HTTP

1. HTTP/1.0

HTTP/1.0 版的主要缺点是，每个TCP连接只能发送一个请求。发送数据完毕，连接就关闭，如果还要请求其他资源，就必须再新建一个连接。

TCP连接的新建成本很高，因为需要客户端和服务器三次握手，并且开始时发送速率较慢（slow start）。所以，HTTP 1.0版本的性能比较差。随着网页加载的外部资源越来越多，这个问题就愈发突出了。

为了解决这个问题，有些浏览器在请求时，用了一个非标准的Connection字段。`Connection: keep-alive` 这个字段要求服务器不要关闭TCP连接，以便其他请求复用。服务器同样回应这个字段 `Connection: keep-alive`。

一个可以复用的TCP连接就建立了，直到客户端或服务器主动关闭连接。但是，这不是标准字段，不同实现的行为可能不一致，因此不是根本的解决办法。

2. HTTP/1.1

2.1. 持久链接

1.1 版的最大变化，就是引入了持久连接（persistent connection），即TCP连接默认不关闭，可以被多个请求复用，不用声明 `Connection: keep-alive`。

客户端和服务器发现对方一段时间没有活动，就可以主动关闭连接。不过，规范的做法是，客户端在最后一个请求时，发送`Connection: close`，明确要求服务器关闭TCP连接 `Connection: close`。

目前，对于同一个域名，大多数浏览器允许同时建立6个持久连接。

2.2. 缺点

虽然1.1版允许复用TCP连接，但是同一个TCP连接里面，所有的数据通信是按次序进行的。服务器只有处理完一个回应，才会进行下一个回应。要是前面的回应特别慢，后面就会有许多请求排队等着。这称为"队头堵塞"（Head-of-line blocking）。

为了避免这个问题，只有两种方法：一是减少请求数，二是同时多开持久连接。这导致了很多的网页优化技巧，比如合并脚本和样式表、将图片嵌入CSS代码、域名分片（domain sharding）等等。如果HTTP协议设计得更好一些，这些额外的工作是可以避免的。

3. HTTP/2

HTTP/2 在 HTTP/1.1 有几处基本的不同：

- HTTP/2 是二进制协议而不是文本协议。不再可读，也不可无障碍的手动创建，改善的优化技术现在可被实施。
- 这是一个多路复用协议。并行的请求能在同一个链接中处理，移除了 HTTP/1.x 中顺序和阻塞的约束。
- 压缩了标头。因为标头在一系列请求中常常是相似的，其移除了重复和传输重复数据的成本。
- 其允许服务器在客户端缓存中填充数据，通过一个叫服务器推送的机制来提前请求。

3.1. 二进制帧

HTTP/1.1 版的头信息肯是文本（ASCII编码），数据体可以是文本，也可以是二进制。HTTP/2 则是一个彻底的二进制协议，头信息和数据体都是二进制，并且统称为"帧"（frame）：头信息帧和数据帧。

二进制协议的一个好处是，可以定义额外的帧。HTTP/2 定义了近十种帧，为将来的高级应用打好了基础。如果使用文本实现这种功能，解析数据将会变得非常麻烦，二进制解析则方便得多。

HTTP/2 将每个请求或回应的所有数据包，称为一个数据流（stream）。每个数据流都有一个独一无二的编号。数据包发送的时候，都必须标记数据流ID，用来区分它属于哪个数据流。另外还规定，**客户端发出的数据流，ID一律为奇数，服务器发出的，ID为偶数。**

3.2. 多路复用协议

多路复用解决了队头阻塞。

HTTP1.1使用的是TCP协议，并且为了节省资源，采用了长连接，长连接引入了队头阻塞的问题。HTTP2引入了流和帧，解决了HTTP层面上的队头阻塞。

举例来说，在一个TCP连接里面，服务器同时收到了A请求和B请求，于是先回应A请求，结果发现处理过程非常耗时，于是就发送A请求已经处理好的部分， 接着回应B请求，完成后，再发送A请求剩下的部分。

3.3. 头信息压缩

HTTP 协议不带有状态，每次请求都必须附上所有信息。所以，请求的很多字段都是重复的，比如Cookie和User Agent，一模一样的内容，每次请求都必须附带，这会浪费很多带宽，也影响速度。

HTTP1.1主要是对Body进行压缩，而头部却没有压缩。

HTTP/2 对这一点做了优化，引入了头信息压缩机制（header compression）。一方面，头信息使用gzip或compress压缩后再发送；另一方面，客户端和服务器同时维护一张头信息表，所有字段都会存入这个表，生成一个索引号，以后就不发送同样字段了，只发送索引号，这样就提高速度了。

3.4. 服务端推送

在 HTTP/1.1 中，只能客户端发起请求，服务器对请求进行响应。而在 HTTP/2 中，服务端可以主动给客户端推送必要的资源，以减少请求延迟时间。

比如当客户端向服务器请求一个 HTML 文件后，服务器除了将此 HTML 文件响应给客户端外，还可以提前主动将此 HTML 中所依赖的 JS 和 CSS 文件推送给客户端，这样客户端在解析 HTML 时，无需耗费额外的请求去得到相应的 JS 和 CSS 文件。

4. HTTP/3

基于 QUIC 的 HTTP 在传输层部分使用 QUIC 而不是 TCP。

QUIC 旨在为 HTTP 连接设计更低的延迟。类似于 HTTP/2，它是一个多路复用协议，但是 HTTP/2 通过单个 TCP 连接运行，所以在 TCP 层处理的数据包丢失检测和重传可以阻止所有流。QUIC 通过 UDP 运行多个流，并为每个流独立实现数据包丢失检测和重传，因此如果发生错误，只有该数据包中包含数据的流才会被阻止。