40 从 HTTP 到 HTTP2.0

更新时间：2020-06-08 09:48:16

> 不想当将军的士兵，不是好士兵。——拿破仑

作为前端，我们在面试中，遇到 HTTP 问题的频率比遇到 TCP/UDP 问题的频率会更高。不过网络层面的知识，逐个盘问者常有，出奇制胜者罕见。我们的应对思路仍然是，解决关键问题、拓展必要能力。

在 HTTP 这个考点上，面向应试，我们需要着重掌握三个方面的考点：

HTTP 请求方法；

HTTP 状态码；

对**无状态**的理解。

这三个方面的知识，不夸张地说，就是背也要背下来。

除此之外，各位也需要明确一个不可忽略的事实：HTTP 协议从未停下它进化的脚步，其版本迭代层出不穷、持续至今。HTTP 面试的考点，也从最初的常识考察、延伸到了对 HTTP 各个版本进行特性考察这个维度。因此，熟悉 HTTP 的发展史、熟悉每个版本解决了哪些重要的问题，也是我们本节的一个重点。

HTTP 通用考点梳理

HTTP状态码速记

1xx：成功接收了请求，但是处理过程还没结束，需要客户端再抛出一个请求才能完成整个过程。这个比较少见，大家了解即可。

2xx：表示成功接收请求、并且已经处理完毕。其中需要重点关注下面这个码：

- 200 OK，标识客户端的请求已经被服务器正确处理

3xx：表示服务器虽然也处理了你的请求，但客户端还需要进一步的工作，才可以完成请求。其中需要大家重点关注的是：

301：永久性重定向，表示资源已被分配了新的 URL

302：临时性重定向，表示资源临时被分配了新的 URL

304，表示服务器校验后发现资源没有改变，提醒客户端直接走缓存来取资源

4xx：客户端错误，意味着请求出错了。需要关注的重点错误码是：

400：请求报文存在语法错误

403：对请求资源的访问被服务器拒绝（多半是没权限）

404，资源不存在，可能是你的路径不对，也可能是这个资源在服务端已经被下掉了

5xx：服务器错误，意味着服务器内部的程序处理有问题。需要关注的重点错误码是：

500：服务器在接受请求后进行处理的过程中，发生了内部错误

502：网关错误

504：网关超时

HTTP 请求方法速记

HTTP 协议常考的请求方法有 GET、POST、HEAD、PUT、DELETE、OPTIONS。

- **GET 方法**：仅用作数据的读取，请求参数以query的形式附加
- **POST 方法**：创建新资源或修改现有资源，请求参数以body的形式传递
- **HEAD 方法**：只请求页面的首部、不请求页面内容。它允许我们单纯获取服务器的响应头信息。
- **PUT 方法**：PUT 在能力上和 POST 类似，区别在于 PUT 的 URI 指向是具体的某个资源，而不能指向资源集合。同时 PUT 对资源的修改是幂等的。
- **DELETE方法**：用于删除指定的资源
- **OPTIONS方法**：用于获取指定服务能够支持的通信选项。

高频考点——对“无状态”的理解

HTTP 协议是无状态的，这个“无状态”到底是指什么呢？

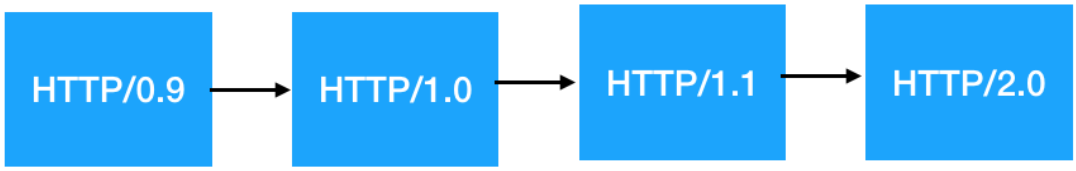
与其说是“无状态”，不如说是“无记忆”。这个 HTTP 协议呀，心非常大，请求与请求之间，是不关心对方的情况的。也就是说你上一秒出去一个 A 请求，下一秒出去一个 B 请求，那么 B 是完全感知不到 A 请求曾经存在过的，更别提了解 A 请求的内容了。总之，两个请求间**毫无瓜葛**。

那么如果我们想维持状态信息，该怎么办呢？两个思路：cookie 和 session。

- **cookie**：cookie 是存储在浏览器的小段文本，会在浏览器每次向同一服务器再发起请求时被携带并发送到服务器上。我们可以把状态信息放在cookie里，带给服务器。
- **session**：session 是存储在服务器的用户数据。浏览器第一次向服务器发起请求时，服务器会为当前会话创建一个session，并且把对应的 session-id 写入 cookie 中，用来标识 session。此后，每次用户的请求都会携带一个包含了 session-id 的 cookie，服务器解析出了 session-id，便能定位到用户的用户信息。

HTTP 的发展史

截止到目前，其实 HTTP3.0 的曙光也已经在眼前。但是面向面试，我们关心的仍然是截止到 HTTP2.0 的版本。在 HTTP 的历史上，截止到 2.0 的版本迭代过程如图所示：

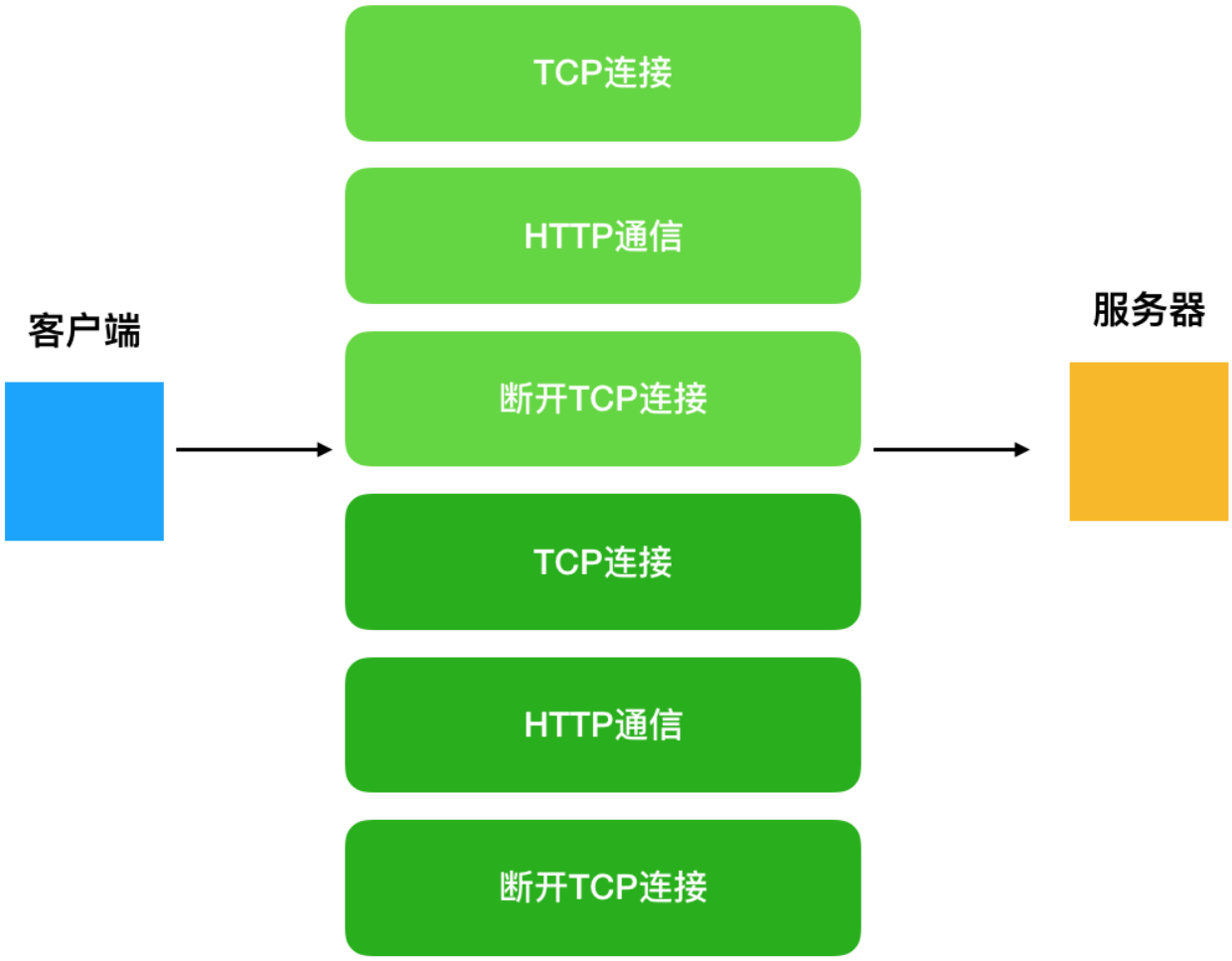


其中，0.9 版本因为年代太过久远（诞生于上世纪80年代末）往往不被关心。关于 HTTP 发展史层面的问题，面试官一般会聚焦于**从 1.1 版本开始，每个版本都解决了什么问题**。

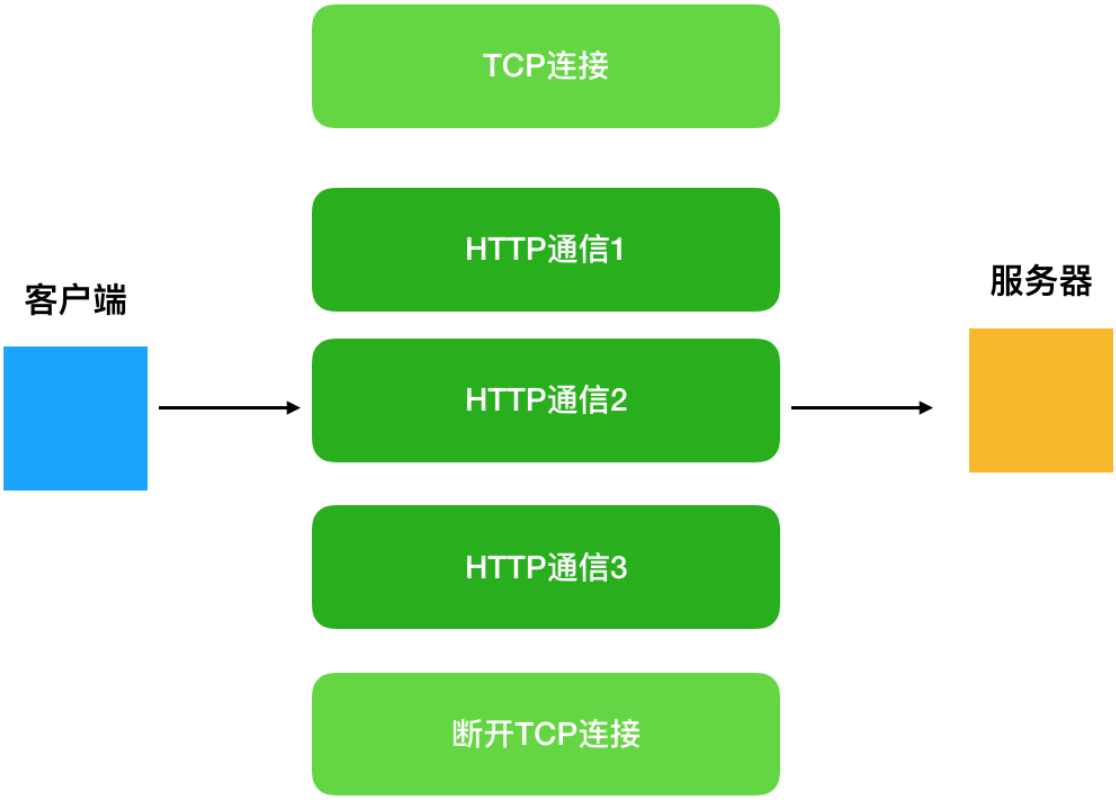
HTTP1.1 解决的问题

要知道 HTTP1.1 解决了什么问题，首先要知道 HTTP1.0 有什么痛点。HTTP1.0 的问题有很多，但最关键的主要是两个。我们这里把问题本身和解决方案放在一起来看：

TCP连接不可复用——HTTP/1.0 每进行一次 HTTP 通信，都需要勤勤恳恳地“三步走”——TCP 连接、HTTP 通信、断开 TCP 连接：



我们看到两次 HTTP 通信就要消耗两次 TCP 连接的资源。当通信的量不大时，计算机资源是吃得消的。但是随着互联网产业的发展，HTTP 通信的量呈指数级上升，通信频率变得非常高。这时人们就希望不同的 HTTP 通信之间不要再徒增一次 TCP 连接的消耗：



像上图这样一个 TCP 连接里可以进行多次 HTTP 通信的机制，就是 HTTP1.1 做的最重头的优化——**实现长连接**。

队头阻塞问题—— HTTP/1.0中，请求与请求间是串行的。如果我发送了 A 请求，那么 A 请求的响应返回之前，你的 B 请求不管多么着急都出不去。如果 A 请求去了一年，那么 B 请求就得耗上一年，活活被堵死。这就是著名的“队头阻塞问题”。

HTTP/1.1 尝试用管线化来解决这个问题。

HTTP/1.1 中的管线化是指，允许多个 HTTP 请求批量地提交给服务器。不过这样做仍然无法从根儿上解决队头阻塞——虽然发送动作可以并行，不过服务器依然需要根据请求顺序来回复浏览器的请求，也就是说响应仍是串行的。A 如果一年没有被响应，那么 B 也休想被响应。

HTTP2.0 对性能的改进

改进性能是 HTTP2.0 的核心目标。我们来看看为此它做了哪些重要的努力：

二进制分帧

在 HTTP1.x 中，数据以文本的格式进行传输，解析起来比较低效。

HTTP2.0 在传输消息时，首先会将消息划分为更小的消息和帧，然后再对其采取二进制格式的编码，确保高效的解析。

头部压缩

HTTP2.0 中，客户端和服务器分别会维护一份相同的静态字典，这个字典用来存储常见的头部名称，以及常见的头部名称和值的组合。同时还会维护一份相同的动态字典，这个字典可以实时被更新。

如此一来，第一次相互通信过后，后面的请求只需要发送与前面请求之间头部不同的地方，其它的头部信息都可以从字典中获取。相对于 HTTP1.x 中每次都要携带整个头部跑来跑去的笨重操作来说，大大节省了网络开销。

服务端推送

在 HTTP1.x 中，如果用户请求了资源 A，结果发现自己如果要用资源 A，那么必须依赖资源 B，这时他不得不再消耗一个请求。

而 HTTP2.0 中，允许服务器主动向客户端 push 资源。也就是说当服务器发现客户端请求了资源 A，却忘了请求资源 A 依赖的资源 B 时，它可以主动将资源 B 顺手推送给客户端。

这样一来，当客户端发现自己漏掉一个必要请求的时候，直接从缓存中就可以读到资源 B 了，而不必再消耗一个请求。

多路复用

前面咱们已经提到，HTTP1.x 并不能真正解决队头阻塞的问题。

HTTP1.x 解决不了的问题，HTTP2.0 来解决！

没错，多路复用其实就是进化版的长连接。

在 HTTP 2.0 中，一次连接建立后，只要这个连接还在，那么客户端就可以在一个链接中批量发起多个请求。同时，请求与请求间完全不阻塞，A 请求的响应就算没回来，也不影响 B 请求收到自己的响应。请求与请求间做到了高度的独立，真正实现了并行请求。由此，彻底规避了队头阻塞问题。

}

39 TCP 与 UDP

41 真正理解 HTTPS