1. HTTP基本概念

HTTP：HyperText Transfer Protocol 超文本传输协议：在两点间（可以是服务器到本地也可以服务器到服务器）传输文字、图片、音视频等超文本（超越文本的文本，关键有超链接）数据的约定和规范

1. HTTP常见的状态码



[200 OK] 最常见，表示一切正常，非HEAD请求服务器返回的响应头有body数据

[204 No Content] 与200相比响应头没有body数据

[206 Partial Content] 应用于分块下载或断点续传，表示响应返回的body数据不是全部

[301 Moved Permanently] 永久重定向，请求资源不存在，需改用新URL访问

[302 Found] 临时重定向，请求资源存在但暂时需要用另一个URL访问

301和302都会在响应头里使用字段Location指明后续要跳转的URL，浏览器会自动重定向

[304 Not Modified] 表示资源未修改，重定向已存在的缓冲文件

[400 Bad Request] 请求报文错误，笼统错误

[403 Forbidden] 服务器禁止访问资源，不是客户端的错

[404 Not Found] 请求的资源在服务器上不存在或未找到

[500 Internal Server Error] 服务器出错，笼统错误

[501 Not Implemented] 客户端请求的功能还不支持（coming soon的意思）

[502 Bad Gateway] 服务器作为网关或代理时返回的错误码，表示服务器自身正常但访问后端服务器发生了错误

[503 Service Unavailable] 服务器正忙

1. HTTP常见字段
2. Host字段：客户端发送请求时用来指定服务器的域名，有了Host字段可以将请求发往同一台服务器上的不同网站 eg：Host:www.baidu.com
3. Content-Length字段：服务器返回数据时用来表明本次回应的数据长度，HTTP协议通过设置回车换行作为HTTP header的边界，通过Content-Length字段作为HTTP body的边界都是为了解决TCP粘包问题（一个报文对应的不一定是一个用户消息，两个消息的某个部分被分到同一个TCP报文时就是粘包问题）
4. Connection 字段：用于客户端要求服务器使用HTTP长连接机制（任意一端没有明确提出断开连接则保持TCP连接状态）以便其他请求复用。HTTP/1.1默认都是长连接，为了兼容老版本HTTP需要指定Connection首部字段的值为Keep-Alive。

这里的Keep-Alive和TCP的Keepalive不是一个东西：

HTTP的Keep-Alive是应用层（用户态）实现的，又叫HTTP长连接，可以使用同一个TCP连接来发送和接收多个HTTP请求或应答，减少了HTTP短链接带来的多次TCP连接建立和释放的开销

TCP的Keepalive是TCP层（内核态）实现的，又叫TCP保活机制，当客户端和服务端长达一定时间没有进行数据交互时，内核为了确保该连接是否有效会发送探测报文来检测对方是否在线，从而决定是否要关闭该连接。

1. Content-Type字段：用于服务器回应时告知客户端本次数据的格式，客户端请求时可以用Accept字段声明自己可以接收的数据格式
2. Content-Encoding字段：说明数据的压缩方法，表示服务器返回的数据使用的压缩格式，客户端用Accept-Encoding指定自己可以接收的压缩方法
3. GET和POST

GET的语义是从服务器获取指定的资源，GET 请求的参数位置一般是写在 URL 中，只支持ASCII，浏览器对URL长度有限制

POST的语义是根据请求负荷（报文body）对指定的资源做出处理，POST请求携带数据的位置一般是写在报文body，浏览器对body大小无限制。eg：留言后提交，浏览器会执行POST请求，把留言的内容放到报文body里，拼接POST请求头，通过TCP协议发送

安全和幂等：

安全：请求方法不会破坏服务器上的资源

幂等：多次执行相同的操作结果都是相同的

GET方法安全且幂等，只读操作不会修改服务器上的资源且每次的结果都相同

POST方法不安全不等，新增或提交操作会修改服务器上的资源，不安全，多次提交会创建多个资源，不幂等（从RFC规范定义的语义来分析，实际开发也有用GET新增删除数据的，这种不在考虑范围）

1. HTTP缓存技术：

缓存实现方式：

强制缓存：浏览器判断缓存没有过期就直接使用浏览器本地的缓存

使用Cache-Control（相对时间，优先级高于Expires），Expires（绝对时间）控制

协商缓存：与服务端协商后判断是否使用本地缓存

使用If-Modified-Since和Last-Modified控制

使用If-None-Match和ETag控制

ETag优先级更高：

没有修改文件内容文件的最后修改时间也可能会改变

If-Modified-Since只支持秒级，ETag可以在1秒内刷新多次

有的服务器不能精确获得文件的最后修改时间

协商缓存要配合强制缓存使用，只用强制缓存未命中才会使用协商缓存

1. HTTP特性

HTTP/1.1优点：

简单、灵活易扩展、跨平台

HTTP/1.1缺点：

无状态：效率高但处理关联操作时很麻烦（可以用Cookie技术解决，在报文中写入Cookie信息来控制客户端状态）

明文传输：方便调试但是不安全

不安全：信息被窃取、篡改或伪装

HTTP/1.1的性能：基于TCP/IP的请求-应答通信模式

长连接：使用一个TCP连接完成请求和响应，双方只要不明确断开连接则保持TCP连接状态

管道网络传输成为可能：可以发送多个请求，不用等响应返回（解决了请求的队头阻塞），服务器必须按照接收的请求顺序发送响应（响应队头阻塞），默认不开启

队头阻塞：一个请求因为某种原因被阻塞时后面排队的所有请求也被阻塞

1. HTTP与HTTPS

HTTP与HTTPS的区别：

HTTP明文传输不安全，HTTPS在应用层和TCP层之间加入了SSL/TLS安全协议，报文加密传输

HTTP在TCP三次握手连接后进行报文传输，HTTPS还需要TLS握手

HTTP默认80端口，HTTPS默认443

HTTPS需要向CA申请数字证书验证身份

HTTPS解决了HTTP的问题：

窃听问题：信息加密

混合加密：使用非对称加密交互会话密钥，使用会话密钥对称加密报文（非对称加密速度慢，对称加密无法做到安全交换密钥）

篡改问题：校验机制

摘要算法（计算内容的哈希值（唯一的无法逆向））+数字签名（私钥加密公钥解密来确认身份，对内容的哈希值加密）

伪造问题：身份证书

向CA注册数字证书：

签发过程：将持有者公钥、颁发者等信息打包，计算hash值，用CA私钥加密hash值形成数字证书

验证过程：使用CA公钥对hash值解密，与自己计算的哈希值比较

证书信任链问题

HTTPS如何建立连接：

SSL/TLS协议基本流程：

客户端向服务端索要并验证服务器公钥

双方协商生成会话密钥

双方使用会话密钥加密通信

TLS握手阶段涉及四次通信，常用密钥交换算法RSA和ECDHE

基于RSA的握手过程：

ClientHello：随机数、TLS版本、支持的密码套件列表

SeverHello：随机数、TLS版本、密码套件（如RSA）、服务器证书

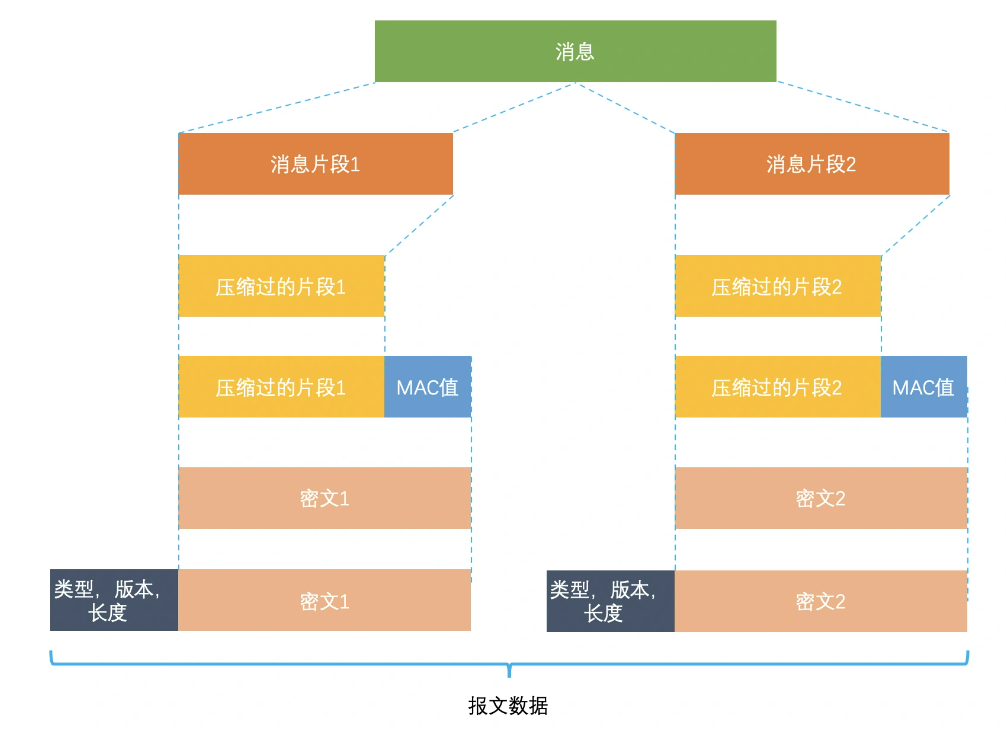
客户端回应：从数字证书中取出服务器公钥，加密报文发送：pre-master key、加密通信算法改变通知、客户端握手结束通知。使用Client Random、Sever Random、pre-master key和双方协商的加密算法生成会话密钥

服务器回应：收到pre-master key解密并计算会话密钥，向客户端发送加密通信算法改变通知、服务器握手结束通知

HTTPS应用数据如何保证完整

握手协议：TLS握手过程，生成对称密钥

记录协议：负责保护应用程序数据并验证其完整性和来源



流程：分割片段、加上消息认证码（MAC值，通过哈希算法生成）、通过对称密钥加密、加上数据类型、版本号、压缩后长度组成的报头

记录协议完成后最终的报文数据通过TCP层传输

HTTPS一定安全吗？

客户端与中间人服务器进行TLS握手时伪造的证书是能被识别出非法的，不信任非法证书，不被恶意导入伪造的根证书就是安全的

为什么抓包工具能截取HTTPS的数据包？

和中间人的工作原理一致，在客户端安装根证书认证抓包工具

如何避免被中间人抓取数据？

不信任非法证书的网站，保护电脑的安全（没有病毒）

HTTPS双向认证（一般是单向认证的（客户端验证服务器，服务器不验证客户端）），服务端发现客户端不可信或客户端发现服务端不可信就中止通信

1. HTTP/1.1，HTTP/2，HTTP/3演变

HTTP/1.1与HTTP/1.0相比提高了什么性能：

长连接、支持管道网络传输（默认不开启）

HTTP/1.1性能瓶颈：

压缩Body但不压缩请求/响应头部Header，首部信息越多延迟越大

每次互相发送相同的首部造成浪费

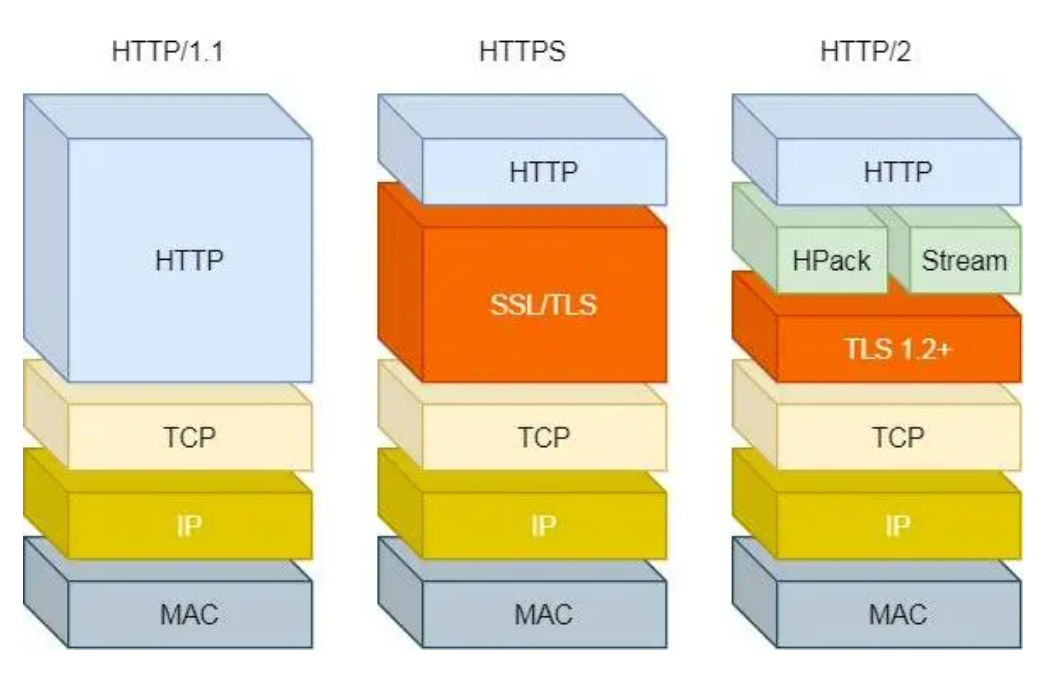
队头阻塞

没有请求优先级控制

请求只能从客户端开始，服务器被动响应

HTTP/2做了什么优化

HTTP/2基于HTTPS，安全性得到保障



相比HTTP/1.1性能上的改进：头部压缩、二进制格式、并发传输、服务器推送

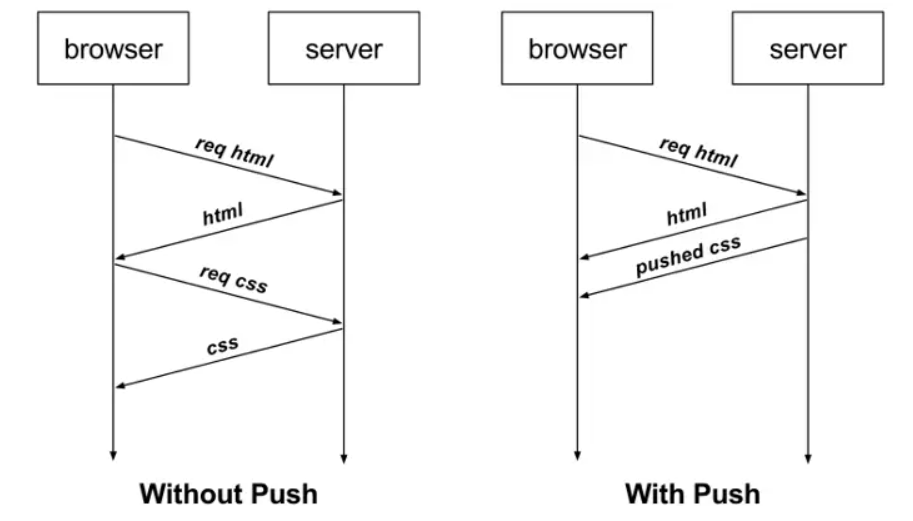
头部压缩：如果同时发出多个请求的头是一样或相似的，协议会消除重复的部分

HPACK算法：在客户端和服务端同时维护一张头信息表，所有字段都存入并生成一个索引号，使用索引号代替字段

二进制格式：HTTP/1.1报文为纯文本形式，HTTP/2使用二进制格式，计算机无需将明文转换成二进制

并发传输：使用Stream概念，多个Stream复用在一条TCP连接，使用独一无二的Stream ID来区分不同的HTTP请求，接收端可以通过Stream ID有序组装，不同的Stream帧可以乱序发送，解决了队头阻塞问题（在HTTP层面，在TCP层面还是存在）

服务器推送：服务端可以主动向客户端发送信息，如



HTTP/2缺陷：

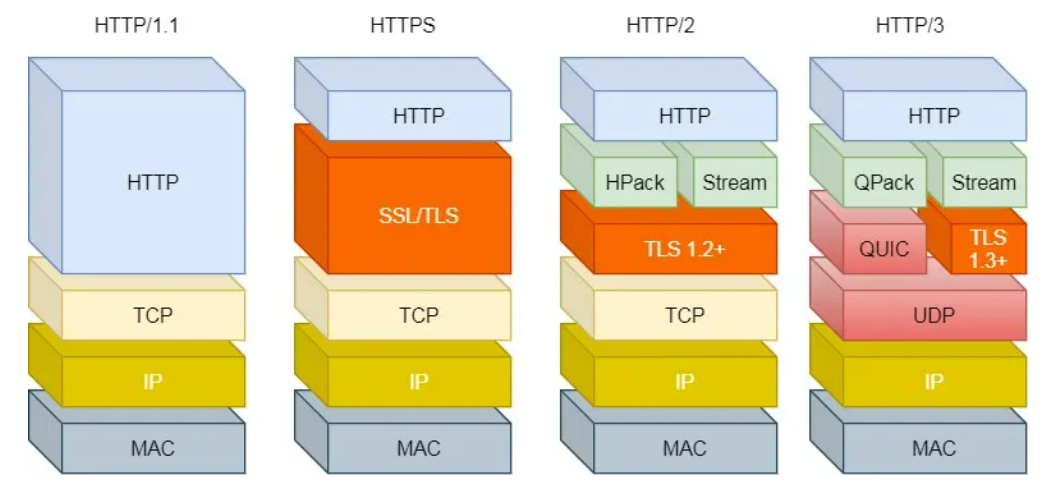
基于TCP协议传输，TCP是字节流协议，TCP层必须保证收到的字节数据是完整且连续的，这是内核才会将缓冲区里的数据返回给HTTP应用，前一个字节数据没有到达时后面的字节数据只能放在内核缓冲区里但HTTP/2应用层不能从内核中拿到数据，只能等前一个字节数据到达（一旦发生丢包现象就会触发TCP重传机制，所有的HTTP请求都必须等待这个丢了的包重传）

HTTP/3做了哪些优化

HTTP/1.1的管道网络通信解决了请求的队头阻塞，没有解决响应的队头阻塞

HTTP/2并发传输解决了HTTP的队头阻塞，但存在TCP层队头阻塞

HTTP/3将TCP协议改成了UDP



基于UDP的QUIC协议可以实现TCP的可靠传输

QUIC特点：无队头阻塞、更快的连接建立、连接迁移

无队头阻塞：类似HTTP/2 Stream多路复用，当某个流发生丢包时只会阻塞这个流，其他流不受影响（QUIC的多个Stream之间并没有依赖）

更快的连接建立：HTTP/1和HTTP/2协议中TCP和TLS是分层的，先TCP握手再TLS握手，QUIC内部包含了TLS

连接迁移：基于TCP的HTTP协议通过四元组（源IP、源端口、目的IP、目的端口）确定一条TCP连接，移动设备网络切换（如4G切换到WIFI）IP地址改变，需要断开并重新建立连接，QUIC协议通过连接ID标记通信两端，即使IP地址改变，保有上下文信息就可以使用原连接，达到了连接迁移功能

QUIC是一个在UDP上的伪TCP+TLS+HTTP/2的多路复用协议