30-时代之风（上）：HTTP2特性概览

在第14讲里，我们看到HTTP有两个主要的缺点：安全不足和性能不高。

刚结束的“安全篇”里的HTTPS，通过引入SSL/TLS在安全上达到了“极致”，但在性能提升方面却是乏善可陈，只优化了握手加密的环节，对于整体的数据传输没有提出更好的改进方案，还只能依赖于“长连接”这种“落后”的技术（参见第17讲）。

所以，在HTTPS逐渐成熟之后，HTTP就向着性能方面开始“发力”，走出了另一条进化的道路。

在第1讲的HTTP历史中你也看到了，“秦失其鹿，天下共逐之”，Google率先发明了SPDY协议，并应用于自家的浏览器Chrome，打响了HTTP性能优化的“第一枪”。

随后互联网标准化组织IETF以SPDY为基础，综合其他多方的意见，终于推出了HTTP/1的继任者，也就是今天的主角“HTTP/2”，在性能方面有了一个大的飞跃。

为什么不是HTTP/2.0

你一定很想知道，为什么HTTP/2不像之前的“1.0”“1.1”那样叫“2.0”呢？

这个也是很多初次接触HTTP/2的人问的最多的一个问题，对此HTTP/2工作组特别给出了解释。

他们认为以前的“1.0”“1.1”造成了很多的混乱和误解，让人在实际的使用中难以区分差异，所以就决定HTTP协议不再使用小版本号（minor version），只使用大版本号（major version），从今往后HTTP协议不会出现HTTP/2.0、2.1，只会有“HTTP/2”“HTTP/3”……

这样就可以明确无误地辨别出协议版本的“跃进程度”，让协议在一段较长的时期内保持稳定，每当发布新版本的HTTP协议都会有本质的不同，绝不会有“零敲碎打”的小改良。

兼容HTTP/1

由于HTTPS已经在安全方面做的非常好了，所以HTTP/2的唯一目标就是改进性能。

但它不仅背负着众多的期待，同时还背负着HTTP/1庞大的历史包袱，所以协议的修改必须小心谨慎，兼容性是首要考虑的目标，否则就会破坏互联网上无数现有的资产，这方面TLS已经有了先例（为了兼容TLS1.2不得不进行“伪装”）。

那么，HTTP/2是怎么做的呢？

因为必须要保持功能上的兼容，所以HTTP/2把HTTP分解成了“语义”和“语法”两个部分，“语义”层不做改动，与HTTP/1完全一致（即RFC7231）。比如请求方法、URI、状态码、头字段等概念都保留不变，这样就消除了再学习的成本，基于HTTP的上层应用也不需要做任何修改，可以无缝转换到HTTP/2。

特别要说的是，与HTTPS不同，HTTP/2没有在URI里引入新的协议名，仍然用“http”表示明文协议，用“https”表示加密协议。

这是一个非常了不起的决定，可以让浏览器或者服务器去自动升级或降级协议，免去了选择的麻烦，让用户

在上网的时候都意识不到协议的切换，实现平滑过渡。

在“语义”保持稳定之后，HTTP/2在“语法”层做了“天翻地覆”的改造，完全变更了HTTP报文的传输格式。

头部压缩

首先，HTTP/2对报文的头部做了一个“大手术”。

通过“进阶篇”的学习你应该知道，HTTP/1里可以用头字段“Content-Encoding”指定Body的编码方式，比如用gzip压缩来节约带宽，但报文的另一个组成部分——Header却被无视了，没有针对它的优化手段。

由于报文Header一般会携带“User Agent”“Cookie”“Accept”“Server”等许多固定的头字段，多达几百字节甚至上千字节，但Body却经常只有几十字节（比如GET请求、204/301/304响应），成了不折不扣的“大头儿子”。更要命的是，成千上万的请求响应报文里有很多字段值都是重复的，非常浪费，“长尾效应”导致大量带宽消耗在了这些冗余度极高的数据上。

所以，HTTP/2把“**头部压缩**”作为性能改进的一个重点，优化的方式你也肯定能想到，还是“压缩”。

不过HTTP/2并没有使用传统的压缩算法，而是开发了专门的“**HPACK**”算法，在客户端和服务器两端建立“字典”，用索引号表示重复的字符串，还采用哈夫曼编码来压缩整数和字符串，可以达到50%~90%的高压缩率。

二进制格式

你可能已经很习惯于HTTP/1里纯文本形式的报文了，它的优点是“一目了然”，用最简单的工具就可以开发调试，非常方便。

但HTTP/2在这方面没有“妥协”，决定改变延续了十多年的现状，不再使用肉眼可见的ASCII码，而是向下层的TCP/IP协议“靠拢”，全面采用二进制格式。

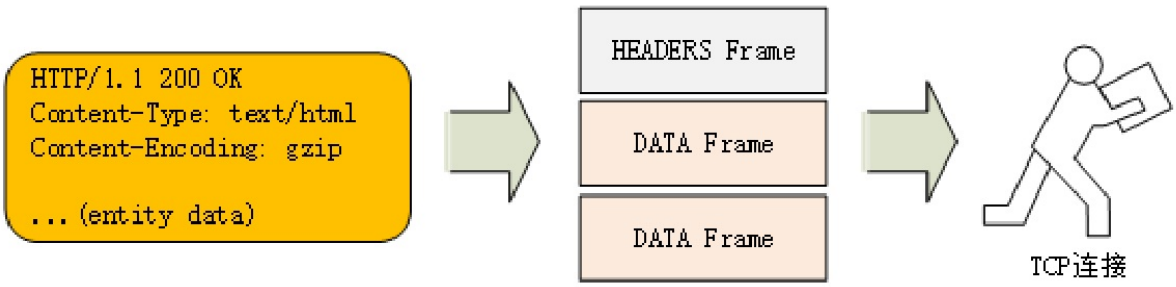
这样虽然对人不友好，但却大大方便了计算机的解析。原来使用纯文本的时候容易出现多义性，比如大小写、空白字符、回车换行、多字少字等等，程序在处理时必须用复杂的状态机，效率低，还麻烦。

而二进制里只有“0”和“1”，可以严格规定字段大小、顺序、标志位等格式，“对就是对，错就是错”，解析起来没有歧义，实现简单，而且体积小、速度快，做到“内部提效”。

以二进制格式为基础，HTTP/2就开始了“大刀阔斧”的改革。

它把TCP协议的部分特性挪到了应用层，把原来的“Header+Body”的消息“打散”为数个小片的**二进制“帧”**（Frame），用“HEADERS”帧存放头数据、“DATA”帧存放实体数据。

这种做法有点像是“Chunked”分块编码的方式（参见第16讲），也是“化整为零”的思路，但HTTP/2数据分帧后“Header+Body”的报文结构就完全消失了，协议看到的只是一个个的“碎片”。



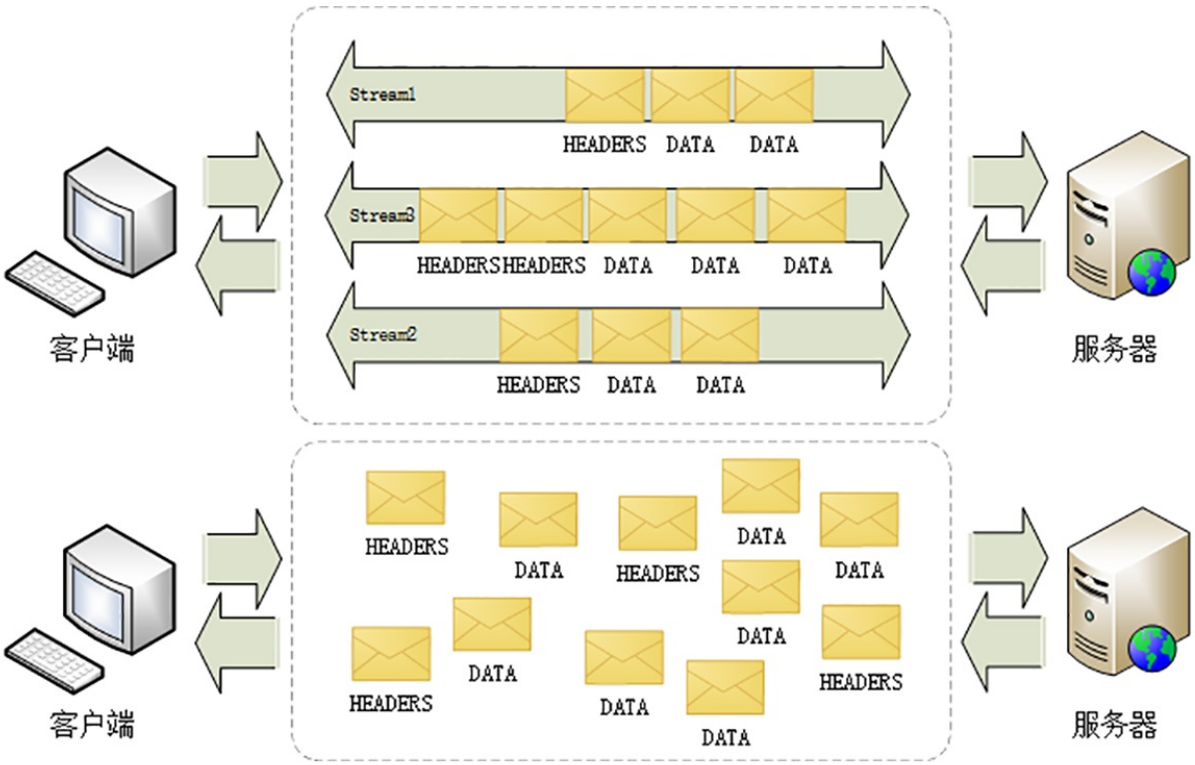
虚拟的“流”

消息的“碎片”到达目的地后应该怎么组装起来呢？

HTTP/2为此定义了一个“**流**”（Stream）的概念，**它是二进制帧的双向传输序列**，同一个消息往返的帧会分配一个唯一的流ID。你可以想象把它成是一个虚拟的“数据流”，在里面流动的是一串有先后顺序的数据帧，这些数据帧按照次序组装起来就是HTTP/1里的请求报文和响应报文。

因为“流”是虚拟的，实际上并不存在，所以HTTP/2就可以在一个TCP连接上用“**流**”同时发送多个“碎片化”的消息，这就是常说的“**多路复用**”（Multiplexing）——多个往返通信都复用一个连接来处理。

在“流”的层面上看，消息是一些有序的“帧”序列，而在“连接”的层面上看，消息却是乱序收发的“帧”。多个请求/响应之间没有了顺序关系，不需要排队等待，也就不会再出现“队头阻塞”问题，降低了延迟，大幅度提高了连接的利用率。



为了更好地利用连接，加大吞吐量，HTTP/2还添加了一些控制帧来管理虚拟的“流”，实现了优先级和流量控制，这些特性也和TCP协议非常相似。

HTTP/2还在一定程度上改变了传统的“请求-应答”工作模式，服务器不再是完全被动地响应请求，也可以新建“流”主动向客户端发送消息。比如，在浏览器刚请求HTML的时候就提前把可能会用到的JS、CSS文件发给客户端，减少等待的延迟，这被称为“**服务器推送**”（Server Push，也叫Cache Push）。

强化安全

出于兼容的考虑，HTTP/2延续了HTTP/1的“明文”特点，可以像以前一样使用明文传输数据，不强制使用加密通信，不过格式还是二进制，只是不需要解密。

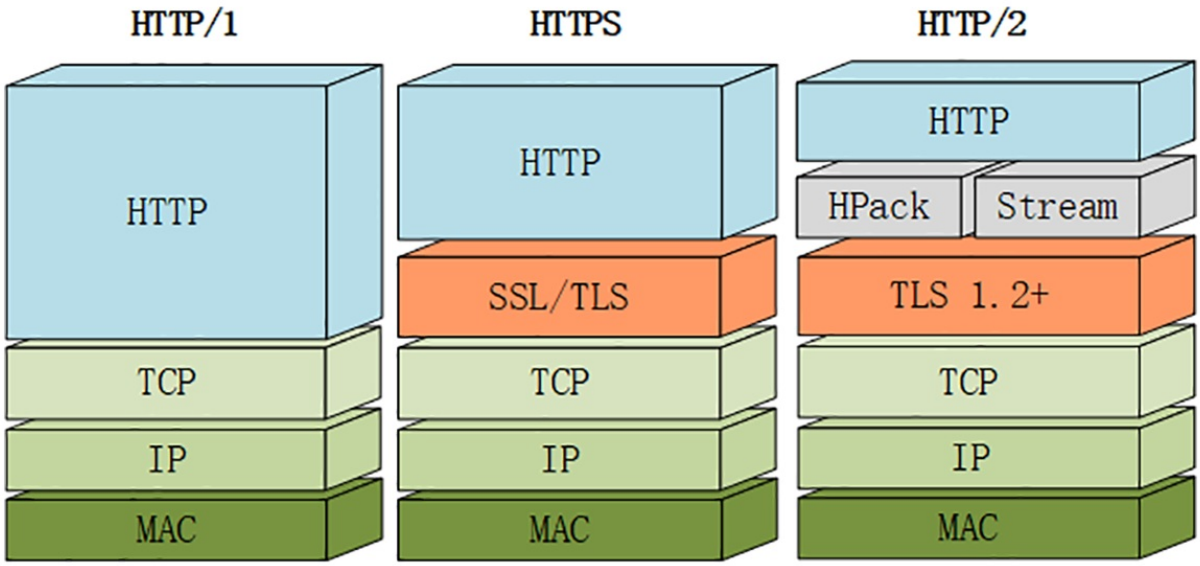
但由于HTTPS已经是大势所趋，而且主流的浏览器Chrome、Firefox等都公开宣布只支持加密的HTTP/2，所以“事实上”的HTTP/2是加密的。也就是说，互联网上通常所能见到的HTTP/2都是使用“https”协议名，跑在TLS上面。

为了区分“加密”和“明文”这两个不同的版本，HTTP/2协议定义了两个字符串标识符：“h2”表示加密的HTTP/2，“h2c”表示明文的HTTP/2，多出的那个字母“c”的意思是“clear text”。

在HTTP/2标准制定的时候（2015年）已经发现了很多SSL/TLS的弱点，而新的TLS1.3还未发布，所以加密版本的HTTP/2在安全方面做了强化，要求下层的通信协议必须是TLS1.2以上，还要支持前向安全和SNI，并且把几百个弱密码套件列入了“黑名单”，比如DES、RC4、CBC、SHA-1都不能在HTTP/2里使用，相当于底层用的是“TLS1.25”。

协议栈

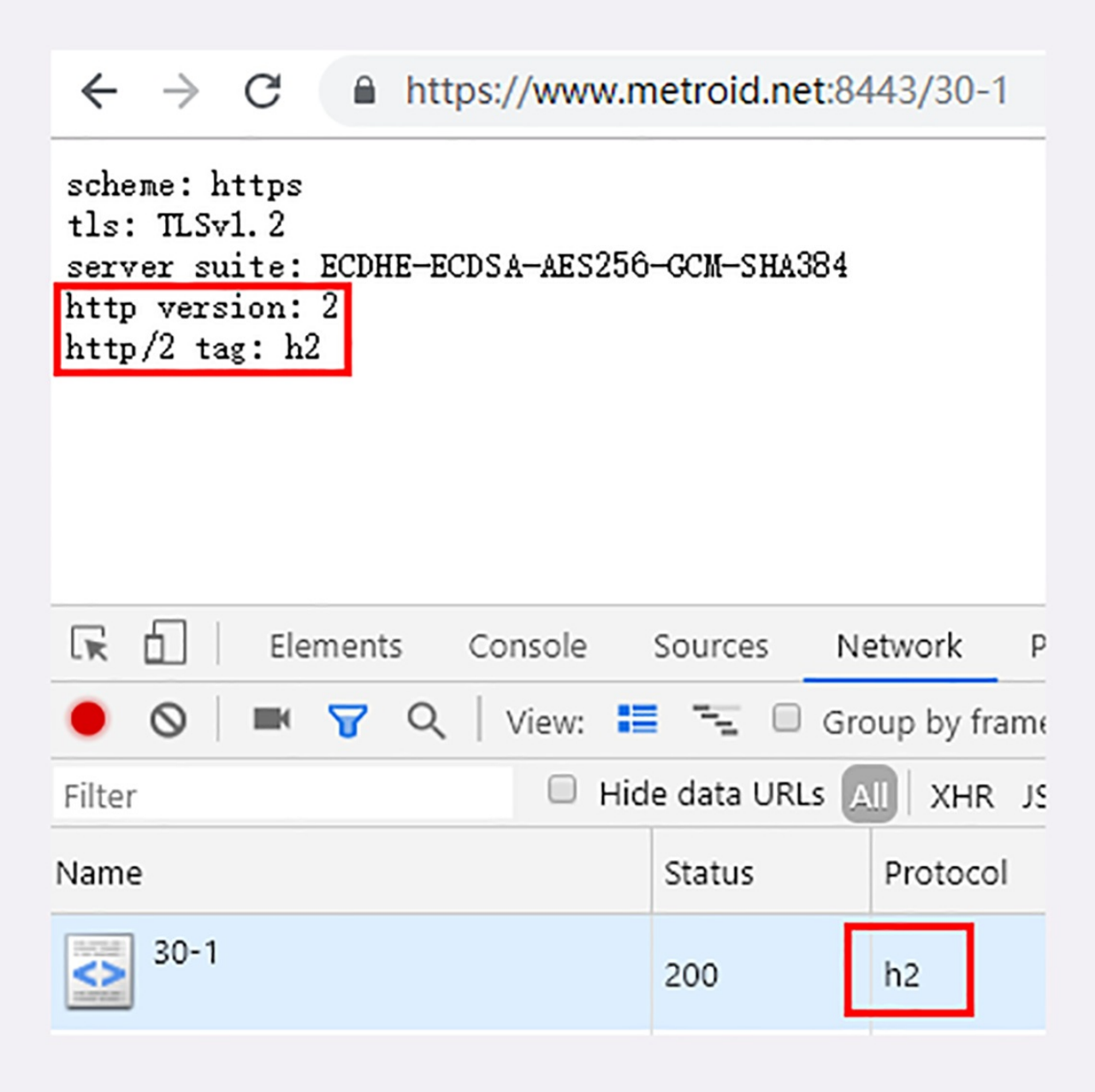
下面的这张图对比了HTTP/1、HTTPS和HTTP/2的协议栈，你可以清晰地看到，HTTP/2是建立在“HPack”“Stream”“TLS1.2”基础之上的，比HTTP/1、HTTPS复杂了一些。



虽然HTTP/2的底层实现很复杂，但它的“语义”还是简单的HTTP/1，之前学习的知识不会过时，仍然能够用得上。

我们的实验环境在新的域名“**www.metroid.net**”上启用了HTTP/2协议，你可以把之前“进阶篇”“安全篇”的测试用例都走一遍，再用Wireshark抓一下包，实际看看HTTP/2的效果和对老协议的兼容性（例如“http://www.metroid.net/11-1”）。

在今天这节课专用的URI“/30-1”里，你还可以看到服务器输出了HTTP的版本号“2”和标识符“h2”，表示这是加密的HTTP/2，如果改用“https://www.chrono.com/30-1”访问就会是“1.1”和空。



你可能还会注意到URI里的一个小变化，端口使用的是“8443”而不是“443”。这是因为443端口已经被“www.chrono.com”的HTTPS协议占用，Nginx不允许在同一个端口上根据域名选择性开启HTTP/2，所以就不得不改用了“8443”。

小结

今天我简略介绍了HTTP/2的一些重要特性，比较偏重理论，下一次我会用Wireshark抓包，具体讲解HTTP/2的头部压缩、二进制帧和流等特性。

1. HTTP协议取消了小版本号，所以HTTP/2的正式名字不是2.0；
2. HTTP/2在“语义”上兼容HTTP/1，保留了请求方法、URI等传统概念；
3. HTTP/2使用“HPACK”算法压缩头部信息，消除冗余数据节约带宽；
4. HTTP/2的消息不再是“Header+Body”的形式，而是分散为多个二进制“帧”；
5. HTTP/2使用虚拟的“流”传输消息，解决了困扰多年的“队头阻塞”问题，同时实现了“多路复用”，提高连接的利用率；

6. HTTP/2也增强了安全性，要求至少是TLS1.2，而且禁用了很多不安全的密码套件。

课下作业

1. 你觉得明文形式的HTTP/2（h2c）有什么好处，应该如何使用呢？
2. 你觉得应该怎样理解HTTP/2里的“流”，为什么它是“虚拟”的？
3. 你能对比一下HTTP/2与HTTP/1、HTTPS的相同点和不同点吗？

欢迎你把自己的学习体会写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

课外小贴士

01 在早期还有一个“HTTP-NG”（HTTP Next Generation）项目，最终失败了。

02 HTTP/2 的“前身”SPDY 在压缩头部时使用了 gzip，但发现会受到“CRIME”攻击，所以开发了专用的压缩算法 HPACK。

03 HTTP/2 里的“流”可以实现 HTTP/1 里的“管道”（pipeline）功能，而且综合性能更好，所以“管道”在 HTTP/2 里就被废弃了。

04 如果你写过 Linux 程序，用过 epoll，就应该知道 epoll 也是一种“多路复用”，不过它是“I/O Multiplexing”。

05 HTTP/2 要求必须实现的密码套件是“TLS_

ECDHE_RSA_WITH_AES_128_GCM_SHA256”，比 TLS1.2 默认的 “TLS_RSA_WITH_AES_128_CBC_SHA” 的安全强度高了很多。

06 实验环境的 “www.metroid.net” 启用了 RSA 和 ECC 双证书，在浏览器里可以看到实际连接时用的会是 ECC 证书。另外，这个域名还用到了第 29 讲里的重定向跳转技术，使用 301 跳转，把 “80/443” 端口的请求重定向到 HTTP/2 的 “8443”。

精选留言：

- magicnum 2019-08-05 12:24:40
  h2c优点是性能，不需要TLS握手以及加解密。可以通过curl工具构造h2c请求；
  h2的流是虚拟的因为它是使用帧传输数据的，相同streamid的帧组成了虚拟消息以及流；
  相同点：都是基于tcp或TLS，并且是基于请求-响应模型，schema还是http或https不会有http2。
  不同点：h2使用二进制传输消息并且通过HPACK压缩请求头，实现流多路复用、服务器推送

[2赞]

作者回复2019-08-05 20:49:56
great!

- -W.LI- 2019-08-06 08:34:52
老师好。之前用MQ的时候，AMPT协议说是只打开一个长链接TCP链接。然后AMPT协议每次都是在这个链接里打开信道进行传输。队列和client(服务器)IP和端口基本固定，如果以TCP链接形式会占用很多端口号，还影响性能，所以就采用了信道。可是信道和信道之间如何实现数据隔离和马上要讲的http2的channel原理差不多么?

- -W.LI- 2019-08-05 21:54:54
课后习题出的很好。可惜我不会坐等答案
1.内网用h2c会比https快么?
2.感觉回答虚拟流之前给先回答啥是真真的流。我对流的理解是有序，切只能读一次。http2支持乱序发，我猜也支持，部分帧重发，所以就是虚拟的了。
3.共同，都是应用层协议，传输成都用的TCP。
不同:https=TLS+HTTP/HTTP2，安全。
http2:二进制传输，对header压缩，通过二进制分帧解决了队头阻塞，传输效率更高，服务端可推数据
http:明文，队头阻塞，半双工。
问题1:一个TCP链接可以打开很多channel是吧，每一个channel都可以传输数据。底层具体怎么实现的啊，是怎么区分channel里的数据谁是谁的?
问题2:我之前看见TPC好像是通过服务端IP,服务端端口号,客户端端IP,客户端端口号。来唯一标识一个链接的。http1的时候队头阻塞，继续要多建http链接。每建立一个链接客户端就用一个不同的端口号么?

作者回复2019-08-06 08:06:14

1.当然，省去了加密的成本。

2.所谓“虚拟的流”，是指流实际上是多个同一序号的帧，并没有真正的流数据结构，这与连接不同。

3.正确。

4.你说的channel应该是http/2里的“流”吧，http/2里没有channel。流是由帧组成的，帧头里有流id标记所属的流，马上会讲具体的细节。

5.标记一个tcp连接要用四元组（客户端ip端口+服务器ip端口），所以肯定要用一个新的端口号，在客户端这是临时分配的，而服务器是固定的端口。

- レイン小雨 2019-08-05 21:20:04
真好

作者回复2019-08-06 06:53:37
thanks。

- 阿锋 2019-08-05 15:12:12
突然想起了一个问题，get和post请求其中一个区别是，post请求会把请求的数据放入请求体（body）中，而get请求是拼接到url后面。get请求是不是一定不能往请求体（body）中放入数据。还是这些都只是客户端和服务端的约定，可以灵活的自定义，没有强制的要求。

作者回复2019-08-05 20:47:49

get也可以有body，post也可以用query参数，区别的关键在于动作语义，一个是取一个是存。

- nb Ack 2019-08-05 12:43:09
  老师好。我想问一下，http2的多路复用和http的长连接效果不是一样吗？

  作者回复2019-08-05 20:48:49
  完全不一样。

  多路复用多个请求没有顺序，而长连接多个请求必须排队，就会队头阻塞。

  可以再看看示意图体会一下。