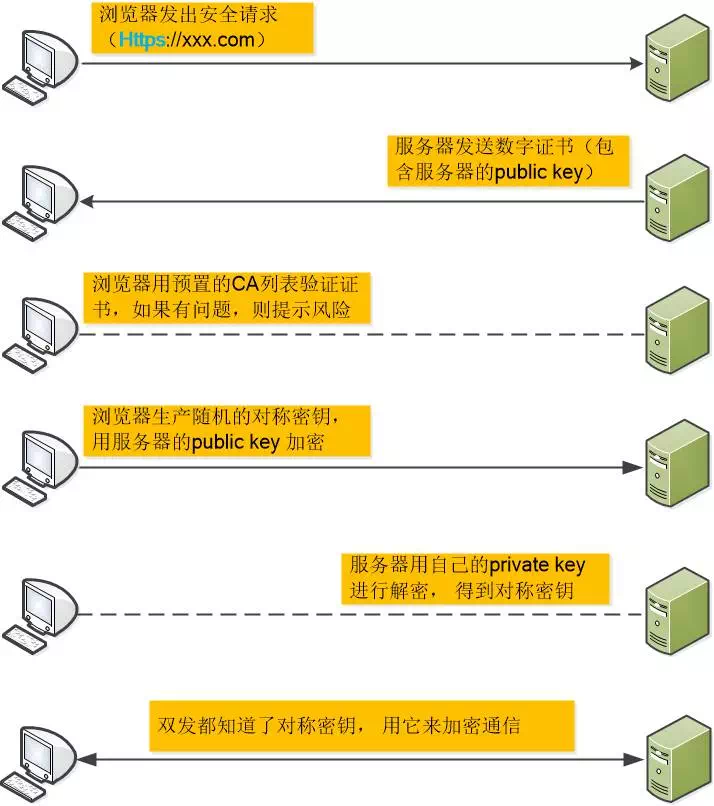
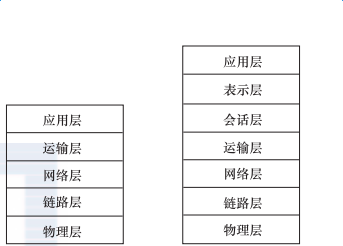
————————————————https————————————————————



https的ssl与tls的安全层协议位于应用层与传输层之间。

———————————————————五层协议————————————————



**网络攻击（CSRF、XSS）**

Csrf:中文名称：跨站请求伪造，也被称为：one click attack/session riding,

利用用户其他网站的cookie进行请求

防止：不用get用post，必要请求加token

Xss

恶意攻击者往Web页面里插入恶意Script代码，当用户浏览该页之时，嵌入其中Web里面的Script代码会被执行，从而达到恶意攻击用户的目的

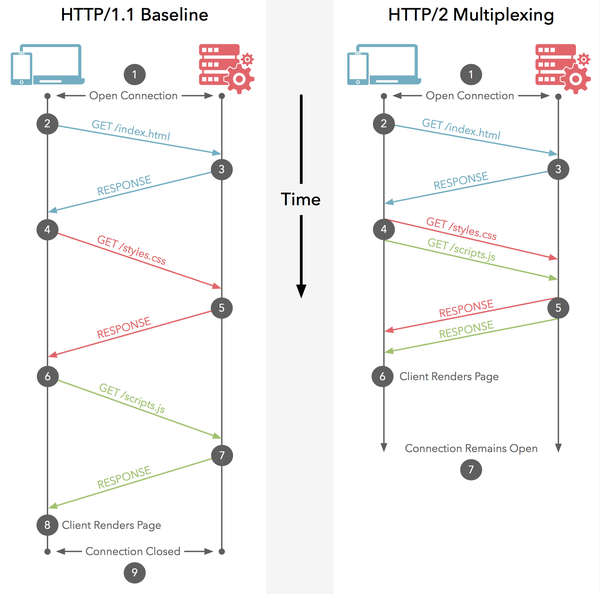
防止：适当过滤

Ddos:泛洪攻击

http2.0/1.0

2.0: 多路复用允许同时通过单一的 HTTP/2 连接发起多重的请求-响应消息。 多流并行减少创建多个tcp链接。

1.0: 在 HTTP/1.1 协议中 「浏览器客户端在同一时间，针对同一域名下的请求有一定数量限制。超过限制数目的请求会被阻塞」。(多个cdn的原因)



2.0:消息头压缩，减少ippackage，为了减少这块的开销并提升性能， HTTP/2会压缩这些首部： HTTP/2在客户端和服务器端使用“首部表”来跟踪和存储之前发送的键－值对，对于相同的数据，不再通过每次请求和响应发送； 首部表在HTTP/2的连接存续期内始终存在，由客户端和服务器共同渐进地更新; 每个新的首部键－值对要么被追加到当前表的末尾，要么替换表中之前的值。

2.0：服务端推送，将客户端需要的内容推送到客户端缓存，一次避免往返

2.0：二进制分帧，数据流以消息的形式发送，而消息由一个或多个帧组成，帧可以在数据流上乱序发送，然后再根据每个帧首部的流标识符重新组装。二进制分帧是HTTP/2的基石，其他优化都是在这一基础上来实现的。

幂等性（规范）

多次请求返回相同结果，同一个请求，发送一次和发送N次效果是一样的。

HTTP GET方法用于获取资源，不应有副作用，所以是幂等的。HTTP DELETE方法用于删除资源，有副作用，但它应该满足幂等性。HTTP POST方法用于创建资源，所对应的URI并非创建的资源本身，而是去执行创建动作的操作者，有副作用，不满足幂等性。HTTP PUT方法用于创建或更新操作，所对应的URI是要创建或更新的资源本身，有副作用，它应该满足幂等性。

有副作用的操作，需要有一个全局唯一标识（像作业帮内部id生成器），然后根据这个标识去check该操作是否已经被执行。比如支付页面就应该同步渲染id

get/post 明文/非明文

|  |  |  |
| --- | --- | --- |
| 方法 | 幂等性 | 安全 |
| OPTIONS | yes | yes |
| GET | yes | yes |
| HEAD | yes | yes |
| PUT | yes | no |
| POST | no | no |
| DELETE | yes | no |
| PATCH | no | no |

安全方法会被缓存

http

**requestheader**:

path:method:cookie:ua:referrer:path

**responseheader**:

content-type: date: etag: status:200 set-cookie

**端口：**

Http/https/:80/433

 TCP/IP

* 三次握手、四次挥手
* 拥塞控制（过程、阈值）
* 流量控制与滑动窗口
* TCP与UDP比较
* 子网划分（一般只有笔试有）
*  (B)IO/NIO/AIO
* 三者原理，各个语言是怎么实现的
* Netty
* Linux内核select poll epoll

什么时候用udp？为什么tcp不行？

Vmstat

一个url到页面全过程（让我能说多详细说多详细，最好从OSI七层的每一层去扩展）http的请求头格式（这个真的记不太清了，只说了几个有印象的标志位）