应用层: HTTP

HTTP协议

组成

请求行

请求方法 | 空格 | 请求路径(URI) | 空格 | HTTP版本号

如GET /api/v1/user HTTP/1.1

请求方法

- GET：用于获取资源
- POST：用于创建资源，或者提交表单
- PUT：用于对资源的覆盖修改，一种完全替换
- DELETE：用于删除资源
- HEAD：用于获取响应头
- PATCH：PUT方法的补充，用于修改资源的部分属性
- OPTIONS：获取目标资源的通信选项，通常用来作为查看服务器的性能。但是在前端跨域请求中，OPTIONS请求也会用来作为嗅探请求，用于判断是否允许跨域

请求方法与其对应的行为仅仅只是一个规范，或者说通用约定
具体的服务端可不遵守，在系统内部逻辑自恰即可

请求路径(URI)：用于标识请求的资源路径

HTTP版本号：目前来讲，HTTP/1.1 是使用最为广泛的HTTP版本，但是由于HTTP/2 的便利性以及较高的性能，已经开始逐渐地取代HTTP/1.1

请求头部

请求头部用于控制请求方与服务端之间的通信协议，如Content-Type用于控制发送的正文格式，Accept-Charset表示客户端可以接受的字符集

底层协议

HTTP协议建立在TCP协议之上，由协议保证数据包的可靠送达

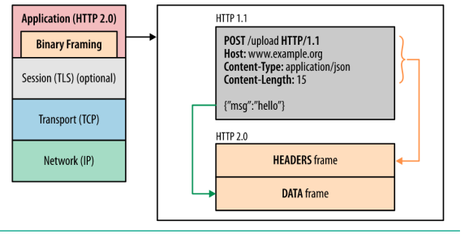
HTTP/2

HTTP/2协议在原有的HTTP协议版本之上进行了大幅度的修改，其传输效率大大提升，并且能够有效地降低服务端的资源消耗

由文本传输改变为二进制数据传输

在HTTP/1中，请求头与数据通过换行符作为分隔符，使用文本进行数据传输

在HTTP/2中，将请求头与数据采用二进制格式，并且将其拆分成更小的帧(frame)进行数据传输，并采用HPACK对请求头进行压缩



传输数据更少

新的二进制分帧机制改变了客户端与服务器之间交换数据的方式

数据流：TCP连接中的双向字节流，可包含一条或多条消息

消息：与逻辑请求或响应消息对应的完整的一系列帧

帧：HTTP/2 通信的最小单位，每个帧都包含帧头，至少也会标识出当前帧所属的数据流

帧可在数据流中乱序传输，抵达目的地后重新组装

解决了HTTP/1.1中头部阻塞问题

HTTP2中的连接为长连接，针对不同的来源建立新的连接，而不是每一个新的请求就建立TCP连接

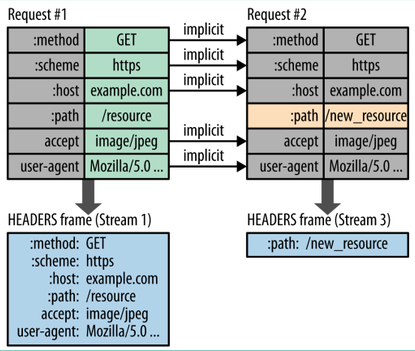
降低了服务端资源消耗，移除了建立多个TCP连接的性能损耗

HTTP/2的HPACK

HTTP/2对请求头部字段的压缩可谓是为HTTP协议带来了巨大的性能提升：在HTTP/1.1中，请求头始终使用纯文本传输，并且每次请求时，都会携带完整的请求头，尽管多个请求头字段存在大量重复

HPACK则会对请求头进行压缩，以及缓存部分请求头以便进行复用

- 霍夫曼压缩：利用霍夫曼压缩技术，在传输时对各个值进行压缩，以减少传输数量
- 标头索引列表：在stream的初次发送时，客户端将传输完整的请求头部字段，并建立索引表。服务端收到消息后建立相同的索引表，以便后续的复用



Google所提供的gRPC在底层同样使用了HTTP/2数据传输，较之于restful API能够为微服务提供更高的性能