https中TLS的四次握手

使用的是RSA和ECDHE算法。

详细流程如下

1. 客户端->服务器 ***ClientHello***

   客户端的TLS版本号+客户端生成的随机数（***ClientRandom***）+支持的密码套件（RSA等）

2. 服务器->客户端 ***ServerHello***

   确认TLS版本号浏览器支持，否则断开，

   生成服务器端随机数（***ServerRandom***）

   确认密码套件

   服务器的数字证书

3. 客户端回应

   确认数字证书，如果正确就拿出服务器的公钥，加密发信息

   加密发送一个随机数（***pre-master key***）

   给个通知，后续都要加密通信了

   通知服务器端结束，把之前的所有消息做个摘要再加密发一遍，用来供服务器端校验

4. 服务器回应

   接收到***pre-master key***之后，通过协商的加密算法，**计算出本次通信的会话秘钥**

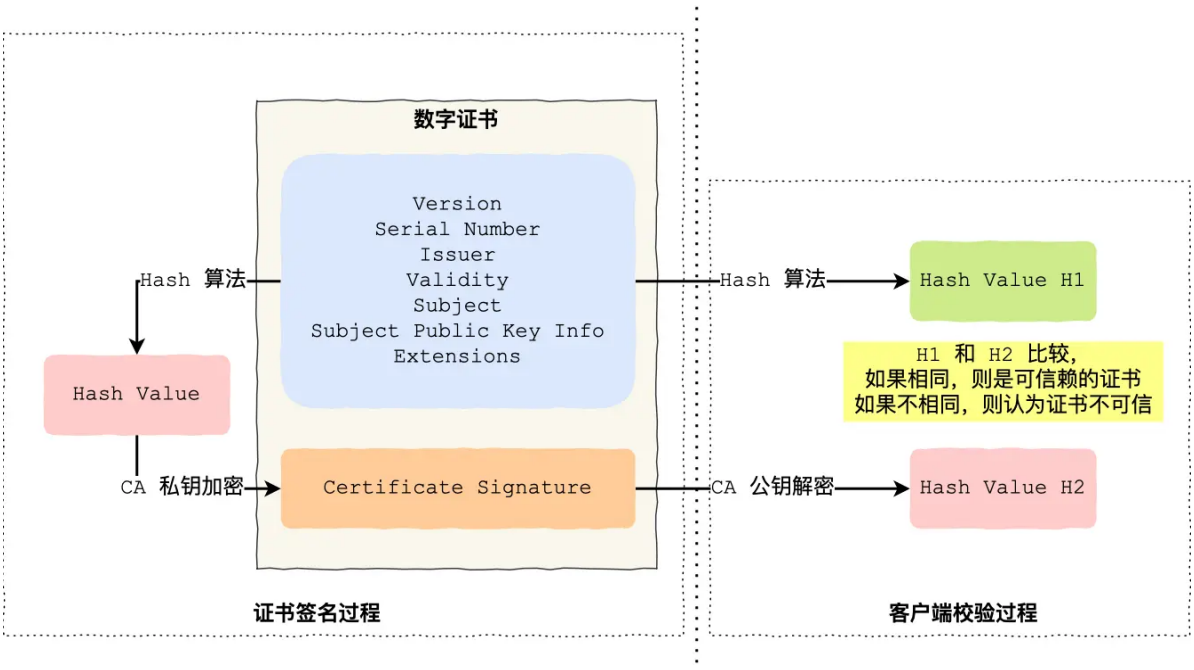
   再发送

   加密通信算法改变通知，后续使用新的会话秘钥

   通知客户端结束，把之前的所有消息做个摘要加密发送一遍，校验一下

RSA的问题：如果服务器端的私钥被破解了，之前所截获的所有加密信息都会被解密，前向安全问题。

核对数字签名部分



CA 签发证书的过程，如上图左边部分：

- 首先 CA 会把持有者的公钥、用途、颁发者、有效时间等信息打成一个包，然后对这些信息进行 Hash 计算，得到一个 Hash 值；
- 然后 CA 会使用自己的私钥将该 Hash 值加密，生成 Certificate Signature，也就是 CA 对证书做了签名；
- 最后将 Certificate Signature 添加在文件证书上，形成数字证书；

客户端校验服务端的数字证书的过程，如上图右边部分：

- 首先客户端会使用同样的 Hash 算法获取该证书的 Hash 值 H1；
- 通常浏览器和操作系统中集成了 CA 的公钥信息，浏览器收到证书后可以使用 CA 的公钥解密 Certificate Signature 内容，得到一个 Hash 值 H2 ；
- 最后比较 H1 和 H2，如果值相同，则为可信赖的证书，否则则认为证书不可信

http的数据完整性怎么保证?

消息先压缩，再加MAC消息验证码的头，它摘要算法生成保证数据的完整性，并进行数据认证。

然后对整个片段进行加密。再传输。

双向认证。

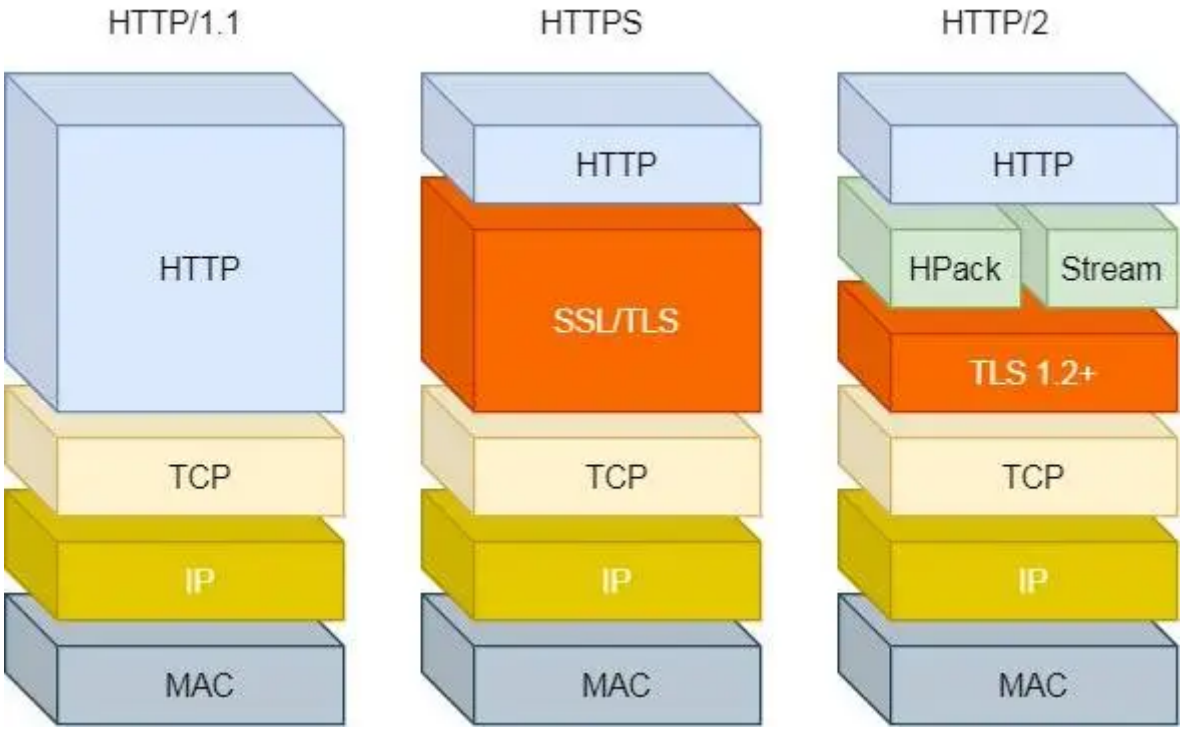
HTTP/1.1 HTTP/2 HTTP/3

HTTP1.1:比1.0多了，长连接+管道传输

HTTP1.1的问题:

- 请求头不压缩，首部信息发送延迟大，只能压缩body
- 服务器按顺序响应，队头阻塞
- 无优先级控制
- 只能从客户端请求服务器端

HTTP/2



是基于HTTPS的。

- 头部压缩

  HPACK算法，在服务端和客户端维护一个头信息的列表，不用每次传

- 二进制格式

  不用文本，用头信息帧和数据帧

- 并发传输

  多个stream共用一个链接。并行交错的传输数据，但是流中的包丢了会互相影响

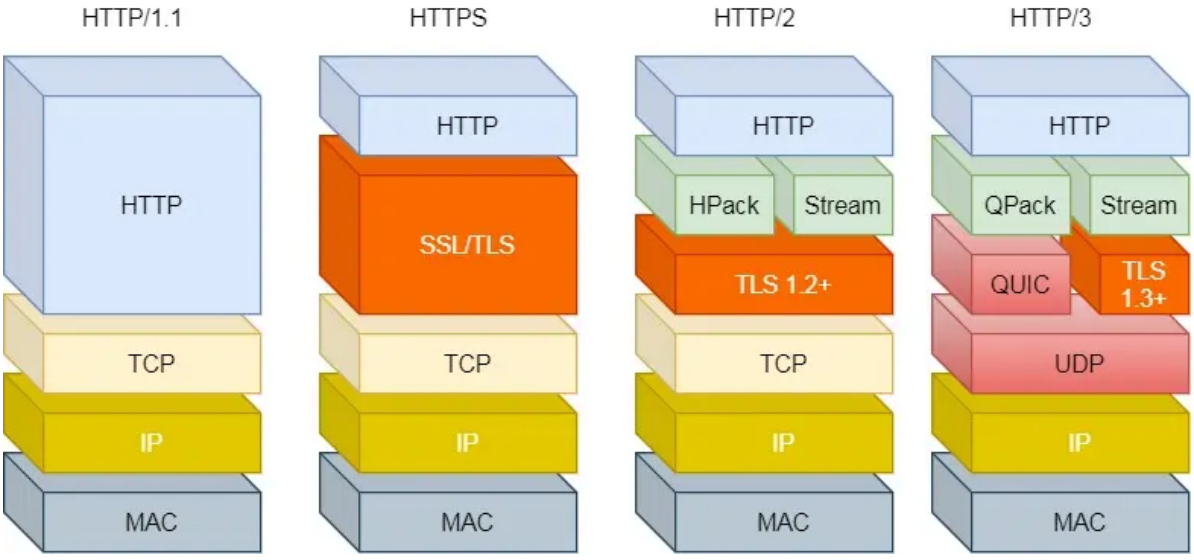
- 服务器主动推送资源

  客户端的Stream是奇数，服务器端的是偶数

问题：

还是队头阻塞，但是是TCP的队头阻塞问题。

HTTP3



针对两种队头阻塞的问题，这里提出的是使用UDP进行通信。

QUIC

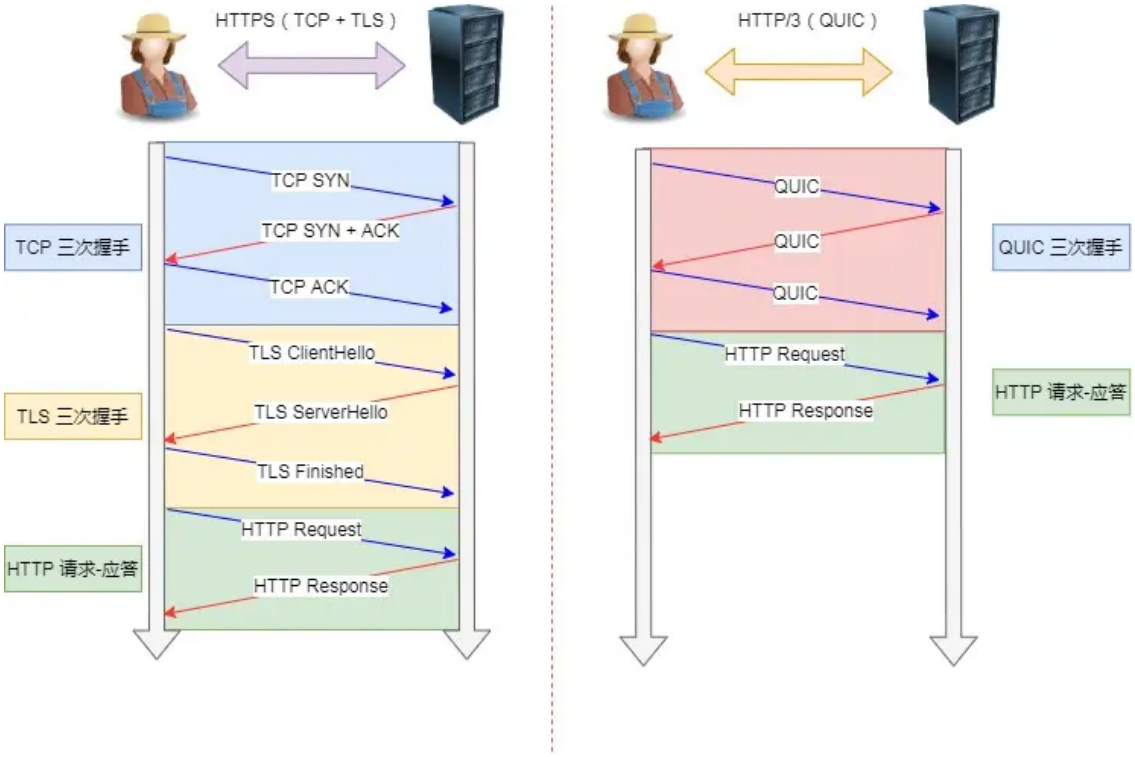
- 无队头阻塞

  改进，只阻塞一个流，其他的不受影响，不存在队头阻塞了

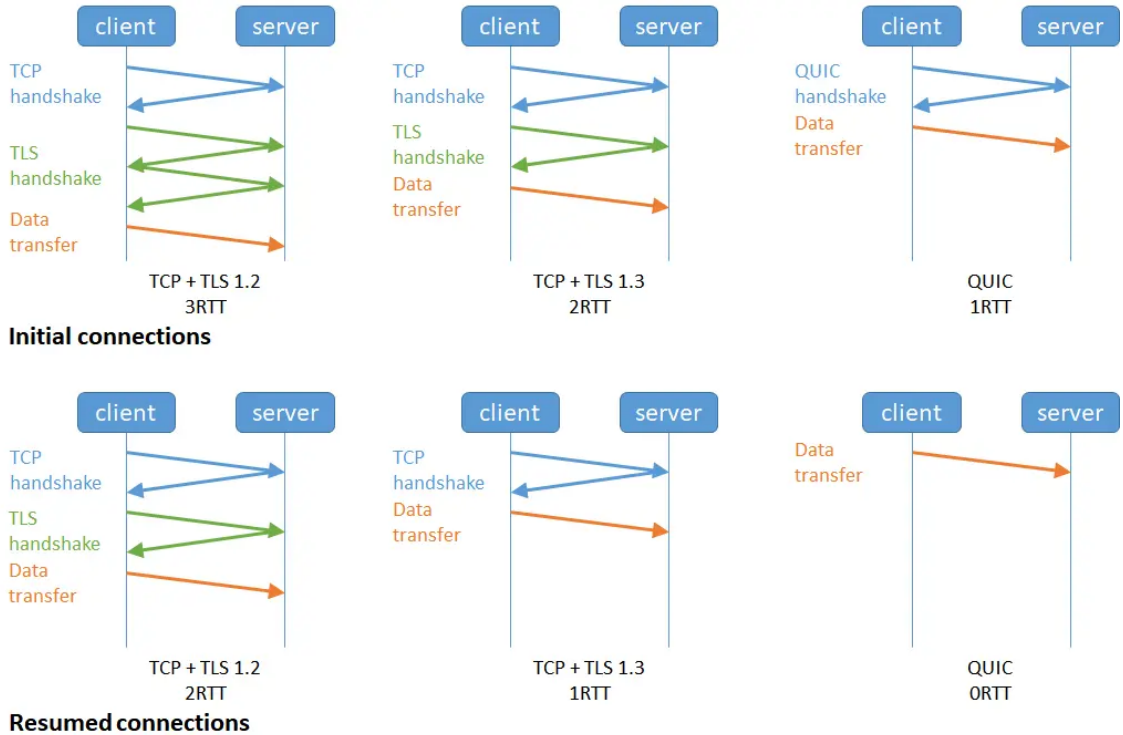
- 更快的连接

  连接之前要进行QUIC连接。握手的目的是为确认双方的「连接 ID」，连接迁移就是基于连接 ID 实现的。

  QUIC中包含了TLS1.3，一次就能够完成建立连接和密钥协商。



  第二次的时候简化，连接信息和TLS一起发送，0RTT



- 连接迁移

因此即使移动设备的网络变化后，导致 IP 地址变化了，只要仍保有上下文信息（比如连接 ID、TLS 密钥等），就可以“无缝”地复用原连接，消除重连的成本，没有丝毫卡顿感，达到了**连接迁移**的功能。

所以， QUIC 是一个在 UDP 之上的**伪** TCP + TLS + HTTP/2 的多路复用的协议d

问题：

一些设备当做UDP扔掉。