HTTP/2

HTTP/1.1性能问题：请求响应模型（队头阻塞），头部巨大且重复、并发连接耗时（每个并发连接都要经过TCP和TLS握手，以及TCP慢启动）、服务器不能主动推送。

兼容HTTP/1.1：

1.对URL没有改动，用户意识不到协议升级

2.只在应用层做了改变，把HTTP分解成语义和语法部分，语义与HTTP/1.1完全一致（请求方法、状态码、头字段等规则不变）在语法层面改变了HTTP报文的传输格式

头部压缩：

HTTP协议由Header + Body构成，HTTP/1.1可以在头字段Content-Encoding指定Body的压缩方式，但没有对Header的压缩方式

HTTP/1.1头部问题：

很多固定的字段，加起来几百上千字节，有必要压缩

大量请求和响应报文里很多字段都是重复的，有必要避免重复性

字段是ASCII编码，方便人类观察但效率低，有必要改成二进制编码

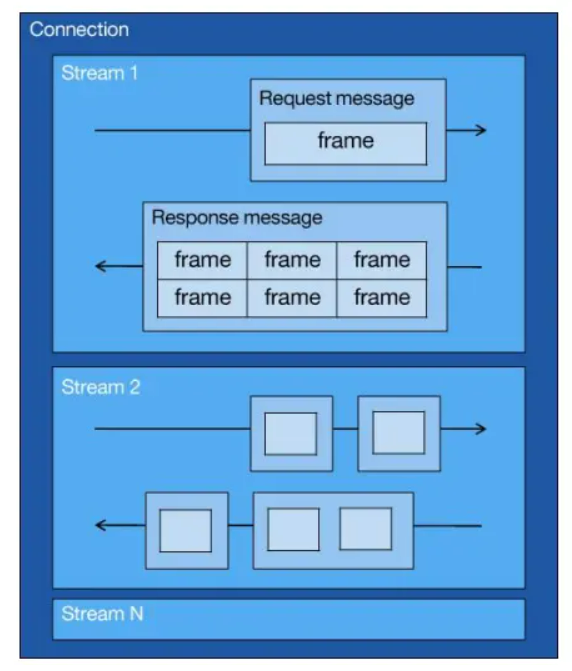
HTTP/2开发HPACK算法压缩头部，HPACK主要包括：静态字典、动态字典、Huffman编码（压缩算法），客户端和服务端都建立和维护字典，用长度较小的索引号表示重复字符串，再用Huffman编码压缩数据，50%-90%的压缩率。

静态表编码：HTTP/2为高频头部字符串建立了静态表（写入HTTP/2框架不会变化）

动态表编码：不在静态表的头部字符串要自行构建动态表，首次发送的字段被加入动态表中并建立Index，之后使用Index代替字符串。（使用动态表的前提是同一个连接上重复传输完全相同的HTTP头部）动态表达到一定大小时会自动断开，防止无限增长占用内存影响并发能力

二进制帧

并发传输：通过Stream设计，多个Stream复用一条TCP连接达到并发效果，解决了HTTP/1.1队头阻塞的问题（在TCP层面也会队头阻塞）



1个TCP连接里包含多个Stream，Stream里包含多个Message，Message里包含多个Frame，Frame是HTTP/2的最小单位。同一个HTTP请求与响应在一个Stream中

不同的Stream的帧可以乱序发送，同一Stream内部的帧必须有序

客户端和服务器双方都可以建立Stream，因此服务器可以主动推送，客户端的Stream为奇数号，服务端为偶数

总结：

HTTP/2在HTTP/1.1基础上：

压缩头部：静态表、动态表、Huffman编码

Stream并发

服务器主动推送

存在问题：TCP层面的队头阻塞，TCP层是字节流协议，必须保证收到的字节数据是完成连续的，这样内核才会将缓冲区数据返回给HTTP应用，前1个字节数据没有到达时后收到的字节数据只能放在缓冲区里，等这个字节的数据到达HTTP/2才能从内核拿到数据