30 | 时代之风（上）：HTTP/2特性概览

2019-08-05 Chrono

《透视HTTP协议》 课程介绍 >

讲述：Chrono

时长 11:03 大小 12.66M

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。

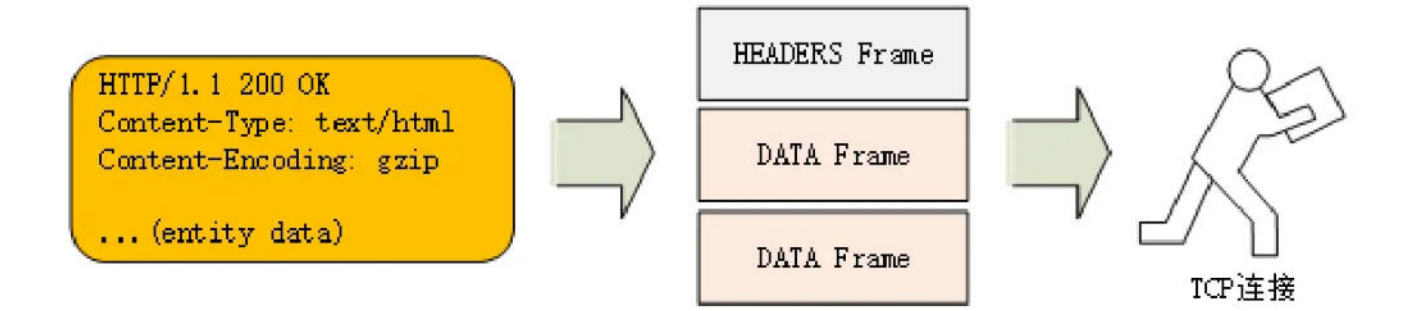
这样虽然对人不友好，但却大大方便了计算机的解析。原来使用纯文本的时候容易出现多义性，比如大小写、空白字符、回车换行、多字少字等等，程序在处理时必须用复杂的状态机，效率低，还麻烦。

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”

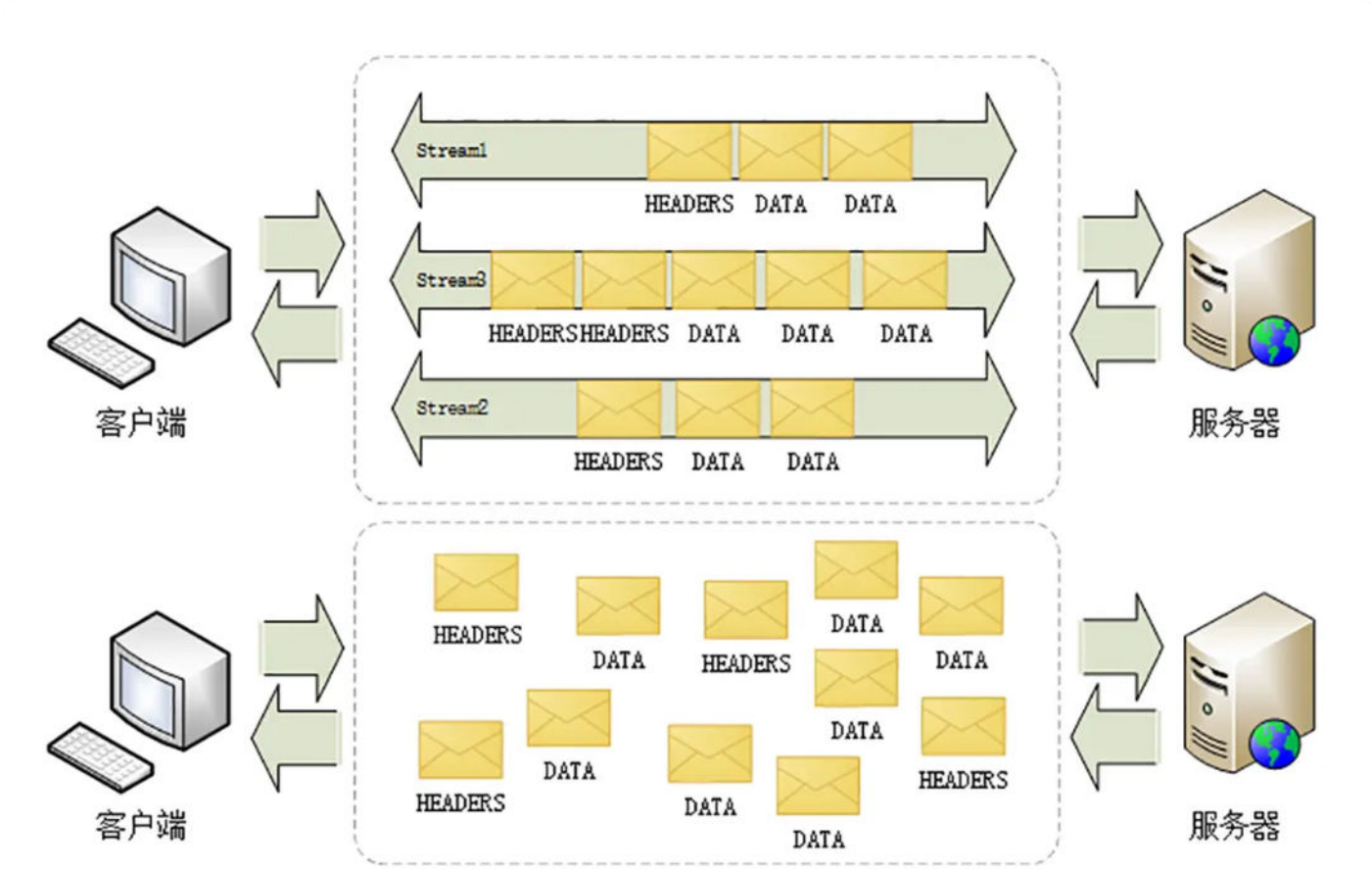
消息的“碎片”到达目的地后应该怎么组装起来呢？

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。你可以把它想象成是一个虚拟的“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。

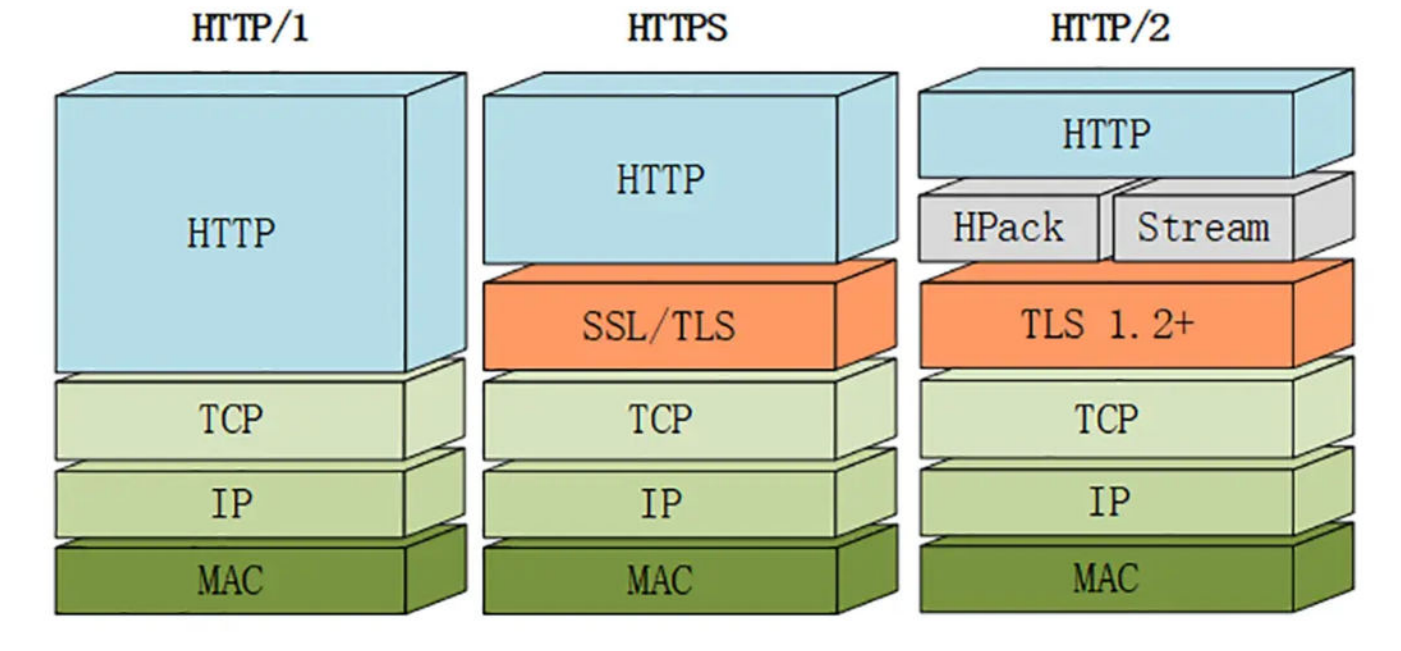
但由于 HTTPS 已经是大势所趋，而且主流的浏览器 Chrome、Firefox 等都公开宣布只支持加密的 HTTP/2，所以“事实上”的 HTTP/2 是加密的。也就是说，互联网上通常所能见到的 HTTP/2 都是使用“https”协议名，跑在 TLS 上面。

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

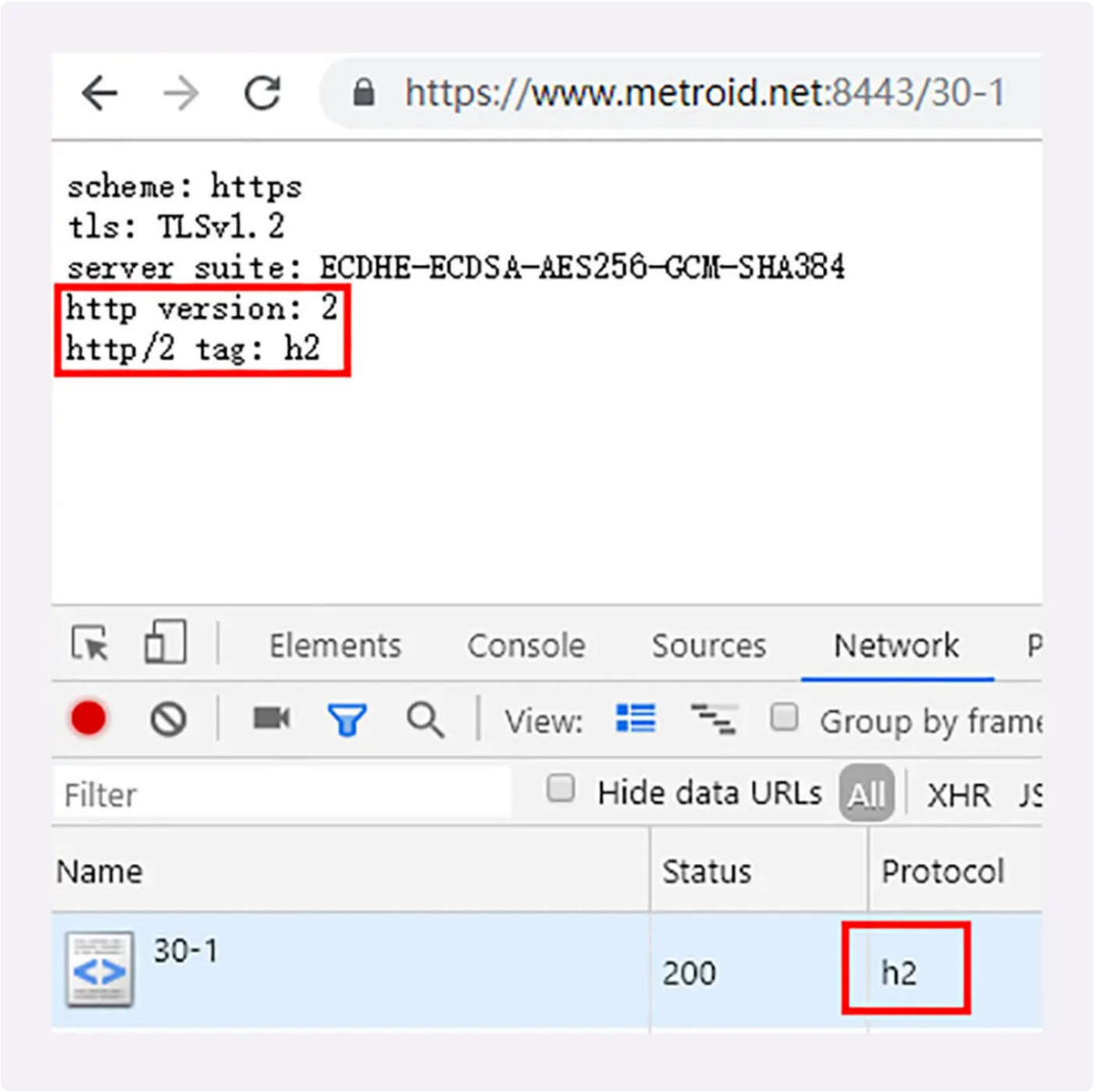
下面的这张图对比了 HTTP/1、HTTPS 和 HTTP/2 的协议栈，你可以清晰地看到，HTTP/2 是建立在“HPack”“Stream”“TLS1.2”基础之上的，比 HTTP/1、HTTPS 复杂了一些。



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

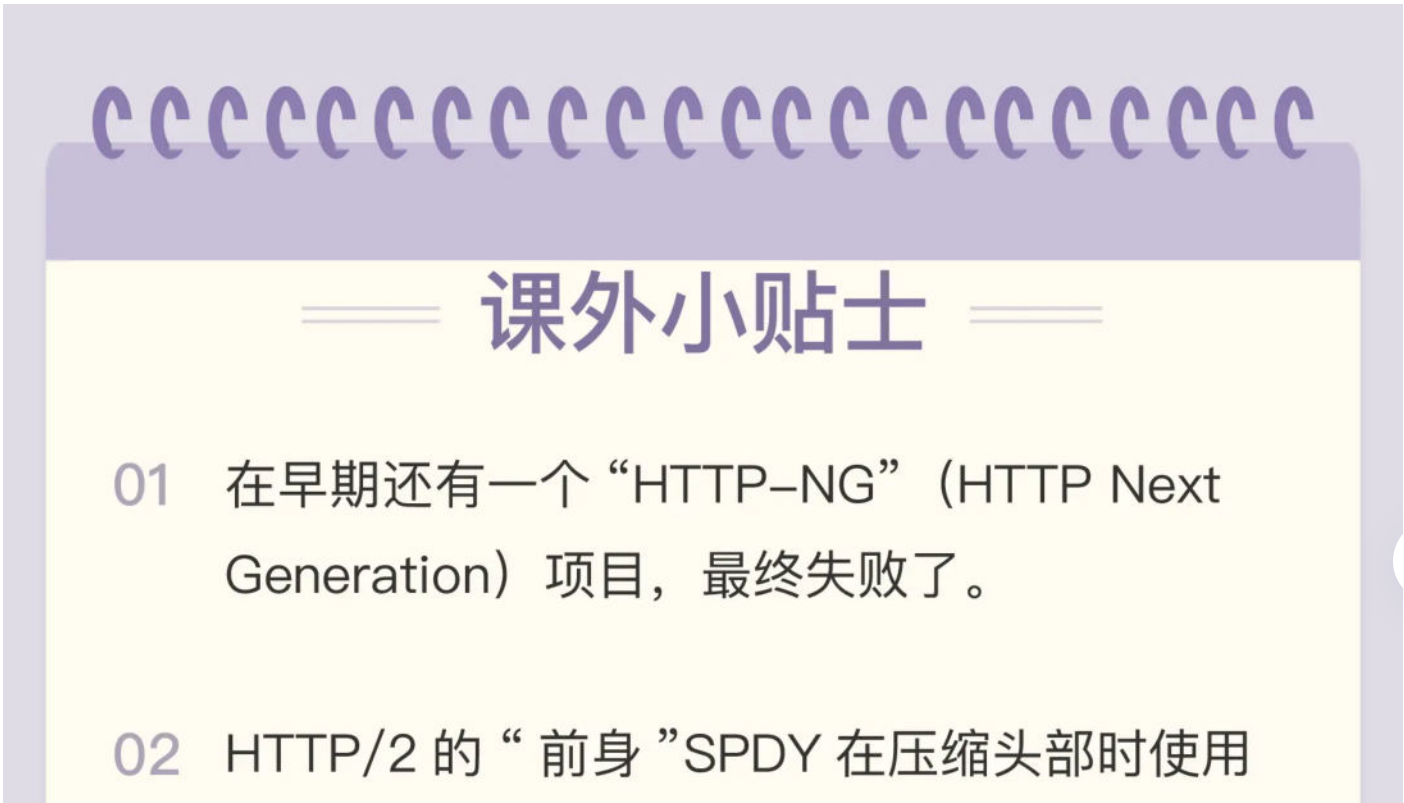
今天我简略介绍了 HTTP/2 的一些重要特性，比较偏重理论，下一次我会用 Wireshark 抓包，具体讲解 HTTP/2 的头部压缩、二进制帧和流等特性。

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。



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”。

上一篇　29 | 我应该迁移到HTTPS吗?

下一篇　31 | 时代之风（下）：HTTP/2内核剖析

精选留言 (29)

写留言

俊伟
2020-01-19

1.h2c使用明文传输，速度更快，不需要TLS握手。
2.客户端将多个请求分成不同的流，然后每个流里面在切成一个个帧，发送的时候是按帧发送的。每个帧存着一个流ID来表示它属于的流。服务端收到请求的时候将帧按流ID进行拼接。从传输的角度来看流是不存在的，只是看到了一个个帧，所以说流是虚拟的。
3.相同点：都是基于TCP和TLS的，url格式都是相同的。都是基于header+body的形式。都是请求–应答模型。
4.不同点： 1.使用了HPACK进行头部压缩。
2.使用的是二进制的方式进行传输。
3.将多个请求切分成帧发送，实现了多路复用。这个感觉上利用了多道程序设计的思想。
4.服务器可以主动向客户端推送消息。充分利用了TCP的全功双通道。

作者回复: 总结的非常好，great。

24

magicnum
2019-08-05

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

作者回复: great!

共 3 条评论> 24

阿锋
2019-08-05

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: 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。

共 2 条评论> 11

–W.LI–
2019-08-05

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:
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。

共 3 条评论>

7

nb Ack

2019-08-05

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: 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。

8

Maske

2020-06-26

1.明文传输时不需要进行加密解密动作，不需要TLS握手，能节约性能。适用于对数据传输安全性要求不高的场景。
2.http2改变了http1.1的“请求–应答”模式，将head+body的请求报文在传输过程中改为 head帧 + data帧，在同一个TCP/IP中，可以将多个请求分解为多个帧，从连接层面来看，这些帧是无序的，为了让接受端准确的将这些帧还原为一个一个独立的请求或响应，就给了每一个帧分配了streamid，streamid相同的即为同一个请求或响应的数据。因此，此处的流并不是真实有序的二进制字节，所以叫‘虚拟流’。
3.http1.1解决的是在万维网中，计算机之间的信息通信的一套规范，包括定义其属于应用层协议，建立在tcp/ip之上，请求响应的报文结构等。https不改变http1.1的原有属性，是在其之上新增了对数据安全性和有效性的特性，解决的是数据安全的问题，通过使用加密解密，数字证书，TLS握手等过程保证了这一点。http2解决的是性能问题，通过头部压缩，使用二进制传输，多路复用，服务器推送等策略使得http的性能更好。http2和https本质上都是对http1.1的扩展和延伸。

作者回复: 理解的很透彻，great。

BoyiKia
2020-05-18

http2 优点
1.兼容性
兼容以前的http1.1 ， https等。

2.性能提升
报文变成了 二进制数据帧，提高传输效率，和减少歧义。
①header 采用了头部压缩，来减小传输体积。
②body数据 放到了 data帧。
a.同一请求或响应的数据帧具有相同的帧标识(流ID)，两端接受到的帧数据可以通过同一帧标识，重新组装成请求或响应数据。
b.不同请求/响应的数据帧可以乱序发，避免生成请求队列造成的队头阻塞。
c.同一个TCP连接上，可以并行发送多种流的数据帧(多路复用，PS： http1的 多路复用是分母效应，同一连接串行增加http通信)。
d.强化了请求响应模式，服务器可以主动发送信息-服务器推送。
3.安全性
①.要求下层必须是 TlS1.2以上，支持前向安全，废除安全性比较低的密码套件。

作者回复: awesome!

4

夏目
2019-12-09

流就是逻辑上将数据帧按id分组了，同组有序，组间无序，本质就是id相同的几个数据帧所以流是虚拟的。在tcp层面还是队首阻塞的吧？需要等待ack

作者回复: 是的，理解的非常正确。

4

潇潇雨歇
2020-11-30

1、明文传输性能更好，省去了加密相关操作

2、流和请求/应答一样，但是流是相同流id的帧组合，不同流可以无序，相同流有序。整个看起来是无序的，请求之间不受影响。这也解决了http1.1的队头阻塞。
3、三者都是基于tcp的，基本语义是一样的。http2在性能上做了提升，比如二进制帧，流，服务器推送，HPACK算法等；https在安全上做了提升，下层多了TLS/SSL，要多做一些握手加密证书验证等操作。

作者回复: 回答的非常好！

3

渴望做梦
2019-08-29

老师，我有个疑问，既然http2是二进制的格式，那我们还能用chrome自带的工具调试吗?

作者回复: 可以的，Chrome会把二进制解码，还原为http/1的文本形式，你可以自己试一下。

2

Maske
2020-11-20

老师我又回来了，按之前的理解是，http2是对同一域名使用单一的TCP连接进行数据传输，多个请求同时进行，既然如此，为什么在chrome调试面板中还能看到资源还是是有请求排队时间的呢?

作者回复: http/2里面的流，就相当于http/1里的并发连接，要开一个新流同样也要有一些准备的工作。Chrome里的排队属于它自己的调度工作，与http/2协议是无关的。

1

思维决定未来
2020-09-07

http2的事实标准就是加密传输的，那是不是跟https重复了?

作者回复: 前面说过，对http的改进有两个方向，一个是安全，一个是效率，http/2是安全和效率兼顾，而https只是传输安全，效率上没有改进。

共 4 条评论>

1

谢一
2019-09-09

老师，既然在连接层，是无序的，那在http/2中是怎么保证frame的有序性的呢?

作者回复: tcp层是有序的，所以一个流里的多个帧会按照顺序依次到达，接收方只要依次接收就可以了。

1

Geek_68d3d2
2021-09-09

请问同一个流可以在一个tcp连接里面发送吗

作者回复: 当然了，流实际上就是多个无序的数据帧，在tcp连接里乱序发送，到了目的地再按照序号组装，就形成了流。

兔嘟嘟
2021-07-29

老师好，我不太理解为什么二进制帧可以提高解析效率，我的理解是这样的：
在HTTP/1.1中，请求方的字符串在TCP层被解码为Unicode二进制，然后应答方在HTTP层编码为utf-8字符串。
而在HTTP/2中，请求方的字符串在HTTP层被解码为二进制，然后应答方在浏览器处编码为字符串。所以好像没有省去时间或者资源。请老师赐教

作者回复: 二进制的好处显而易见，用位来表示信息，要比字符串表示简单，比如用01表示host，而用字符串就需要4个字节，而且要用状态机去检测单词，非常麻烦。

http/2在底层是二进制，解析起来快速方便，然后再到应用层对字符串做个映射就行了，不是再编码。

可以再用hpack来理解一下。

共 3 条评论>

疯琴
2021-07-16

请问老师，同一个流里面不同序号的帧可以乱序到达统一组装么？

作者回复: 不会的，tcp会保证有序送达，多个流是并行乱序发，但看单个流，它里面的帧还是有序的。

共 2 条评论>

Mingyan
2021-06-04

我有疑问，http2.0如果遇到服务器主动关闭tcp链接会理会回ack，再发fin等ack去关闭tcp链接还是不理会继续使用被关闭的tcp链接了？

作者回复: tcp连接关闭后就消失了，再建连就是一个新的连接，不存在复用。


hao
2021-06-04

为何新增了流的概念，就可以实现服务器主动推送呢？那HTTP2可以像websocket那样实现即使通信（服务器主动推送）？

作者回复: 在http/1里，请求只能由客户端发起，在http/2里，服务器端也可以发起流，所以服务器就可以用这样的流向客户端推送数据。

但http/2的推送目的不是即时通讯，功能上不如websocket那么强大。


ThinkerWalker
2020-12-19

老师，http2使用一个tcp连接传输多个流（多个请求），http1.1用的是多个tcp连接多个请求，会不会因为连接少而导致整体吞吐量降低？

作者回复: 不会，http/1虽然是多个连接，但单个连接的利用率很低，而http/2的一个连接利用率很高，还有压缩功能，所以相当于http/1的三四个连接甚至更多。


尿布
2020-09-10

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