 条评论 >

2



月翎魂雨





2019-11-12

文中“目前浏览器中对于同一个域名，默认允许同时建立 6 个 TCP 持久连接”，看老师画的图是一个tcp持久链接有6个请求的意思吧。而且前面第03章文中“同一个域名同时最多只能建立 6 个 TCP 连接，如果在同一个域名下同时有 10 个请求发生，那么其中 4 个请求会进入排队等待状态”。所以我就很迷糊了，这个请求数和tcp链接关系到底是啥？

作者回复: http/1.1中的一个tcp链接同时只能发起一个http请求！

浏览器会让每个域名同时最多建立6个tcp链接，也就是说同一个域名同时能支持6个http请求！

共 3 条评论 >



3



大可

2021-04-28

作为一名后端 觉得老师把浏览器讲的很明白



1



小炭

2020-10-23

为什么HTML 格式的文件，使用 ASCII 字节码来传输是最合适的？

共 2 条评论 >



2



Lorin

2020-05-28

同一个TCP连接中最多支持多少个HTTP请求呢？

共 1 条评论 >



1



blueBean

2020-03-06

Chunk transfer 机制，应该是每个数据块发送时会附上下个数据块的长度吧？

共 4 条评论 >



1



--|||

2019-12-16

同一个域名默认最多6个tcp链接，每个链接每次只能维护一个http链接，这样数据的交换就又局限；明文传输容易被攻击。不能像app那样接收推送信息。



1



铭



2019-10-13

http1.1 虚拟主机支持那部分，host 可以继续展开来啊。“虚拟主机使得多个域名可以映射到同一IP”，跟“针对不同域名IP做不同处理”，感觉没get到内涵啊

共 1 条评论 >

👍 1



tokey

2019-10-10

- 1、对头阻塞
- 2、容易遭受dos攻击
- 3、对小文件传输效率低



👍 1



dxp

2021-07-14

只能是客户端发起，一来一往。



SXL

2021-03-15

1如果当前页面手动刷新，TCP持久连接会不会断开？

共 1 条评论 >



啊哈哈

2020-12-21

1. TCP 数量限制
2. 队头阻塞

