从浏览器中输入 URL 到页面加载的发生了什么?

张太国 2018/12

作 者

姓名

张太国

信念

科技改变生活，
科技改变世界。
生命在于运动，
生活在于折腾。

邮箱

confach@gmail.com

个人微信：terryisme

公众号：terrymemo

目 录

背景

这是一道经典的面试题，涉及到的知识面非常多，但作为一个自认为对网络知识掌握的比较好的老码农来说，回答这个问题自然不在话下。如果这道题目如果在面试出现，对我来说就是送分题啊。尽管如此，我还是愿意花一些时间回答一下这个题目，看我自己到底掌握的有多深，同时也把自己的知识梳理一下。

想起一件往事，这道题有点类似于在手机上浏览器上输入一个 URL，手机到底做了一些什么？我当时转做通信学习核心网给自己提出的一个问题。

当然，我非常愿意将这个面试题的答案共享出来，一是希望得到大家的意见，二是也希望对那些不是特别熟的人起到一些帮助。

从本文里学到什么？

正如前面所说，这篇文章涉及到的知识面非常丰富，我相信您绝对可以从本文里学到很多基础知识，还有一些高级话题。

1. DNS 的解析原理，常用命令，端口等
2. TCP/IP 模型，三次握手，四次挥手。
3. HTTP/HTTPS 的原理和解析。
4. 浏览器 render 一个页面
5. Web 安全性问题
6. 抓包，分析 TCP 模型，三次握手，SSL/TLS，让学起来不再枯燥。
7. 其他一些高级话题。

总概：几大步骤

总的来说，当你输入在浏览器里输入一个 URL 到页面加载，发生的顺序如下：

1. DNS 查询
2. TCP 连接
3. 发送 HTTP 请求
4. Server 处理 HTTP 请求并返回 HTTP 报文
5. 浏览器解析并 render 页面
6. HTTP 连接断开

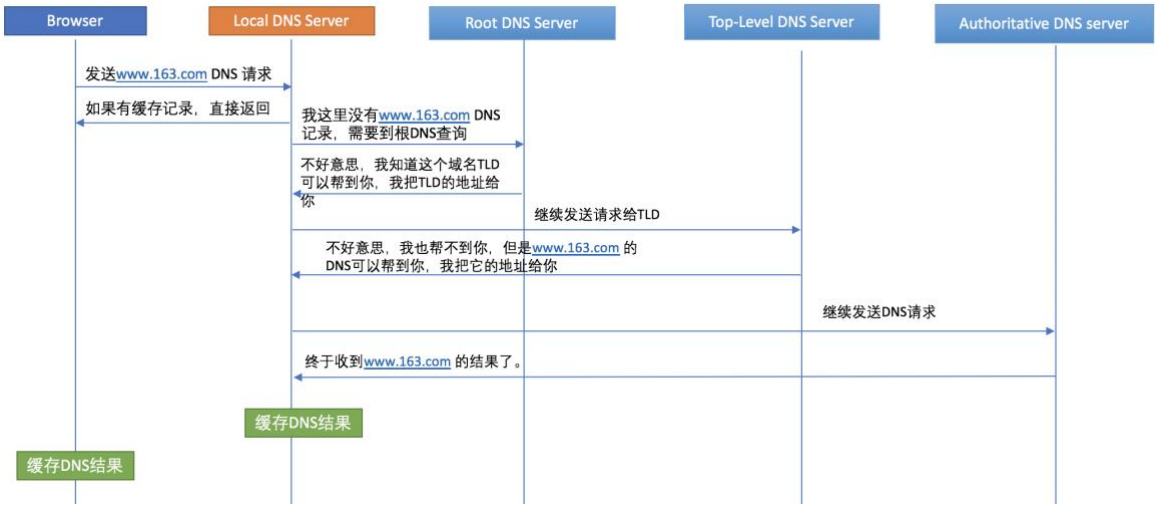
后面将对以上步骤详细介绍。

DNS 查询

DNS 解析流程

假设输入的 URL 是包含域名的，那肯定会涉及到 DNS 解析。当然，如果 URL 仅仅是 IP，那就不会涉及到 DNS 的。域名的出现是为了方便记忆，因为域名比 IP 好记。我们这里假设 URL 包含域名。

DNS 解析的步骤大致如下图：



上述流程是查找 www.163.com 的 Web Server 的 IP 地址过程。

首先，在**本地域名服务器**中根据域名查询 IP 地址，如果没有找到的情况下，本地域名服务器会向**根域名服务器**发送一个请求。
如果根域名服务器也不存在该域名时，本地域名会向 com **顶级域名服务器**（TLD）发送一个请求，依次类推下去。
直到最后本地域名服务器得到 google 的 IP 地址并把它缓存到本地，供下次查询使用。

可以参考页面 https://www.verisign.com/en_US/website-presence/online/how-dns-works/index.xhtml ，该页面诠释了 DNS 的过程。

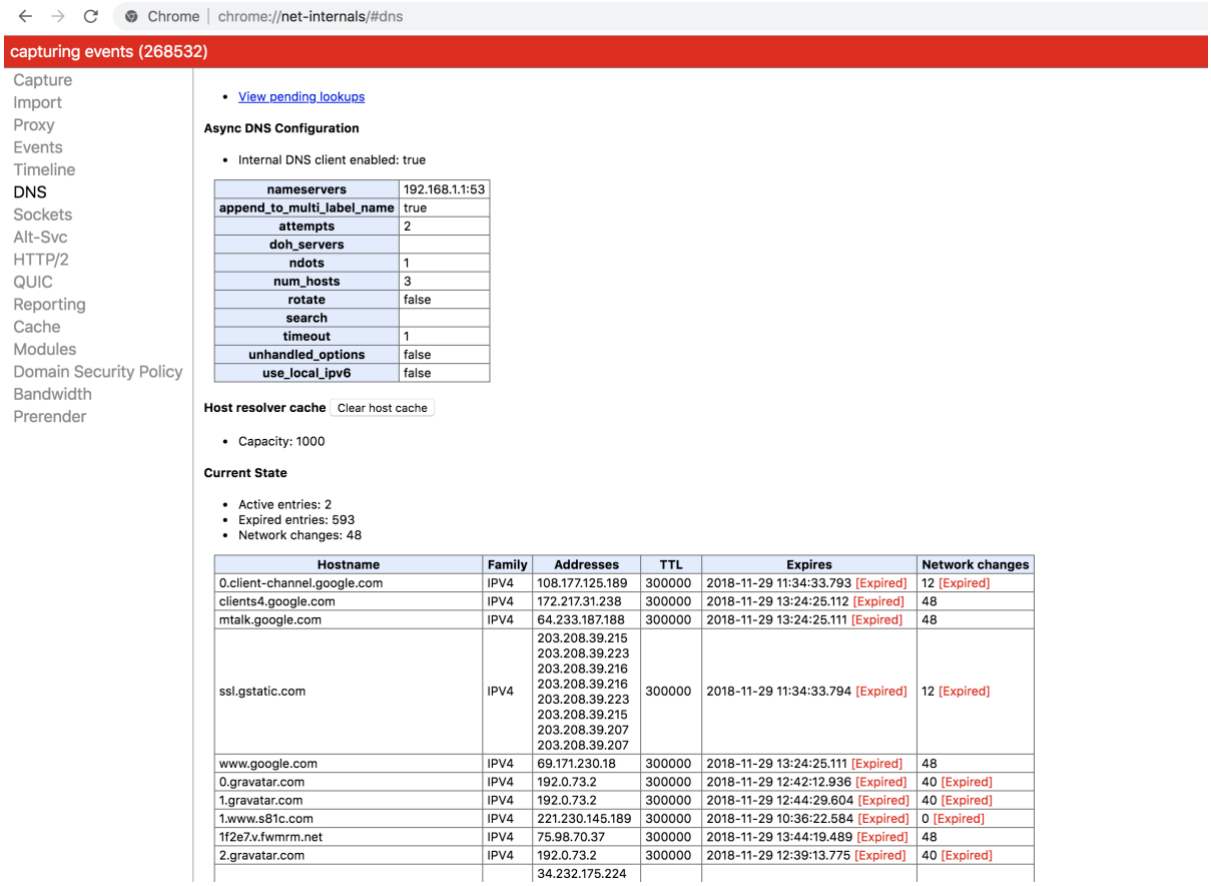
需要说明的是 Root DNS Server 一般有 13 个，后面有个点（.），别忘了。
a.root-servers.net.
c.root-servers.net.
j.root-servers.net.
b.root-servers.net.
i.root-servers.net.
d.root-servers.net.
k.root-servers.net.
f.root-servers.net.
l.root-servers.net.
h.root-servers.net.
m.root-servers.net.
g.root-servers.net.
e.root-servers.net.

DNS 的优化

我们发现，一个 DNS 查询在没有缓存的情况下会有 6 步，这将是一个耗时的过程，如果 DNS 查询时间过长，甚至会影响到用户体验。

那么现阶段是怎么优化的呢？缓存。DNS 是存在着多级缓存，从离浏览器的距离排序的话，有以下几种: 浏览器缓存，系统缓存，路由器缓存，IPS 服务器缓存，根域名服务器缓存，顶级域名服务器缓存，主域名服务器缓存。

我们以 Chrome 为例子，输入 ***chrome://net-internals/#dns***，我们会看到如下界面：



这里的 *Capacity: 1000*，代表缓存 1K 条，那么每条记录缓存多久呢？参看

timeout	1

这个 1 代表 1 分钟。

如果是系统缓存，一般分 2 种情况：

Linux 操作系统

见下图，一般在/etc/hosts 下。


```
192:~ terryzhang$ more /etc/hosts
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
```

Windows 操作系统

一般在 C:**Windows**\System32**Drivers****etc****hosts**。

谈到这里，我给大家分享一个特别有用的技巧。

我们在做开发和测试时，会有一种情况，经常会去访问某个（测试）URL，例如 http://192.168.1.8:8080/admin/login, 如果这样的话，我们极有可能存在一个问题，觉得每次输入 ip 真的很麻烦。怎么解决？那么我们可以这么做，利用 DNS 的原理，将某个伪 hostname（abc-test）加入到 hosts 里，只需加入一条记录：

192.168.1.8 . abc-test

这样就可以用 *http://abc-test:8080/admin/login* 去访问了，简单易用，对吧。

DNS 负载均衡

不知道大家有没有注意或思考过一个问题，Google 在全球都有服务器，不同的国家或区域访 Google，是不是都很快，这就是 Google GSLB 的作用。也就是说不同的区域访问 Google 返回的 IP 是不一样的，甚至在同一个办公室访问 Google，返回回来的 IP 也有可能是不一样的。

为什么呢？这里用的就是 DNS 负载均衡，不然 Google 怎么去支持全球几十亿客户的请求呢。总之，DNS 会根据你的位置或 IP 返回一个合适的 IP 给你用。除了 Google，一些 CDN 的 SP 例如 Akamai、AWS、Azure、阿里云都有类似的服务。

DNS Record（记录）

DNS 记录是一个非常重好的概念。DNS 记录类型如下表。

记录类型	含义简介
A（Address）	指定域名对应的 IPv4 地址
AAAA	指定域名对应的 IPv6 地址

NS（Name Server）	指定该域名由哪个 DNS 服务器来进行解析
MX（Mail Exchanger）	邮件交换记录，用于电子邮件系统发邮件时根据收信人的地址后缀来定位邮件服务器
CNAME	别名记录，多个域名映射到同一台计算机（如同一主机提供 mail 和 www 服务）
TXT	主机名或域名的说明
TTL（Time-To-Live）	DNS 服务器中保存的时间
PTR	将一个主机地址映射到对应的域名
HINFO	说明映射到特定 DNS 主机名的 CPU 类型和操作系统类型

在这里只介绍常用的 A Record， CNAME，MX，NS。

A 记录

A 记录是用的最多的一种类型。

A (Address) 记录是用来指定主机名（或域名）对应的 IP 地址记录。用户可以将该域名下的网站服务器指向到自己的 Web Server 上。

同时也可以设置该域名的子域名。通俗来说 A 记录就是服务器的 IP,域名绑定 A 记录就是告诉 DNS,当你输入域名的时候给你引导向设置在 DNS 的 A 记录所对应的服务器。

后面的抓包分析会对 A 记录进行分析，让您有直观认识。

CNAME 记录

CNAME 记录是另一种用的比较多的记录，可以将一个域名或者子域名指向另外一个主机名。

最常用的使用场景是什么呢？没错，CDN 就是这种方式。举个例子，例如公司 A 想把自己的图片放在 Akamai 的 CDN 上，A 的子域名是 img.abc.com, 而 Akamai 的 CDN 服务域名是 img.akaimacdn.com. 但是 A 公司期望用自己的域名吗，而不是 Akamai 的域名。为了实现这个目

标，怎么办？是的，使用 CNAME，只需要将子域名 img.abc.com 指向到 img.akaimacdn.com。
问题又来了，在哪里设置呢？肯定是在公司 A 这边的 DNS server 上，而不是 Akamai 那边。
后面的抓包分析会对 MX 进行分析，让您有直观认识。

MX 记录

MX（Mail Exchanger）记录，字面意思很直观，知道它用来做邮件路由，用户可以将域名下的邮件服务器指向到自己的邮件服务器上，然后可以自己操控所有的邮箱设置。所以只需在线填写服务器的 IP 地址，即可以将域名下的邮件全部转到您自己设定相应的邮件服务器上。

NS 记录

NS 记录用来解析服务器记录，表明由哪台服务器对该域名进行解析，这里的 NS 记录只对子域名生效。

例如用户希望由 12.34.56.78 这台服务器解析 sub1.mydomain.com，则需要设置 sub1.mydomain.com 的 NS 记录。

这里涉及到一个问题，细心的我们会发现 A 记录也有该功能，这里就涉及到优先级的问题了。NS 记录优先于 A 记录。如果一个主机地址同时存在 NS 记录和 A 记录，则 A 记录不生效。

DNS 常用命令和工具

只介绍 2 个常用的命令 *dig* 和 *nslookup*。

dig

dig 是一个 DNS 查询工具。

```
[warren]$ dig

; <<>> DiG 9.9.5-3ubuntu0.16-Ubuntu <<>>
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57775
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       16265   IN      NS      a.root-servers.net.
```



```
.                       16265   IN      NS      c.root-servers.net.
.                       16265   IN      NS      j.root-servers.net.
.                       16265   IN      NS      b.root-servers.net.
.                       16265   IN      NS      i.root-servers.net.
.                       16265   IN      NS      d.root-servers.net.
.                       16265   IN      NS      k.root-servers.net.
.                       16265   IN      NS      f.root-servers.net.
.                       16265   IN      NS      l.root-servers.net.
.                       16265   IN      NS      h.root-servers.net.
.                       16265   IN      NS      m.root-servers.net.
.                       16265   IN      NS      g.root-servers.net.
.                       16265   IN      NS      e.root-servers.net.

;; Query time: 0 msec
;; SERVER: 208.113.157.202#53(208.113.157.202)
;; WHEN: Thu Nov 29 18:04:06 PST 2018
;; MSG SIZE  rcvd: 239
```

再看看 dig 京东的域名 www.jd.com 会有什么效果。

```
dig www.jd.com

; <<>> DiG 9.10.6 <<>> www.jd.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2675
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.jd.com.                    IN      A

;; ANSWER SECTION:
www.jd.com.             300     IN      CNAME   www.jd.com.gslb.qianxun.com.
www.jd.com.gslb.qianxun.com. 300 IN     CNAME   www.jdcdn.com.
www.jdcdn.com.          300     IN      A       61.174.55.1

;; Query time: 3 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Fri Nov 30 10:07:37 CST 2018
;; MSG SIZE  rcvd: 106
```

红色部分为有 CNAME 记录和 A 记录。

nslookup

nslookup 适用于 Windows，Linux，macOS 等操作系统。

```
nslookup www.jd.com
Server:         192.168.1.1
Address:    192.168.1.1#53

Non-authoritative answer:
www.jd.com      canonical name = www.jd.com.gslb.qianxun.com.
www.jd.com.gslb.qianxun.com     canonical name = www.jdcdn.com.
Name:   www.jdcdn.com
Address: 61.174.55.1
```

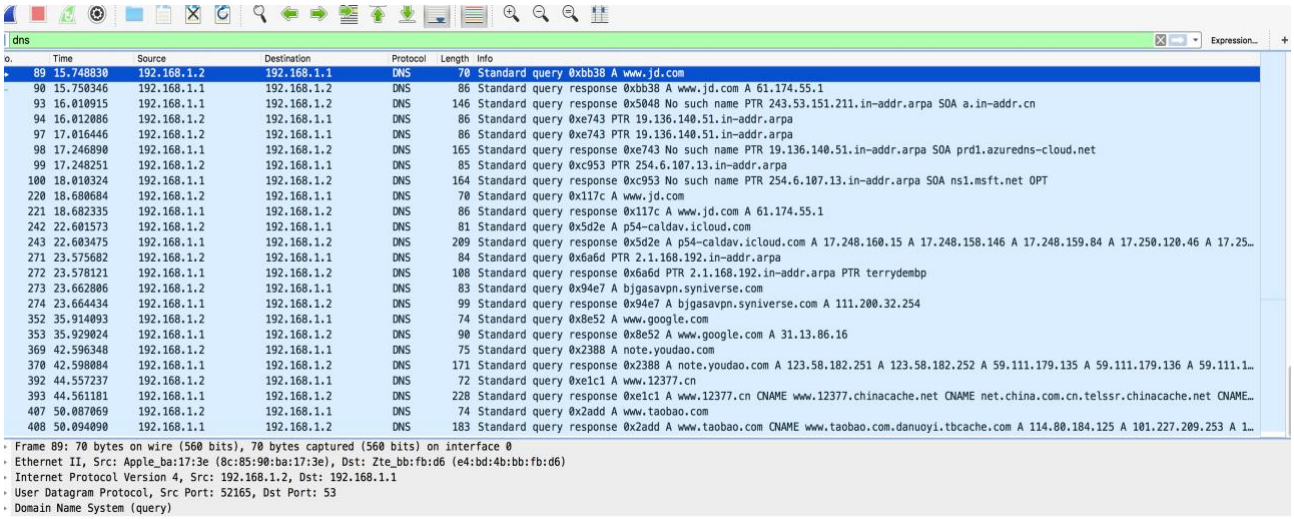
DNS 抓包分析

说了这么多，有些同学可能还是觉得只有概念，没法完全理解。所以，我们还是来点实战更为实际，这也是我更进一步了解 DNS 的工作方式，因为在我看来想了解根本，最有效的方法是抓包，然后深入查探 DNS 的 packet。

本文都会使用 Wireshark，相信大家都用过这个工具吧。如果没用过，可以下载一个，入门很简单的，别担心。

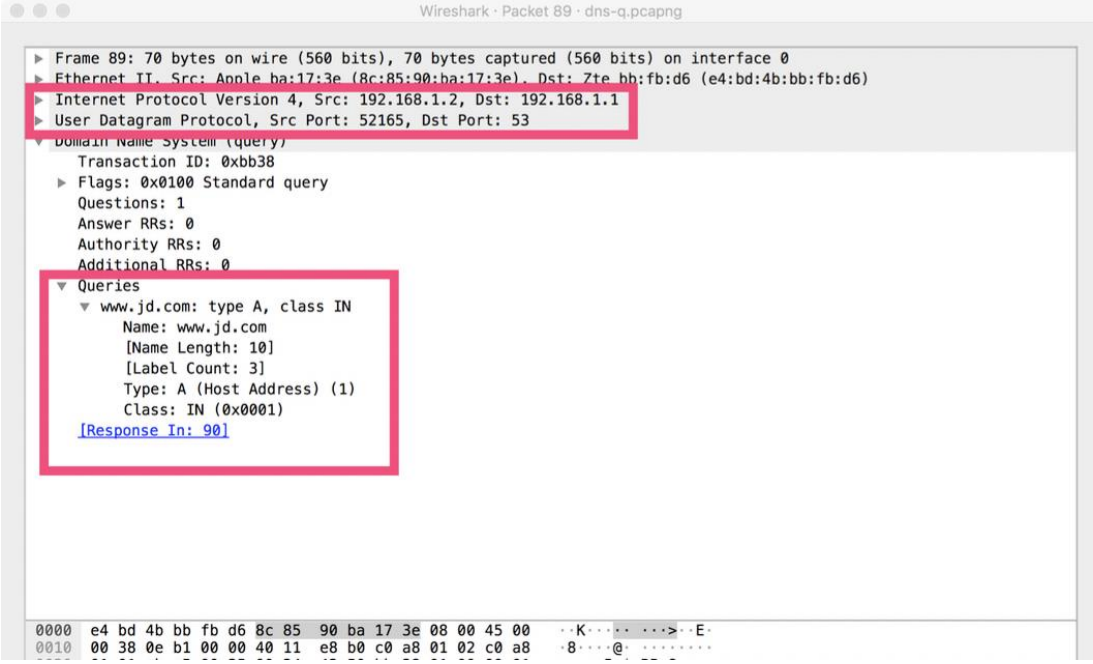
A 记录

下面是抓包图，过滤条件是 dns，即 protocol 为 dns。



请参看第 89 和 90 包，这是查询 www.jd.com 的 request 和 response。

第 89 包细节图如下：

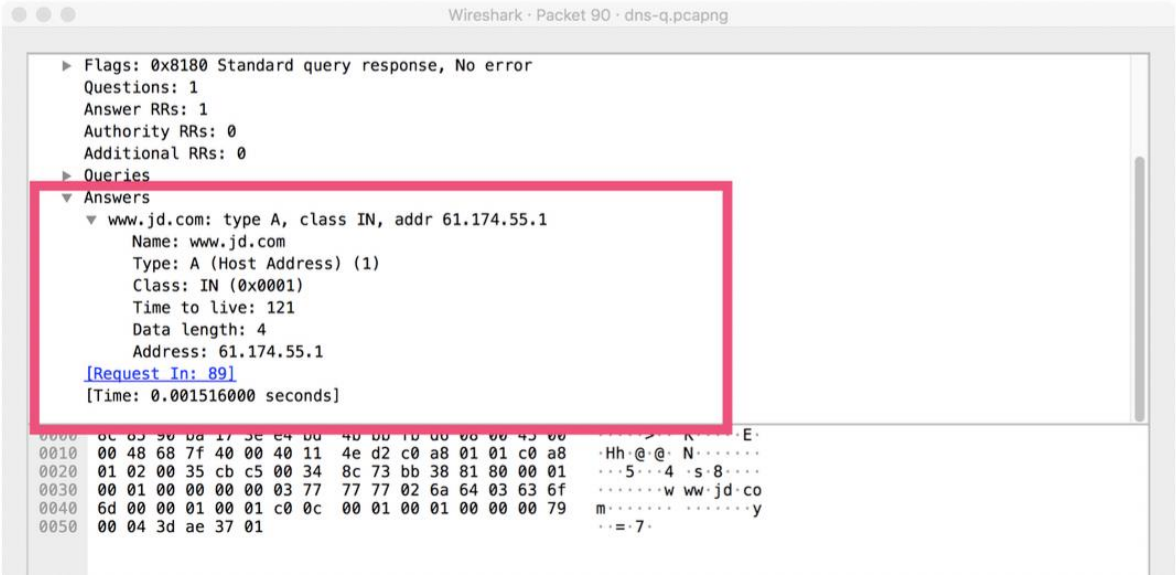


我们可以看到上图，Name 是 www.jd.com, Type 是 A 记录。

另外，我们可以看到其他信息，DNS 是通过 UDP 传送。其实，有时候也是通过 TCP 来传送。那什么时候用 TCP，什么时候用 UDP？很简单，当 response 包长度大于 512 字节时，就用 TCP，反之，则用 UDP。回头再看看 89 包，长度为 70，所以用 UDP 了。那再多问一个问题，什么情况下 DNS 查询包超过 512？CNAME 就有可能。可以参考下面的 CNAME 抓包。

DNS 使用的端口是多少呢？53，是的，DNS 用的是 53 端口，非常重要，一般防火墙要打开，否则 DNS 解析不了，也意味着无法访问域名的 website。

那么返回什么呢？继续看第 90 包。
我们可以看到 www.jd.com 的 DNS A 记录，IP 地址是 61.174.55.1。

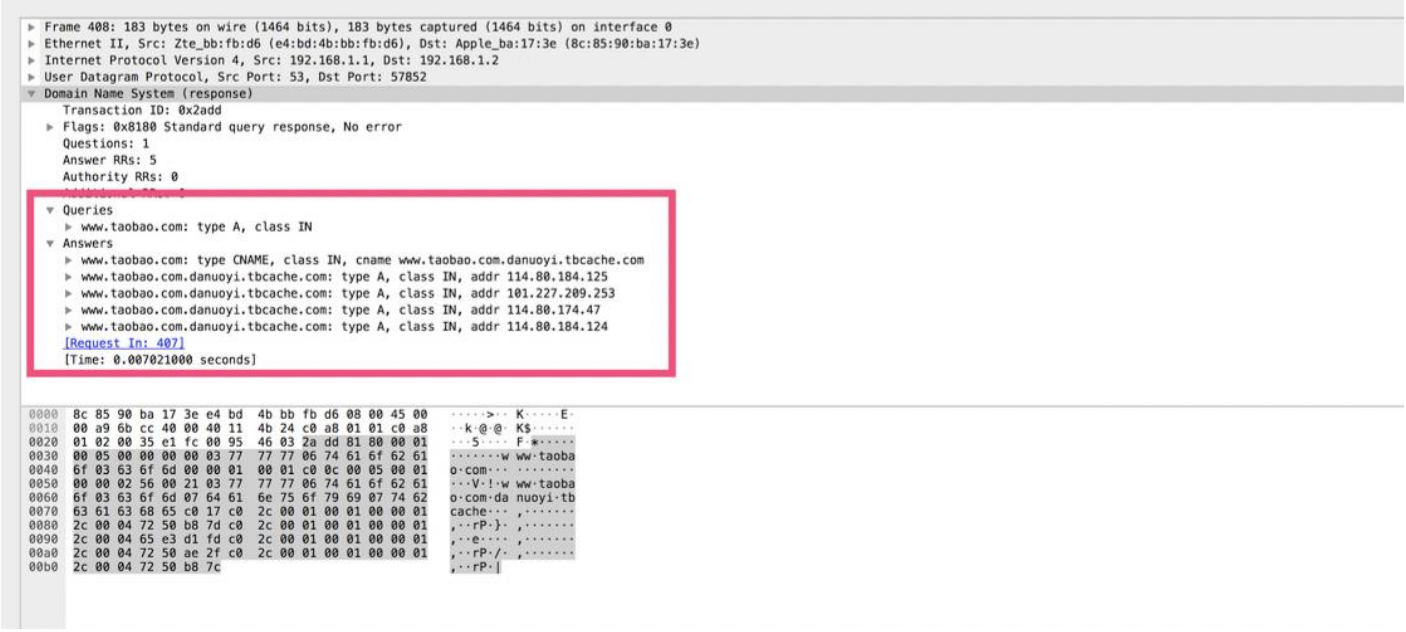


CNAME

下图是查询 www.taobao.com 的抓包。

我们看到 www.taobao.com 转到了 www.taobao.com.danuoyi.tbcache.com。

同时我们也可以看到 www.taobao.com.danuoyi.tbcache.com 有多条 A 记录，这里就是传说的 DNS 负载均衡。



DNS 标准和协议

DNS 是有标准和协议的，就跟 HTTP 类似的。DNS 标准和协议在 IETF 里。大家可以参考 https://en.wikipedia.org/wiki/Domain_Name_System 的 RFC documents 部分。

我认为学习协议，框架，或技术，最佳的方法就是阅读官方资料。例如，这里学习 DNS 可以直接看 RFC 文档，学习 Angular，可以访问其官网。

DNS 10 问

如果面试，下面 10 问基本可以覆盖全了，答案在上面已经说过了：

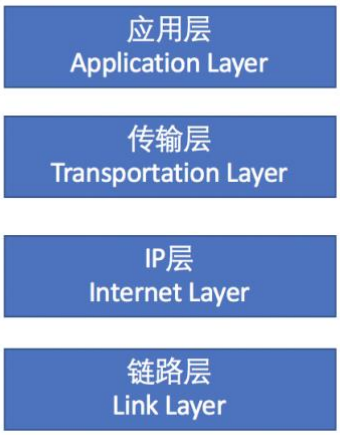
1. 为什么要用域名？
2. DNS 解析的基本流程？
3. DNS 的根域名是什么，有几个 Server？TLD DNS 是什么？
4. DNS 的优化策略是什么？在各个环节怎么做的？Chrome 和各个操作系统怎么做的？
5. DNS 负载均衡是什么，为什么要用？
6. DNS 的记录类型有哪些？CNAME 一般用在哪些场合？举例子说明一下。
7. DNS 的常用工具和命令有哪些？
8. DNS 查询是用 TCP 还是 UDP？一般用哪个端口？
9. DNS 抓包抓过吗？Wireshark 有用过吗？
10. 请说明一下 www.google.com 和 google.com 的区别。如何设置它们的 DNS？

TCP 连接

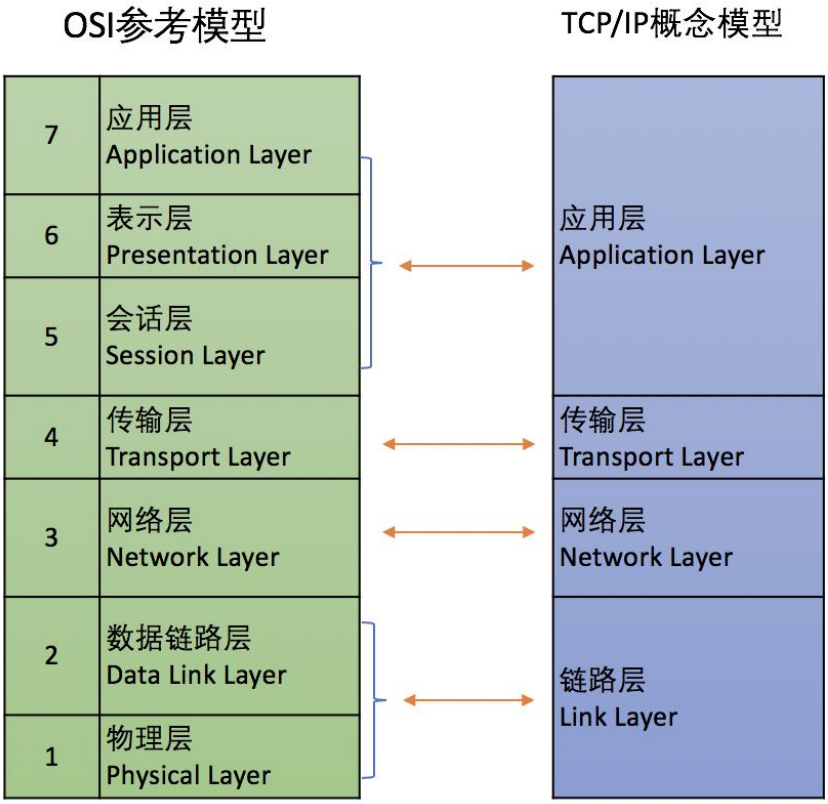
DNS 解析返回域名的 IP 之后，接下来就是浏览器要和该 IP 建立 TCP 连接了。为什么是 TCP 而不是 UDP？那是因为 HTTP 是基于 TCP 上的。这里涉及到另外一个话题：TCP/IP 模型。这个已经在大学的课本上学过了，我们再复习一下。

TCP/IP 模型

TCP/IP 模型一般分为 4 层，下面是我用 PPT 画的。

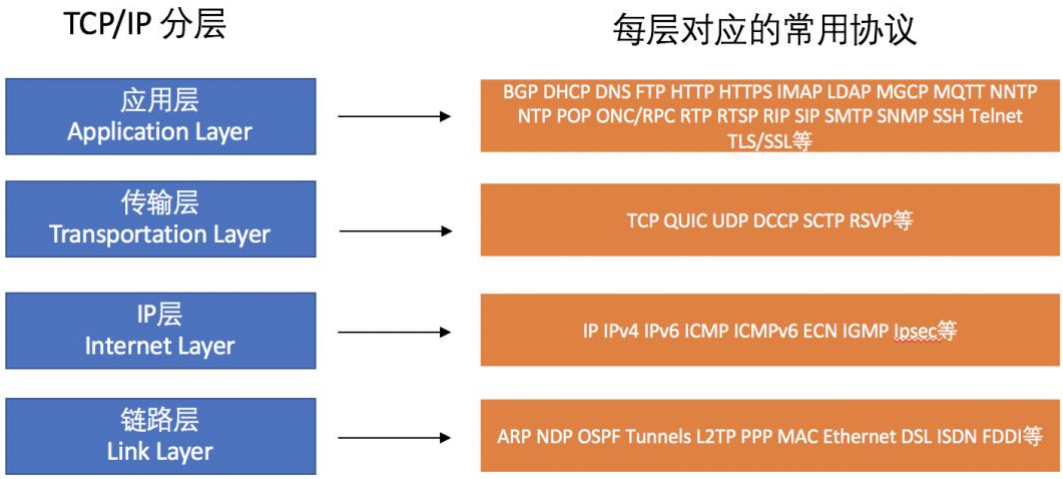


在这里不得不说 OSI 七层参考模型，和 TCP/IP 模型的区别和联系见下图：



不多解释，OSI 的 7/6/5 层和 TCP/TP 的应用层对应，2/1 层和链路层对应。在实际的应用中，主要还是 TCP/IP 概念模型，后面的内容主要讲它。

这些都是大学里课本上的，也许忘了（毕竟不是天天用到这些嘛），没关系，我们以最简单的方式来讲解。为了便于理解每层的含义和作用，先看每层有哪些协议，看看有没有自己熟悉的协议。有熟悉的协议，先体会一下。



应用层

我们可以看到，有常用的 HTTP/HTTPS/IMAP/SSH/Telnet 等都在应用层上（题外话，这一层你用的协议越多，说明你知识越开阔）。相信每个人都用过 HTTP/HTTPS，所以我上面说 HTTP/HTTP 是基于 TCP 上的。

Wikipedia 这么解释：

The application layer is the scope within which applications create user data and communicate this data to other applications on another or the same host. The applications, or processes, make use of the services provided by the underlying, lower layers, especially the Transport Layer which provides reliable or unreliable *pipes*to other processes. The communications partners are characterized by the application architecture, such as the client-server model and peer-to-peer networking. This is the layer in which all *higher level* protocols, such as SMTP, FTP, SSH, HTTP, operate. Processes are addressed via ports which essentially represent services.

传输层

没错，最常见的 TCP 和 UDP 就在这里，TCP 三次握手也在这里。

Wikipedia 这么解释：

The transport layer performs host-to-host communications on either the same or different hosts and on either the local network or remote networks separated by routers.[22] It provides a channel for the communication needs of applications. UDP is the basic transport layer protocol, providing an unreliable datagram service. The Transmission Control Protocol provides flow-control, connection establishment, and reliable transmission of data.

IP 层

IP 层非常重要，可能这么说还不太懂，看看其他协议。大家知道 ICMP 吗？估计很多人还是说不上来 ICMP 是什么东西。大家肯定用过 ping 命令吧，它就是用的 ICMP。说到这里，应该有感性的认识了吧。

Wikipedia 这么解释：

The internet layer exchanges datagrams across network boundaries. It provides a uniform networking interface that hides the actual topology (layout) of the underlying network connections. It is therefore also referred to as the layer that establishes internetworking. Indeed, it defines and establishes the Internet. This layer defines the addressing and routing structures used for the TCP/IP protocol suite. The primary protocol in this scope is the Internet Protocol, which defines IP addresses. Its function in routing is to transport datagrams to the next IP router that has the connectivity to a network closer to the final data destination.

链路层

这个非常底层了，ARP，NDP，Ethernet 都很常见，如果认真看过 HTTP 抓包，熟悉 LVS，Ngnix 等提供的负载均衡，应该对 ARP 不陌生，是的 ARP 用来查找设备的 MAC 地址，在 LVS 做负载均衡时会用到，因为进来的 stream 的包 MAC 地址要和出去的 Stream 包的 MAC 地址保持一致，因为做负载均衡，有可能会变化，如何解决这个问题，则不在本文的讨论范围内，如有兴趣可以参看 LVS 的文档。

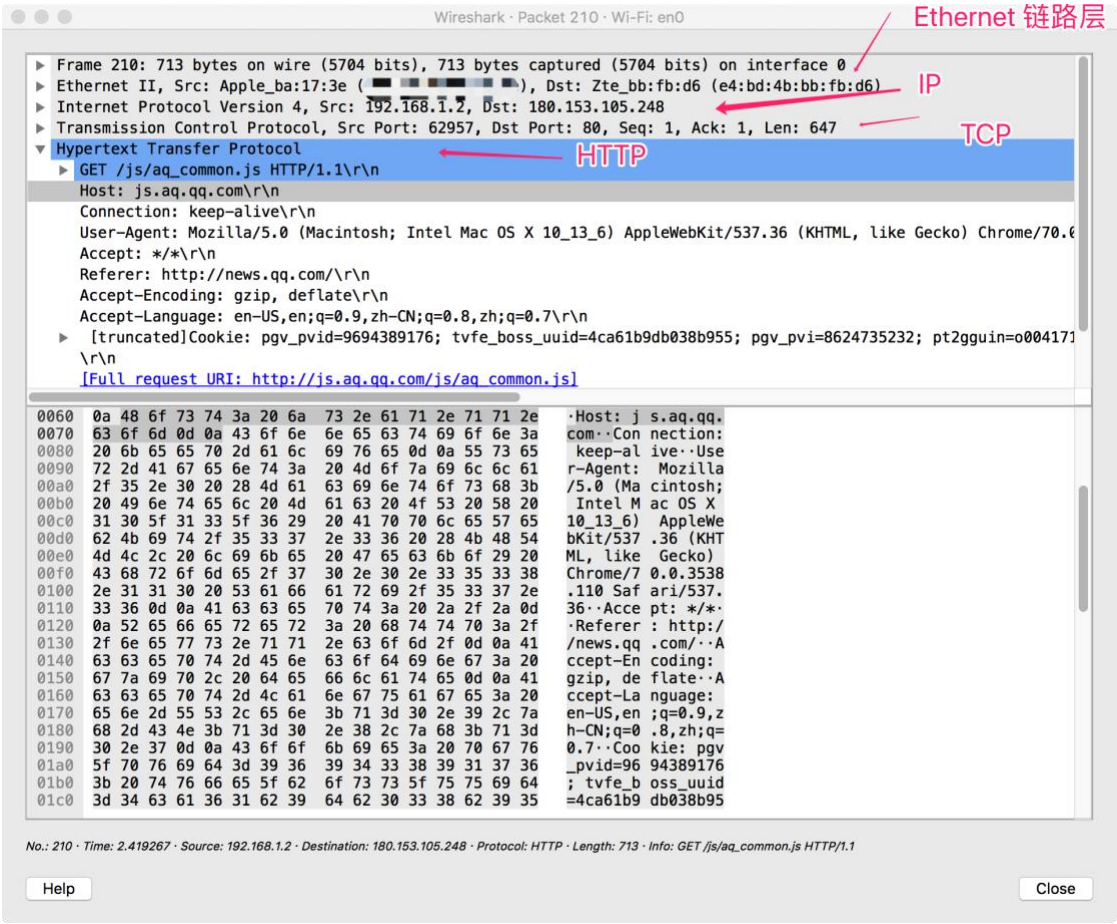
Wikipedia 这么解释：

The link layer defines the networking methods within the scope of the local network link on which hosts communicate without intervening routers. This layer includes the protocols used to describe the local network topology and the interfaces needed to effect transmission of Internet layer datagrams to next-neighbor hosts.

TCP/IP 抓包分析

看了前面的内容，还是觉得抽象吗？如果是，不要紧，也在预期内。让我们抓个包，分析认识一下就清楚了。

先访问 www.qq.com 这个主页，抓到的包如下：



看到这里，应该开始有感觉了吧。

HTTP，即应用层，正在访问 *js.aq.qq.com*.

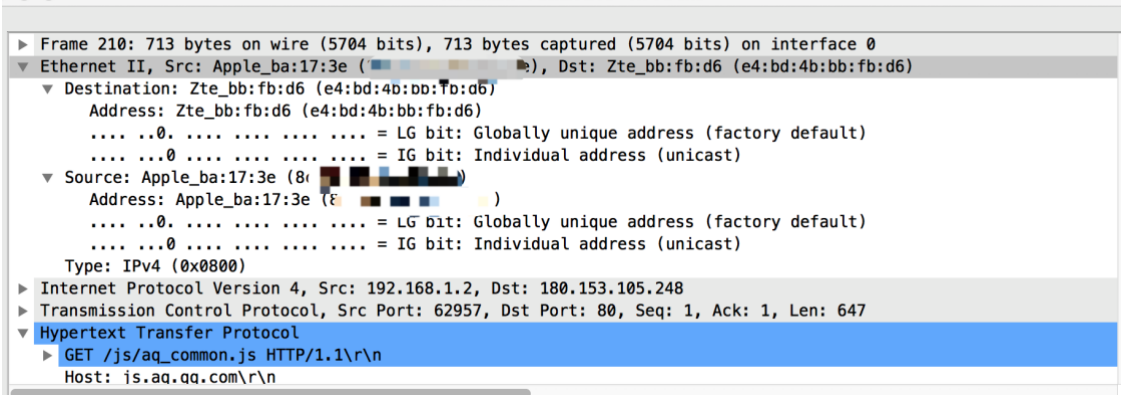
TCP 层，src port 是 62957， dst port 是 80 端口，因为 js.aq.qq.com 的端口是 80。

IP 层，用的是 IPV4，我的计算机 IP 地址是 192.168.1.2，目标 IP 是 180.153.105.248.

链路层， 我使用的是 Apple 电脑，我的 MAC 地址我隐藏了，对端是 ZTE 设备。

再截几个图大家仔细看一下。

链路层



IP 层


```
▼ Internet Protocol Version 4, Src: 192.168.1.2, Dst: 180.153.105.248
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 699
    Identification: 0x0000 (0)
  ▶ Flags: 0x4000, Don't fragment
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x5801 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.1.2
    Destination: 180.153.105.248
```

TCP 层

```
▼ Transmission Control Protocol, Src Port: 62957, Dst Port: 80, Seq: 1, Ack: 1, Len: 647
    Source Port: 62957
    Destination Port: 80
    [Stream index: 24]
    [TCP Segment Len: 647]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 648    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 4133
    [Calculated window size: 4133]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xc636 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▶ [SEQ/ACK analysis]
  ▶ [Timestamps]
    TCP payload (647 bytes)
```

应用层（HTTP）

```
▼ Hypertext Transfer Protocol
  ▶ GET /js/aq_common.js HTTP/1.1\r\n
    Host: js.aq.qq.com\r\n
    Connection: keep-alive\r\n
    User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, l
    Accept: */*\r\n
    Referer: http://news.qq.com/\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7\r\n
  ▶  [truncated]Cookie: pgv_pvid=9694389176; tvfe_boss_uuid=4ca61b9db038b955; pgv_pvi=862473
    \r\n
    [Full request URI: http://js.aq.qq.com/js/aq_common.js]
    [HTTP request 1/1]
    [Response in frame: 217]
```

在这里不加以详解，后面会对 TCP 以及 HTTP 层详详细说明。

TCP 三次握手与四次挥手

以下部分内容节选自 https://hit-alibaba.github.io/interview/basic/network/TCP.html。

TCP 三次握手

所谓三次握手(Three-way Handshake)，是指建立一个 TCP 连接时，需要客户端和服务器总共发送 3 个包。

三次握手的目的是连接服务器指定端口，建立 TCP 连接，并同步连接双方的序列号和确认号，交换 TCP 窗口大小信息。在 socket 编程中，客户端执行 `connect()` 时。将触发三次握手。

- 第一次握手(SYN=1, seq=x):

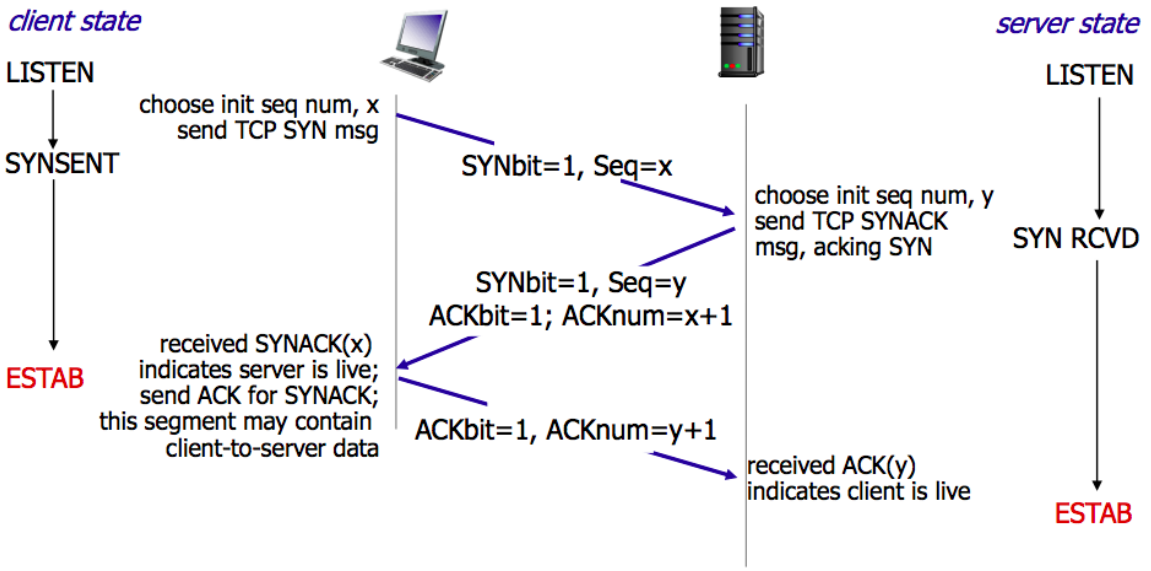
  客户端发送一个 TCP 的 SYN 标志位置 1 的包，指明客户端打算连接的服务器的端口，以及初始序号 X,保存在包头的序列号(Sequence Number)字段里。

  发送完毕后，客户端进入 `SYN_SEND` 状态。
- 第二次握手(SYN=1, ACK=1, seq=y, ACKnum=x+1):

  服务器发回确认包(ACK)应答。即 SYN 标志位和 ACK 标志位均为 1。服务器端选择自己 ISN 序列号，放到 Seq 域里，同时将确认序号(Acknowledgement Number)设置为客户的 ISN 加 1，即 X+1。 发送完毕后，服务器端进入 `SYN_RCVD` 状态。
- 第三次握手(ACK=1，ACKnum=y+1)

  客户端再次发送确认包(ACK)，SYN 标志位为 0，ACK 标志位为 1，并且把服务器发来 ACK 的序号字段+1，放在确定字段中发送给对方，并且在数据段放写 ISN 的+1

  发送完毕后，客户端进入 `ESTABLISHED` 状态，当服务器端接收到这个包时，也进入 `ESTABLISHED` 状态，TCP 握手结束。
  三次握手的过程的示意图如下：



确实比较抽象，让我们继续通过抓包来分析(访问 baidu.com)

No.	Time	Source	Destination	Protocol	Length	Info
270	18.613174	192.168.1.2	115.239.211.112	TCP	78	56717 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 W
274	18.624940	115.239.211.112	192.168.1.2	TCP	78	443 → 56717 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
275	18.625018	192.168.1.2	115.239.211.112	TCP	54	56717 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0

这里有三个包。

192.168.1.2 是我机器的 IP，115.239.211.112 是 baidu 的 IP，是不是和上图一致。

具体看 270 包，192.168.1.2 发送 SYN 到 115.239.211.112，seq=0；

如下图：


```
▼ Transmission Control Protocol, Src Port: 56717, Dst Port: 443, Seq: 0, Len: 0
    Source Port: 56717
    Destination Port: 443
    [Stream index: 19]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    [Next sequence number: 0    (relative sequence number)]
    Acknowledgment number: 0
    1011 .... = Header Length: 44 bytes (11)
  ▶ Flags: 0x002 (SYN)
    Window size value: 65535
    [Calculated window size: 65535]
    Checksum: 0x8cee [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▼ Options: (24 bytes), Maximum segment size, No-Operation (NOP), Window scale, No
    ▶ TCP Option - Maximum segment size: 1460 bytes
    ▶ TCP Option - No-Operation (NOP)
    ▶ TCP Option - Window scale: 5 (multiply by 32)
    ▶ TCP Option - No-Operation (NOP)
    ▶ TCP Option - No-Operation (NOP)
    ▶ TCP Option - Timestamps: TSval 1361579357, TSecr 0
    ▶ TCP Option - SACK permitted
    ▶ TCP Option - End of Option List (EOL)
```

274 包图，115.239.211.112 返回 SYN 和 ACK 给 192.168.1.2， seq =0， 但是 ACK 等于 SYN 里的 seq（为 0）+1，所以为 1

```
▼ Transmission Control Protocol, Src Port: 443, Dst Port: 56717, Seq: 0, Ack: 1, Len: 0
    Source Port: 443
    Destination Port: 56717
    [Stream index: 19]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    [Next sequence number: 0    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    1011 .... = Header Length: 44 bytes (11)
  ▶ Flags: 0x012 (SYN, ACK)
    Window size value: 8192
    [Calculated window size: 8192]
    Checksum: 0xd807 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (24 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Ope
  ▼ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 270]
      [The RTT to ACK the segment was: 0.011766000 seconds]
      [iRTT: 0.011844000 seconds]
```

275 包图，192.168.1.2 收到 ACK 包后，给 115.239.211.112 再回一个 ACK，ACK#为 1：


```
▼ Transmission Control Protocol, Src Port: 56717, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
    Source Port: 56717
    Destination Port: 443
    [Stream index: 19]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  ▶ Flags: 0x010 (ACK)
    Window size value: 8192
    [Calculated window size: 262144]
    [Window size scaling factor: 32]
    Checksum: 0x4de1 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▼ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 274]
      [The RTT to ACK the segment was: 0.000078000 seconds]
      [iRTT: 0.011844000 seconds]
  ▶ [Timestamps]
```

对于协议的理解，还是多观察，多比对，就知道是怎么回事了。

TCP 四次挥手

TCP 连接的拆除需要发送四个包，因此称为四次挥手(Four-way handshake)，也叫做改进的三次握手。客户端或服务器均可主动发起挥手动作，在 socket 编程中，任何一方执行 `close()` 操作即可产生挥手操作。

- 第一次挥手(FIN=1，seq=x)

  假设客户端想要关闭连接，客户端发送一个 FIN 标志位置为 1 的包，表示自己已经没有数据可以发送了，但是仍然可以接受数据。

  发送完毕后，客户端进入 `FIN_WAIT_1` 状态。
- 第二次挥手(ACK=1，ACKnum=x+1)

  服务器端确认客户端的 FIN 包，发送一个确认包，表明自己接受到了客户端关闭连接的请求，但还没有准备好关闭连接。

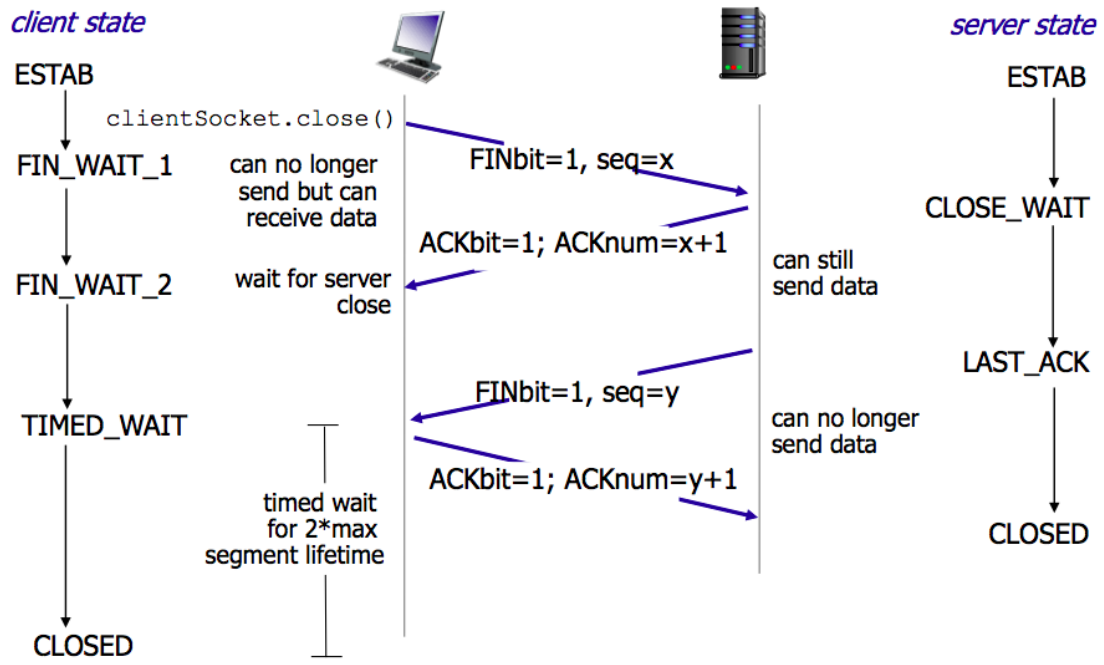
  发送完毕后，服务器端进入 `CLOSE_WAIT` 状态，客户端接收到这个确认包之后，进入 `FIN_WAIT_2` 状态，等待服务器端关闭连接。
- 第三次挥手(FIN=1，seq=y)

  服务器端准备好关闭连接时，向客户端发送结束连接请求，FIN 置为 1。

  发送完毕后，服务器端进入 `LAST_ACK` 状态，等待来自客户端的最后一个 ACK。
- 第四次挥手(ACK=1，ACKnum=y+1)

客户端接收到来自服务器端的关闭请求，发送一个确认包，并进入 TIME_WAIT 状态，等待可能出现的要求重传的 ACK 包。
服务器端接收到这个确认包之后，关闭连接，进入 CLOSED 状态。
客户端等待了某个固定时间（两个最大段生命周期，2MSL，2 Maximum Segment Lifetime）之后，没有收到服务器端的 ACK ，认为服务器端已经正常关闭连接，于是自己也关闭连接，进入 CLOSED 状态。

四次挥手的示意图如下：

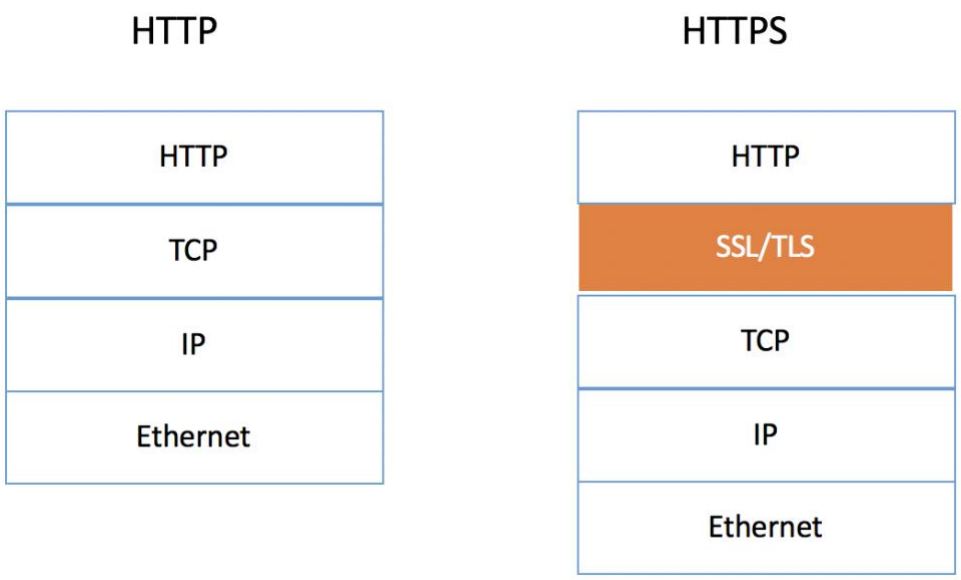


这里就不抓包了，可以自行抓包对比看看。

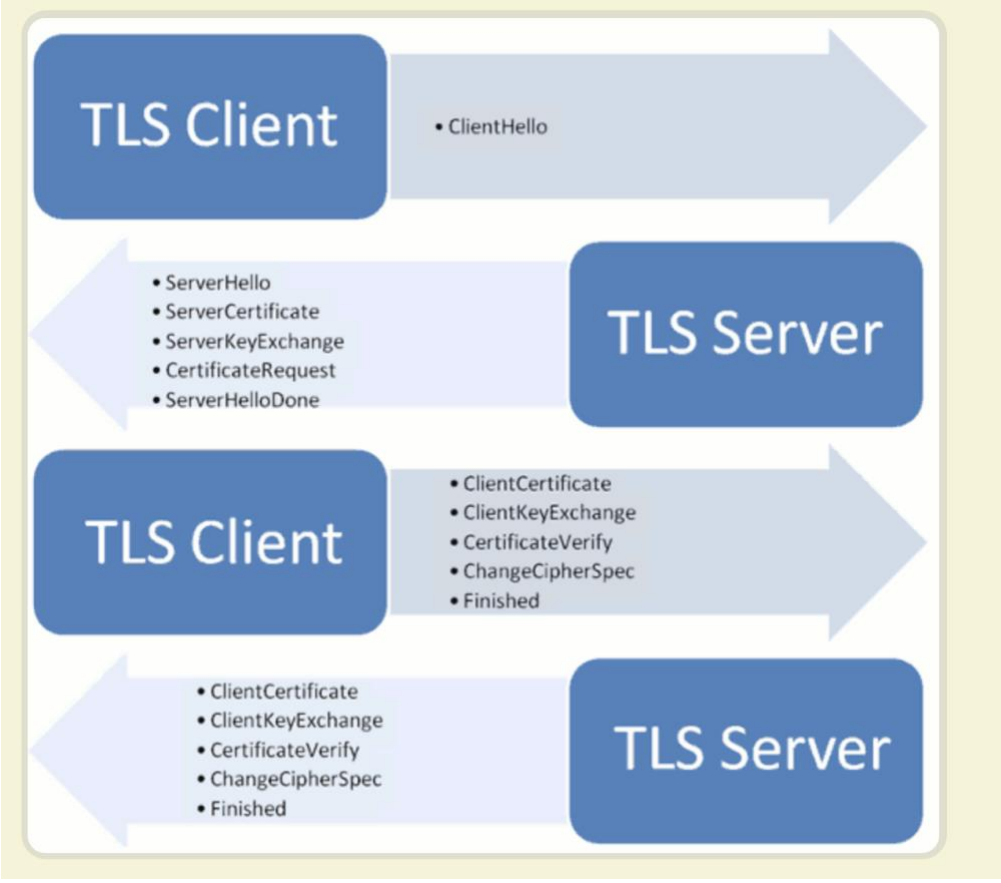
HTTPS 证书

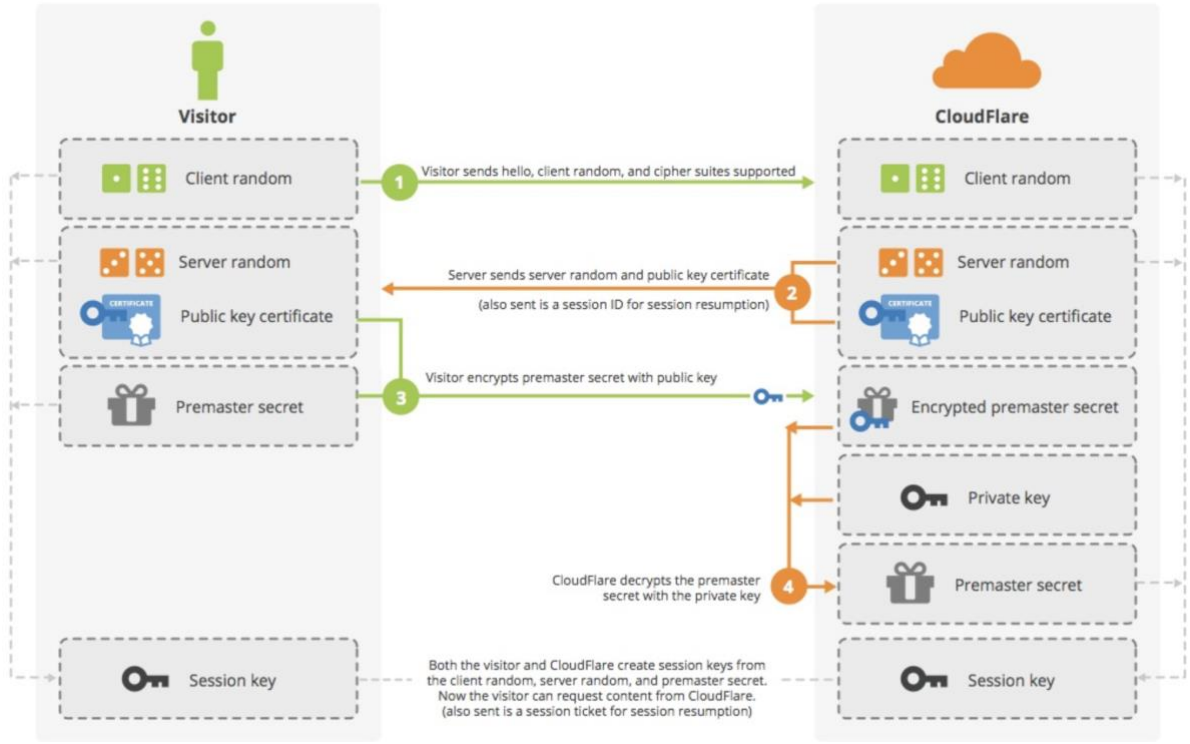
越来越多的网站开始使用 HTTPS（Apple 要求 App 都须用 HTTPS）。对于 HTTPS，需要有一个 SSL/TLS 的鉴权/认证，才能建立 TCP 链接。

下图描述了 HTTP 和 HTTPS 的区别。



关于 SSL/TLS 如何交互等，可以参看阮一峰老师的文章 http://www.ruanyifeng.com/blog/2014/02/ssl_tls.html 和 http://www.ruanyifeng.com/blog/2014/09/illustration-ssl.html，写非常清楚的。借用一张阮一峰老师的图：





该图来自于 https://blog.cloudflare.com/keyless-ssl-the-nitty-gritty-technical-details/。

我们还是抓包体会一下。

tcp.stream eq 18 and ssl

No.	Time	Source	Destination	Protocol	Length	Info
273	18.622930	192.168.1.2	115.239.211.112	TLSv1.2	571	Client Hello
278	18.636374	115.239.211.112	192.168.1.2	TLSv1.2	122	Server Hello
283	18.641411	115.239.211.112	192.168.1.2	TLSv1.2	656	Certificate
294	18.685677	115.239.211.112	192.168.1.2	TLSv1.2	392	Server Key Exchange
295	18.685680	115.239.211.112	192.168.1.2	TLSv1.2	63	Server Hello Done
298	18.686329	192.168.1.2	115.239.211.112	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
299	18.692906	192.168.1.2	115.239.211.112	TLSv1.2	933	Application Data
307	18.711871	115.239.211.112	192.168.1.2	TLSv1.2	229	New Session Ticket
308	18.711875	115.239.211.112	192.168.1.2	TLSv1.2	60	Change Cipher Spec
309	18.711875	115.239.211.112	192.168.1.2	TLSv1.2	99	Encrypted Handshake Message
327	19.324400	115.239.211.112	192.168.1.2	TLSv1.2	331	Application Data
330	19.324983	115.239.211.112	192.168.1.2	TLSv1.2	62	Application Data
333	19.325495	115.239.211.112	192.168.1.2	TLSv1.2	78	Application Data
337	19.326503	115.239.211.112	192.168.1.2	TLSv1.2	90	Application Data

对着阮一峰老师图片的流程看：

273 包： Client 发送 Client Hello 给 Server。

278 包： Server 回 Server Hello 给 Client。

283 包： Server 继续回 Server Certificate 给 Client，要交换证书。

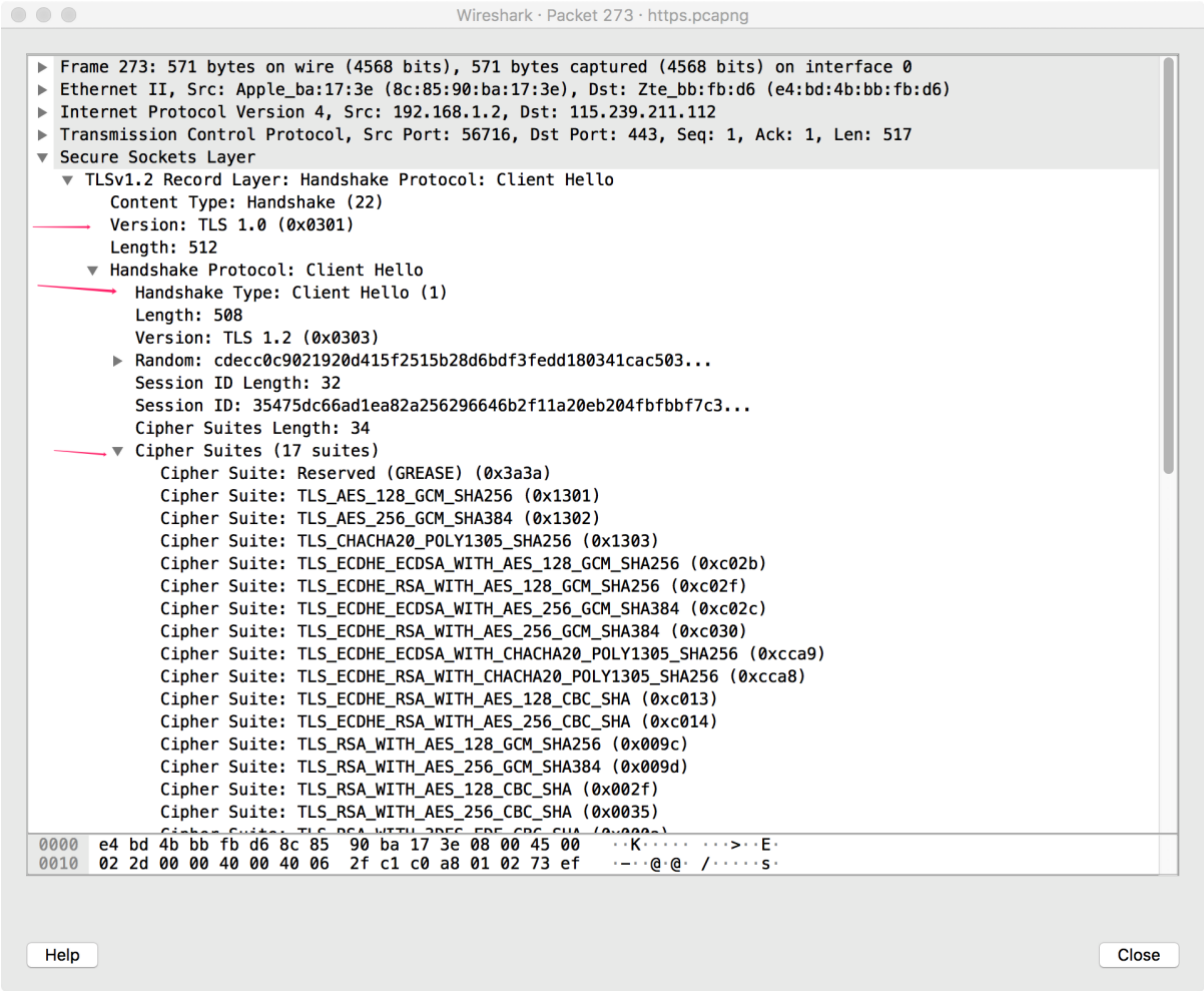
294 包：Server 继续回 Server Key Exchange，交换 key。

295 包：Server 返回 Server Hello Done。

截止这里，Server 已经做了不少事，接下来轮到 Client 了。

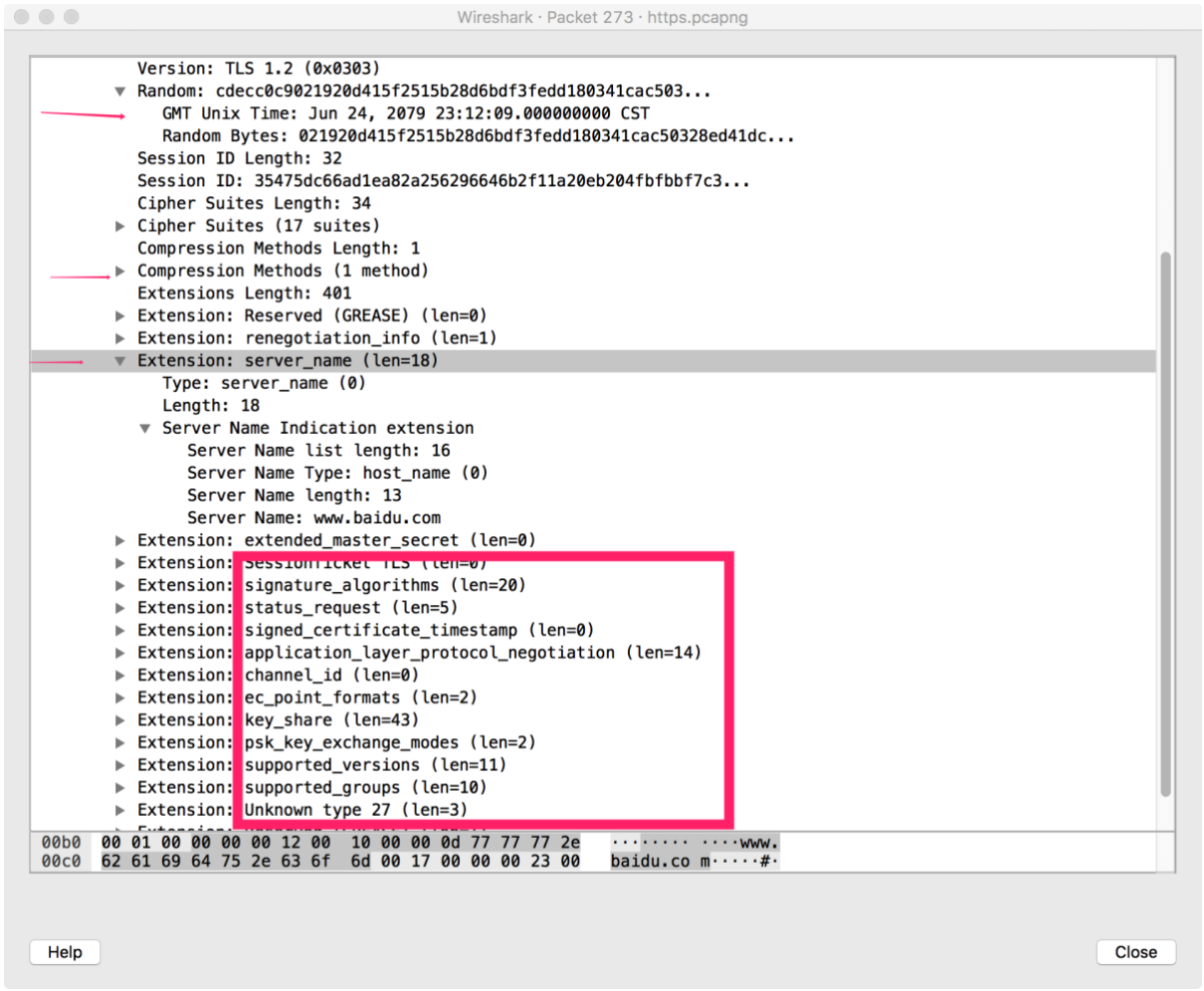
298 包：Client Key Exchange，Change Cipher Spec，Encrypted Handshake Message。到这里就结束了。

我们再仔细看看 Client Hello 的 273 包，注意标记部分。

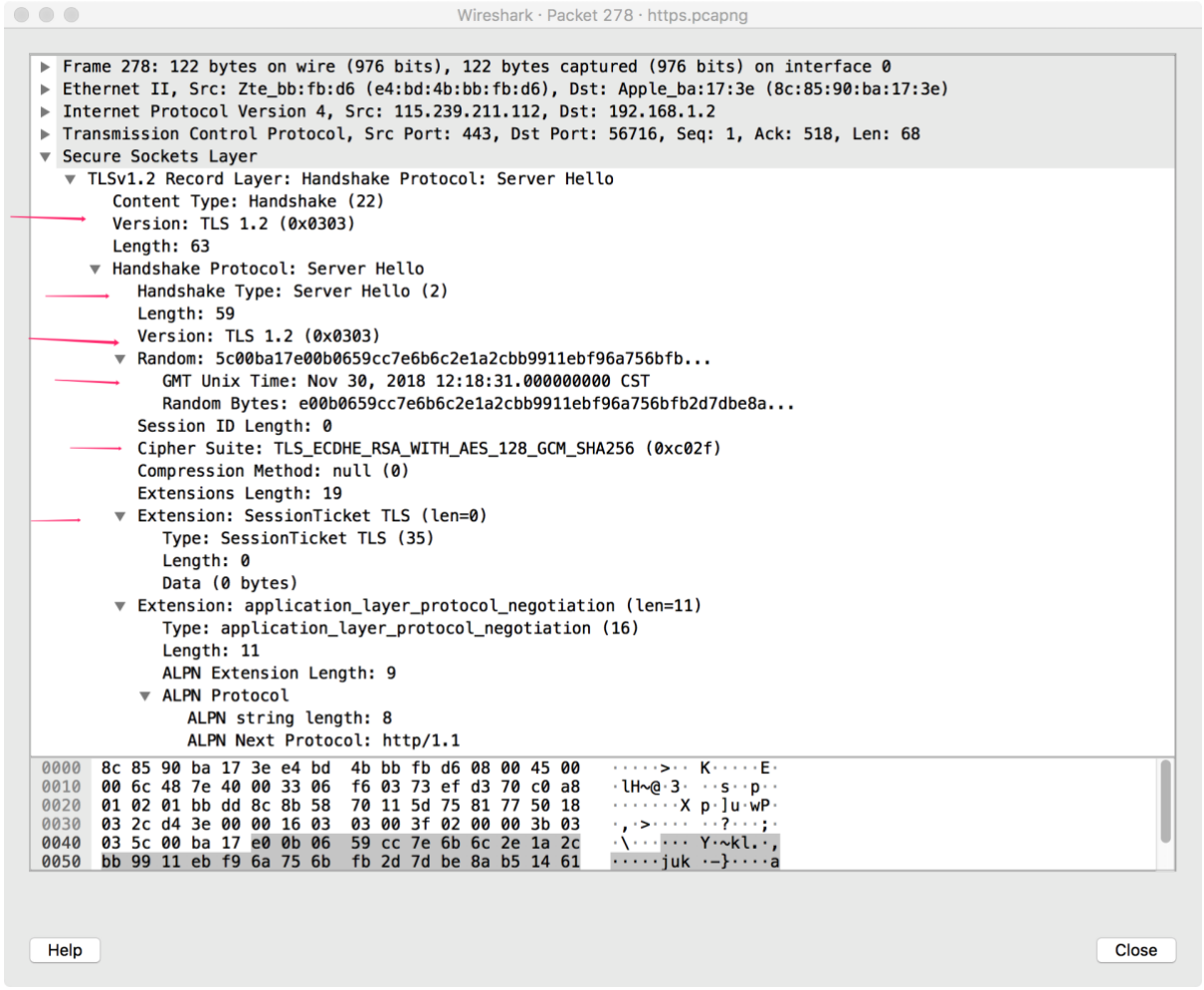


接下来再看看 Extension，有很多对不对。这里专门说一下 SNI（Server Name Indication），server_name, 我们可以看见它的值是 www.baidu.com. 也就是说浏览器发送过来的证书是给 baidu 这个域名签发的。SNI 用来校验该证书是不是为 server name 提供的域名签发的。如果不是，就会报错。

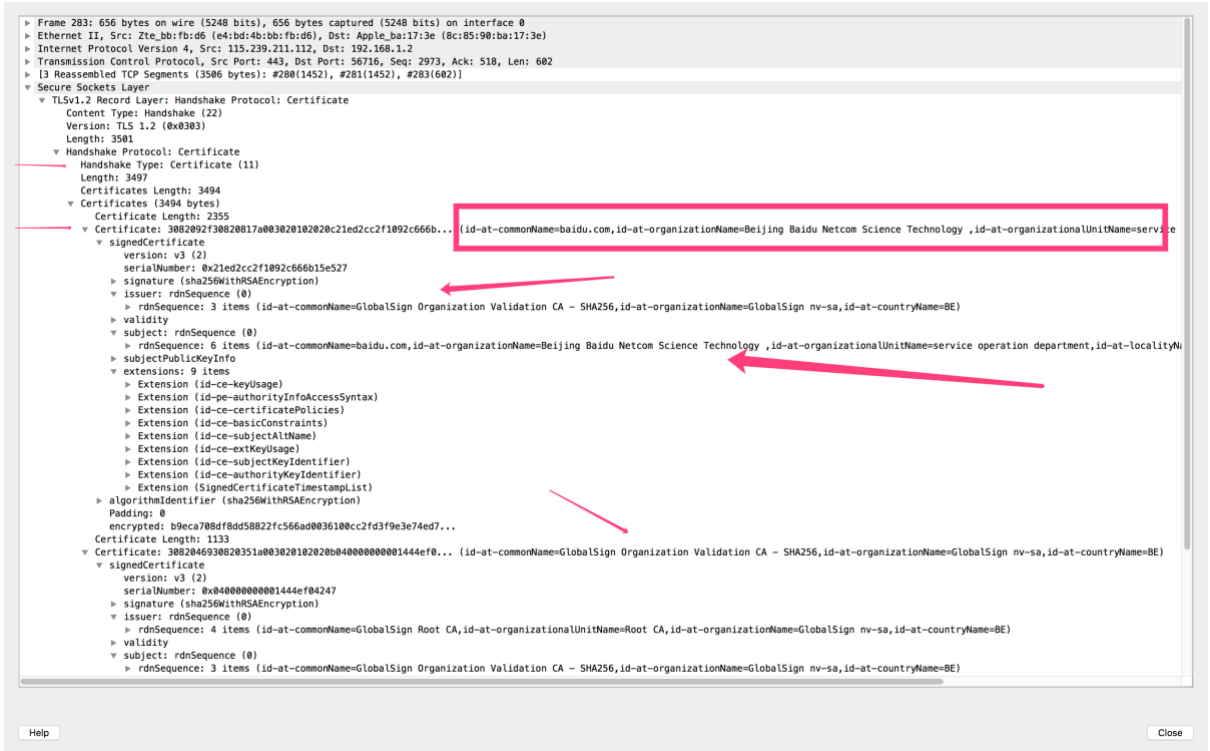
有一种情况特别需要注意，在早期版本的浏览器或者 HTTP 客户端，SNI 可能不包含在该包里的，那么怎么处理呢？如果 Server 端只有一个证书部署，那简单，就是按照部署的那个证书去判断。如果有多个证书部署呢？比如部署了 aaa.com bbb.com ccc.com 三个域名的证书，那么就会按照缺省的去匹配，取决于软件（Apache，Tomcat 等）和硬件（F5，Netscaler）怎么配置了。
但是如果已经商用，去改默认配置，会对商用服务有影响，那么可不可以在 Client 有一些改进呢？如果您正在使用某个 HTTP library，可以考虑升级版本是否支持。



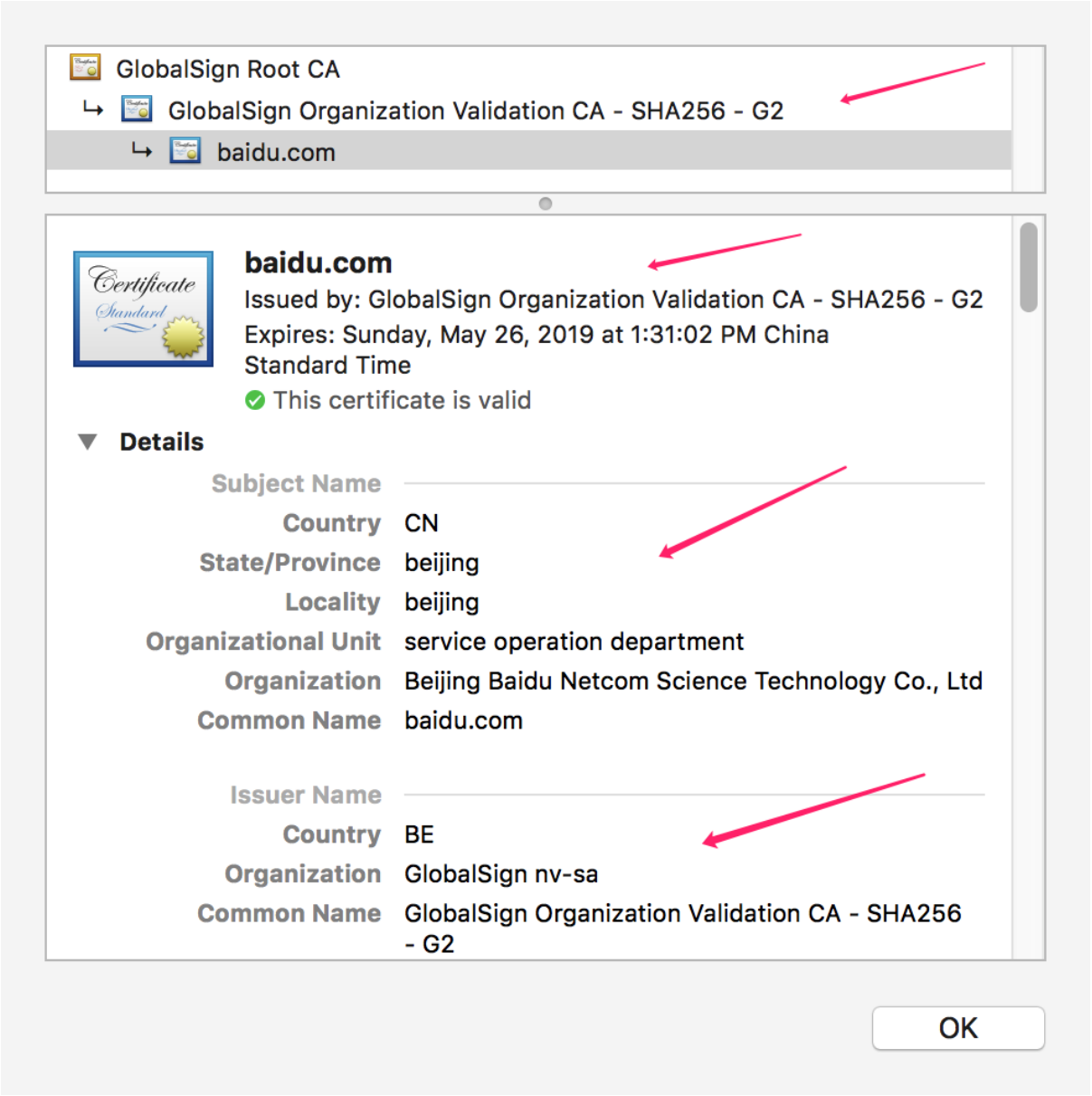
278 包的 Server Hello



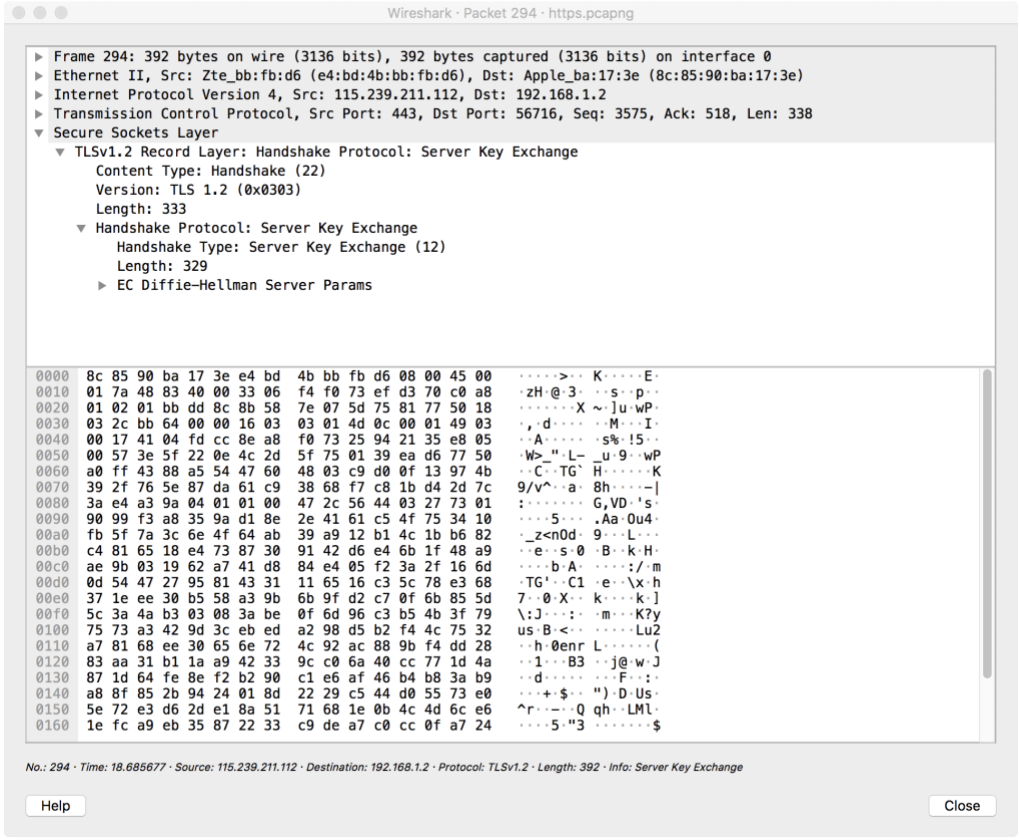
继续看看 Server Certificate，即 283 包



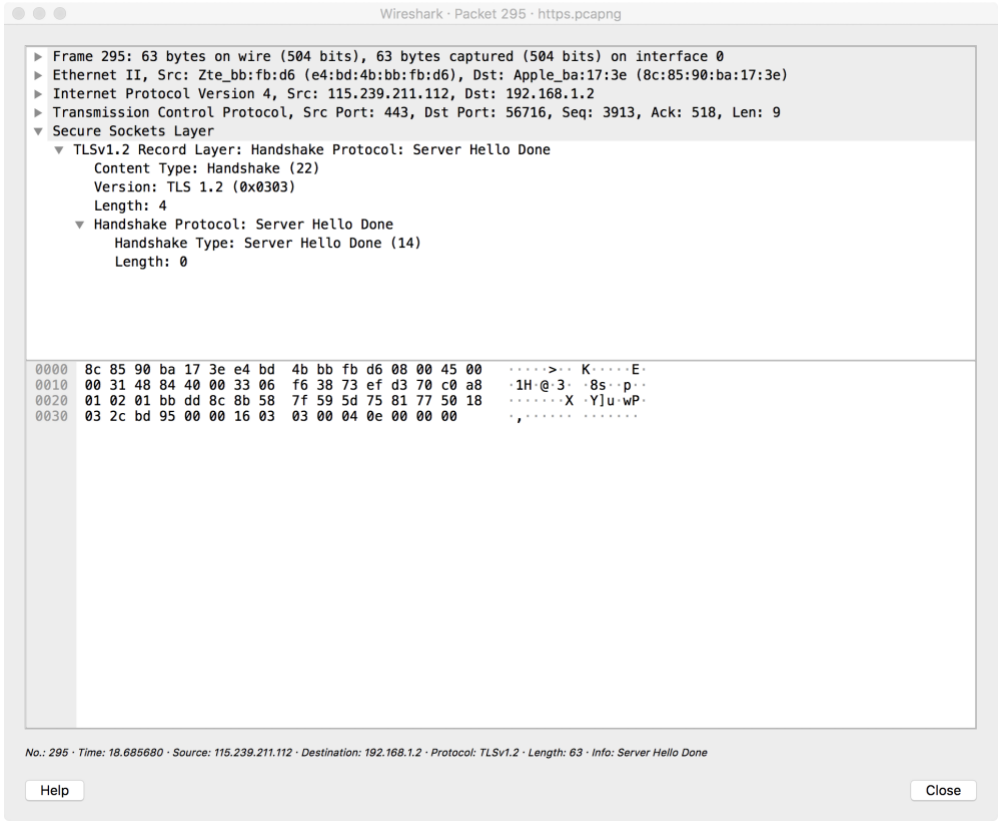
我们看看 baidu 的证书，打开 Chrome 就可以看到了，对比一下两图的基本信息，这时是不是觉得更容易理解？我相信答案是肯定的。



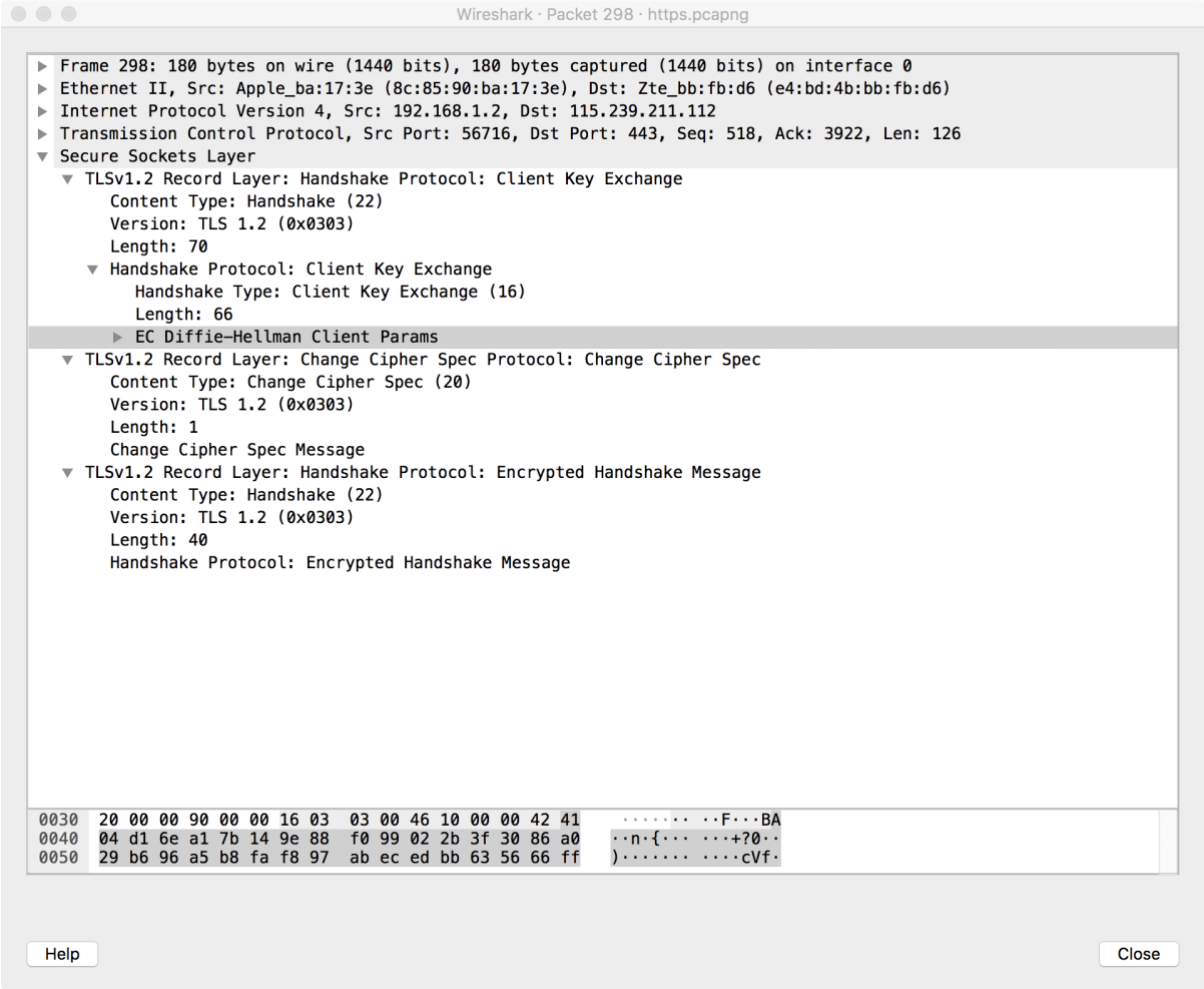
294 包，Server key Exchange



295 包，Server Hello Done。



298 包，Client Key Exchange/Change Cipher Spec/Encrypted Handshake Message。



TCP/IP 其他

上面都是最基本的东西，在实际的过程中还有包重试，包拼装等，太底层了，大家有兴趣可以找资料看看。

TCP/IP 10 问

以下几个问题大部分都可以找到答案。

1. TCP/IP 的 4 层模型了解吗？每层有哪些常见协议？
2. TCP/IP 的三次握手了解吗？四次挥手是什么，了解多少？
3. HTTP 和 HTTPS 在 TCP 握手上有什么不同？SSL/TLS 握手流程了解吗？
4. SSL/TLS 的版本有哪些？当前浏览器支持哪些版本？
5. SNI 了解多少？如果 SNI 没有，该如何校验证书？
6. TCP 与 UDP 区别在哪里？
7. 为什么 TCP 经常会组装包？如何保证包的完整性？
8. TCP 滑动窗口原理是什么？TCP 有哪些状态？

9. MAC 地址的是如何定义的？（*这个问题太 Edge 了*）
10. SSL/TLS 证书和端口有关系吗？为什么？

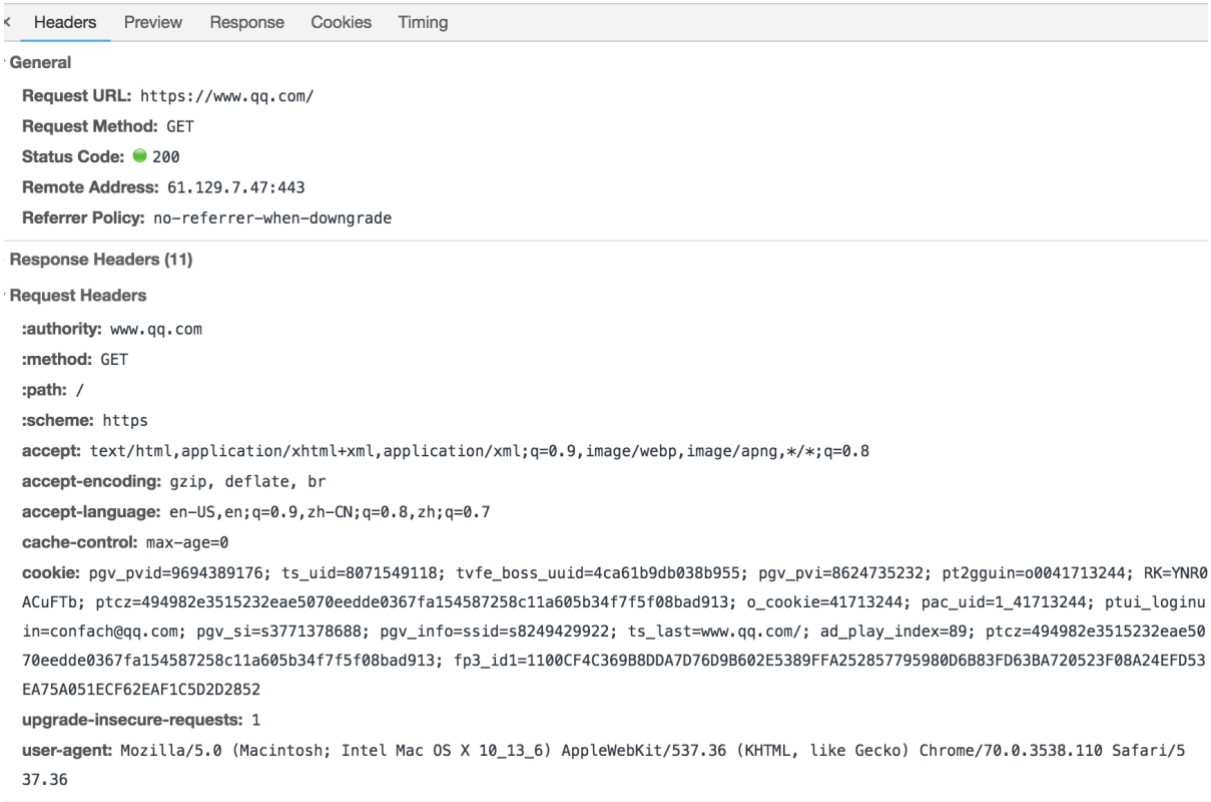
HTTP/HTTPS 请求和响应

前面 TCP 连接已经建立好了，意味着桥已经搭好了，下一步就该传输 HTTP 消息了。因为 HTTP 我们都很熟悉，很常见，也不是那么底层，理解起来轻松不少。

还是抓包来分析，不过这次不用 Wireshark 来抓，因为不太直观，这次直接用 Chrome 自带的 Developer Tools。

HTTP 请求

下图展示了 HTTP 请求 Header



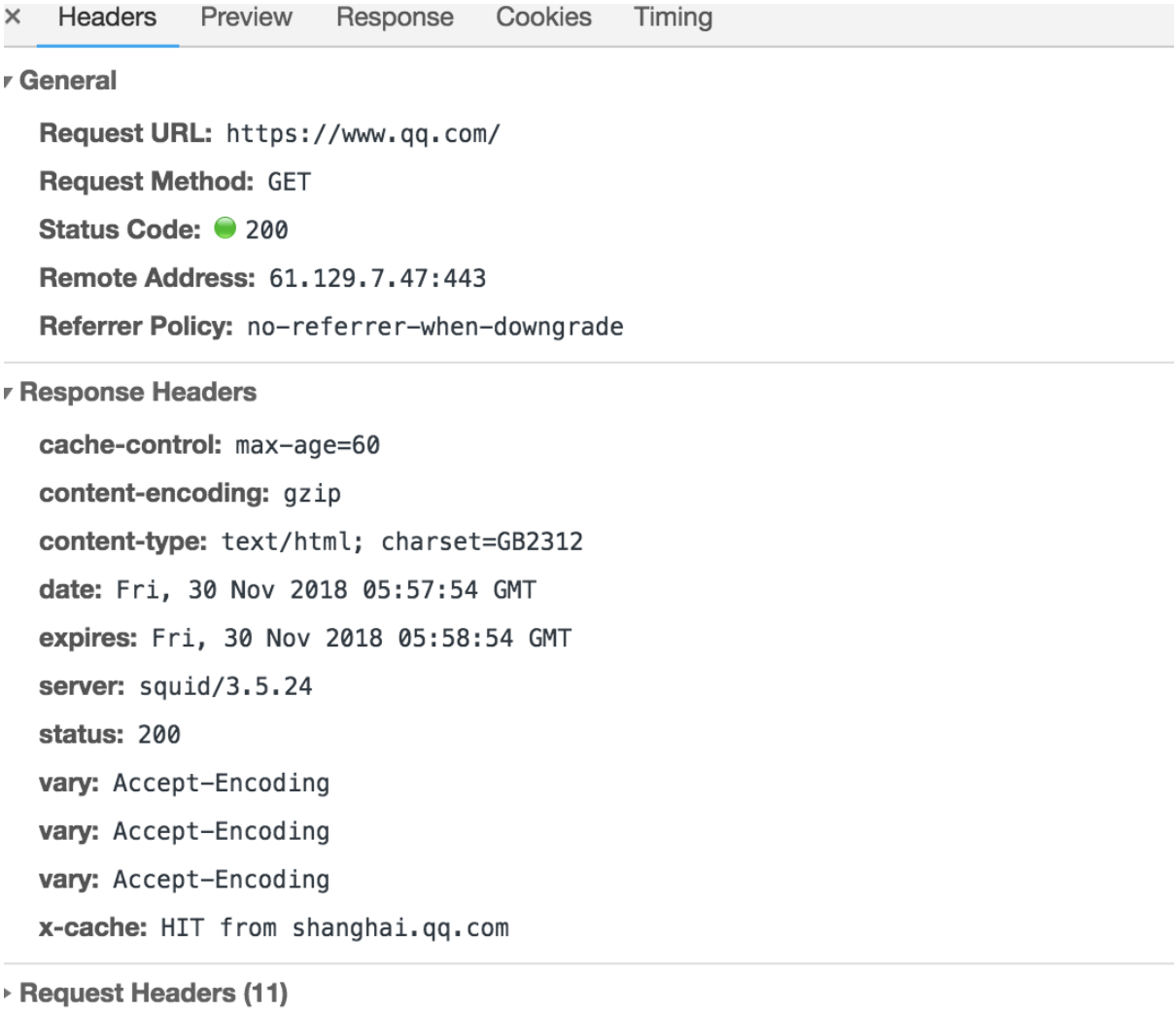
Request URL：就是请求的 URL

Request Method： GET， POST， PUT，DELETE， OPTIONS， HEAD

接下来就是其他报文头，常见的请求报头有: Accept, Accept-Charset, Accept-Encoding, Accept-Language, Content-Type, Authorization, Cookie, User-Agent 等。

HTTP 响应

下图展示了 HTTP 响应 Header



最常见的就是 Status Code（ 200， 302， 307， 404， 500），server 等。

HTTP 10 问

HTTP 问题简单，那就直接列举几个问题，有些问题我给出详细答案。

1. **HTTP METHOD 有哪几种，分别是什么？**
   常见问题，不多解释。

2. **HTTP 的 PUT/DELETE/等使用时需要注意什么？**
   有一点需要特别注意，有的浏览器是不支持的，所以在使用和实现时需要仔细评估好自己客户端的能力。

3. **HTTP 的 OPTIONS 用来做什么？**
   OPTIONS 一般用来获取目标资源的通信选项，例如确认允许的 HTTP Method.下面的这个例子来在 Mozilla 官网 https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/OPTIONS。

   请求：
   curl -X OPTIONS http://example.org -i

响应：

HTTP/1.1 200 OK
Allow: OPTIONS, GET, HEAD, POST
Cache-Control: max-age=604800
Date: Thu, 13 Oct 2016 11:45:00 GMT
Expires: Thu, 20 Oct 2016 11:45:00 GMT
Server: EOS (lax004/2813)
x-ec-custom-error: 1
Content-Length: 0

看红色的返回部分，意思是说该 URL 允许的 HTTP Method 为 OPTIONS、GET、HEAD 以及 POST。

那么 OPTIONS 一般用在哪里呢？是的，CORS。如果您开发过 SPA，或者您的前端的逻辑和交互完全用 JavaScript 来实现的，肯定会碰到此问题。所谓 JavaScript 实现，目前比交流行的有 VueJS，AngularJS，React 等流行框架。如何解决这个问题？最常用的方法是是前端和后端在同一个域名下。如果不在同一个域名下，需要在后端实现支持 CORS 的功能，目前 Spring/Spring Boot 已经有类似功能支持 CORS，实现起来蛮简单的，具体可以参看 https://spring.io/blog/2015/06/08/cors-support-in-spring-framework，不在赘述。

关于 CORS，最难的不是这里，在我遇到的 Case 里，比这个更复杂，如果前端是 SPA，而且在某种情况下，访问任何前端页面都会进行跳转，这是需求。但是因为是 SPA 的架构，会出现 OPTIONS（这是浏览器关于 CORS 支持的流程），这会使整个 HTTP 流程变得紊乱，也是一个很棘手的问题。

不管怎么说，CORS 属于安全性这一块，后面在软件安全（包括 Web）方面专门写吧。在过去几年里，因为公司和客户在产品安全和数据安全放在了一个很高的位置，而我又是去负责这一块，所以自己在安全方面积累了大量的实战经验，非常宝贵。

4. **上述访问 qq 的 HTTP response 里的 server 哪里不合适，需要注意什么？**
server 的值是 *squid/3.5.24*，不合适之处在哪里？也许部分人是不清楚的。其实很简单，不应该把 squid 的版本信息放在这里。为什么？

如果 web server 软件是自己公司开发的，私有的，不开源的，这也罢了，没人知道你的 web server 是怎么实现的，但这并不代表 web server 没有漏洞，不代表别人发现不了漏洞。

但如果用的是开源的，例如 Apache HTTPd，Apache Tomcat，Ngnix 等，就需要注意了，每个版本都有安全漏洞，而且这些安全漏洞都有专门的记录，看看隔三差五报告的 CVE，就知道漏洞多少了。所以，如果使用开源软件，很容易将自己的 web server 处于一个具有潜在风险的位置，所以不要加上版本号。

腾讯作为全球顶级的大公司，这方面不注意，实在是有点说不过去。如果有腾讯的朋友看到这里，还是改过来吧。

和上面一样，这也是属于安全性问题，有时间我再写吧。

5. **Status Code 302 与 307 的区别是什么？**

   都属于跳转，但是区别在哪里呢？

   我们看看 307 在协议里是怎么定义的？参看 rfc2616 第 10 章节

   10.3.8 307 Temporary Redirect

   The requested resource resides temporarily under a different URI. Since the redirection MAY be altered on occasion, the client SHOULD continue to use the Request-URI for future requests. This response is only cacheable if indicated by a Cache-Control or Expires header field.

   The temporary URI SHOULD be given by the Location field in the response. Unless the request method was HEAD, the entity of the response SHOULD contain a short hypertext note with a hyperlink to the new URI(s) , since many pre-HTTP/1.1 user agents do not understand the 307 status. Therefore, the note SHOULD contain the information necessary for a user to repeat the original request on the new URI.

   If the 307 status code is received in response to a request other than GET or HEAD, the user agent MUST NOT automatically redirect the request unless it can be confirmed by the user, since this might change the conditions under which the request was issued.

   在 GET、HEAD 这些幂等的请求方式上，302、307 没区别，但对于 POST 就不同了，大部分浏览器 都会 302 会将 POST 请求转为 GET，而 307 则不一样，规范要求浏览器继续向 Location 的地址 POST 内容。

   举个例子解释一下，假设正在 POST 一个消息，里面的 Body 有 1M 内容，在 307 的情况下，这 1M 的内容会继续发过去，但在 302 的情况下，则不会。

6. **在 HTTP Response 里 Connection 的用法需要注意什么？**

   不解释过多，keep-alive，closed 用法不一样，根据实际情况而定，在优化网络时经常用到。

需要注意的是，Connection 在通信领域会一些场景下会造成一些麻烦，尤其是在监控某个 HTTP Session 的 flow 时。

7. **在 HTTP Response 里 Strict-Transport-Security（HSTS）怎么用？**
   HSTS 一般大公司都会用，在我实际的项目涉及到 HTTPS，为了实现某个功能，HSTS 成了一个跨不过去的坎，遇到过很大问题。大家自己多看看吧。

8. **可以抓 HTTPS 的包了解 HTTP 请求和响应吗？有什么方法？**
   不多解释，但提供 2 个软件名字，Fiddler，HTTP Analyzer。

9. **为什么对性能要求高的场景下不使用 HTTP 作为协议？例如在商用里，RPC 开源项目一般不使用 HTTP 作为传输协议？而在 5G 下使用 HTTP 协议呢？**
   有一点是需要注意的，HTTP 的消息头太多了，会造成消息体特别大，影响性能，而且有些场景下这些消息头大部分都是无用的。
   但是在 5G 里，为什么 3GPP 组织会采用 HTTP 协议作为各个 reference point 的 interface 的实现呢？大家体会一下。

10. **HTTP 协议（包括 HTTP/2.0）有看过吗？**
    不多解释，但是 HTTP/2.0 还是需要了解一下，优势和缺点。

浏览器解析和渲染页面

现在浏览器接收到了 server 的返回内容，接下来浏览器该把内容呈现给用户了。

Server 返回的内容有哪些呢？这里只以 HTML 页面为例（API 返回的 JSON 数据或 XML 数据不在讨论范围内）。

一个页面一般包含 HTML、CSS、 JS、 图片等文件，那么浏览器收到这些文件后该如何渲染（render）他们呢？

以下部分很多参考下面 2 篇文章：

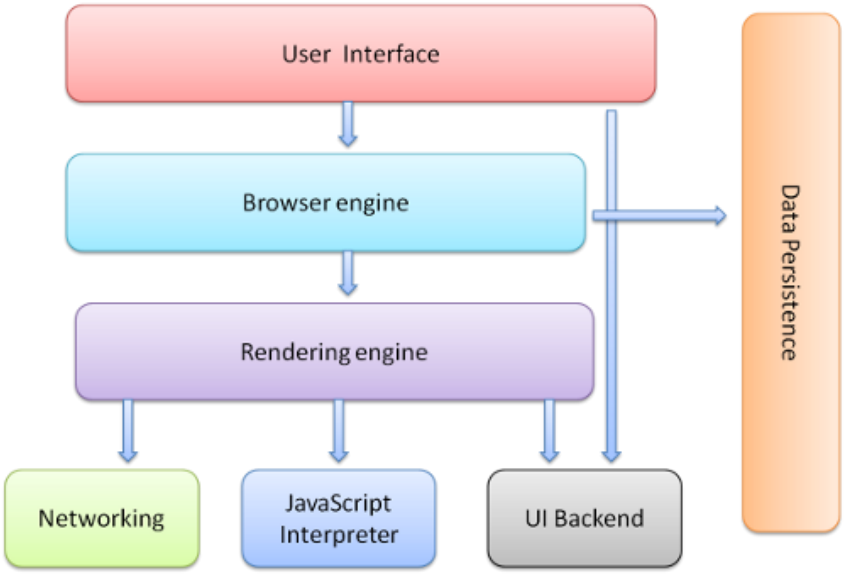
- https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/
- https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-tree-construction

浏览器的组成

首先我们先了解一下浏览器的组件构成，以及每个组件的功能，下图是浏览器包括的几个部分：

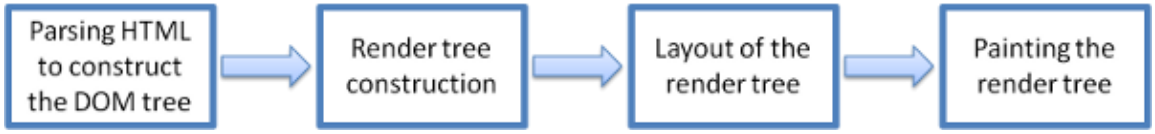
1. **User Interface**: UI 组件包括地址栏，前进/后退按钮，书签菜单等。
2. **Browser Engine**: 在 UI 组件和渲染引擎间采取一些 action.
3. **Rendering engine** : 负责显示请求的内容。例如，如果是 HTML 页面，它将解析 HTML，CSS，并将解析的内容显示在屏幕上。

   不同的浏览器使用不同的渲染引擎：
   - IE 使用 Trident
   - Firefox 使用 Gecko
   - Safari 使用 WebKit
   - Chrome 和 Opera（版本 15 开始）使用 Blink。它是基于 Webkit 开发的。
4. **Networking**: 负责网络调用，例如 HTTP 请求。在不同的平台有不同的实
5. **UI backend**: 主要用来绘画基本的 UI 元素，例如下拉框，Windows 等。这个 UI 后台暴露一些通用的接口，并不依赖平台的。
6. **JavaScript interpreter**. 用来解析和运行 JavaScript code。
7. **Data storage**. 数据持久化的那一层。浏览器可能需要存储各种各样的数据，例如 Cookie。浏览器也得支持我们常用的 LocalStorage， IndexedDB，WebSQL 以及 FileSystem。



渲染页面的主要流程

下面是浏览器的渲染引擎的主要步骤。



渲染引擎解析 HTML 文档，并将 HTML 包含的元素转化为一个个 DOM，并构建为一个 **DOM 树**。然后引擎开始解析来自 CSS 文件或直接嵌在 HTML 页面的 CSS 样式数据，这些样式信息又会构建另外一个树：**渲染树**。

渲染树包含了多个矩形，这些矩形包含了颜色，大小，位置等属性，而且会按照对应的顺序显示在屏幕上。

当渲染树构造完毕后，接下来进入布局的程序，在这个程序里，渲染引擎会给每个 DOM 元素安排精确的坐标，并根据坐标在屏幕上显示。

接下来是遍历渲染树，UI Backend 层会将一个个 DOM 元素绘画在屏幕上绘画出来。

需要注意的是，上面是一个渐进的过程，理解这一点非常重要。但是为了得到更好的用户体验，浏览器会边解析边渲染，它并不会等到所有 HTML 解析完了才开始构造和布局渲染树。当部分内容正在解析渲染时，另外一部分正从网络那边下载下来呢。

下面 2 个图是 WebKit 和 Gecko 的渲染引擎的流程，我们发现他们大致相同的。

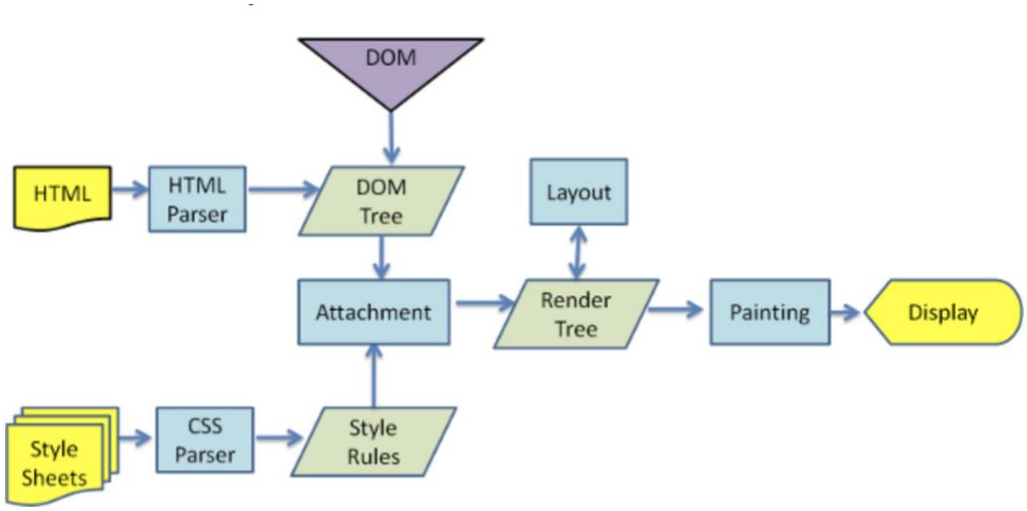


Figure : WebKit main flow

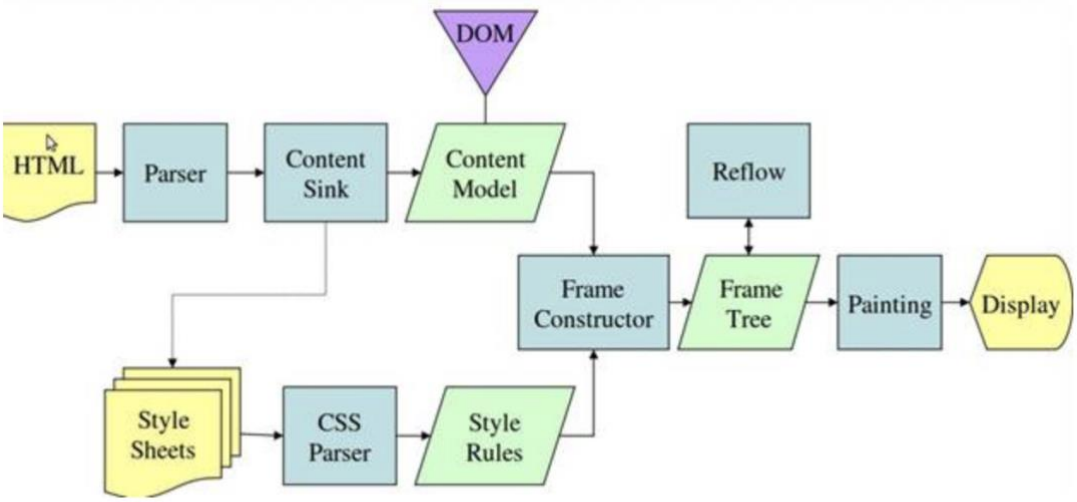
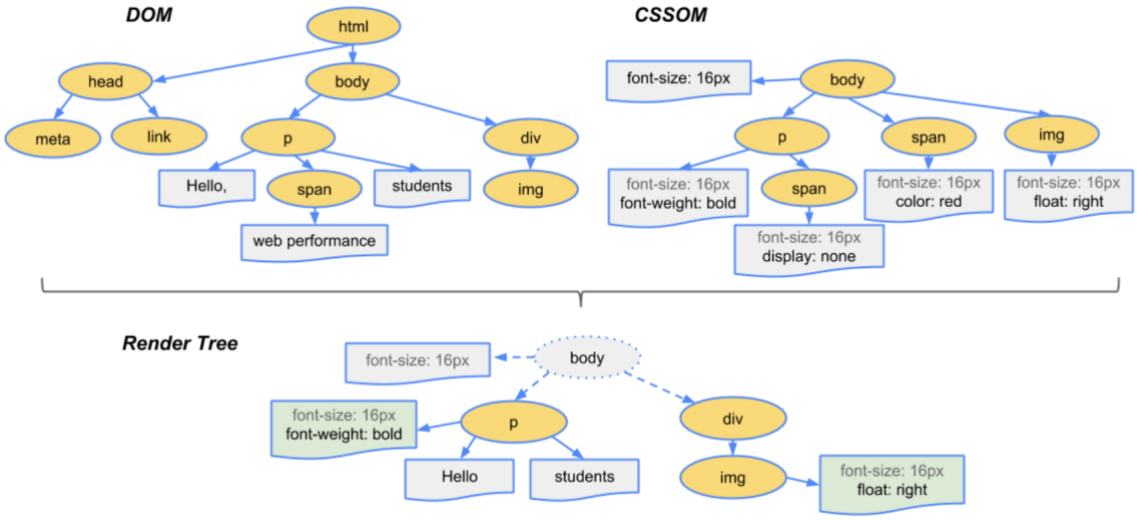


Figure : Mozilla's Gecko rendering engine main flow (3.6)

下面是 DOM 树，渲染树的树形结构。



渲染引擎是单线程工作的，除了网络操作，其他所有的都是单线程的。在 Firefox 和 Safari，它们自己就是主线程，而 Chrome 就是每个 tab 处理主线程。

网络操作则由多个并列线程去执行，但数量也是受限的，一般在 2-6 个。

浏览器的主线程是一个无限的事件循环，而且一直保持进程 alive，一直等着各种事件（例如绘画事件，布局事件），并处理他们。

浏览器渲染 10 问

1. 浏览器的组成部分是什么？
2. 各个主流浏览器的渲染引起是什么？
3. 浏览器显示页面的主要流程是什么？
4. 您在做开发和测试时，有哪些浏览器（包括手机）存在兼容性问题较多？

遇见最多的是 Samsung 手机和 Huawei 手机，有时一个兼容性需要花费大量时间去调研和修复，可能是 Samsung 和 Huawei 定制的太厉害了，不兼容一些特性吧。

5. 渲染引擎是单线程还是多线程？
6. 浏览器的网络操作一般由几个线程去执行？
7. DOM 树，渲染树是什么？
8. 为了获得更好的用户体验，我们应该在页面做些什么改进？
9. 浏览器是如何打开 PDF，Word 等文档的？
10. 如果让你开发一个浏览器，设计思路有哪些？

Web 优化

我们知道，人的耐心是有限的，一个页面如果超过 8s，人基本上不会等了，这会对业务产生巨大影响。我们该如何去优化页面呢？

思路很简单，就是按照我们前面介绍的几大步骤去优化。我们先回顾一下几大步骤：

1. DNS 查询
2. TCP 连接
3. 发送 HTTP 请求
4. Server 处理 HTTP 请求并返回 HTTP 报文
5. 浏览器解析并 render 页面
6. HTTP 连接断开

当我想总结一下的时候，雅虎在 10 多年就总结出来一些经验，参看这里（https://developer.yahoo.com/performance/rules.html）。
这些都是 10 多年前的经验，随着科技的发展，一些新的经验又出现了，以便适应当前的场景，在页面上和架构上都有。

比较容易想到的是：

1. 尽量将 server 离用户近一些，例如人处在中国访问 Apple，应该是 Apple 中国站提供服务，GSLB 很重要。
2. 不要把 layout 嵌入一层又一层，简单说就是嵌套别太深，不然影响解析和渲染性能。
3. 有些数据可以在后台处理的，就不要在前端通过 JavaScript 处理了。
4. 如果请求过大，Load Balance 这些手段还是要上的。
5. 保持 HTTP 连接，合理设置 Connection。
6. 后台事件性能要高，能够及时将结果返回给用户。

当然涉及到高可用、高性能等那是另外一组话题，不在这里叙述。

总结

这道面试题非常经典，考察的知识非常丰富，跨度较大，如果没有几年的经验，是很难完全掌握的，所以想答好其实不容易的。对这个问题，基本上可以根据我的经验回答，也算是对这些年来在这方面的知识的一个总结。但同时，我也参考了一些资料，感谢他们。在参考的地方，我都把 URL 列出来了。

写这篇文章耗费了大量时间，我觉得挺有意义，但是也不能保证里面的内容全都是正确或准确的，如果您有任何问题可以通过以下方式联系我，以便我进一步改正并更新。

个人微信：terryisme

公众号：terrymemo