HTTP协议比较

Links

- QUIC/HTTP3 协议简析 - 又拍云
- HTTP/2 - MDN Web 文档术语表：Web 相关术语的定义 | MDN
- HTTP/2 - 维基百科，自由的百科全书
- HTTP 的发展 - HTTP | MDN

HTTP

HTTP 是基于文本的 (所有的通信都以纯文本的形式进行) 以及无状态的 (当前通信状态不会发现以前的通信状态)，该特性极大方便了在 www 上浏览网页的人。除此之外，HTTP 也可以用于构建服务器之间交互的 REST web 服务，以及使得网站内容更加动态化的 AJAX 请求。

由于 HTTP/0.9 协议的应用十分有限，浏览器和服务器迅速扩展内容使其用途更广：

- 协议版本信息现在会随着每个请求发送（`HTTP/1.0` 被追加到了 `GET` 行）。
- 状态码会在响应开始时发送，使浏览器能了解请求执行成功或失败，并相应调整行为（如更新或使用本地缓存）。
- 引入了 HTTP 标头的概念，无论是对于请求还是响应，允许传输元数据，使协议变得非常灵活，更具扩展性。
- 在新 HTTP 标头的帮助下，具备了传输除纯文本 HTML 文件以外其他类型文档的能力（凭借 `Content-Type` 标头）。

HTTP/1.1

HTTP/1.1 消除了大量歧义内容并引入了多项改进：

- 连接可以复用，节省了多次打开 TCP 连接加载网页文档资源的时间。
- 增加管线化技术，允许在第一个应答被完全发送之前就发送第二个请求，以降低通信延迟。
- 支持响应分块。（boundery request）
- 引入额外的缓存控制机制。
- 引入内容协商机制，包括语言、编码、类型等。并允许客户端和服务器之间约定以最合适的内容进行交换。
- 凭借 `Host` 标头，能够使不同域名配置在同一个 IP 地址的服务器上。

HTTP/2

HTTP/2 是 HTTP 网络协议的一个重要版本。HTTP / 2 的主要目标是通过启用完整的请求和响应多路复用来减少延迟，通过有效压缩 HTTP 标头字段来最小化协议开销，并增加对请求优先级和服务器推送的支持。

HTTP/2 不会修改 HTTP 协议的语义。HTTP 1.1 中的所有核心概念（例如 HTTP 方法，状态码，URI 和 headers）都得以保留。而是修改了 HTTP/2 数据在客户端和服务器之间的格式（帧）和传输方式，这两者都管理整个过程，并在新的框架层内隐藏了应用程序的复杂性。所以，所有现有的应用程序都可以不经修改地交付。

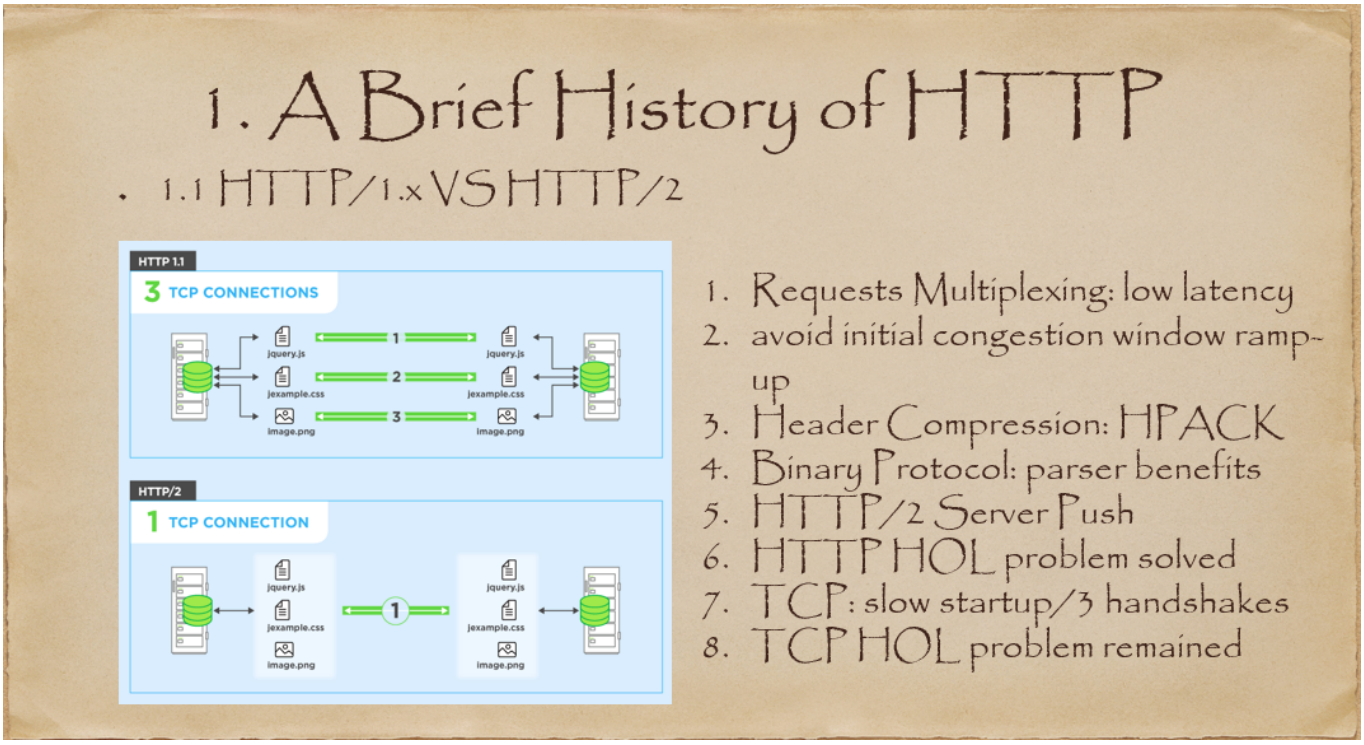
主要基于 SPDY （speedy）协议

SPDY（发音如英语：speedy），一种开放的网络传输协议，由Google开发，用来发送网页内容。基于传输控制协议（TCP）的应用层协议。SPDY也就是HTTP/2的前身。Google最早是在Chromium中提出的SPDY协议。被用于Google Chrome浏览器中来访问Google的SSL加密服务。SPDY并不是首字母缩略字，而仅仅是"speedy"的缩写。SPDY现为Google的商标。HTTP/2的关键功能主要来自SPDY技术，换言之，SPDY的成果被采纳而最终演变为HTTP/2。

主要改动：

- 使用二进制来传送数据
- 对 HTTP头字段 进行 数据压缩 （即HPACK算法）；
- HTTP/2服务端推送（Server Push）；
- 请求 流水线；
- 修复HTTP/1.0版本以来未修复的 队头阻塞 问题；
- 对数据传输采用 多路复用，让多个请求合并在同一 TCP 连接内。
- 支持现有的HTTP应用场景，包括桌面和移动设备浏览器、网络API、不同规格的 网络服务器和正向代理、反向代理 服务器软件、 防火墙和 CDN 等。
- 标准没有但是浏览器厂商都实现的 TSL 加密，废弃 h2c

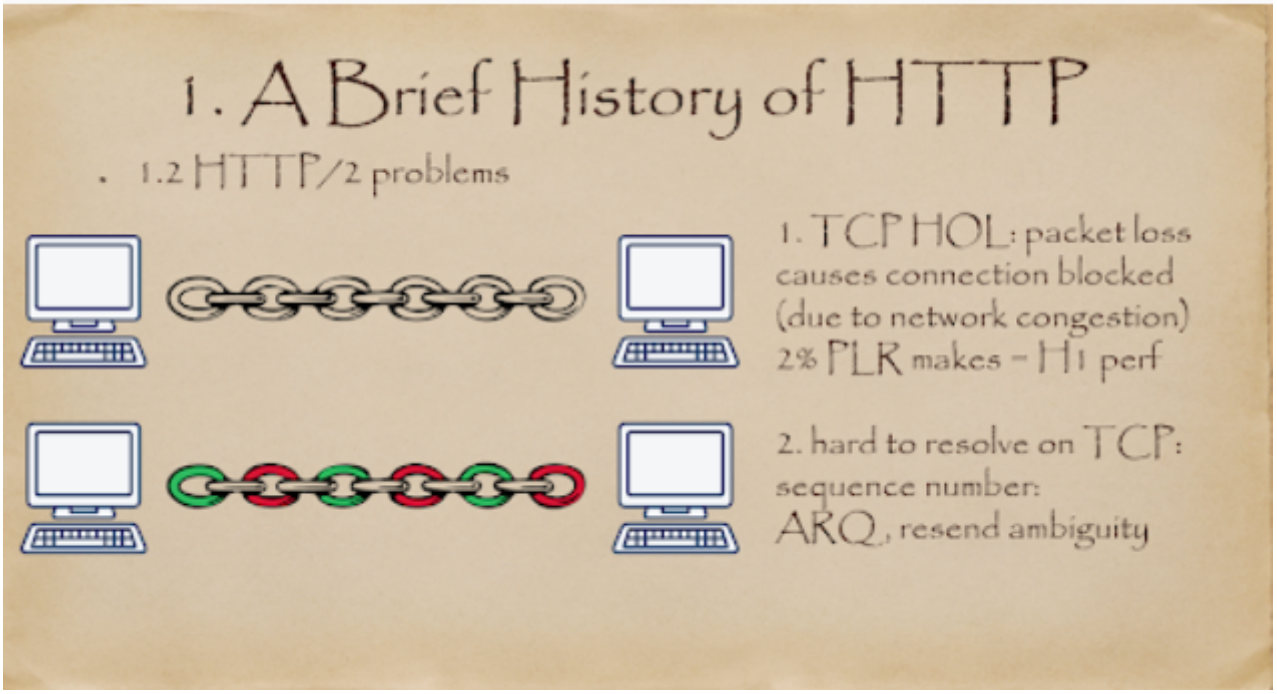
新特性



在HTTP/2的第一版草案（对SPDY协议的复刻）中，新增的性能改进不仅包括HTTP/1.1中已有的多路复用，修复队头阻塞问题，允许设置设定请求优先级，还包含了一个头部压缩算法（HPACK）。此外，HTTP/2采用了二进制而非明文来打包、传输客户端和服务器之间的数据。

帧、消息、流和TCP连接

> 解决了 HTTP 对头阻塞问题（HOL，Head-of-Line blocking），但是没有解决 TCP 对头阻塞问题



有别于HTTP/1.1在连接中的明文请求，HTTP/2与SPDY一样，将一个TCP连接分为若干个流（Stream），每个流中可以传输若干消息（Message），每个消息由若干最小的二进制帧（Frame）组成)。这也是HTTP/1.1与HTTP/2最大的区别所在。HTTP/2中，每个用户的操作行为被分配了一个**流编号（S**tream ID），这意味着用户与服务端之间创建了一个TCP通道；协议将每个请求分割为二进制的控制帧与数据帧部分，以便解析。这个举措在SPDY中的实践表明，相比HTTP/1.1，新页面加载可以加快11.81%到47.7%

HPACK 算法

HPACK算法是新引入HTTP/2的一个算法，用于对HTTP头部做压缩。其原理在于：

- 客户端与服务端根据RFC 7541的附录A，维护一份共同的静态字典（Static Table），其中包含了常见头部名及常见头部名称与值的组合的代码；
- 客户端和服务端根据先入先出的原则，维护一份可动态添加内容的共同动态字典（Dynamic Table）；
- 客户端和服务端根据RFC 7541的附录B，支持基于该静态哈夫曼码表的哈夫曼编码（Huffman Coding）。

服务器推送

网站为了使请求数减少，通常采用对页面上的图片、脚本进行 极简化 处理。但是，这一举措十分不方便，也不高效，依然需要诸多HTTP链接来加载页面和页面资源。

HTTP/2引入了**服务器推送**，即服务端向客户端发送比客户端请求更多的数据。这允许服务器直接提供浏览器渲染页面所需资源，而无须浏览器在收到、解析页面后再提起一轮请求，节约了加载时间。

h2c（HTTP/2 without TSL）的支持度

HTTP/2的设计本身允许非加密的HTTP协议，也允许使用 TLS 1.2 或更新版本协议进行加密。协议本身未要求必须使用加密，、Chrome、Safari、Opera、IE和Edge等）的开发者声明，他们只会实现通过TLS加密的HTTP/2协议，这使得经TLS加密的HTTP/2成为了事实上的强制标准，而h2c事实上被主流浏览器废弃。

总结

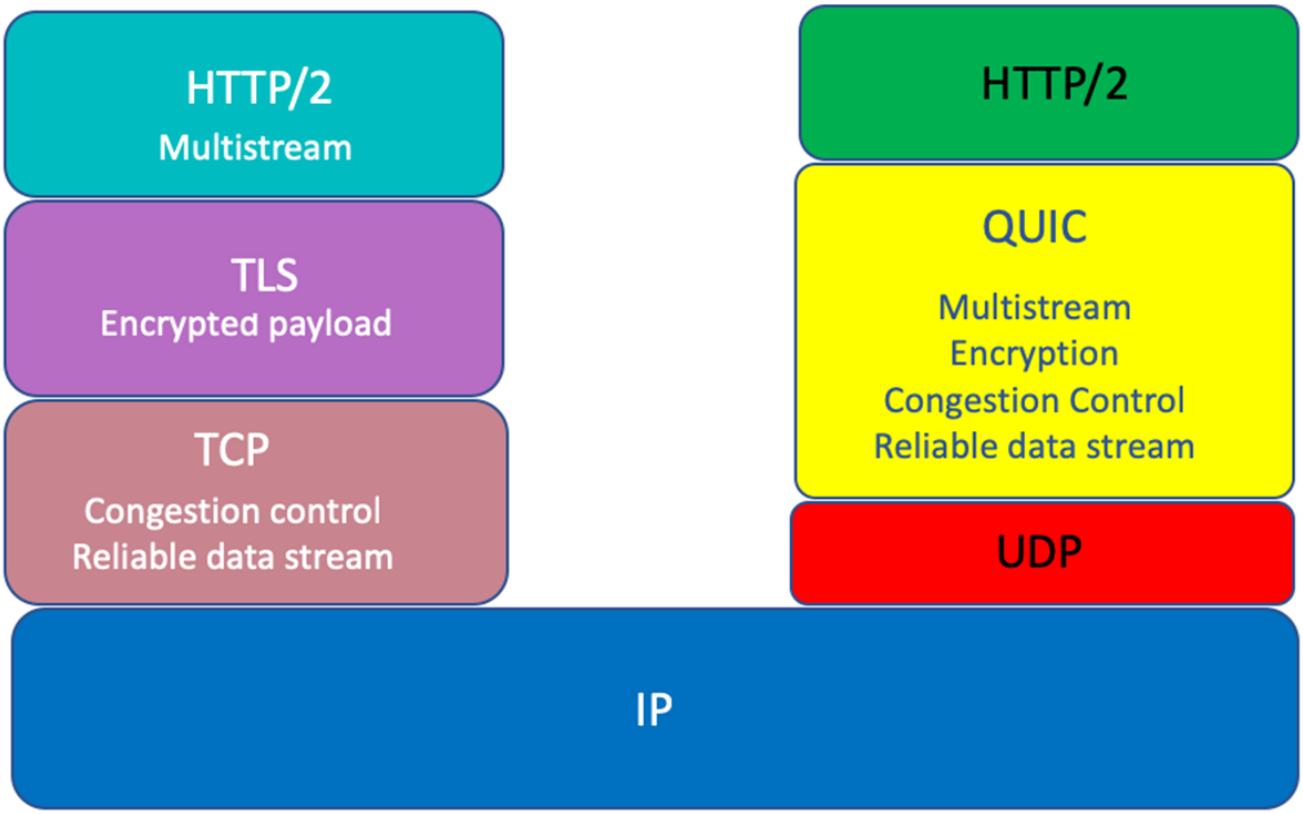
HTTP/2 与 HTTP/1 的区别

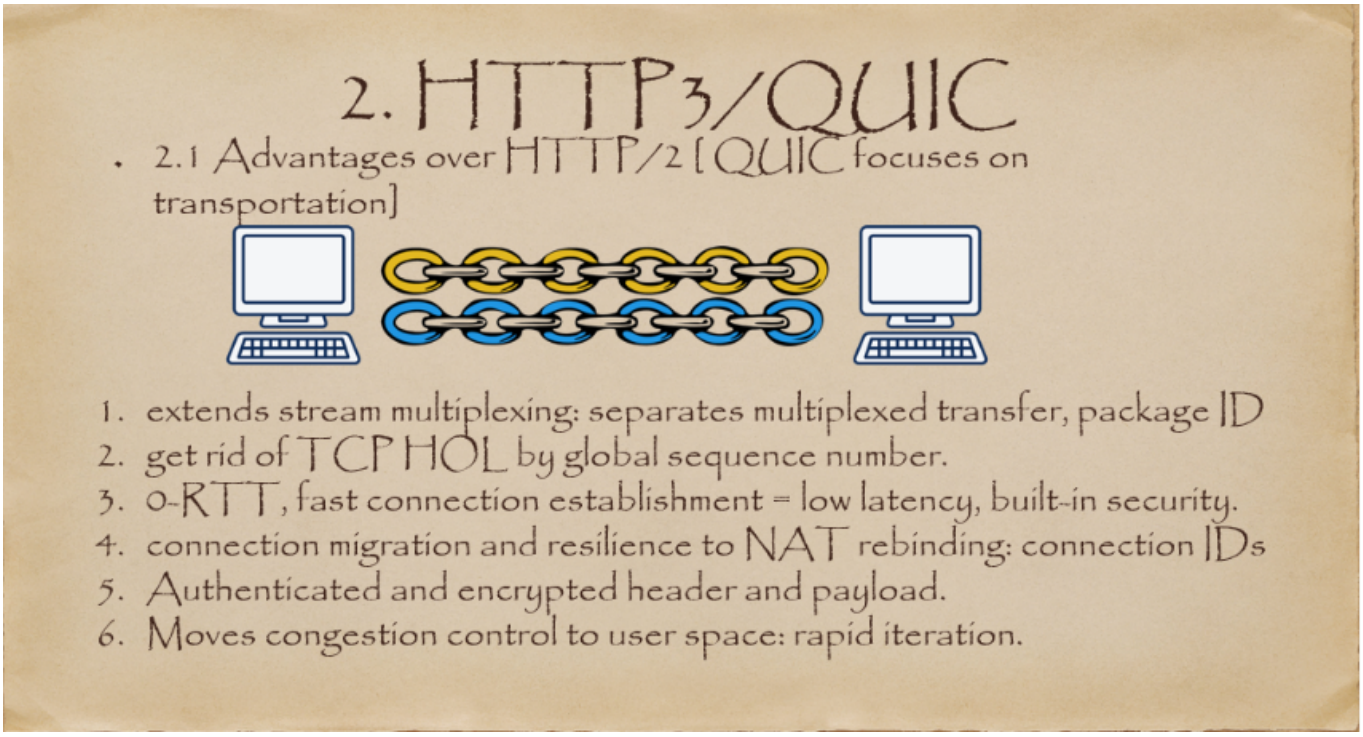
- 使用二进制传输客户端和服务器之间的数据
- 使用 HPACK 压缩 HTTP 请求头
- 使用 Pipeline（请求流水线）技术解决了 HTTP HOL （HTTP 对头阻塞）问题，服务器不必按照客户端发送请求的数据来回复请求。
- 使用 TSL 对数据进行加密（非标准）

- 多路复用，将多个请求合并在一个 TCP 连接中
- 服务端推送，其允许服务器在客户端缓存中填充数据

HTTP/3

- 弃用 TCP 协议，改为使用基于 UDP协议的 QUIC协议实现。解决了 TCP 对头阻塞问题。
- HTTP3 使用 stream 进一步扩展了 HTTP2 的多路复用。在 HTTP3 模式下，一般传输多少个文件就会产生对应数量的 stream。当这些文件中的其中一个发生丢包时，你只需要重传丢包文件的对应 stream 即可。
- HTTP3 更换成了兼容 HPACK 的 QPACK 压缩方案
- HTTP3 含有一个包括验证、加密、数据及负载的 built-in 的TLS安全机制。（标准？）
- 通过引入 Connection ID，使得 HTTP3 支持连接迁移以及 NAT 的重绑定。（在连接过程中切换网络，会话可以继续）
- QUIC 是应用层实现，通过用户空间来实现。这样做的好处就是不再需要等待内核更新可以实现很方便的进行快速迭代。





HTTP 流水线

HTTP流水线（英语：HTTP pipelining）是将多个HTTP请求（request）整批提交的技术，而在发送过程中不需先等待服务器的回应。

请求结果流水线使得 HTML 网页加载时间动态提升，特别是在具体有高延迟的连接环境下，如卫星上网。在宽带连接中，加速不是那么显著的，因为需要服务端要遵循 HTTP/1.1 协议，必须按照客户端发送的请求顺序来回复请求，这样整个连接还是先进先出的，队头阻塞（HOL blocking）可能会发生，造成延迟。未来的 HTTP/2.0 或者SPDY中的异步操作将会解决这个问题。因为它可能将多个 HTTP 请求填充在一个TCP数据包内，HTTP 流水线需要在网络上传输较少的 TCP 数据包，减少了网络负载。

流水线机制须透过永久连线（persistent connection）完成，并且只有 GET 和 HEAD 等要求可以进行流水线，非幂等的方法，例如POST将不会被管线化。连续的 GET 和 HEAD 请求总可以管线化的。一个连续的幂等请求，如 GET，HEAD，PUT，DELETE，是否可以被管线化取决于一连串请求是否依赖于其他的。此外，初次创建连线时也不应启动流水线机制，因为对方（服务器）不一定支持 HTTP/1.1 版本的协议。

HTTP 管线化同时依赖于客户端和服务器的支持。遵守 HTTP/1.1 的服务器支持管线化。这并不是意味着服务器需要提供管线化的回复，而只是要求在收到管线化的请求时候不会失败。

极简化

极简化（另称缩小化），在编程语言 (尤其是 JavaScript) 的范畴里，指的是在不影响功能的情况下，移除所有非功能性必要之源代码字元（如：空白、换行、注解、以及些许的区块办识子），因为虽然它们有助于提升源代码的易读性，但在实际运行时却不是必要的部分。

举以下的 JavaScript 为例子

```
var array = [];
for (var i = 0; i < 20; i++) {
  array[i] = i;
}
```

与下面极简化后的源代码等价

```
for(var a=[i=0];++i<20;a[i]=i);
```