

32 | 未来之路：HTTP/3展望

2019-08-09 Chrono

《透视HTTP协议》

课程介绍 >



讲述：Chrono

时长 10:29 大小 14.41M



在前面的两讲里，我们一起学习了 HTTP/2，你也应该看到了 HTTP/2 做出的许多努力，比如头部压缩、二进制分帧、虚拟的“流”与多路复用，性能方面比 HTTP/1 有了很大的提升，“基本上”解决了“队头阻塞”这个“老大难”问题。

HTTP/2 的“队头阻塞”

等等，你可能要发出疑问了：为什么说是“基本上”，而不是“完全”解决了呢？

这是因为 HTTP/2 虽然使用“帧”“流”“多路复用”，没有了“队头阻塞”，但这些手段都是在应用层里，而在下层，也就是 TCP 协议里，还是会发生“队头阻塞”。

这是怎么回事呢？

领资料



让我们从协议栈的角度来仔细看一下。在 HTTP/2 把多个“请求 – 响应”分解成流，交给 TCP 后，TCP 会再拆成更小的包依次发送（其实在 TCP 里应该叫 segment，也就是“段”）。

在网络良好的情况下，包可以很快送达目的地。但如果网络质量比较差，像手机上网的时候，就有可能会丢包。而 TCP 为了保证可靠传输，有个特别的“丢包重传”机制，丢失的包必须要等待重新传输确认，其他的包即使已经收到了，也只能放在缓冲区里，上层的应用拿不出来，只能“干着急”。

我举个简单的例子：

客户端用 TCP 发送了三个包，但服务器所在的操作系统只收到了后两个包，第一个包丢了。那么内核里的 TCP 协议栈就只能把已经收到的包暂存起来，“停下”等着客户端重传那个丢失的包，这样就又出现了“队头阻塞”。

由于这种“队头阻塞”是 TCP 协议固有的，所以 HTTP/2 即使设计出再多的“花样”也无法解决。

Google 在推 SPDY 的时候就已经意识到了这个问题，于是就又发明了一个新的“QUIC”协议，让 HTTP 跑在 QUIC 上而不是 TCP 上。

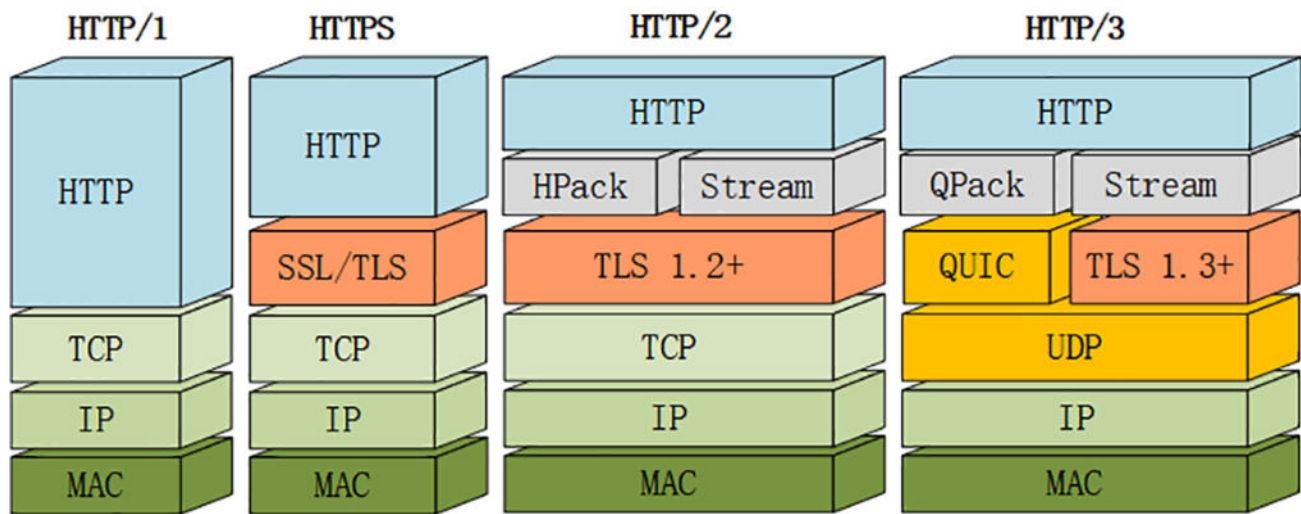
而这个“HTTP over QUIC”就是 HTTP 协议的下一个大版本，**HTTP/3**。它在 HTTP/2 的基础上又实现了质的飞跃，真正“完美”地解决了“队头阻塞”问题。

不过 HTTP/3 目前还处于草案阶段，正式发布前可能会有变动，所以我今天尽量不谈那些不稳定的细节。

这里先贴一下 HTTP/3 的协议栈图，让你对它有个大概的了解。

领资料





QUIC 协议

从这张图里，你可以看到 HTTP/3 有一个关键的变化，那就是它把下层的 TCP“抽掉”了，换成了 UDP。因为 UDP 是无序的，包之间没有依赖关系，所以就从根本上解决了“队头阻塞”。

你一定知道，UDP 是一个简单、不可靠的传输协议，只是对 IP 协议的一层很薄的包装，和 TCP 相比，它实际应用的较少。

不过正是因为它简单，不需要建连和断连，通信成本低，也就非常灵活、高效，“可塑性”很强。

所以，QUIC 就选定了 UDP，在它之上把 TCP 的那一套连接管理、拥塞窗口、流量控制等“搬”了过来，“去其糟粕，取其精华”，打造出了一个全新的可靠传输协议，可以认为是“新时代的 TCP”。



QUIC

领资料



QUIC 最早是由 Google 发明的，被称为 gQUIC。而当前正在由 IETF 标准化的 QUIC 被称为 iQUIC。两者的差异非常大，甚至比当年的 SPDY 与 HTTP/2 的差异还要大。

gQUIC 混合了 UDP、TLS、HTTP，是一个应用层的协议。而 IETF 则对 gQUIC 做了“清理”，把应用部分分离出来，形成了 HTTP/3，原来的 UDP 部分“下放”到了传输层，所以 iQUIC 有时候也叫“QUIC-transport”。

接下来要说的 QUIC 都是指 iQUIC，要记住，它与早期的 gQUIC 不同，是一个传输层的协议，和 TCP 是平级的。

QUIC 的特点

QUIC 基于 UDP，而 UDP 是“无连接”的，根本就不需要“握手”和“挥手”，所以天生就要比 TCP 快。

就像 TCP 在 IP 的基础上实现了可靠传输一样，QUIC 也基于 UDP 实现了可靠传输，保证数据一定能够抵达目的地。它还引入了类似 HTTP/2 的“流”和“多路复用”，单个“流”是有序的，可能会因为丢包而阻塞，但其他“流”不会受到影响。

为了防止网络上的中间设备（Middle Box）识别协议的细节，QUIC 全面采用加密通信，可以很好地抵御篡改和“协议僵化”（ossification）。

而且，因为 TLS1.3 已经在去年（2018）正式发布，所以 QUIC 就直接应用了 TLS1.3，顺便也就获得了 0-RTT、1-RTT 连接的好处。

但 QUIC 并不是建立在 TLS 之上，而是内部“包含”了 TLS。它使用自己的帧“接管”了 TLS 里的“记录”，握手消息、警报消息都不使用 TLS 记录，直接封装成 QUIC 的帧发送，省掉了一次开销。

QUIC 内部细节

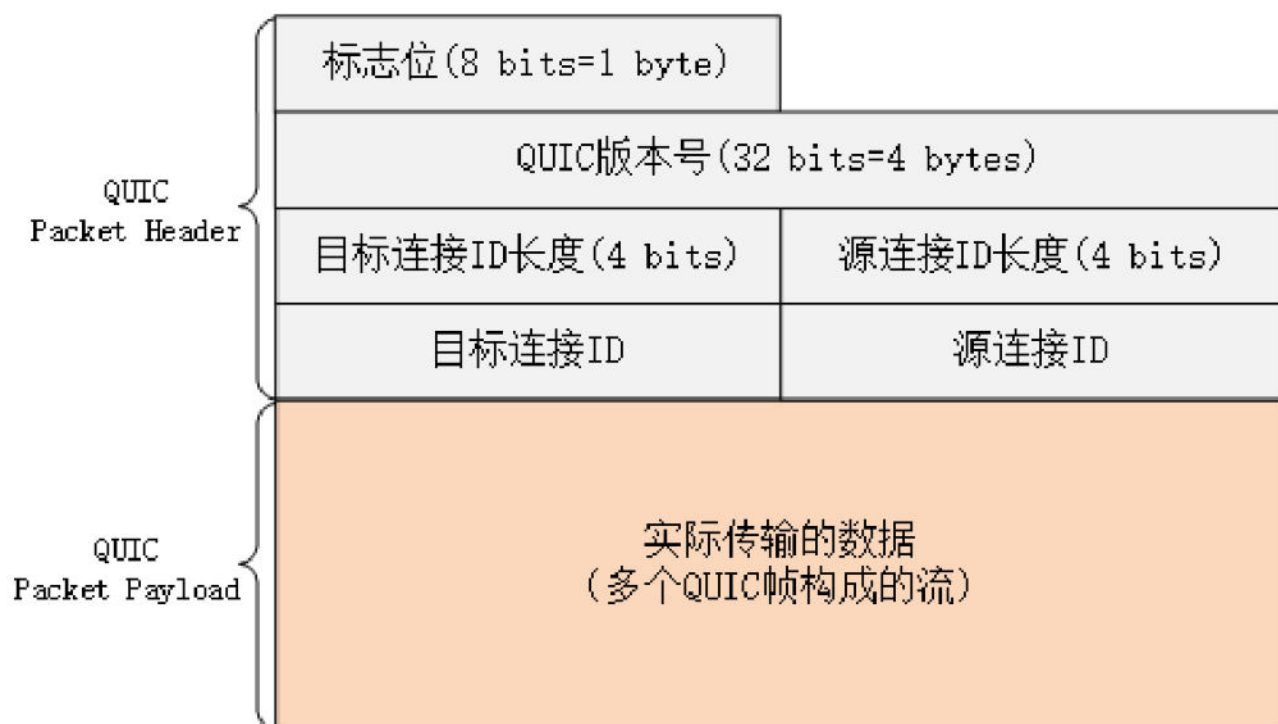
由于 QUIC 在协议栈里比较偏底层，所以我只简略介绍两个内部的关键知识点。

QUIC 的基本数据传输单位是**包**（packet）和**帧**（frame），一个包由多个帧组成，包面向的是“连接”，帧面向的是“流”。

领资料



QUIC 使用不透明的“**连接 ID**”来标记通信的两个端点，客户端和服务端可以自行选择一组 ID 来标记自己，这样就解除了 TCP 里连接对“IP 地址 + 端口”（即常说的四元组）的强绑定，支持“**连接迁移**”（Connection Migration）。



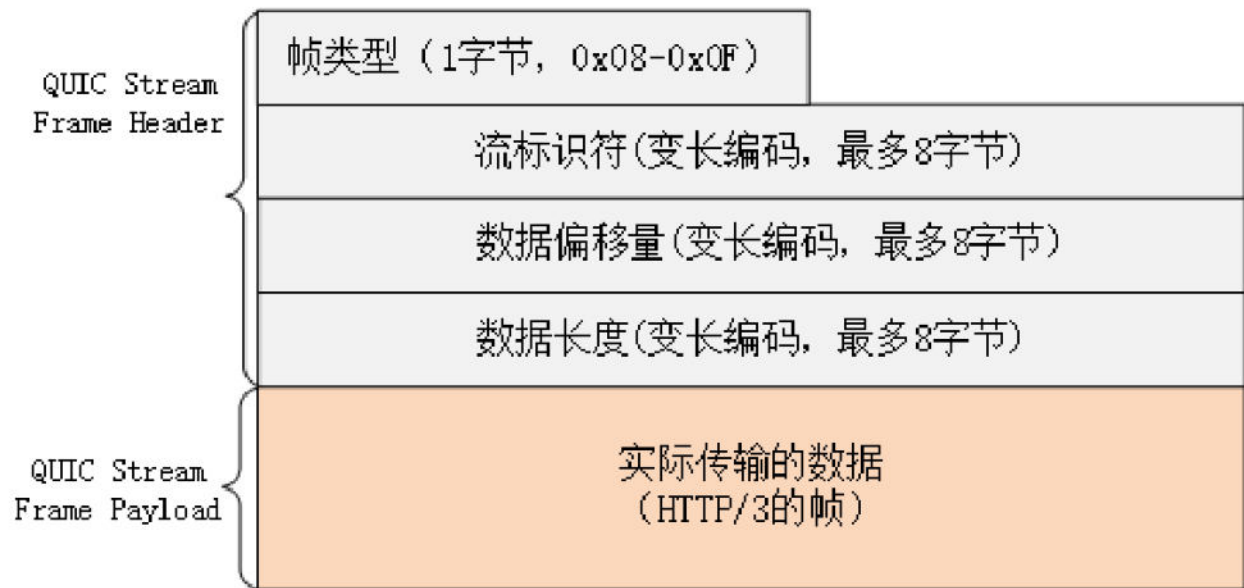
比如你下班回家，手机会自动由 4G 切换到 WiFi。这时 IP 地址会发生变化，TCP 就必须重新建立连接。而 QUIC 连接里的两端连接 ID 不会变，所以连接在“逻辑上”没有中断，它就可以在新的 IP 地址上继续使用之前的连接，消除重连的成本，实现连接的无缝迁移。

QUIC 的帧里有多种类型，PING、ACK 等帧用于管理连接，而 STREAM 帧专门用来实现流。

QUIC 里的流与 HTTP/2 的流非常相似，也是帧的序列，你可以对比着来理解。但 HTTP/2 里的流都是双向的，而 QUIC 则分为双向流和单向流。

领资料





QUIC 帧普遍采用变长编码，最少只要 1 个字节，最多有 8 个字节。流 ID 的最大可用位数是 62，数量上比 HTTP/2 的 2^{31} 大大增加。

流 ID 还保留了最低两位用作标志，第 1 位标记流的发起者，0 表示客户端，1 表示服务器；第 2 位标记流的方向，0 表示双向流，1 表示单向流。

所以 QUIC 流 ID 的奇偶性质和 HTTP/2 刚好相反，客户端的 ID 是偶数，从 0 开始计数。

HTTP/3 协议

了解了 QUIC 之后，再来看 HTTP/3 就容易多了。

因为 QUIC 本身就已经支持了加密、流和多路复用，所以 HTTP/3 的工作减轻了很多，把流控制都交给 QUIC 去做。调用的不再是 TLS 的安全接口，也不是 Socket API，而是专门的 QUIC 函数。不过这个“QUIC 函数”还没有形成标准，必须要绑定到某一个具体的实现库。

HTTP/3 里仍然使用流来发送“请求 - 响应”，但它自身不需要像 HTTP/2 那样再去定义流，而是直接使用 QUIC 的流，相当于做了一个“概念映射”。

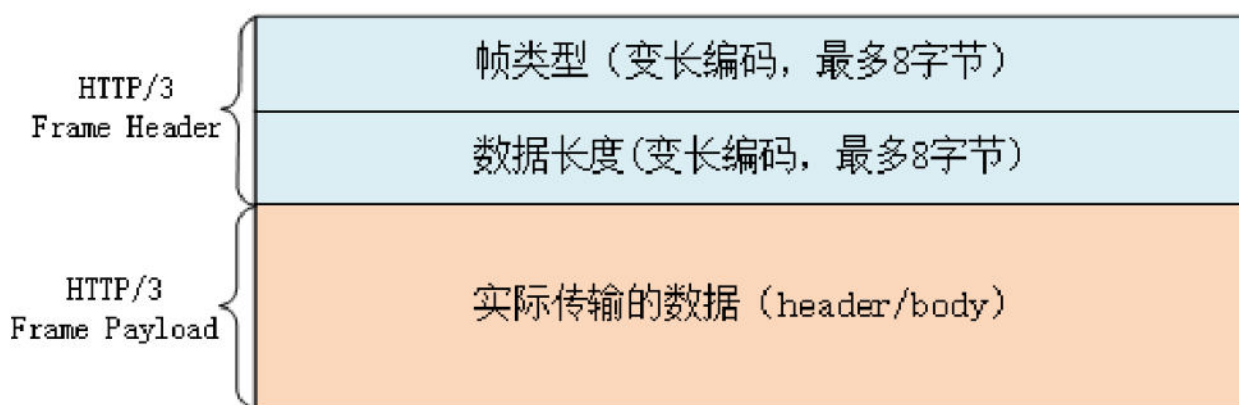
HTTP/3 里的“双向流”可以完全对应到 HTTP/2 的流，而“单向流”在 HTTP/3 里用来实现控制和推送，近似地对应 HTTP/2 的 0 号流。

由于流管理被“下放”到了 QUIC，所以 HTTP/3 里帧的结构也变简单了。

领资料



帧头只有两个字段：类型和长度，而且同样都采用变长编码，最小只需要两个字节。



HTTP/3 里的帧仍然分成数据帧和控制帧两类，HEADERS 帧和 DATA 帧传输数据，但其他一些帧因为在下层的 QUIC 里有了替代，所以在 HTTP/3 里就都消失了，比如 RST_STREAM、WINDOW_UPDATE、PING 等。

头部压缩算法在 HTTP/3 里升级成了“**QPACK**”，使用方式上也做了改变。虽然也分成静态表和动态表，但在流上发送 HEADERS 帧时不能更新字段，只能引用，索引表的更新需要在专门的单向流上发送指令来管理，解决了 HPACK 的“队头阻塞”问题。

另外，QPACK 的字典也做了优化，静态表由之前的 61 个增加到了 98 个，而且序号从 0 开始，也就是说“:authority”的编号是 0。

HTTP/3 服务发现

讲了这么多，不知道你注意到了没有：HTTP/3 没有指定默认的端口号，也就是说不一定非要在 UDP 的 80 或者 443 上提供 HTTP/3 服务。

那么，该怎么“发现”HTTP/3 呢？

这就要用到 HTTP/2 里的“扩展帧”了。浏览器需要先用 HTTP/2 协议连接服务器，然后服务器可以在启动 HTTP/2 连接后发送一个“**Alt-Svc**”帧，包含一个“h3=host:port”的字符串，告诉浏览器在另一个端点上提供等价的 HTTP/3 服务。

浏览器收到“Alt-Svc”帧，会使用 QUIC 异步连接指定的端口，如果连接成功，就会断开 HTTP/2 连接，改用新的 HTTP/3 收发数据。

领资料



小结

HTTP/3 综合了我们之前讲的所有技术（HTTP/1、SSL/TLS、HTTP/2），包含知识点很多，比如队头阻塞、0-RTT 握手、虚拟的“流”、多路复用，算得上是“集大成之作”，需要多下些功夫好好体会。

1. HTTP/3 基于 QUIC 协议，完全解决了“队头阻塞”问题，弱网环境下的表现会优于 HTTP/2；
2. QUIC 是一个新的传输层协议，建立在 UDP 之上，实现了可靠传输；
3. QUIC 内含了 TLS1.3，只能加密通信，支持 0-RTT 快速建连；
4. QUIC 的连接使用“不透明”的连接 ID，不绑定在“IP 地址 + 端口”上，支持“连接迁移”；
5. QUIC 的流与 HTTP/2 的流很相似，但分为双向流和单向流；
6. HTTP/3 没有指定默认端口号，需要用 HTTP/2 的扩展帧“Alt-Svc”来发现。

课下作业

1. IP 协议要比 UDP 协议省去 8 个字节的成本，也更通用，QUIC 为什么不构建在 IP 协议之上呢？
2. 说一说你理解的 QUIC、HTTP/3 的好处。
3. 对比一下 HTTP/3 和 HTTP/2 各自的流、帧，有什么相同点和不同点。

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 根据当前的标准草案，QUIC 已经不再是“Quick UDP Internet Connections”（快速

领资料




QUIC (Quick UDP Internet Connections) (快速 UDP 互联网连接) 的缩写了, “QUIC” 就是 “QUIC”。

- 02 QUIC 早期还有一个 “前向纠错” (Forward Error Correction) 的特性, 通过发送 xor 冗余数据来实现数据校验和恢复, 但目前此特性已经被 “搁置”, 也许会在以后的版本里出现。
- 03 QUIC 虽然是个传输层协议, 但它并不由操作系统内核实现, 而是运行在用户空间, 所以能够不受操作系统的限制, 快速迭代演化, 有点像 Intel 的 DPDK。
- 04 QUIC 里的包分为 “长包” 和 “短包” 两类, “长包” 的第一个字节高位是 1, 格式比较完整, 而短包只有目标连接 ID。
- 05 QUIC 和 HTTP/3 的变长编码使用第一个字节的高两位决定整数的长度, 最多是 8 个字节 (64 位), 所以最大值是 2^{62} 。
- 06 HTTP/3 的帧不再需要 END_HEADERS 标志位和 CONTINUATION 帧, 因为帧的长度足够

大 (2^{62}), 无论是多大的头都可以用一个帧传输。

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 10

 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 31 | 时代之风 (下) : HTTP/2内核剖析

下一篇 33 | 我应该迁移到HTTP/2吗?

学习推荐

JVM + NIO + Spring

各大厂面试题及知识点详解

限时免费 



领资料





许童童

2019-08-09

IP 协议要比 UDP 协议省去 8 个字节的成本，也更通用，QUIC 为什么不构建在 IP 协议之上呢？

直接利用UDP，兼容性好。

说一说你理解的 QUIC、HTTP/3 的好处。

彻底解决队头阻塞，用户态定义流量控制、拥塞避免等算法，优化慢启动、弱网、重建连接等问题。

对比一下 HTTP/3 和 HTTP/2 各自的流、帧，有什么相同点和不同点。

HTTP/3在QUIC层定义流、帧，真正解决队头阻塞，HTTP/2流、帧是在TCP层上抽象出的逻辑概念。

相同点是在逻辑理解上是基本一致的，流由帧组成，多个流可以并发传输互不影响。

作者回复: great。

共 2 条评论 >

18



lesserror

2019-12-25

老师，以下问题，麻烦回答一下，谢谢：

1.它使用自己的帧“接管”了 TLS 里的“记录”，握手消息、警报消息都不使用 TLS 记录，直接封装成 QUIC 的帧发送，省掉了一次开销。省掉的一次开销是什么？

2.解决了 HPACK 的“队头阻塞”问题。 没明白这句话。

作者回复:

1.省去了TLS封装步骤，也就没有了TLS帧头等相关信息，字节数等成本就减少了。

2.HPACK的字典需要双方传输同步，如果有的字典没有传过来就会阻塞，而HTTP/3解决了这个问题。



10

领资料



阿锋

2019-08-09

(1) http的队头阻塞，和tcp的队头阻塞，怎么理解？是由于tcp队头阻塞导致http队头阻塞，还是http本身的实现就会造成队头阻塞，还是都有。感觉有点模糊？

(2) 看完了QUIC，其流内部还是会产生队头阻塞，感觉没啥区别，QUIC内部还不是要实现tcp的重传那一套东西。QUIC没看出来比tcp好在哪里。

(3) 队头阻塞在http，tcp，流等这几个概念中是怎么理解和区分的，很迷惑。

作者回复:

1.队头阻塞在tcp和http层都存在，原因不同。http/2解决了http的队头阻塞，在http层是没有问题了，但在tcp层还有队头阻塞，所以会影响传输效率。

2.一个流就是一个请求响应，它阻塞不会影响其他流，所以不会发生队头阻塞。

3.QUIC有很多优于tcp的新特性，例如连接迁移、多路复用，加密。

4.队头阻塞确实不太好理解，可以再看看之前的课程，结合示意图来加深体会。



👍 5



-W.LI-

2019-08-09

- 1.传输层TCP和UDP就够了，在多加会提高复杂度，基于UDP向前兼容会好一些。
- 2.在传输层解决了队首阻塞，基于UDP协议，在网络拥堵的情况下，提高传输效率
- 3.http3在传输层基于UDP真正解决了队头阻塞。http2只是部分解决。

作者回复: good。



👍 5



moooofly

2019-08-31

我是不是可以这样理解，QUIC 之所以解决了队头阻塞，是基于UDP的乱序，无连接，以包为单位进行传出的特性，即当发生丢包时，当前流中对应的请求或应答就彻底“丢失”了，之后只需要通过在UDP基础实现的“可靠传输”功能，重传就好了，这样就避免了接收端死等尚未接收到的数据的“干着急”状态；

作者回复: 基本正确，udp的各个包之间没有依赖关系，所以就不会出现队头阻塞。

quic里的另一个关键是流概念，它和http/2的一样，把多个请求响应解耦，互不干扰。

这样在上层应用层和下层的传输层就都没有了队头阻塞，但因为丢包而重传该等还是会等，只是不会影响其他的流。

领资料





cake

2021-10-05

老师 请问下这句话怎么理解呢 HTTP/2 那样再去定义流

作者回复: 在quic里已经有流的概念了, 所以http/3就不需要再用一大段文字来定义流的格式、行为等, 直接引用quic就可以。

而http/2里的流概念是全新的, 下面的tcp、tls没有, 就必须花上很多篇幅去说明。



1



钱

2020-04-04

浏览器需要先用 HTTP/2 协议连接服务器, 然后服务器可以在启动 HTTP/2 连接后发送一个“Alt-Svc”帧, 包含一个“h3=host:port”的字符串, 告诉浏览器在另一个端点上提供等价的 HTTP/3 服务。

老师, 这里的意思是指HTTP/3包含了HTTP/2的这部分功能, 还是HTTP/3的使用必须依赖HTTP/2?

另外, QUIC中的包是一个完整的请求或响应报文? 否则多个包的内容才能组成一个完整的请求或响应报文, 必然也需要等待所有包都到齐了, 组装一下吧? 假如你一个包, 这个包得多大?

作者回复:

1.目前的http/3草案来看, 这个是http/3建立连接的要求, 也就是要从http/2升级。

2.quic的一个包里有多个帧, 多个帧组成一个报文, 包只是把帧整合在一起而已, 实际上用的还是帧。



1



Hills录

2020-02-27

课后1: QUIC 不基于 IP 协议, 是因为没有设备认识它

课后2: HTTP/3 端口不固定、内容天然加密、连接迁移等特性, 让互联网回归自由

作者回复: good。



1

领资料



chao

2019-10-22

老师，文中有一张包的结构图，Quic Package Payload 里面说『实际传输的数据是多个帧构成的流』，这里怎么理解呢？

是这样吗，Quic里面有帧、流、包的概念，流上传输的是帧，Quic是把多个流结合为包然后传递给UDP吗，因为每个流是一个消息，丢包的时候会一次丢多个消息吗

作者回复: quic里的流由多个帧组成，这些帧会组合在一个包里，也就是说一个包里会有多个流的帧，但不是一个包就包含了完整的流，而是会有多个包（里面多个帧）然后才能是一个完整的流。

丢包只会丢失部分帧，quic也只会重传这些丢失的帧。



1



mooolfly

2019-08-31

“下班回家，手机会自动由 4G 切换到 WiFi。这时 IP 地址会发生变化，TCP 就必须重新建立连接。而 QUIC 连接里的两端连接 ID 不会变，所以连接在“逻辑上”没有中断，它就可以在新的 IP 地址上继续使用之前的连接，消除重连的成本，实现连接的无缝迁移”，我觉得这里是不是应该强调一下，QUIC 是基于无连接概念的 UDP 协议，因此也就没有所谓的“中断”和“重连”概念，进而才能实现在新的 ip 地址上的无缝迁移；

作者回复: 在udp协议之上也可以实现有连接的协议，比如有的公司的自定义协议。

quic底层的udp提供了无连接的基础，但它是有连接的，连接迁移的功能还是quic自己实现的。



1



mooolfly

2019-08-31

一个小建议：既然 TLS1.3 是被“包含”在 QUIC 协议中的，那么文章中给出的 HTTP/3 协议栈图，就有点容易让人产生误会，图示给人的感觉是 QUIC 和 TLS1.3 是一个级别的对等存在，让人感觉 QPack 是基于 QUIC 的，而 Stream 是基于 TLS1.3 实现的

作者回复: 是的，quic和tls的关系比较复杂，不是直接的上下级关系，但画图为了“美观”给画成了这样，所以示意图只是“示意”，还是要结合文字来全面理解。



1

领资料



QQ怪

2019-08-09

老师能否分享下要更新换代http3，其上层的服务协议是否也要更新还是都能够兼容？

作者回复: 和http/2一样, http/3是完全兼容的, 因为语义还是保持不变。



1



功夫熊猫

2021-10-31

udp虽然可以节省时间和速度比tcp快, 但是如果传输的是那种很机密的东西的时候, 但是如何保证udp传输的数据是没有丢失的, (所以udp一般是传输视频, 图片之类的东西吧) 是换tcp还是对udp进行改装, 还是http/3有什么特殊的方法

作者回复: 在udp之上, quic有自己的丢包重传机制。

其实在tcp里也是有丢包的, 它自己来保证数据的完整可达, 类似可以去理解quic。



Jasmine

2021-10-11

老师, 文中QUIC Packet Header的目标连接\源连接ID的长度为什么单独拎出来画? 是ID长度单独花8bit来表示, 具体ID再是ID, 还是连接ID就是各4位来表示?

作者回复: 这里依据的quic版本已经过时了, 现在的正式版本格式和课程正文里的差距比较大, 现在长度是8位(1字节), 而且大小不得超过20。

可参考特别放送来进一步了解。



cake

2021-10-05

HTTP/2 那样再去定义流 这句话中的 "定义" 如何理解呢

作者回复: 在quic里已经有流的概念了, 所以http/3就不需要再用一大段文字来定义流的格式、行为等, 直接引用quic就可以。

而http/2里的流概念是全新的, 下面的tcp、tls没有, 就必须花上很多篇幅去说明。



Unknown element

2021-06-12

领资料



老师再问下课外小贴士里说QUIC和HTTP3的变长编码使用第一个字节的高两位决定整数的长度，这里的“整数”是指哪个整数呢

作者回复: 就是这个整数本身。

比如开头两位是00，那么这个整数就只是一个字节，剩下6位最大到63。

如果开头是11，那么整数就是8个字节，可用的位数就是 $64-2=62$ 位。



Unknown element

2021-06-12

gQUIC 混合了 UDP、TLS、HTTP，是一个应用层的协议。而 IETF 则对 gQUIC 做了“清理”，把应用部分分离出来，形成了 HTTP/3，原来的 UDP 部分“下放”到了传输层，所以 iQUIC 有时候也叫“QUIC-transport”。接下来要说的 QUIC 都是指 iQUIC，要记住，它与早期的 gQUIC 不同，是一个传输层的协议，和 TCP 是平级的

老师问下这一段最后为什么说iQUIC是传输层协议？本来gQUIC是应用层协议，去掉传输层部分后反而变成了传输层协议吗？

作者回复: gQUIC把udp/tls/http混合在一起，所以是应用层协议。而iQUIC只专注于数据传输，是gQUIC的udp+tls部分，http那部分变成了http/3。

可以去看rfc9000，里面写的很清楚。



Unknown element

2021-05-30

老师问下既然gQUIC 混合了 UDP、TLS、HTTP那可不可说gQUIC就是构建在ip协议之上呢

作者回复: 看来我说的不够严谨，我的意思是它基于udp，又加入了tls、http，把这些协议糅合在一起才是gQUIC。

但它不是基于ip协议的，因为它的底层是利用udp来发送数据，单位是一个个的udp包，而没有在ip之上新发明一个传说协议。

可以再看一下协议栈图，可能就会比较清楚了。

领资料





小童

2021-04-28

老师这个QUIC 是如何保证UDP的 可靠传输？ 还是没看明白。

作者回复: 底层的细节比较复杂，一下子也说不清楚，可以类比一下tcp和ip，quic就是在udp之上加了tcp的那套重传校验机制。



Rick

2021-03-20

请问连接迁移是如何做到的？毕竟它依赖于udp，而udp使用了ip/port。当一个连接的一端从一个ip/port转移到另外一个ip/port上的时候，怎么通知对端呢？需要使用QUIC的控制帧来完成吗？

作者回复: 这个跟控制端无关，因为在quic里标记连接的是连接id，所以两边都用这个id来标记连接，在quic这层是看不到ip+port的，只有id，这个道理和四元组是一样的。

至于具体如何做到，那就要看rfc了，不过我个人觉得如果是做应用，了解的太细不是很有必要。



领资料

