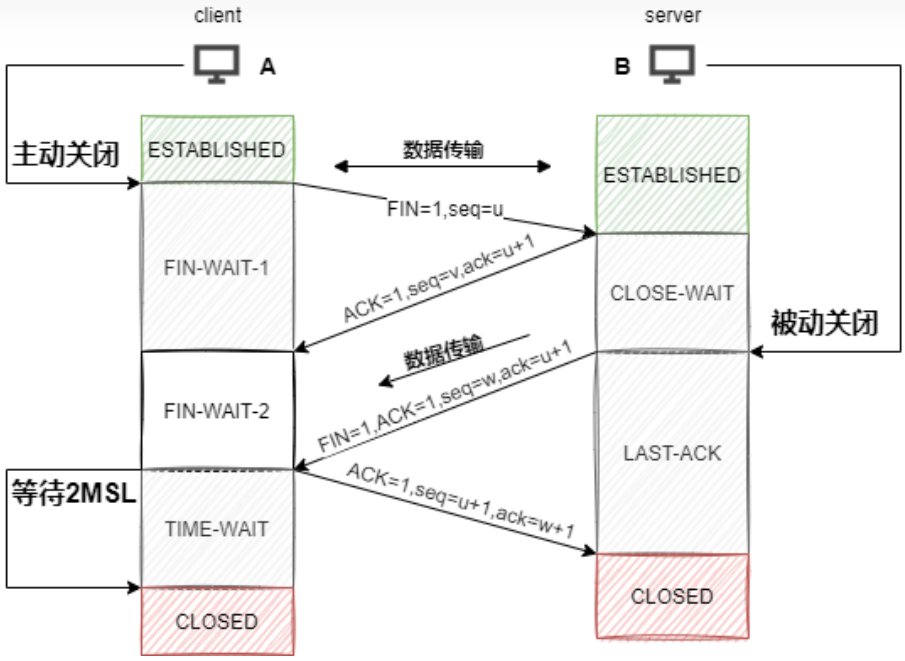


面试八股文——网络篇

程序员大彬

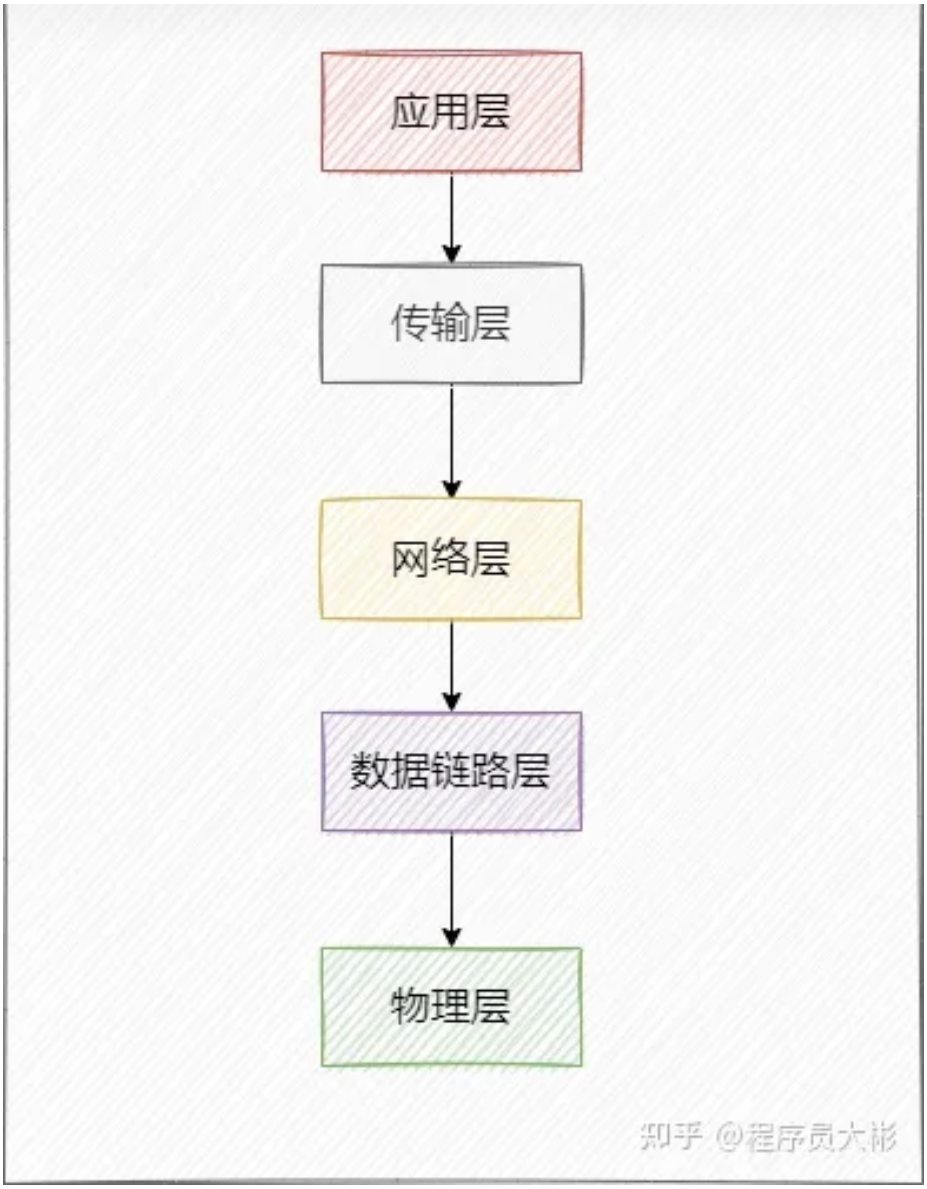
今天给大家分享计算机网络常见的面试题，希望对正在找工作的小伙伴有所帮助~

另外我将**大厂常见的面试题汇总成一份面试手册**，包括Java核心、并发、JVM、计算机网络、MySQL、Redis等，需要的小伙伴可以自行下载：

大厂高频面试题总结
mp.weixin.qq.com/s?__biz=Mzg2OTY1Nz...

网络分层结构

计算机网络体系大致分为三种，OSI七层模型、TCP/IP四层模型和五层模型。一般面试的时候考察比较多的是五层模型。

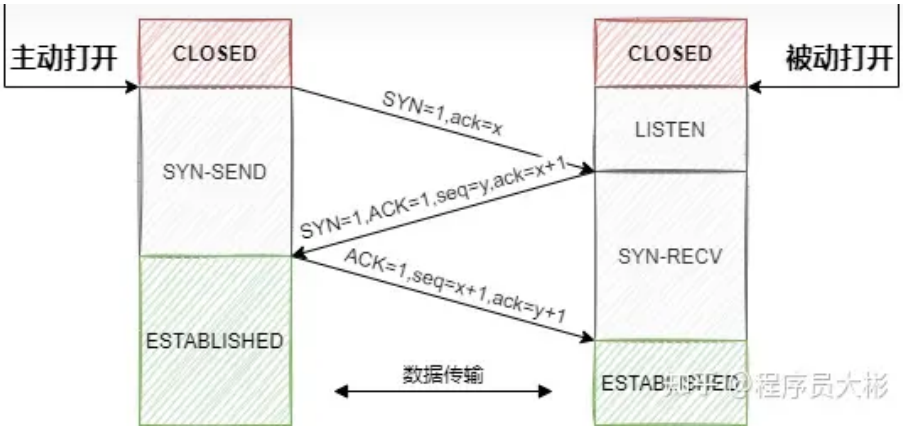


TCP/IP五层模型：应用层、传输层、网络层、数据链路层、物理层。

- **应用层**：为应用程序提供交互服务。在互联网中的应用层协议很多，如域名系统DNS、HTTP协议、SMTP协议等。
- **传输层**：负责向两台主机进程之间的通信提供数据传输服务。传输层的协议主要有传输控制协议TCP和用户数据协议UDP。
- **网络层**：选择合适的路由和交换结点，确保数据及时传送。主要包括IP协议。
- **数据链路层**：在两个相邻节点之间传送数据时，**数据链路层将网络层交下来的 IP 数据报组装成帧**，在两个相邻节点间的链路上传送帧。
- **物理层**：实现相邻节点间比特流的透明传输，尽可能屏蔽传输介质和物理设备的差异。

三次握手

假设发送端为客户端，接收端为服务端。开始时客户端和服务端的状态都是 `CLOSED` 。



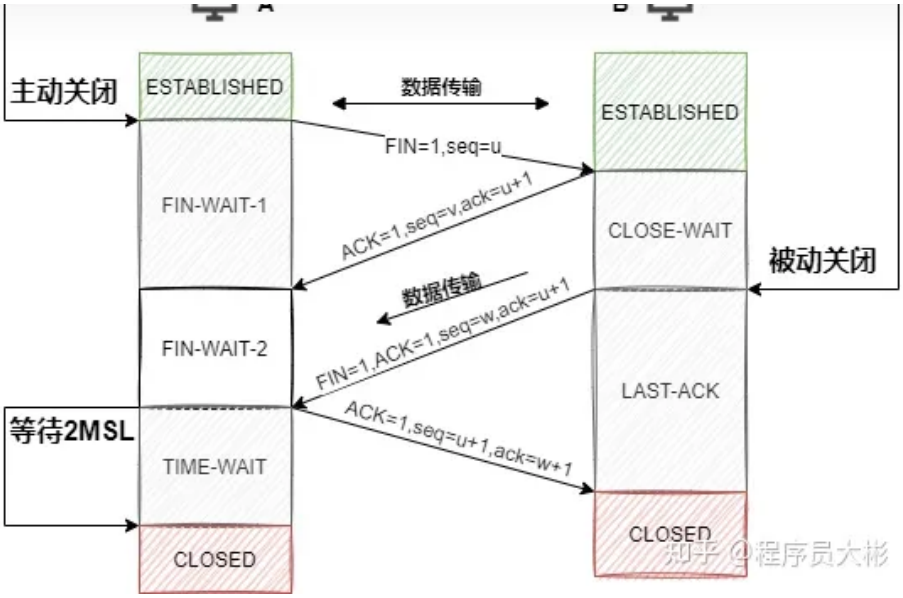
1. 第一次握手：客户端向服务端发起建立连接请求，客户端会随机生成一个起始序列号x，客户端向服务端发送的字段中包含标志位 `SYN=1` ，序列号 `seq=x` 。第一次握手前客户端的状态为 `CLOSE` ，第一次握手后客户端的状态为 `SYN-SENT` 。此时服务端的状态为 `LISTEN` 。
2. 第二次握手：服务端在收到客户端发来的报文后，会随机生成一个服务端的起始序列号y，然后给客户端回复一段报文，其中包括标志位 `SYN=1` ， `ACK=1` ，序列号 `seq=y` ，确认号 `ack=x+1` 。第二次握手前服务端的状态为 `LISTEN` ，第二次握手后服务端的状态为 `SYN-RCVD` ，此时客户端的状态为 `SYN-SENT` 。（其中 `SYN=1` 表示要和客户端建立一个连接， `ACK=1` 表示确认序号有效）
3. 第三次握手：客户端收到服务端发来的报文后，会再向服务端发送报文，其中包含标志位 `ACK=1` ，序列号 `seq=x+1` ，确认号 `ack=y+1` 。第三次握手前客户端的状态为 `SYN-SENT` ，第三次握手后客户端和服务端的状态都为 `ESTABLISHED` 。**此时连接建立完成。**

两次握手可以吗?

第三次握手主要为了**防止已失效的连接请求报文段**突然又传输到了服务端，导致产生问题。

- 比如客户端A发出连接请求，可能因为网络阻塞原因，A没有收到确认报文，于是A再重传一次连接请求。
- 连接成功，等待数据传输完毕后，就释放了连接。
- 然后A发出的第一个连接请求等到连接释放以后的某个时间才到达服务端B，此时B误认为A又发出一次新的连接请求，于是就向A发出确认报文段。
- 如果不采用三次握手，只要B发出确认，就建立新的连接了，**此时A不会响应B的确认且不发送数据，则B一直等待A发送数据，浪费资源。**

四次挥手



1. A的应用进程先向其TCP发出连接释放报文段（ FIN=1，seq=u ），并停止再发送数据，主动关闭TCP连接，进入 FIN-WAIT-1 （终止等待1）状态，等待B的确认。
2. B收到连接释放报文段后即发出确认报文段（ ACK=1，ack=u+1，seq=v ），B进入 CLOSE-WAIT （关闭等待）状态，此时的TCP处于半关闭状态，A到B的连接释放。
3. A收到B的确认后，进入 FIN-WAIT-2 （终止等待2）状态，等待B发出的连接释放报文段。
4. B发送完数据，就会发出连接释放报文段（ FIN=1，ACK=1，seq=w，ack=u+1 ），B进入 LAST-ACK （最后确认）状态，等待A的确认。
5. A收到B的连接释放报文段后，对此发出确认报文段（ ACK=1，seq=u+1，ack=w+1 ），A进入 TIME-WAIT （时间等待）状态。此时TCP未释放掉，需要经过时间等待计时器设置的时间 2MSL （最大报文段生存时间）后，A才进入 CLOSED 状态。B收到A发出的确认报文段后关闭连接，若没收到A发出的确认报文段，B就会重传连接释放报文段。

第四次挥手为什么要等待2MSL?

- **保证A发送的最后一个ACK报文段能够到达B。**这个 ACK 报文段有可能丢失，B收不到这个确认报文，就会超时重传连接释放报文段，然后A可以在 2MSL 时间内收到这个重传的连接释放报文段，接着A重传一次确认，重新启动2MSL计时器，最后A和B都进入到 CLOSED 状态，若A在 TIME-WAIT 状态不等待一段时间，而是发送完ACK报文段后立即释放连接，则无法收到B重传的连接释放报文段，所以不会再发送一次确认报文段，B就无法正常进入到 CLOSED 状态。
- **防止已失效的连接请求报文段出现在本连接中。**A在发送完最后一个 ACK 报文段后，再经过2MSL，就可以使这个连接所产生的所有报文段都从网络中消失，使下一个新的连接中不会出现旧的连接请求报文段。

为什么是四次挥手?

因为当Server端收到Client端的 SYN 连接请求报文后，可以直接发送 SYN+ACK 报文。**但是在关闭连接时，当Server端收到Client端发出的连接释放报文时，很可能并不会立即关闭SOCKET**，所以Server端先回复一个 ACK 报文，告诉Client端我收到你的连接释放报文了。只有等到Server端所有的报文都发送完了，这时Server端才能发送连接释放报文，之后两边才会真正的断开连接。故需要四次挥手。

TCP有哪些特点?

- TCP是**面向连接**的运输层协议。
- **点对点**，每一条TCP连接只能有两个端点。
- TCP提供**可靠交付**的服务。
- TCP提供**全双工通信**。

TCP和UDP的区别?

1. TCP**面向连接**；UDP是无连接的，即发送数据之前不需要建立连接。
2. TCP提供**可靠的服务**；UDP不保证可靠交付。
3. TCP**面向字节流**，把数据看成一连串无结构的字节流；UDP是面向报文的。
4. TCP有**拥塞控制**；UDP没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如实时视频会议等）。
5. 每一条TCP连接只能是**点到点**的；UDP支持一对一、一对多、多对一和多对多的通信方式。
6. TCP首部开销20字节；UDP的首部开销小，只有8个字节。

HTTP协议的特点?

1. HTTP允许传输**任意类型**的数据。传输的类型由Content-Type加以标记。
2. **无状态。**对于客户端每次发送的请求，服务器都认为是一个新的请求，上一次会话和下一次会话之间没有联系。
3. 支持**客户端/服务器模式。**

HTTP报文格式

HTTP请求由**请求行、请求头部、空行和请求体**四个部分组成。

- **请求行**：包括请求方法，访问的资源URL，使用的HTTP版本。 `GET` 和 `POST` 是最常见的HTTP方法，除此以外还包括 `DELETE`、`HEAD`、`OPTIONS`、`PUT`、`TRACE` 。
- **请求头**：格式为“属性名:属性值”，服务端根据请求头获取客户端的信息，主要有 `cookie`、`host`、`connection`、`accept-language`、`accept-encoding`、`user-agent` 。
- **请求体**：用户的请求数据如用户名，密码等。

请求报文示例：

```
POST /xxx HTTP/1.1 请求行
Accept:image/gif.image/jpeg, 请求头部
Accept-Language:zh-cn
Connection:Keep-Alive
Host:localhost
User-Agent:Mozila/4.0(compatible;MSIE5.01;Window NT5.0)
Accept-Encoding:gzip,deflate

username=dabin 请求体
```

HTTP响应也由四个部分组成，分别是：**状态行、响应头、空行和响应体。**

- **状态行**：协议版本，状态码及状态描述。
- **响应头**：响应头字段主要有 `connection`、`content-type`、`content-encoding`、`content-length`、`set-cookie`、`Last-Modified`，、`Cache-Control`、`Expires` 。
- **响应体**：服务器返回给客户端的内容。

响应报文示例：

```
HTTP/1.1 200 OK
Server:Apache Tomcat/5.0.12
Date:Mon,6Oct2003 13:23:42 GMT
Content-Length:112

<html>
    <body>响应体</body>
</html>
```

1xx	服务器收到请求，需要请求者继续执行操作
2xx	请求正常处理完毕
3xx	重定向，需要进一步操作已完成请求
4xx	客户端错误，服务器无法处理请求
5xx	服务器处理请求出错

HTTP1.0和HTTP1.1的区别?

- **长连接**：HTTP1.0默认使用短连接，每次请求都需要建立新的TCP连接，连接不能复用。**HTTP1.1支持长连接，复用TCP连接，允许客户端通过同一连接发送多个请求。**不过，这个优化策略也存在问题，当一个队头的请求不能收到响应的资源时，它将会阻塞后面的请求。这就是“**队头阻塞**”问题。
- **断点续传**：HTTP1.0 **不支持断点续传。**HTTP1.1 新增了 **range** 字段，用来指定数据字节位置，**支持断点续传。**
- **错误状态响应码**：在HTTP1.1中新增了24个错误状态响应码，如 `409（Conflict）` 表示请求的资源与资源的当前状态发生冲突、 `410（Gone）` 表示服务器上的某个资源被永久性的删除。
- **Host头处理**：在HTTP1.0中认为每台服务器都绑定一个唯一的IP地址，因此，请求消息中的URL并没有传递主机名。到了HTTP1.1时代，虚拟主机技术发展迅速，在一台物理服务器上可以存在多个虚拟主机，并且它们共享一个IP地址，故HTTP1.1增加了HOST信息。

HTTP1.1和 HTTP2.0的区别?

HTTP2.0相比HTTP1.1支持的特性：

- **新的二进制格式**：HTTP1.1 基于文本格式传输数据；HTTP2.0采用二进制格式传输数据，解析更高效。
- **多路复用**：在一个连接里，允许同时发送多个请求或响应，**并且这些请求或响应能够并行的传输而不被阻塞**，避免 HTTP1.1 出现的”队头堵塞”问题。
- **头部压缩**，HTTP1.1的header带有大量信息，而且每次都要重复发送；HTTP2.0 把header从数据中分离，并封装成头帧和数据帧，**使用特定算法压缩头帧**，有效减少头信息大小。并且HTTP2.0**在客户端和服务器端记录了之前发送的键值对，对于相同的数据，不会重复发送。**比如请求a发送了所有的头信息字段，请求b则**只需要发送差异数据**，这样可以减少冗余数据，降低开销。
- **服务端推送**：HTTP2.0允许服务器向客户端推送资源，无需客户端发送请求到服务器获取。

HTTPS与HTTP的区别?

1. HTTP是超文本传输协议，信息是**明文传输**；HTTPS则是具有**安全性**的ssl加密传输协议。
2. HTTP和HTTPS用的端口不一样，HTTP端口是80，HTTPS是443。
3. HTTPS协议**需要到CA机构申请证书**，一般需要一定的费用。
4. HTTP运行在TCP协议之上；HTTPS运行在SSL协议之上，SSL运行在TCP协议之上。

什么是数字证书?

服务端把证书传输给浏览器，浏览器从证书里取公钥。证书可以证明该公钥对应本网站。

数字签名的制作过程：

1. CA使用证书签名算法对证书内容进行**hash运算。**
2. 对hash后的值**用CA的私钥加密**，得到数字签名。

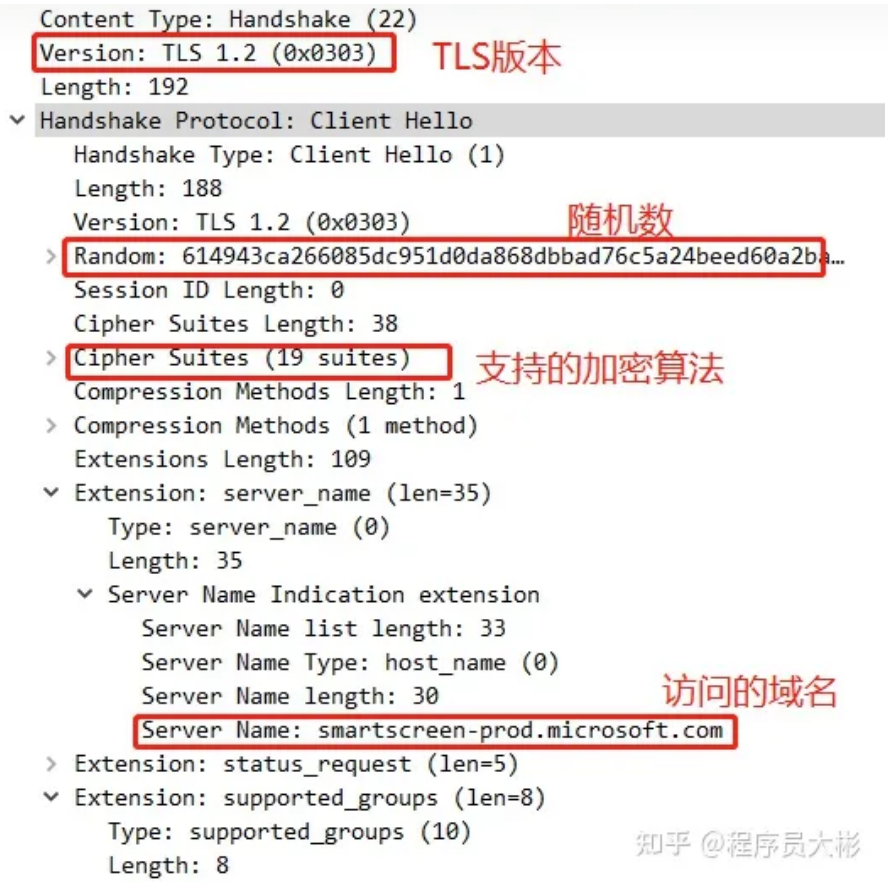
浏览器验证过程：

1. 获取证书，得到证书内容、证书签名算法和数字签名。
2. 用CA机构的公钥**对数字签名解密**（由于是浏览器信任的机构，所以浏览器会保存它的公钥）。
3. 用证书里的签名算法**对证书内容进行hash运算。**
4. 比较解密后的数字签名和对证书内容做hash运算后得到的哈希值，相等则表明证书可信。

HTTPS原理

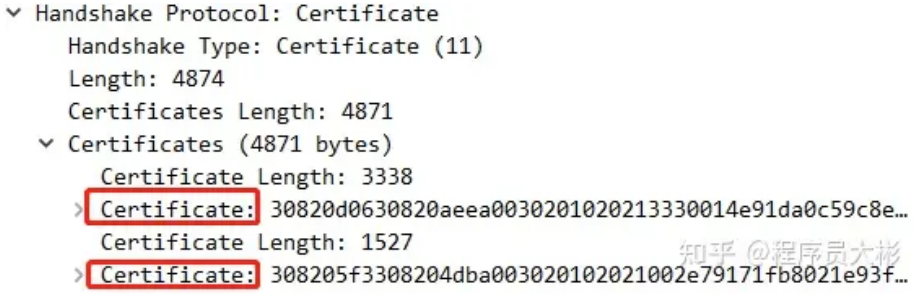
首先是TCP三次握手，然后客户端发起一个HTTPS连接建立请求，客户端先发一个 `Client Hello` 的包，然后服务端响应 `Server Hello` ，接着再给客户端发送它的证书，然后双方经过密钥交换，最后使用交换的密钥加解密数据。

1. **协商加密算法** 。在 `Client Hello` 里面客户端会告知服务端自己当前的一些信息，包括客户端要使用的TLS版本，支持的加密算法，要访问的域名，给服务端生成的一个随机数（Nonce）等。需要提前告知服务器想要访问的域名以便服务器发送相应的域名的证书过来。

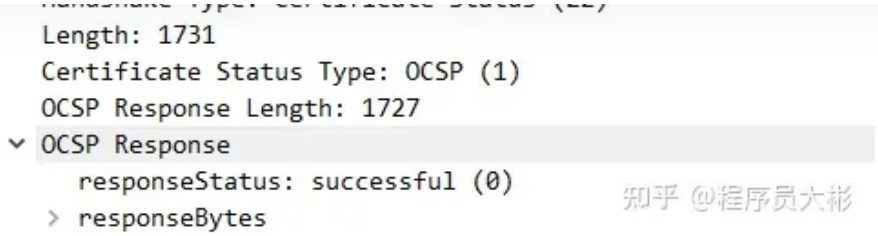


1. 服务端响应 `Server Hello` ，告诉客户端服务端**选中的加密算法。**

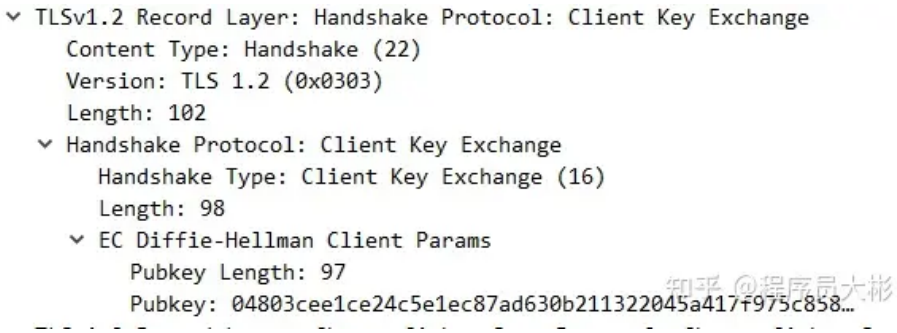
1. 接着服务端给客户端发来了2个证书。第二个证书是第一个证书的签发机构（CA）的证书。



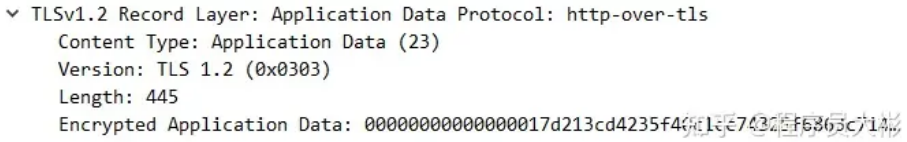
1. 客户端使用证书的认证机构CA公开发布的RSA公钥**对该证书进行验证**，下图表明证书认证成功。



1. 验证通过之后，浏览器和服务器通过**密钥交换算法**产生共享的**对称密钥。**



1. 开始传输数据，使用同一个对称密钥来加解密。



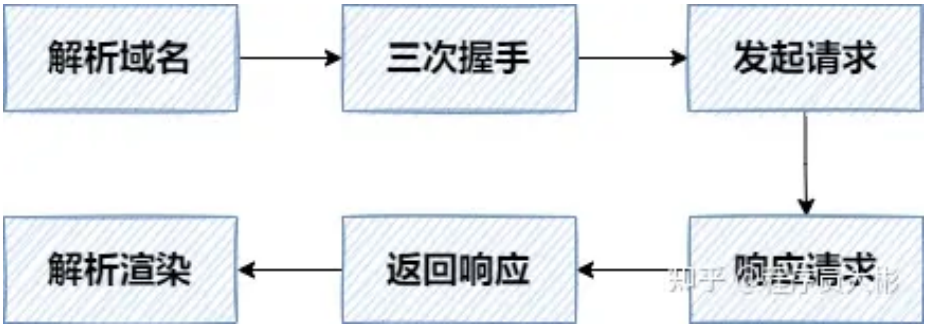
DNS 的解析过程?

1. 浏览器搜索**自己的DNS缓存**
2. 若没有，则搜索**操作系统中的DNS缓存和hosts文件**
3. 若没有，则操作系统将域名发送至**本地域名服务器**，本地域名服务器查询自己的DNS缓存，查找成功则返回结果，否则依次向**根域名服务器、顶级域名服务器、权限域名服务器**发起查询请求，最终返回IP地址给本地域名服务器
4. 本地域名服务器将得到的IP地址返回给**操作系统**，同时自己也**将IP地址缓存起来**
5. 操作系统将 IP 地址返回给浏览器，同时自己也将IP地址缓存起来
6. 浏览器得到域名对应的IP地址

浏览器中输入URL返回页面过程?

1. **解析域名**，找到主机 IP。

3. 建立 TCP 连接后，浏览器向主机发起一个HTTP请求。
4. 服务器**响应请求**，返回响应数据。
5. 浏览器**解析响应内容，进行渲染**，呈现给用户。



Cookie和Session的区别?

- **作用范围不同**，Cookie 保存在客户端，Session 保存在服务器端。
- **有效期不同**，Cookie 可设置为长时间保持，比如我们经常使用的默认登录功能，Session 一般失效时间较短，客户端关闭或者 Session 超时都会失效。
- **隐私策略不同**，Cookie 存储在客户端，容易被窃取；Session 存储在服务端，安全性相对 Cookie 要好一些。
- **存储大小不同**， 单个 Cookie 保存的数据不能超过 4K；对于 Session 来说存储没有上限，但出于对服务器的性能考虑，Session 内不要存放过多的数据，并且需要设置 Session 删除机制。

什么是对称加密和非对称加密?

对称加密：通信双方使用**相同的密钥**进行加密。特点是加密速度快，但是缺点是密钥泄露会导致密文数据被破解。常见的对称加密有 AES 和 DES 算法。

非对称加密：它需要生成两个密钥，**公钥和私钥。**公钥是公开的，任何人都可以获得，而私钥是私人保管的。公钥负责加密，私钥负责解密；或者私钥负责加密，公钥负责解密。这种加密算法**安全性更高**，但是**计算量相比对称加密大很多**，加密和解密都很慢。常见的非对称算法有 RSA 和 DSA 。

> 本文已经收录到github仓库，此仓库用于分享**互联网大厂高频面试题、Java核心知识总结**，包括Java基础、并发、MySQL、Springboot、MyBatis、Redis、RabbitMQ等等，面试必备！欢迎大家star！
> github地址：github.com/Tyson0314/Ja...
> 如果github访问不了，可以访问gitee仓库
> gitee地址：gitee.com/tysondai/Java...

码字不易，如果觉得对你有帮助，可以**点个赞**鼓励一下！

我是@程序员大彬，专注Java后端硬核知识分享，欢迎大家关注~

编辑于 2021-11-27 15:33

秋招 Java 计算机网络

不是很能搞懂，明明是一个工科或者理科的专业，现在的面试变成了文科的形式，看谁能背书，背的多，背的准。

2021-10-13 回复 3

雨天

市场供大于求，就使劲儿卷了。

2021-10-13 回复 1

贝宁

为什么不能两次握手的答案是错的哦，应该是因为只有两次的话服务端B没办法确认他上一次发过去的请求是否被A成功收到（可能会丢失）

2021-10-07 回复 2

佳闻

没错，你这也是一个原因，拜占庭问题

2022-09-24 回复 喜欢

风中一匹狼 ‣ **贝宁**

牛逼 教科书错了，你对

2022-04-18 回复 喜欢

展开其他 3 条回复 >

肥鱼张

谢谢，已收藏，能唬住面试官就要40k🥹

2021-10-07 回复 4

程序员大彬 作者

小伙伴们留个赞再走🥹

2021-10-05 回复 1

达摩克利斯

想问下，复习八股有必要系统过一遍408专业课吗

2022-07-11 回复 喜欢

程序员大彬 作者 ‣ **达摩克利斯**

可以的，关键是理解哈

2022-07-11 回复 喜欢

达摩克利斯 ‣ **程序员大彬**

喔喔，那直接八股文开背完事的嘛?

2022-07-11 回复 喜欢

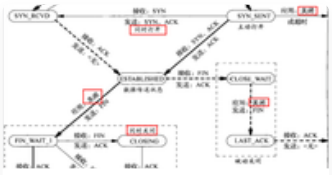
展开其他 2 条回复 >

文章被以下专栏收录

Java基础知识总结

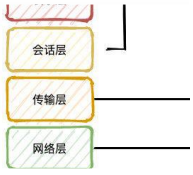
硬核Java知识总结，你值得拥有

推荐阅读



日常知识点之网络面试八股文（tcp,惊群现象，协

最常问的网络基础面试问题整理



关于网络面试题你只