如何快速入门网络基础知识（TCP/IP 和 HTTP）

前言

在写之前，先给这篇文章做一个明确定位，读完这篇文章后，希望你能够：

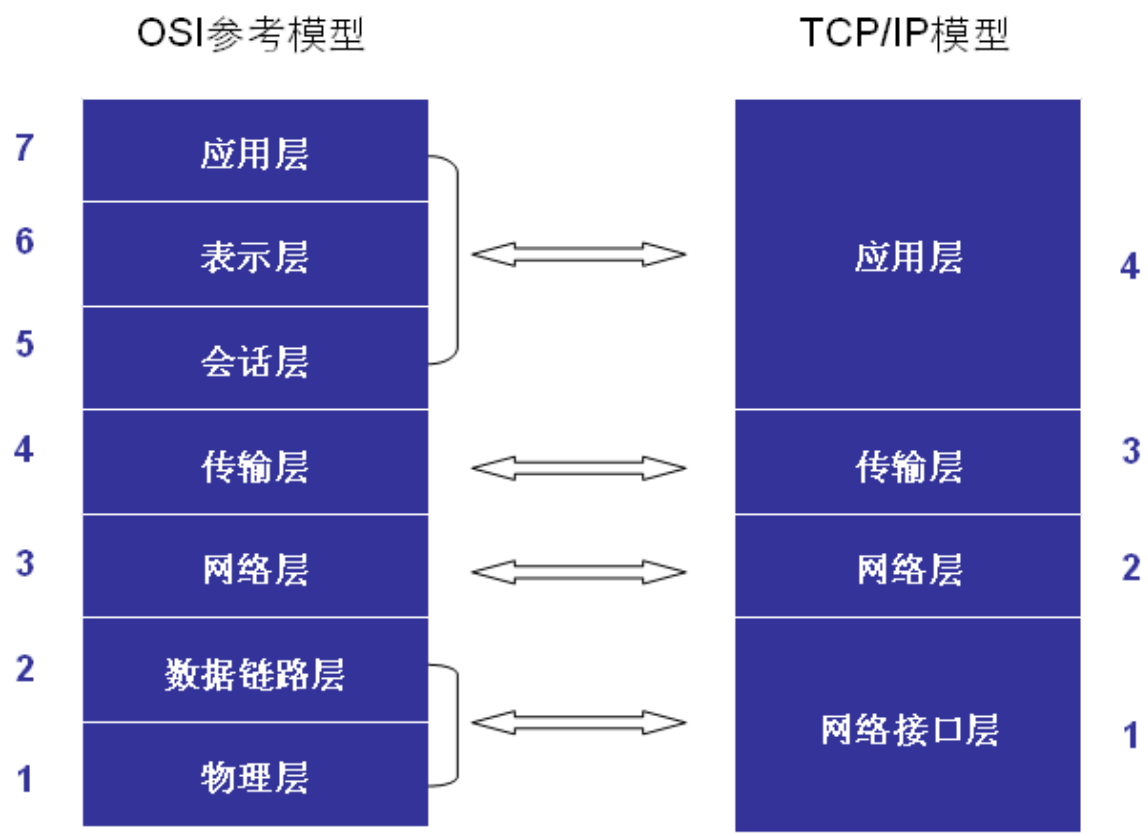
- 对于计算机网络有初步的认识和了解，了解一些经典专业术语，如三次握手、四次挥手、DNS解析的含义。
- 了解一些应用层协议，如传统的HTTP、HTTPS协议，以及业界近几年开始逐步普及的HTTP2、QUIC协议。
- 通过实际生产环境下的例子，了解网络优化在项目中的实际意义以及带来的效果。

课前准备

为了能够更好地理解这篇文章的内容，建议阅读之前做以下几个准备：

- 了解TCP/IP的基本概念，推荐去阅读《TCP/IP协议详解》中的第3章和第17章，这里给大家一个链接，可以免费下载，传送门。
- 其次希望在学习的过程中，为了能够更好地看到效果，请下载抓包工具，这里推荐的是Wireshark和Charles，大家可以下载后，结合文章内容并尝试抓包，更容易理解整个网络转发过程。

网络模型



可以看到，OSI七层模型和TCP/IP五层模型存在一个对应关系，并且传输层以下的完全一致(TCP模型中的网络接口层就是数据链路层和物理层的集合)，因此可以说将OSI模型中的会话层、表示层与应用层合并为TCP/IP模型中的应用层后，二者基本一致。

上述两个网络模型都属于通用网络模型，相对来说，TCP/IP模型更为普遍一些，所以我们也主要以TCP/IP模型为网络模型开展论述，这也是为什么这节课的名字TCP/IP的由来。

那么每一层都对应哪些协议呢？请看下图：

TCP/IP协议集

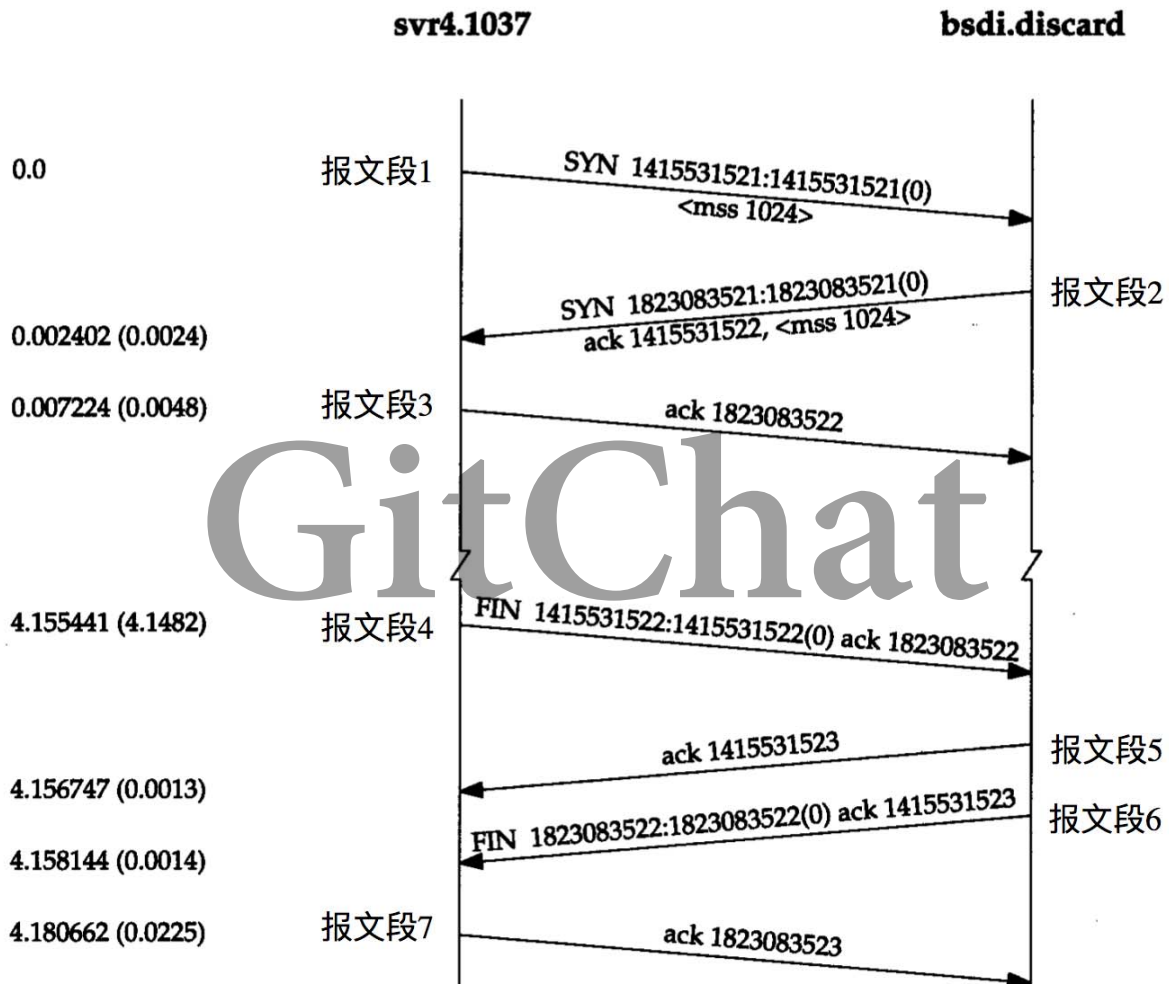
应用层	Telnet, FTP, SMTP, DNS, HTTP 以及其他应用协议
传输层	TCP， UDP
网络层	IP, ARP, RARP, ICMP
网络接口	各种通信网络接口（以太网等） （物理网络）

可以看到，我们熟知的一些协议，IP协议位于网络层，TCP协议位于传输层，而HTTP协议则位于应用层，其余还有比较熟悉的DNS协议，FTP协议等等，都有其所属的层级。

从上往下依次是Frame帧头、以太帧头、IP协议头、TCP协议头和HTTP协议头，最后的一行则是本次请求的数据，其格式为JSON

三握四挥

所谓三握四挥是指三次握手和四次挥手，也就是TCP协议建立连接和断开连接的过程，之所以叫做三次握手，是因为建立连接的双方需要经过三次数据交互以后才能完成连接的建立，同样的，四次挥手是指在断开连接时需要四次数据交互，其交互过程图如下：



> 小S：喂，小C，我有点累啦，今天要不就这样吧
> 小C：好呀，你休息下，我再说两句
> 小C：哎呀，我也好累呀，今天就到这里吧
> 小S：好，那就到这吧，886

然后小S和小C就挂了电话，我们注意到，在四次挥手的过程中，小S先提出了断开连接，但实际上他们的对话并没有结束，后面小C确认这个消息后，并没有立马断开连接，而是继续对话，这是因为TCP协议具备全双工特性，简单点说就是一个连接，存在小C——小S和小S到小C两条线路，而小S提出并由小C确认关闭的只是小S——小C这条线路，因此小C还可以继续向小S发消息，直到小C也觉得要关闭连接并由小S确认后，两人的所有连接才彻底关闭。

那么你肯定会问啦，为什么TCP要这么设计呢，这是因为TCP是一个全双工的协议，全双工(Full Duplex)是通讯传输的一个术语。通信允许数据在两个方向上同时传输，我们在上面的例子也提到了在一次TCP交互中，需要维持两条线路，因此无论是在建立和断开的时候，都要确保两条线路的状态正确。

上面的阐述还是建立在理论阶段，为了能更好地巩固知识，我们利用Wireshark在实际生产环境下抓包看下:

> 客户端IP为：10.2.203.93
> 服务端IP为：10.108.21.2

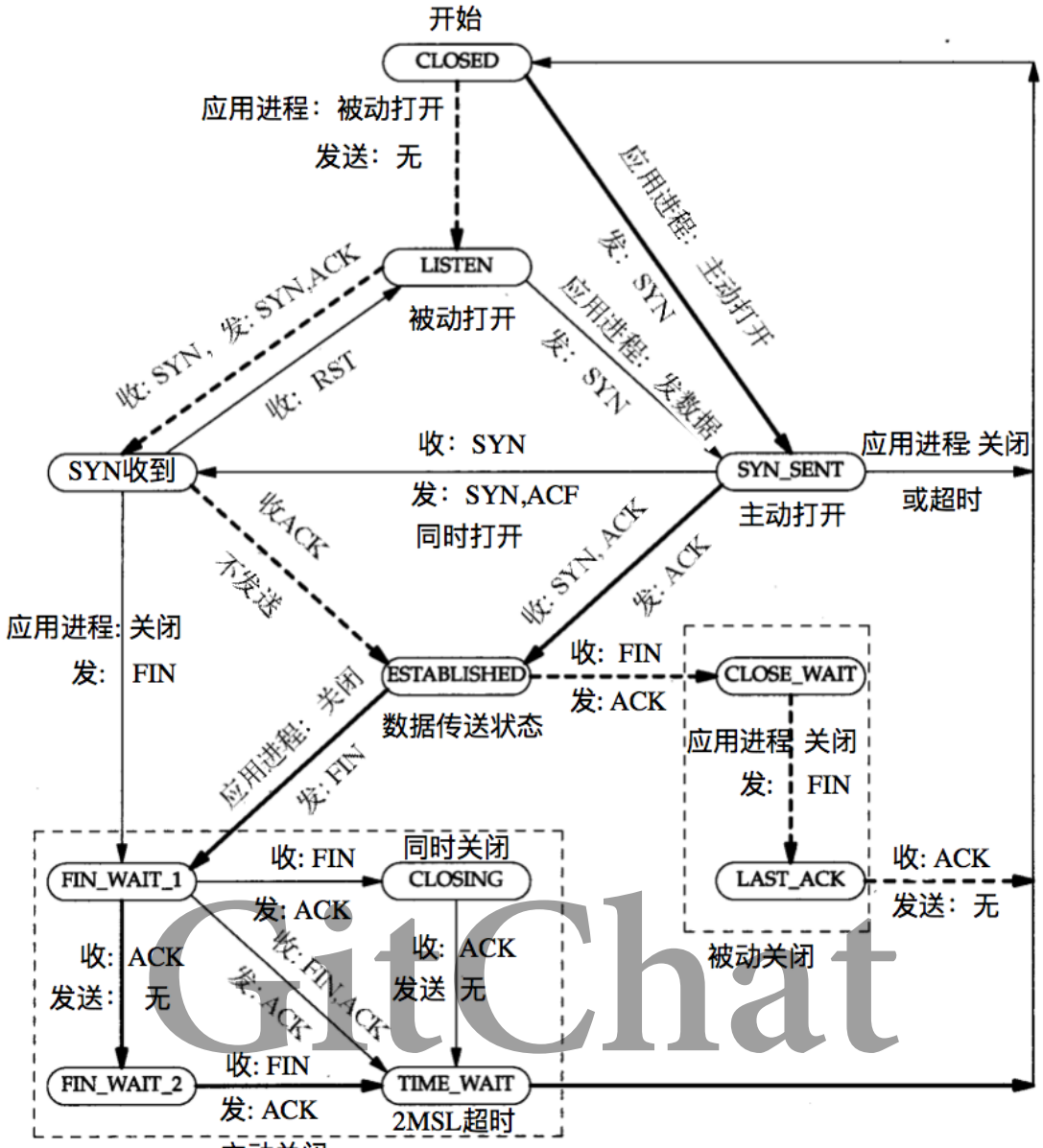
客户端和服务端建立连接时的抓包情况：

((ip.src == 10.2.203.93 && ip.dst == 10.108.21.2) || (ip.dst == 10.2.203.93 && ip.src == 10.108.21.2)) && tcp

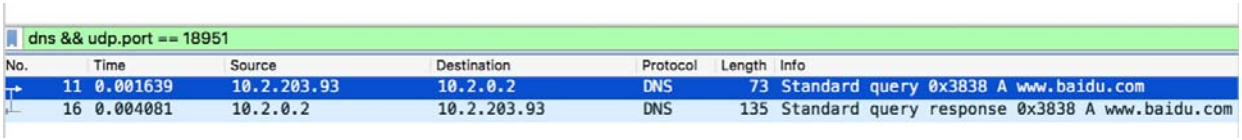
	Time	Source	Destination	Protocol	Length	Info
36	2.568156	10.2.203.93	10.108.21.2	TCP	78	63979→80 [SYN] Seq=0 Win=65535 Len=0 MSS=1382 WS
38	2.619578	10.108.21.2	10.2.203.93	TCP	74	80→63979 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0
39	2.619638	10.2.203.93	10.108.21.2	TCP	66	63979→80 [ACK] Seq=1 Ack=1 Win=131520 Len=0 TSva

可以看到由客户端首先发SYN报文，服务端收到并回应SYN ACK报文，客户端最后再回一个ACK报文，连接就算建立完毕了。

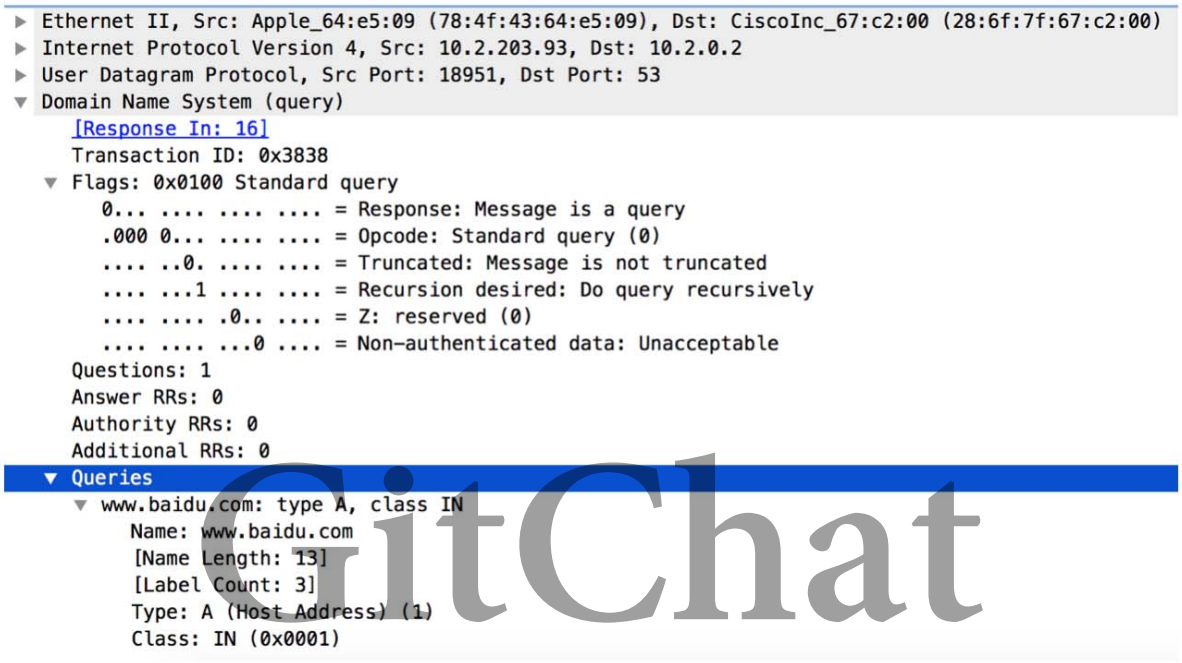
再来看看断开连接时的情况：



我们在浏览器中输入www.baidu.com时，会向服务器发送DNS请求报文，当服务器端处理完这个请求以后，就会发送DNS响应报文，其中就包含我们关心的IP地址，可以看到我们抓到两个报文，前者我们称之为DNS请求报文，后者称之为DNS响应报文，注意我们的筛选条件，通过UDP端口来过滤更加方便：



先来看DNS请求报文：



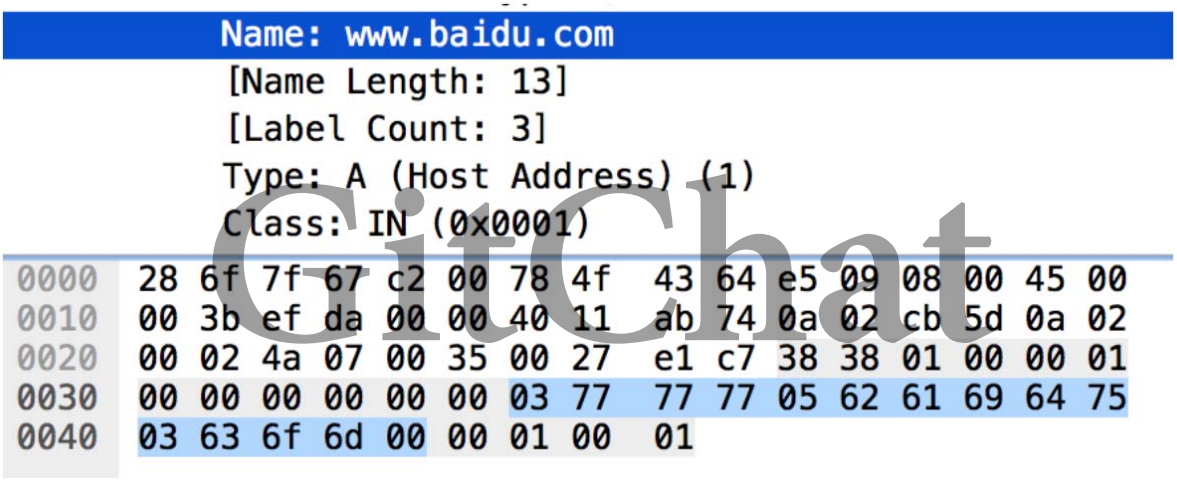
可以看到DNS的传输层协议是UDP，而不是TCP，并且其端口号为53。紧接着的是Transaction ID(2字节)，这个ID可以作为DNS请求的一个唯一ID来使用，也就是说对于一个请求和应答报文，这个ID是相同的，因此也可以借助这个ID来查找请求报文相对应的应答报文。

Flags字段长度也是2字节，可以看到，16bit被分成了以下几部分，依次为：

1 格式差错
2 问题在域名服务器上
3 域参照问题
4 查询类型不支持
5 在管理上被禁止
6 – 15 保留

紧接着Flags下面的几个字段分别是：queries、answers、auth_r、add_rr，其相应的中文含义为问题数、资源记录数、授权资源记录数和额外资源记录数，它们的长度都是2字节，一般来说queries为1，其余的字段值为0。

接下来就是报文的正文部分，这里包括要查询的域名，查询类型和相应的查询类，这里的域名的格式比较特别，在这里的域名是www.baidu.com，而标记为蓝色的部分则是报文中的表示，可以看到，03是代表3个字节，而紧跟着3个77，如果转换为ASIC码的话，就是0x77，因此对于www.baidu.com，首先是以“.”为分隔符，分成3个部分后，用相应的段长度再加上域名段的ASIC组成一个段，这样就构成了一个完整的域名。



后续的两个字段分别是Type和Class，在这里两个字段都为1，其中Type为A则代表此次请求类型是通过域名获取IP地址，也是最为常见的一种DNS请求形式。而Class字段为1，则代表这里查询的数据是internet数据，也是最为常见的一种形式。

介绍完了请求报文，接下来再看一下响应报文：

```
▶ Frame 16: 135 bytes on wire (1080 bits), 135 bytes captured (1080 bits) on interface 0
▶ Ethernet II, Src: CiscoInc_67:c2:00 (28:6f:7f:67:c2:00), Dst: Apple_64:e5:09 (78:4f:43:64:e5:09)
▶ Internet Protocol Version 4, Src: 10.2.0.2, Dst: 10.2.203.93
▶ User Datagram Protocol, Src Port: 53, Dst Port: 18951
▼ Domain Name System (response)
    [Request In: 11]
    [Time: 0.002442000 seconds]
    Transaction ID: 0x3838
  ▼ Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 3
    Authority RRs: 0
    Additional RRs: 0
  ▶ Queries
  ▼ Answers
    ▶ www.baidu.com: type CNAME, class IN, cname www.a.shifen.com
    ▶ www.a.shifen.com: type A, class IN, addr 61.135.169.125
    ▶ www.a.shifen.com: type A, class IN, addr 61.135.169.121
```

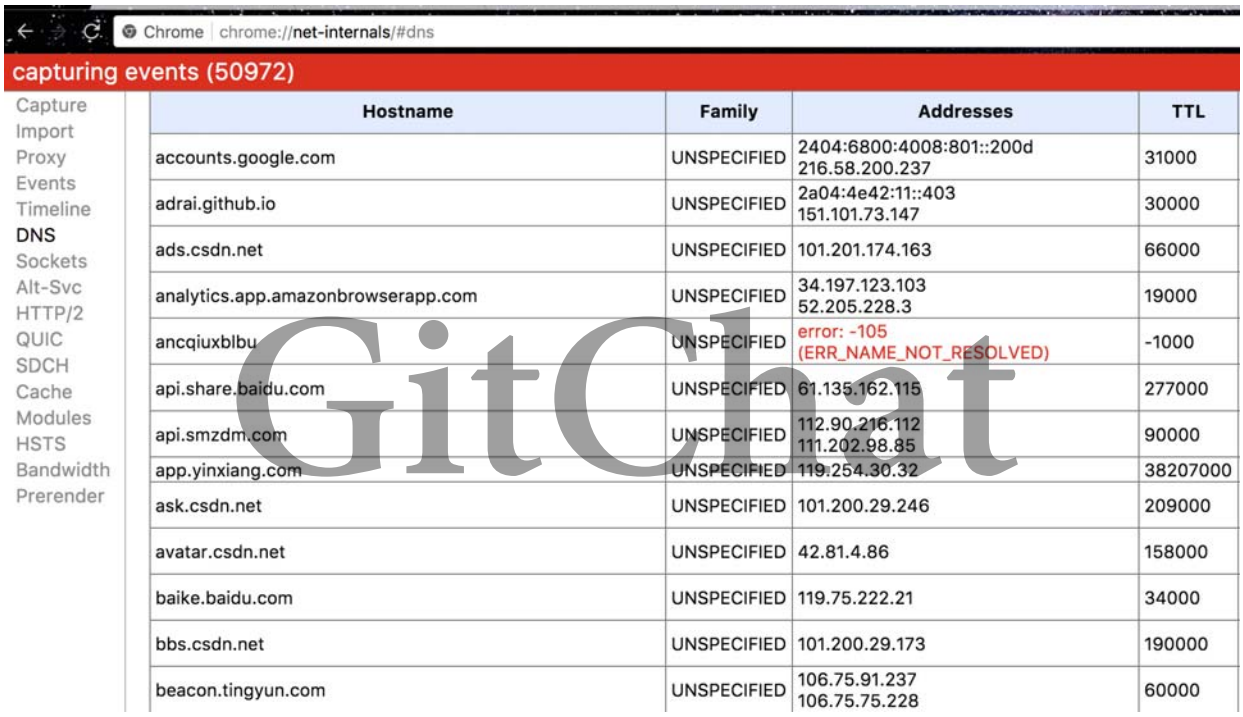
响应报文和应答报文相同的部分就不再赘述了，可以看到Flags中的Response值为1，就说明这是一个响应报文，同时Transaction ID也和请求报文中的ID一致，说明这就是上面那个请求报文所对应的响应报文。

请求报文正文中最主要的部分就是Answers字段，这里面包括了我们想要的IP地址，但是我们也注意到，对于www.baidu.com这一个域名，响应字段居然有3条，那么究竟以哪一条为准呢？我们一条条来看。

首先是第1条Answer，这里的Type类型为CNAME，这里的CNAME表示这个回应是请求报文中查询的域名的一个别名，也就是此处返回的将是www.baidu.com的一个别名，也就是www.a.shifen.com，紧接着后面两个Answer，Type类型为A，代表返回值将是一个IPV4地址，其他的比较常见的Type类型还有AAAA——IPV6地址，PTR——IP地址转换为域名，NS——名字服务器。

可以看到，对于同一个域名，可以返回多个IP地址，在上面的响应报文中，返回了2个IP地址，分别是61.135.169.125和61.135.169.121，这就是我们最终想要的结果，为了防止其中某个IP地址出现异常，因此通常对于一个域名，都会有两个甚至以上的IP地址与其对

对于使用chrome浏览器的同学，可以输入chrome://net-internals/#dns 来查看浏览器DNS解析列表：



Hostname	Family	Addresses	TTL
accounts.google.com	UNSPECIFIED	2404:6800:4008:801::200d 216.58.200.237	31000
adrai.github.io	UNSPECIFIED	2a04:4e42:11::403 151.101.73.147	30000
ads.csdn.net	UNSPECIFIED	101.201.174.163	66000
analytics.app.amazonbrowserapp.com	UNSPECIFIED	34.197.123.103 52.205.228.3	19000
ancqiuxblbu	UNSPECIFIED	error: -105 (ERR_NAME_NOT_RESOLVED)	-1000
api.share.baidu.com	UNSPECIFIED	61.135.162.115	277000
api.smzdm.com	UNSPECIFIED	112.90.216.112 111.202.98.85	90000
app.yinxiang.com	UNSPECIFIED	119.254.30.32	38207000
ask.csdn.net	UNSPECIFIED	101.200.29.246	209000
avatar.csdn.net	UNSPECIFIED	42.81.4.86	158000
baike.baidu.com	UNSPECIFIED	119.75.222.21	34000
bbs.csdn.net	UNSPECIFIED	101.200.29.173	190000
beacon.tingyun.com	UNSPECIFIED	106.75.91.237 106.75.75.228	60000

在这里可以看到你访问过的网站，以及相应的解析记录，这里还有一栏TTL，代表域名解析结果的生存时间，简单点说就是当我们解析完毕一个域名以后，会将其记录缓存起来，在TTL时间之内的访问，我们都直接从缓存中获取，而不再去进行DNS解析，这样带

介绍完TCP协议和DNS协议之后，我们就要开始介绍处于TCP/IP模型中最上层的应用层协议了。应用层协议也是和用户交互最密切的，因此对用户感知影响也是最直接的，下面就以此介绍几种比较常见的应用层协议。

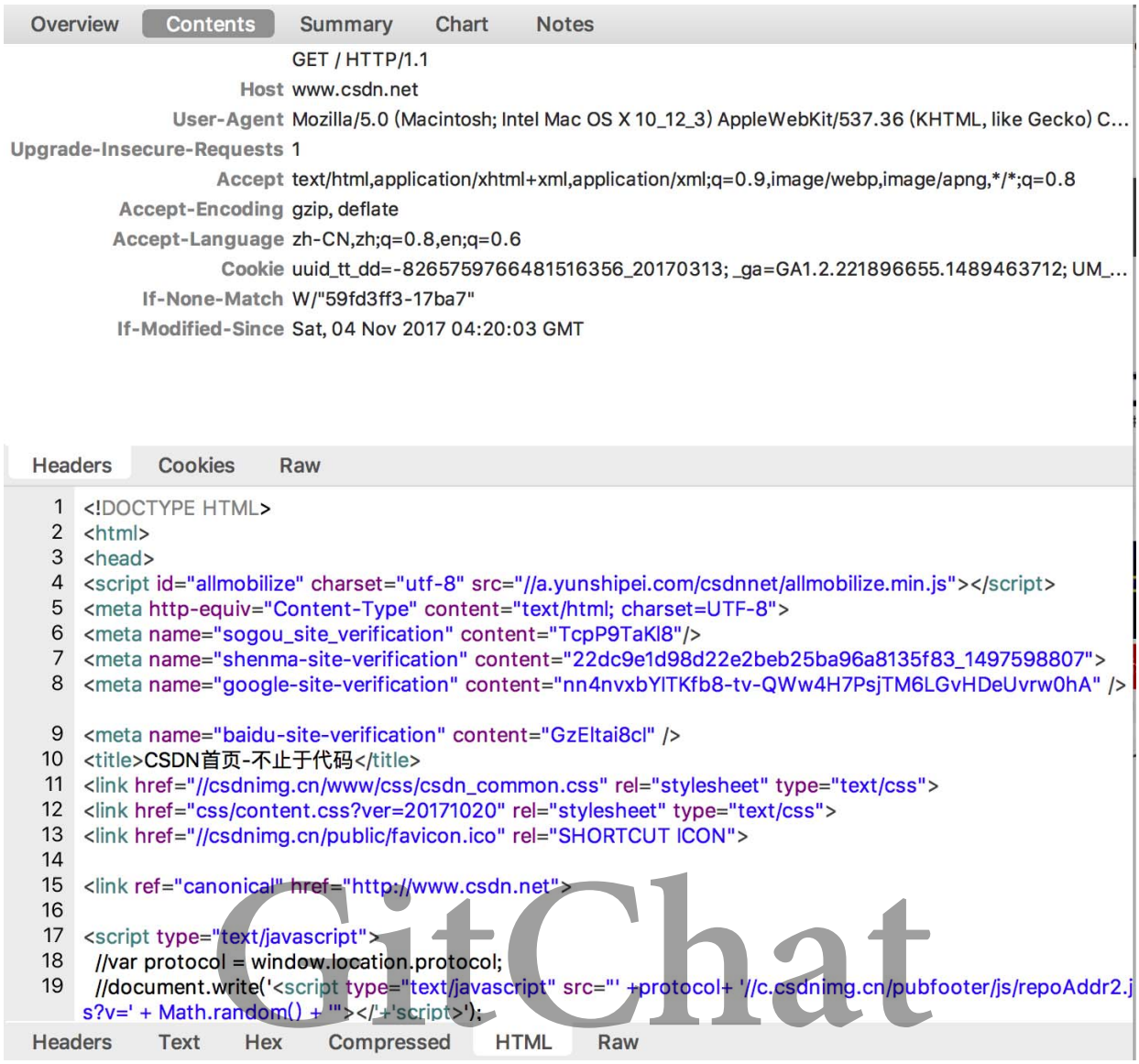
HTTP

> HTTP（HyperText Transport Protocol）是超文本传输协议的缩写，它用于传送WWW方式的数据，关于HTTP协议的详细内容请参考RFC2616。HTTP协议采用了请求/响应模型。客户端向服务器发送一个请求，请求头包含请求的方法、URL、协议版本、以及包含请求修饰符、客户信息和内容的类似于MIME的消息结构。服务器以一个状态行作为响应，响应的内容包括消息协议的版本，成功或者错误编码加上包含服务器信息、实体元信息以及可能的实体内容。

上面是关于HTTP的权威描述，HTTP可以说是整个互联网当中最普遍也是最重要的一个协议了，包括你现在能看到我写的这篇文章，也是利用HTTP进行数据传输的，那么纠结是如何进行工作的呢？这次我们借助另外一款抓包神器——Charles来进行抓包分析。

当我们利用浏览器访问http://www.csdn.net/时，浏览器会在我们输入地址并敲下回车后在页面显示：

这是一个在平常不过的操作了，此时我们用Charles进行抓包，得到下面的结果：



上面是整个完整的交互，实际上包含了两部分，首先是HTTP request请求，也就是上面中上半部分，可以看到，在request请求中几个关键点是GET、HTTP/1.1、Host、User-Agent、Accept以及Cookie，这些关键字构成了一个request请求的报文头，代表客户端想通过本次请求得到服务端的哪些数据，服务端在收到request后，作出的回应便是HTTP response报文，也就是上图中下半部分，因为我们访问的是csdn的主页，并且在Accept里也指定了html是一种请求数据，所以response报文返回的数据里便包含了HTML数据，当然，response报文也有其它组成部分，如下图所示：

	HTTP/1.1 200 OK
Server	openresty
Date	Sat, 04 Nov 2017 08:02:10 GMT
Content-Type	text/html; charset=utf-8
Keep-Alive	timeout=20
Vary	Accept-Encoding
Last-Modified	Sat, 04 Nov 2017 08:00:03 GMT
Vary	Accept-Encoding
ETag	W/"59fd7383-17ba7"
Content-Encoding	gzip
Transfer-Encoding	chunked
Proxy-Connection	Keep-alive

Headers Text Hex Compressed HTML Raw

这里面就包括了服务端对于本次请求的回应数据，其中最关键的便是200 OK这个字段，这是响应状态码，最常见的就是200，也就是表明请求OK，还有比较常见的就是404和502，前者代表客户端非法请求，后者代表服务端响应失败，比如说我们输入http://www.csdn.net/test123时，页面就会提示：

HTTPS

> HTTPS（全称：Hyper Text Transfer Protocol over Secure Socket Layer），是以安全为目标的HTTP通道，简单讲是HTTP的安全版。即HTTP下加入SSL层，HTTPS的安全基础是SSL，因此加密的详细内容就需要SSL。 它是一个URI scheme（抽象标识符体系），句法类同http:体系。用于安全的HTTP数据传输。https:URL表明它使用了HTTP，但HTTPS存在不同于HTTP的默认端口及一个加密/身份验证层（在HTTP与TCP之间）

从上面的阐述中我们可以得到HTTPS = HTTP + TLS这样一个简单的结论，也就试试哦HTTPS是比HTTP更加安全的协议，并且目前已经有不少网站开始支持HTTPS了。比如说百度：

可以看到，使用的是HTTPS协议，同时浏览器会提示安全，我们再看另外几个例子：

哈，建行主页竟然还没有使用HTTPS协议，那是不是就说明建行不安全了呢？

其实不然，我们点击登陆按钮：

可以看到使用的协议还是HTTPS协议，这说明我们的登陆操作依然是有安全保障的，大大降低了账号信息被盗用的可能

HTTP2

> HTTP/2（超文本传输协议第2版，最初命名为HTTP 2.0），简称为h2（基于TLS/1.2或以上版本的加密连接）或h2c（非加密连接）[1]，是HTTP协议的的第二个主要版本，使用于万维网。
>
> HTTP/2是HTTP协议自1999年HTTP 1.1发布后的首个更新，主要基于SPDY协议。它由互联网工程任务组（IETF）的Hypertext Transfer Protocol Bis（httpbis）工作小组进行开发。[2]该组织于2014年12月将HTTP/2标准提议递交至IESG进行讨论[3]，于2015年2月17日被批准。[4]

HTTP/2标准于2015年5月以RFC 7540正式发表。[5]HTTP/2的标准化工作由Chrome、Opera、Firefox[6]、Internet Explorer 11、Safari、Amazon Silk及Edge等浏览器提供支持。[7]

多数主流浏览器已经在2015年底支持了该协议。[8]此外，根据W3Techs的数据，在2017年5月，在排名前一千万的网站中，有13.7%支持了HTTP/2。

可以看到HTTP2协议已经在互联网占有一席之地，那么它究竟比HTTP强在哪里呢？总结了一下，大致有以下几点。相比 HTTP/1.x，HTTP/2 在底层传输做了很大的改动和优化：

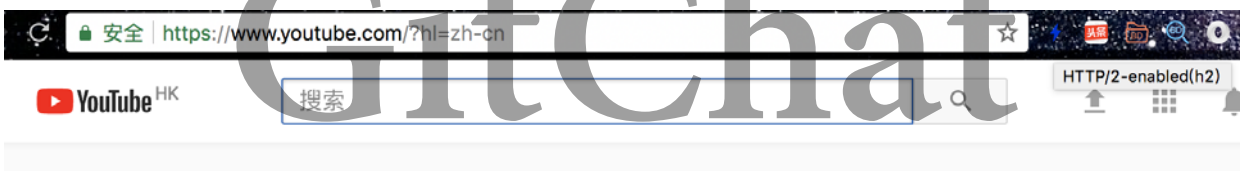
QUIC相比于上述介绍的HTTP、HTTPS和HTTP2协议最大的不同就在于，其传输层采用的是UDP协议而不是TCP协议，因此其具备的特性有以下几点：

- 0-RTT 建联(首次建联除外)
- 类似TCP的可靠传输
- 类似TLS的加密传输，支持完美前向安全
- 用户空间的拥塞控制，最新的BBR算法
- 支持h2的基于流的多路复用， 但没有TCP的HOL问题
- 前向纠错FEC
- 类似MPTCP的Connection migration

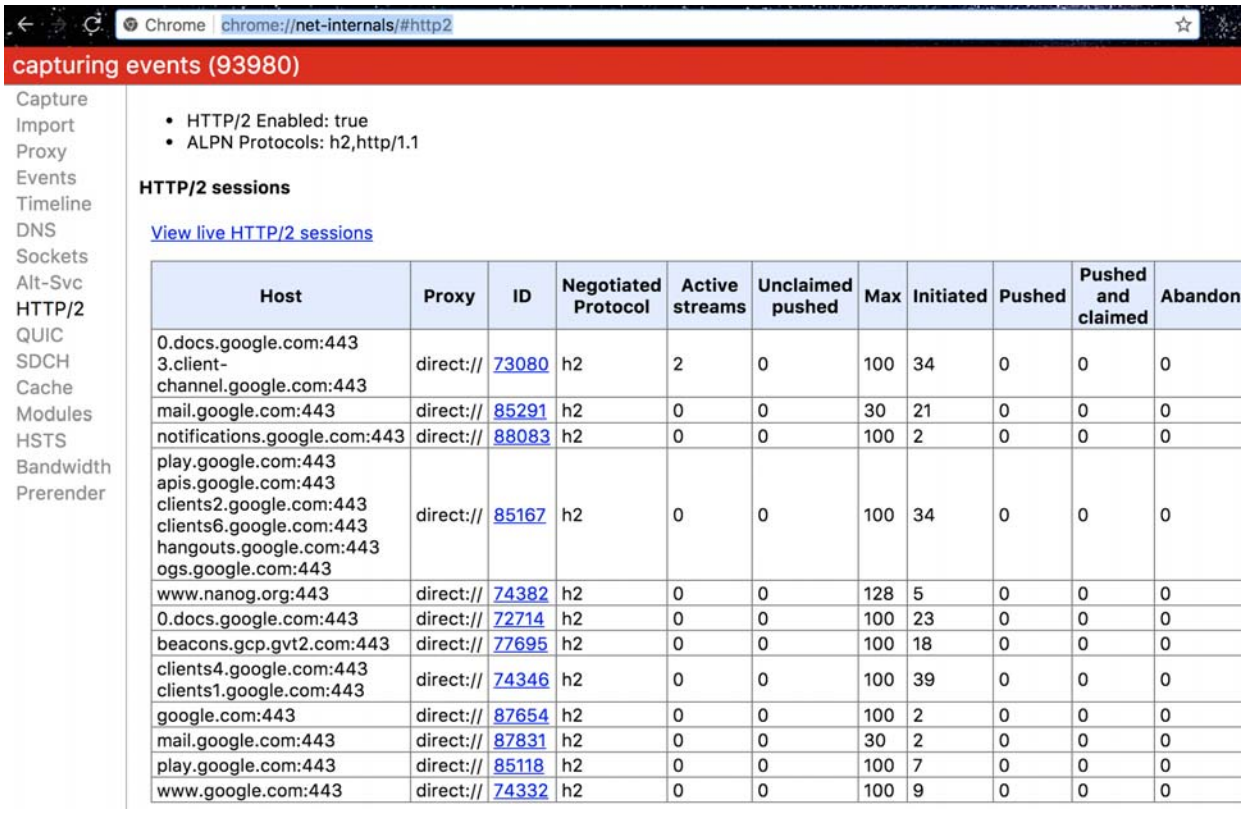
那在实际环境中，如何知道哪些访问使用了HTTP2、哪些访问使用了QUIC协议呢？

这里就要提到chrome的一个插件——HTTP/2 and SPDY indicator，当下载该插件并成功访问后，我们就可以看到浏览器地址栏右侧会多一个⚡标志：

访问百度时，这个⚡标志是白色的，当我们访问YouTube时：



会发现标志变为蓝色，鼠标移到该标志时，提示HTTP2已经使能，这说明在YouTube上面已经开始使用HTTP2协议了，在chrome浏览器中输入chrome://net-internals/#http2就可以看到具体哪些网站使用了HTTP2和QUIC：



Host	Proxy	ID	Negotiated Protocol	Active streams	Unclaimed pushed	Max	Initiated	Pushed	Pushed and claimed	Abandon
0.docs.google.com:443 3.client-channel.google.com:443	direct://	73080	h2	2	0	100	34	0	0	0
mail.google.com:443	direct://	85291	h2	0	0	30	21	0	0	0
notifications.google.com:443	direct://	88083	h2	0	0	100	2	0	0	0
play.google.com:443 apis.google.com:443 clients2.google.com:443 clients6.google.com:443 hangouts.google.com:443 ogs.google.com:443	direct://	85167	h2	0	0	100	34	0	0	0
www.nanog.org:443	direct://	74382	h2	0	0	128	5	0	0	0
0.docs.google.com:443	direct://	72714	h2	0	0	100	23	0	0	0
beacons.gcp.gvt2.com:443	direct://	77695	h2	0	0	100	18	0	0	0
clients4.google.com:443 clients1.google.com:443	direct://	74346	h2	0	0	100	39	0	0	0
google.com:443	direct://	87654	h2	0	0	100	2	0	0	0
mail.google.com:443	direct://	87831	h2	0	0	30	2	0	0	0
play.google.com:443	direct://	85118	h2	0	0	100	7	0	0	0
www.google.com:443	direct://	74332	h2	0	0	100	9	0	0	0

总结

本期的内容到这里也就告一段落了，希望读完本篇文章后，可以让你对网络有更深入的了解，并且能够在实际生活中，去留意这些知识，尤其是抓包分析网络问题，可以说是学习网络知识和分析网络问题的最大利器。

后续我会在这一期的基础上，再推出一期网络知识进阶篇，敬请期待！