
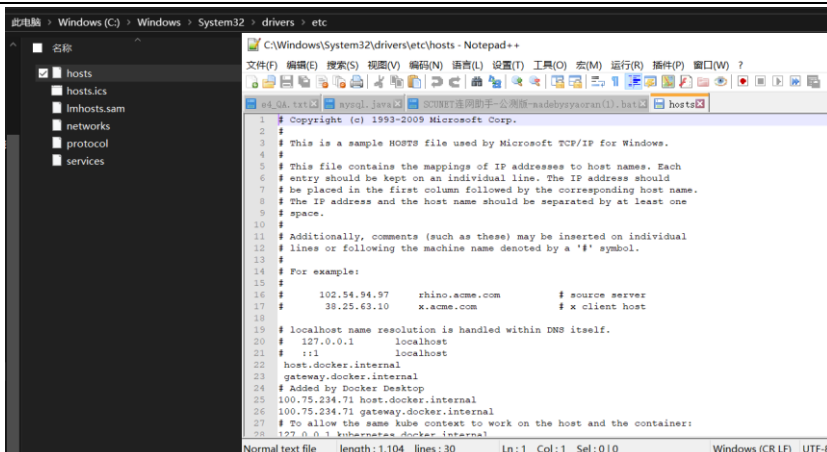


四川大学计算机学院、软件学院

实验报告

学号：2017141051019 姓名：王崇智 专业：计算机科学与技术 班级：173041014 第 8 周

课程名称	计算机网络课程实验	实验课时	2 课时
实验项目	DNS 服务器的配置与分析	实验时间	2019/10/22
实验目的	1) 掌握 DNS 服务器地址、hosts 文件的配置； 2) 了解 DNS 域名解析的工作流程。		
实验环境	Windows10 + wireshark + 小米手机		
实验内容 (算法、程序、步骤和方法)	<p>(一) 基础分</p> <p># 清除 DNS 缓存</p> <p>在 cmd 中输入 ipconfig/flushdns 即可完成</p>  <p>#</p> <p>本机实验环境为 windows 操作系统，经资料查阅后了解到 hosts 文件即本地的一个缓存文件存储在 C:\Windows\System32\drivers\etc 中，利用 notepad++打开文件后发现如下结果</p>		



```
1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host name.
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 # 102.54.94.97 rhino.acme.com # source server
17 # 38.25.63.10 x.acme.com # x client host
18 #
19 # localhost name resolution is handled within DNS itself.
20 # 127.0.0.1 localhost
21 # ::1 localhost
22 host.docker.internal
23 gateway.docker.internal
24 # Added by Docker Desktop
25 100.75.234.71 host.docker.internal
26 100.75.234.71 gateway.docker.internal
27 # To allow the same kube context to work on the host and the container:
28 127.0.0.1 kubernetec.docker.internal
```

以管理员身份打开文件后在下方的空白行中添加

202.108.22.5 www.baidu.com

这样的语句即可，即要保证 ip 地址与域名之间至少保留一个空格，保存之后。即完成了在本地的 hosts 文件中建立百度的 IP 地址与百度的域名之间的联系，即最终实现了加快域名解析的作用

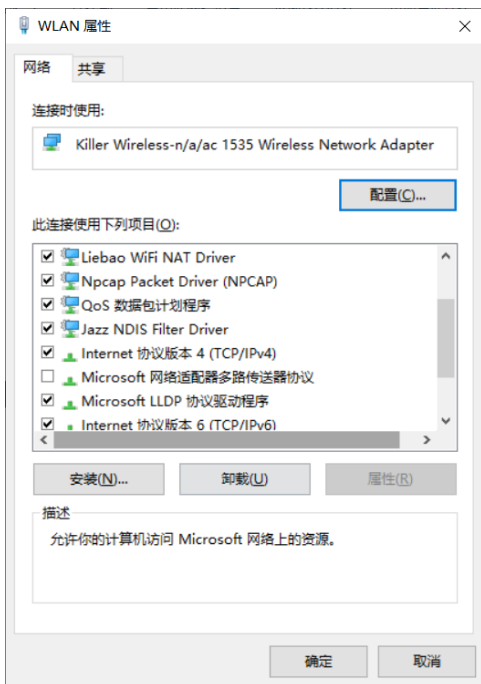
#

在 windows 中设置首选 dns 服务器的 ip 地址

在控制面板中打开**网络和 Internet**



当前实验环境处于连接手机热点的状态，右键该 WLAN



双击 ipv4 的设置



可以看出当前默认设置获得 ip 地址与 dns 服务器地址都是自动的操作。点击下方使用下面的 DNS

服务器地址，并在首选 DNS 服务器中输入自己想要使用的 ip，这里试用 8.8.8.8 作为例子。

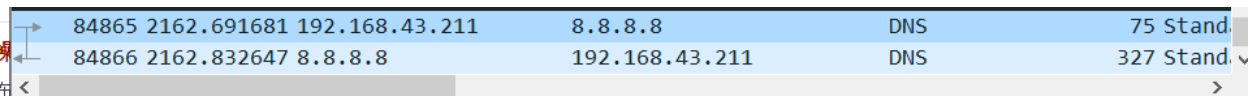


点击重新连接 WLAN 观察设置效果

属性	
SSID:	ow
协议:	Wi-Fi 4 (802.11n)
安全类型:	WPA2-个人
网络频带:	2.4 GHz
网络通道:	6
IPv6 地址:	240e:d8:7d2:4b68:fc12:b3ee:a266:c7bd
本地链接 IPv6 地址:	fe80::fc12:b3ee:a266:c7bd%12
IPv4 地址:	192.168.43.211
IPv4 DNS 服务器:	8.8.8.8
制造商:	Qualcomm Atheros Communications Inc.
描述:	Killer Wireless-n/a/ac 1535 Wireless Network Adapter
驱动程序版本:	12.0.0.722
物理地址(MAC):	9C-B6-D0-1E-C1-2D

观察到下方的 IPv4 DNS 服务器地址已经变为预先设置的 DNS 地址，即说明到此设置首选服务器的操作成功。但是我们知道 8.8.8.8 为谷歌的服务器地址，正常情况下国内网络条件发出的请求应该不会被允许，即发生超时等失败情况。而实验在浏览器中输入希望的域名，

请求应该不会收到反馈。但是利用 wireshark 抓包后发现，以 www.hupu.com 为例



84865 2162.691681 192.168.43.211 8.8.8.8 DNS 75 Standard query 0x6392 AAAA d.docs.live.net

实际上是可以发送并在本地接收的，这个问题在接下来的实验中看看能不能找到原因。

由于又在校园网环境下进行实验，不得不要了解一下公有地址与私有地址的区别与联系。

我们平时用 ipconfig 查出来的地址是本机的 ip 地址，也是内网的私有地址，这类地址仅在局域网使用，不能联通外网。而百度查 ip 的时候反馈的是公有地址，这个时候通常不是你主机的地址，而是运行商分给你的地址，以用于连接互联网

如 ping baidu.com 时，在过滤条件为 `ip.src == 10.132.6.179` (连接校园网条件下通过 ipconfig 查询到此时被分配到的 ip 地址为 10.132.6.179) 时，结果如下

```
C:\Users\Chris>ping baidu.com
正在 Ping baidu.com [39.156.69.79] 具有 32 字节的数据:
来自 39.156.69.79 的回复: 字节=32 时间=40ms TTL=49
来自 39.156.69.79 的回复: 字节=32 时间=42ms TTL=49
来自 39.156.69.79 的回复: 字节=32 时间=39ms TTL=49
来自 39.156.69.79 的回复: 字节=32 时间=39ms TTL=49

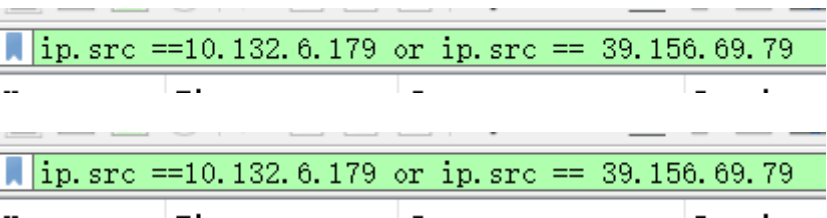
39.156.69.79 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 39ms, 最长 = 42ms, 平均 = 40ms
```

52989	1104.291673	10.132.6.179	39.156.69.79	ICMP	74 Echo (ping) request	id=0x0001, seq=62/15872, ttl=128 (reply in 16974)
53815	1105.212849	10.132.6.179	202.108.22.5	TCP	54 [TCP Retransmission]	59557 → 443
53820	1105.295866	10.132.6.179	39.156.69.79	ICMP	74 Echo (ping) request	id=0x0001, seq=62/15872, ttl=128 (reply in 16973)
53823	1105.409332	10.132.6.179	202.108.22.5	TCP	54 [TCP Retransmission]	59529 → 443
53824	1105.410312	10.132.6.179	202.108.22.5	TCP	54 [TCP Retransmission]	59528 → 443
53834	1105.622615	10.132.6.179	13.227.53.116	TCP	55 [TCP keep-alive]	59509 → 443 [ACK] Seq=1874 Ack=5
53855	1106.263082	10.132.6.179	182.61.200.166	TCP	54 59556 → 443 [ACK]	Seq=1874 Ack=5
53860	1106.299644	10.132.6.179	39.156.69.79	ICMP	74 Echo (ping) request	id=0x0001, seq=63/16128, ttl=128 (reply in 16978)
53881	1107.098232	10.132.6.179	107.182.177.1...	UDP	261 55388 → 63318	len=219

命令行中显示想 ip 地址为 (39.156.69.79) 发送了四次数据包, 而右侧的 wireshark 中也抓到了这个数据传输的过程

16971	569.930130	10.132.38.45	202.115.32.39	DNS	69 Standard query 0xa939 A baidu.com
16972	569.933740	202.115.32.39	10.132.38.45	DNS	271 Standard query response 0xa939 A baidu.com A 220.181.38.148 A 39.156.69.79 NS ns3.ba
16973	569.941603	10.132.38.45	220.181.38.148	ICMP	74 Echo (ping) request id=0x0001, seq=62/15872, ttl=128 (reply in 16974)
16974	569.976045	220.181.38.1...	10.132.38.45	ICMP	74 Echo (ping) reply id=0x0001, seq=62/15872, ttl=45 (request in 16973)
16977	570.946394	10.132.38.45	220.181.38.148	ICMP	74 Echo (ping) request id=0x0001, seq=63/16128, ttl=128 (reply in 16978)
16978	570.981514	220.181.38.1...	10.132.38.45	ICMP	74 Echo (ping) reply id=0x0001, seq=63/16128, ttl=45 (request in 16977)
16979	571.251805	120.204.17.1...	10.132.38.45	OICQ	393 OICQ Protocol
16980	571.252185	10.132.38.45	120.204.17.122	OICQ	97 OICQ Protocol

而我们所 ping 的域名需要先经过应用层 dns 处理得到真正的 ip 地址后，客户端才能真正捕获到 ICMP 的结果。



而在过滤捕获条件为上图时，抓包内容抓到了四次如下的内容。及说明发送接收确实都成功进行。

64900	1401.289982	10.132.6.179	39.156.69.79	ICMP	74 Echo (ping) request	id=0x0001, seq=42/10752, ttl=128 (reply in 64905)
64905	1401.330189	39.156.69.79	10.132.6.179	ICMP	74 Echo (ping) reply	id=0x0001, seq=42/10752, ttl=49 (request in 64900)

以 649001401.289982 10.132.6.179 39.156.69.79 ICMP 74 Echo (ping) request id=0x0001, seq=42/10752, ttl=128 (reply in 64905)为例，双击该记录，弹出如下窗口

Wireshark · 分组 64900 · WLAN

```
> Frame 64900: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
> Ethernet II, Src: RivetNet_1e:c1:2d (9c:b6:d0:1e:c1:2d), Dst: RuijieNe_4c:47:53 (58:69:6c:4c:47:53)
> Internet Protocol Version 4, Src: 10.132.6.179, Dst: 39.156.69.79
> Internet Control Message Protocol
```

0000	58 69 6c 4c 47 53 9c b6 d0 1e c1 2d 08 00 45 00	XillGS... ..E.
0010	00 3c 29 c3 00 00 80 01 92 dc 0a 84 06 b3 27 9c	..<).....'.
0020	45 4f 08 00 4d 31 00 01 00 2a 61 62 63 64 65 66	E0..M1.. ..*abcdef
0030	67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76	ghijklmn opqrstuv
0040	77 61 62 63 64 65 66 67 68 69	wabcdefg hi

除了十六进制数据，需要格外关注的就是封包的详细信息
具体来说

Frame: 物理层的数据帧概况

Ethernet II: 数据链路层以太网的头部信息

Internet Protocol Version 4: 互联网层 IP 包头部信息

Transmission Control Protocol: 传输层 T 的数据段头部信息，这里说的时 ICMP

因为对这里新出现 ICMP 协议不太了解，所以进行了必要的资料查阅工作

- ICMP(internet control message protocol) 即 internet 控制报文协议。它是 TCP/IP 协议族的一个子协议，通常用于在 IP 主机、路由器之间传递控制消息。而控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。虽然这类控制消息并不传输用户数据，但是对于用户数据能否传递起着很重要的作用。

- ICMP 提供一致易懂的出错报告信息。发送的出错报文返回到发送原数据的设备，因为只有发送设备才是出错报文的逻辑接受者。发送设备随后可根据 ICMP 报文确定发生错误的类型，并确定如何才能更好地重发之前失败的数据包

- 我们“ping”操作的过程实际上就是 icmp 协议工作的过程，另外其他的网络命令如跟踪路由的 tracert 命令也是基于 ICMP 协议的。

- ICMP 协议是 IP 的附属协议，介于 IP 层和 TCP 层之间，一般认为其属于 IP 层协议。而 IP 协议用它来与其他主机或路由器交换错误保温和其他的一些网络情况。

通常详细信息中也会含有 hypertext transfer protocol: 即应用层的信息，如 http 协议

同时，还要明确递归解析的思路，即在这种解析方式中，如果客户端配置的本地服务器不能解析的话，则后面的查询全由本地名称服务器代替 DNS 客户端进行查询，直到本地名称服务器从权威域名服务器得到正确的解析结果，然后由本地名称服务器告诉 DNS 客户端查询结果。

下利用 nslookup 进行尝试，不指定 dns-server，利用系统默认的 dns 服务器进行尝试

iP

本机IP: 221.10.55.148 四川省成都市 联通

请输入ip地址

查询

下图为一个完整的查询流程

C:\Users\Chris>nslookup www.baidu.com	1556... 33.642076	10.132.6.179	202.115.32.39	DNS	73 Standard query 0x0002 A www.baid
服务器: dart.scu.edu.cn	1556... 33.645670	202.115.32.39	10.132.6.179	DNS	302 Standard query response 0x0002 A
Address: 202.115.32.39	1556... 33.646422	10.132.6.179	202.115.32.39	DNS	73 Standard query 0x0003 AAAA www.b
非权威应答:	1556... 33.649640	202.115.32.39	10.132.6.179	DNS	157 Standard query response 0x0003 A
名称: www.a.shifen.com	1563... 59.435992	10.132.6.179	202.115.32.39	DNS	88 Standard query 0xb42e A chshap.b
Addresses: 182.61.200.7	1563... 59.436223	10.132.6.179	202.115.32.39	DNS	88 Standard query 0x323f AAAA chsha
182.61.200.6	1563... 59.441572	202.115.32.39	10.132.6.179	DNS	164 Standard query response 0x323f N
Aliases: www.baidu.com	1563... 59.441868	202.115.32.39	10.132.6.179	DNS	164 Standard query response 0xb42e N

▼ Domain Name System (query)

Transaction ID: 0x0003

> Flags: 0x0100 Standard query

Questions: 1

Answer RRs: 0

Authority RRs: 0

Additional RRs: 0

> Queries

[\[Response In: 155612\]](#)

点击该记录，出现如下结果。

其中该图片即显示了应用层（DNS）的详细信息，注意这是向 dns 服务器发送的请求，

现选择 1556... 33.649640 202.115.32.39 10.132.6.179 DNS 157 Standard query response 0x0003。即 dns 服务器发送回的数据

```


▼ Domain Name System (response)
  Transaction ID: 0x0003
  > Flags: 0x8180 Standard query response, No error
  Questions: 1
  Answer RRs: 1
  Authority RRs: 1
  Additional RRs: 0
  ▼ Queries
    > www.baidu.com: type AAAA, class IN
  ▼ Answers
    > www.baidu.com: type CNAME, class IN, cname www.a.shifen.com
  ▼ Authoritative nameservers
    > a.shifen.com: type SOA, class IN, mname ns1.a.shifen.com
    [Request In: 155610]
    [Time: 0.003218000 seconds]

```

```

▼ Authoritative nameservers
  > a.shifen.com: type SOA, class IN, mname ns1.a.shifen.com

```

即为权威域名服务器  [Time: 0.003218000 seconds] 位于最下方的这条记录即为请求的生存时间

查询一个新的网站，如 hupu

88 4.113822	10.132.6.179	202.115.32.39	DNS	72 Standard query 0x6a3c A www.hupu.com
89 4.114033	10.132.6.179	202.115.32.39	DNS	72 Standard query 0xec88 AAAA www.hupu.com
90 4.140941	10.132.6.179	202.115.32.36	DNS	72 Standard query 0xec88 AAAA www.hupu.com
91 4.140941	10.132.6.179	202.115.32.36	DNS	72 Standard query 0x6a3c A www.hupu.com
92 4.149891	202.115.32.39	10.132.6.179	DNS	516 Standard query response 0x6a3c A www.hupu.com CNAME www.hupu.com.w.kunlungr.com A 222.22.29.84 A 222.22.29.96 A 222.22.29.98 A...
93 4.149980	202.115.32.39	10.132.6.179	DNS	162 Standard query response 0xec88 AAAA www.hupu.com CNAME www.hupu.com.w.kunlungr.com SOA ns3.kunlungr.com
95 4.178257	202.115.32.36	10.132.6.179	DNS	162 Standard query response 0xec88 AAAA www.hupu.com CNAME www.hupu.com.w.kunlungr.com SOA ns3.kunlungr.com

该过程所有的请求即接受结果如下，10.132.6.179 为公网 IP，202.114.32.39 为首选 DNS 服务器，202.155.32.36 为备选服务器，而封包中 type=A 表示为 ipv4，type=AAAA 表示为 ipv6 只选其中一组进行研究。

序号 88:

```

▼ Domain Name System (query)
  Transaction ID: 0x6a3c
  > Flags: 0x0100 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  ▼ Queries
    > www.hupu.com: type A, class IN
    [Response In: 92]

```


序号 92:

```

  Domain Name System (response)
    Transaction ID: 0x6a3c
    > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 17
    Authority RRs: 3
    Additional RRs: 6
  Queries
    > www.hupu.com: type A, class IN
  Answers
    > www.hupu.com: type CNAME, class IN, cname www.hupu.com.w.kunlungr.com
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.84
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.96
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.98
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.99
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.82
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.101
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.100
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.94
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.86
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.81
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.95
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.80
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.85
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.97
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.83
    > www.hupu.com.w.kunlungr.com: type A, class IN, addr 222.22.29.87
  Authoritative nameservers
    > w.kunlungr.com: type NS, class IN, ns ns3.kunlungr.com
    > w.kunlungr.com: type NS, class IN, ns ns5.kunlungr.com
    > w.kunlungr.com: type NS, class IN, ns ns4.kunlungr.com
  Additional records
    [Request In: 88]
    [Time: 0.036069000 seconds]
    - Transaction id: 标识符字段、唯一的
    - flages: 标志位
      ○ response: 消息，是一个请求查询
      ○ opcode: 标准信息查询
      ○ truncated: 截断，可判断消息是否被阶段
      ○ Recursion desired: 是否期望递归查询
    - Questions: 问题个数
    - Answer RRs: 回答问题个数
    - Queries: 问题区
      ○ 即可问该域名的 ip 地址是多少
    - Answers: 回答区
  【Time】即为记录的生存
    > www.hupu.com: type CNAME, class IN, cname www.hupu.com.w.kunlungr.com

```

代表具备规范名称,而

```
type A, class IN, addr 222.22.29.84
type A, class IN, addr 222.22.29.96
type A, class IN, addr 222.22.29.98
type A, class IN, addr 222.22.29.99
type A, class IN, addr 222.22.29.82
type A, class IN, addr 222.22.29.101
type A, class IN, addr 222.22.29.100
type A, class IN, addr 222.22.29.94
type A, class IN, addr 222.22.29.86
type A, class IN, addr 222.22.29.81
type A, class IN, addr 222.22.29.95
type A, class IN, addr 222.22.29.80
type A, class IN, addr 222.22.29.85
type A, class IN, addr 222.22.29.97
type A, class IN, addr 222.22.29.83
type A, class IN, addr 222.22.29.87
```

为查询到的 ip 结果。通常处于使用状态的即为前一两条 ip

该实验操作表明，DNS 客户端向本地 DNS 服务器发送一次请求后即返回权威域名服务器地址，说明之前已预先被缓存。所以不必执行完整的递归解析流程。

其中 authoritative nameservers 即为权威域名服务器；这个逐步解析必要的内容，每次从上层结点查询的时候，其查询的对象即为权威域名服务器内的表。

下方的 Time：即为各记录的生存时间；

通常情况下，DNS 客户端首先向本地名称服务器发出解析 www.hupu.com 域名的 DNS 请求报文，然后其先查看本地缓存。如果查到了该域名的对应记录，则直接向 DNS 客户返回其查询的内容。由于本次使用的校园网连接，且利用自动分配的 IP，本地名称服务器内已存有大量常用的 ip 地址；

但通常情况下，假设没有查到该域名在本地缓存中的记录，则本地名称服务器就会向所配置的跟名称服务器发出解析请求，再通过查询得到的顶级域名服务器检查其缓存，假设也没有该域名的记录，则继续向本地名称服务器发送含有而域名服务器对应的地址的请求解析 www.hupu.com 域名的新 DNS 请求报文。一直持续该行为至权威域名服务器。而在权威域名服务器收到 DNS 请求后，在他的 DNS 区域数据库中进行查找，最终得到了 www.hupu.com 域名所对应的 ip 地址。然后向本地名称服务器返回该 DNS 应答报文。最后本地名称服务器接受权威域名服务器信息后，向 DNS 客户端返回一条 DNS 应答报文，并告诉其域名所对应的 IP 地址。

传输层详细信息如下

```
▼ User Datagram Protocol, Src Port: 51277, Dst Port: 53
  Source Port: 51277
  Destination Port: 53
  Length: 39
  Checksum: 0x6448 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 1504]
  ▼ [Timestamps]
    [Time since first frame: 0.000000000 seconds]
    [Time since previous frame: 0.000000000 seconds]
```

使用的是 UDP 协议，src 端口号为 51277，dst 端口号为 53

（二） 扩展加分点

1. dns 为什么使用 udp 而不是 tcp

a. udp 传输速度更快。tcp 传输相对较慢，因为它要求必须实现三次握手。为了实现负载均衡，避免过多连接同时向一个服务器发起，采用不需要保持连接的 udp 更合适

b. DNS 传输时的信息量要求不大 udp 片段就可以很好满足其需求

c. udp 不够可靠，但其可靠性可以额外的加入到应用层中。一个应用可以使用 udp，并且在添加超时和重发之后其能变为可靠的。

d. 面向无连接的 udp 虽然快，加快了解析速度。但是实际上近些年来 dns 逐渐转向利用 tcp 传输，因为 udp 传输信息有限，也有时经常出现非重传的裸 udp，丢包现象严重，以及伪造等安全因素的考虑。udp 的使用率开始下降。

2. 解释递归查询和迭代查询，并画出示意图

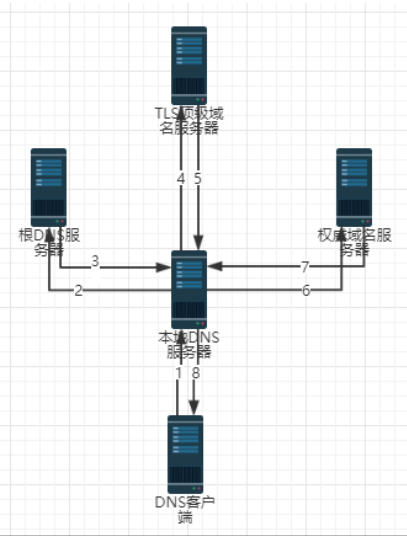
（1）递归查询

递归查询是一种 DNS 服务器的查询模式，在该模式下 DNS 服务器接收到客户机请

求，必须使用一个准确的查询结果回复客户机。如果 DNS 服务器本地没有存储查询 DNS 信息，那么该服务器会询问其他服务器，并将返回的查询结果提交给客户机。

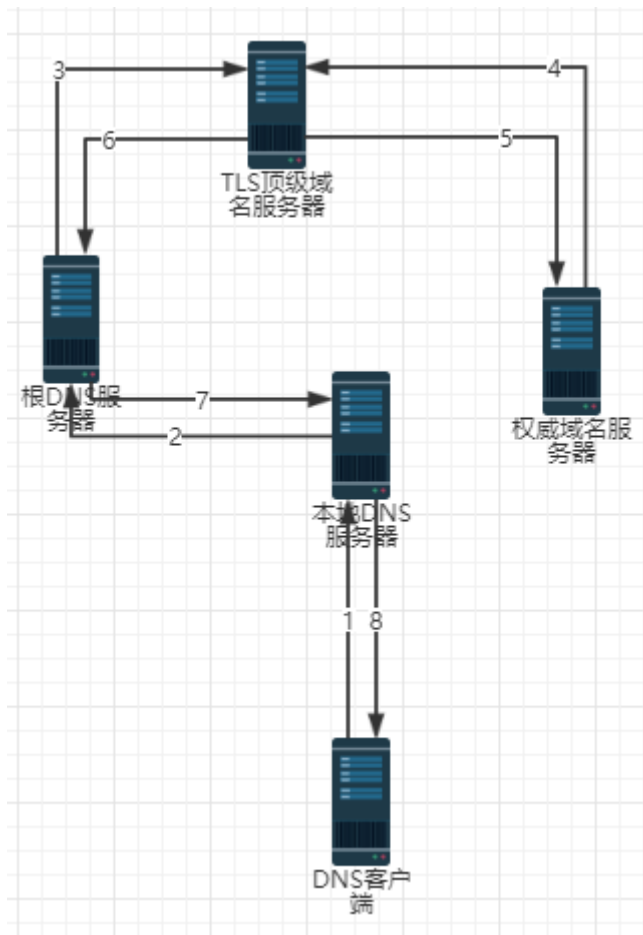
(2) 迭代查询

DNS 服务器另外一种查询方式为迭代查询，DNS 服务器会向客户机提供其他能够解析查询请求的 DNS 服务器地址，当客户机发送查询请求时，DNS 服务器并不直接回复查询结果，而是告诉客户机另一含有目标地址的 DNS 服务器地址，客户机再向这台 DNS 服务器提交请求，依次循环直到返回查询的结果为止



该图为递归与迭代方式相结合。 注意到该图中从 DNS

客户端到本地 DNS 服务器的查询为递归查询，因为该查询请求是本地 DNS 服务器以自己的名义获得的映射，其他三个查询为迭代查询，因为这个三个查询对应的回答都是直接返回给查询发起的一方，即本地 DNS 服务器。



递归方式，该图显示了一条查询链，最初由 DNS 客户机先向本地 DNS 服务器发送请求，通常来说如果存在缓存，则会直接向 DNS 客户机返回请求结果，如果没有缓存，则会从当前服务器继续根服务器请求解析，而如果仍未找到满足请求的地址则会继续向下一层服务器发出查询请求，即向 TLS 发送请求，此时之前发出请求的服务亲均未收到回答。如果此时仍未在对应的表中找到对应的 ip 则会继续向下一层即权威域名服务器发送请求，直到得到结果，沿着查询经过的服务器路径，层层回传解析结果。最后由本地 DNS 服务器传送给 DNS 客户端。同时还会再 DNS 服务器中对该之前未保存的请求进行缓存工作，以方便下一次任意 DNS 客户端发出的相同请求，加快解析，节约时间。但纯递归方式不符合实际，因为全球的根服务器，顶级域名服务器数量及幸能有限。如果全世界的用户同时通过其本地 DNS 服务器发送请求，且均未预先缓存的情况下，所有请求将会在根或顶级域名服务器处排队拥塞，破坏负载均衡。同时会影响客户端自身的性能。

3. 列举集中与 DNS 相关的网络攻击方式

a. 分布式拒绝服务（DDoS）带宽洪泛攻击

i. 由于通常服务器的接入速率有限，只要 R bps 不是很大，单一攻击源就可以产生足够大的流量来上海服务器。但是如果 R 非常大时，攻击者可以采用分布式的方式进行攻击，即其利用受害主机的僵尸网络，让每个源向目标猛烈发送流量。

- ii. 其原理即攻击者向目标主机发送大量的分组，分组总数量之多使得目标的接入链路变得拥塞，所有请求都在队列中等待，使得合法的分组长时间无法到达服务器而影响正常工作。
- iii. 但实际上，许多 DNS 根服务器受到了分组过滤器的保护，配置的分组过滤器阻挡了所有只想根服务器的 PING icmp 报文。而且大多数本地 DNS 服务器缓存了顶级域名服务器的 IP 地址，使得这些大量的请求过程通常绕过了 DNS 根服务器
- iv. 所以，更有效的 DDoS 攻击方式为向顶级域名服务器发送大量的 dns 请求，因为过滤指向 DNS 服务器的 DNS 请求更加困难，且不会像根服务器那样容易绕过
- b. 中间人攻击
 - i. 攻击者截获来自主机的请求并返回伪造的回答
- c. DNS 毒害攻击
 - i. 攻击者向一台 DNS 服务器发送伪造的回答，诱使服务器在它的缓存中接收伪造的记录。
- d. 充分利用 DNS 基础设施对目标主机发起 DDoS 攻击
 - i. 攻击者直接向权威 DNS 服务器发送 DNS 请求，但是每个请求具有目标攻击服务器的假冒源地址。这样，权威 DNS 服务器在解析请求之后直接向目标攻击主机返回其回答。如果这些请求的一些参数被精心修改，即响应的字节数远大于请求的字节数，攻击者就不必自行产生大量的流量即可满足获得大量的流量来淹没目标主机

4. 利用电脑工具，猎豹 wifi 助手，开启热点，利用自己的小米手机连接该热点，利用 wireshark 尝试抓包

109370	1159.215370	10.132.249.183	117.156.18.55	HTTP	752	GET	/gchatpic_new/1921048037304074523C18905EE2462CF6E85A9F51F418A9621D6A768ED444089153869040837FCED27F802C789C5C827FE15C49122EAE80B3F1B2687F05726FA128A54AA6E2...
109378	1159.245811	10.132.249.183	117.156.18.55	HTTP	752	GET	/gchatpic_new/1921048037304074523C18905EE2462CF6E85A9F51F418A9621D6A768ED444089153869040837FCED27F802C789C5C827FE15C49122EAE80B3F1B2687F05726FA128A54AA6E2...
109380	1159.271268	117.156.18.55	10.132.249.183	HTTP	1448	HTTP/1.1 206	Partial Content (image/jpeg)

设定过滤后找到一条 get 请求

标记/取消标记 分组(M)

忽略/取消忽略 分组(I)

设置/取消设置 时间参考

时间平移...

分组注释...

编辑解析的名称

作为过滤器应用

准备过滤器

对话过滤器

对话着色

SCTP

追踪流

复制

协议首选项

解码为(A)...

在新窗口显示分组(W)

Ctrl+M

Ctrl+D

Ctrl+T

Ctrl+Shift+T

Ctrl+Alt+C

右键追踪流并选择 http

105316 - 199.772736	10.132.249.183	117.156.18.55	TCP	74 12565 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=4000918988 TSecr=0 WS=512
105317 - 199.744555	117.156.18.55	10.132.249.183	TCP	66 80 → 12565 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1394 SACK_PERM=1 WS=128
105318 - 199.748267	10.132.249.183	117.156.18.55	TCP	54 12565 → 80 [ACK] Seq=1 Ack=1 Win=88064 Len=0
105319 - 199.739769	10.132.249.183	117.156.18.55	HTTP	752 GET /gchatpic_new/A6701E3C3CC002A658BA93A0526542EC00BF928EC67A25A2A5EF30D4C833A0562098307978048803A0D1ABC8B8DF33E48EBFF28738FA4239EB77338328747499D0A075C65F43A1A
105320 - 199.712342	117.156.18.55	10.132.249.183	TCP	66 80 → 12565 [ACK] Seq=1 Ack=699 Win=16000 Len=0
105321 - 199.710222	117.156.18.55	10.132.249.183	HTTP	1448 HTTP/1.1 206 Partial Content (image/jpeg)
105322 - 199.710202	117.156.18.55	10.132.249.183	HTTP	1448 Continuation
105323 - 199.710201	117.156.18.55	10.132.249.183	HTTP	241 Continuation
105324 - 199.706280	10.132.249.183	117.156.18.55	TCP	54 12565 → 80 [ACK] Seq=699 Ack=1395 Win=90624 Len=0
105325 - 199.705972	10.132.249.183	117.156.18.55	TCP	54 12565 → 80 [ACK] Seq=699 Ack=2789 Win=93184 Len=0

过滤后的新结果如下，get 之前的封包记录

10.132.249.183	117.156.18.55	TCP	74 12565 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=4000918988 TSecr=0 WS=512
117.156.18.55	10.132.249.183	TCP	66 80 → 12565 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1394 SACK_PERM=1 WS=128
10.132.249.183	117.156.18.55	TCP	54 12565 → 80 [ACK] Seq=1 Ack=1 Win=88064 Len=0

简单来说，这就是三次握手。即，第一次握手，a 向 b 发送信息，b 收到信息后可以确认自己的收信能力和 a 的发信能力没有问题。第二次握手，b 向 a 发送信息，a 可以确认自己的发信能力和收信能力没有问题，但此时 b 不知道自己的发信能力如何，所以需要第三次通信。第三次握手 a 向 b 发送信息，b 就可以确定自己的发信能力没有问题。而上图也成功表示 wireshark 接我了三手握手的三个数据包，第四个包为 http，说明 http 确实使用 tcp 协议来建立连接的。

注意到第一个记录中

```
Sequence number: 0      (relative sequence number)
[Next sequence number: 0      (relative sequence number)]
Acknowledgment number: 0
1010 .... = Header Length: 40 bytes (10)
Flags: 0x002 (SYN)
```

第二条记录中

```
Sequence number: 0      (relative sequence number)
[Next sequence number: 0      (relative sequence number)]
Acknowledgment number: 1      (relative ack number)
1000 .... = Header Length: 32 bytes (8)
Flags: 0x012 (SYN, ACK)
```

第三条记录中，返回 ack 1，并把服务器发来的字段 sequence number+1


```
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)
... - - -
```

可以从这三条记录中看除 flag 在不断进行变化

在连接之后，向目标服务器发送 get 请求

```
Hypertext Transfer Protocol
v [truncated]GET /gchatpic_new/A67D1E3C3CCB02A658BA93A0526542EC00BF928EC67A25A2A5EF3DD4C833A056209B3D7978B48803AAD1ABC888DF33E48EBFF28738AFA239EB7733832B747499D0A075C65F434
> [ [truncated]Expert Info (Chat/Sequence): GET /gchatpic_new/A67D1E3C3CCB02A658BA93A0526542EC00BF928EC67A25A2A5EF3DD4C833A056209B3D7978B48803AAD1ABC888DF33E48EBFF28738AFA
Request Method: GET
> Request URI [truncated]: /gchatpic_new/A67D1E3C3CCB02A658BA93A0526542EC00BF928EC67A25A2A5EF3DD4C833A056209B3D7978B48803AAD1ABC888DF33E48EBFF28738AFA239EB7733832B747499E
Request Version: HTTP/1.1
> Cookie: ST=00015DAF9FC10058FB080F8156901A040F2E17061C97965171C58AA2A29F0384A59FED6AA1B4EB9FAFB780EA519B4E60A37B6907AF12F14266C9D881D8EB311B40EA48BC16E5558FD21DE65D8E8796EE
Referer: http://im.qq.com/mobileqq/\r\n
Accept-Encoding: identity\r\n
Range: bytes=0-\r\n
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; MI 9 MIUI/V10.2.35.0.PFAC10X)\r\n
Host: 117.156.18.55\r\n
Connection: Keep-Alive\r\n
\r\n
[Full request URI [truncated]: http://117.156.18.55/gchatpic_new/A67D1E3C3CCB02A658BA93A0526542EC00BF928EC67A25A2A5EF3DD4C833A056209B3D7978B48803AAD1ABC888DF33E48EBFF28738AFA239EB7733832B747499E
[HTTP request 1/2]
```

上半部分即为请求的相关信息，如 request method 表明其类型为 GET，其目标 uri 和使用的 http 协议版本均在上面显示出来。而下半部分即为该请求携带的 headers 信息。从 user-agent 可以看出这个数据包是从手机端抓下来的。host 即为其请求的目标服务器，connection 为 keep-alive 即使用持久化连接的方式，以使得建立一次 tcp 连接可以连续传输文件。

注意到其 response 部分

117.156.18.55	10.132.249.183	TCP	60 80 → 12565 [ACK] Seq=1 Ack=699 Win=16000 Len=0
117.156.18.55	10.132.249.183	HTTP	1448 HTTP/1.1 206 Partial Content (image/jpeg)
117.156.18.55	10.132.249.183	HTTP	1448 Continuation
117.156.18.55	10.132.249.183	HTTP	241 Continuation

目标服务器先向本服务器发送一次 ack 为 1 的 tcp 信息，（是否是向本地服务器发送确认信息，告知其发信能力没有问题。）然后进行 response 行为：

该例子中返回了三次 http 信息，且长度分别为 1448，1448，241. 可以知道该条件下网络传输 http 最长的长度为 1448，过长的内容将被分包传输。

打开其应用层详细信息


```

Hypertext Transfer Protocol
  HTTP/1.1 206 Partial Content\r\n
    > [Expert Info (Chat/Sequence): HTTP/1.1 206 Partial Content\r\n]
      Response Version: HTTP/1.1
      Status Code: 206
      [Status Code Description: Partial Content]
      Response Phrase: Partial Content
      Server: ImgHttp3.0.0\r\n
      Connection: keep-alive\r\n
      Content-Type: image/jpeg\r\n
    > Content-Length: 2610\r\n
      Last-Modified: Tue, 22 Oct 2019 11:42:32 GMT\r\n
      Cache-Control: max-age=2592000\r\n
      X-Delay: 568 us\r\n
      X-Info: real data\r\n
      X-Cpt: filename=0\r\n

```

即反映了其具体的 response 信息。其类型为 partial content，传输文件的类型，以及协议，server 的等的信息都在里面被记录了。

而与这三次分包传输相对应的即为

10.132.249.183	117.156.18.55	TCP	54 12565 → 80 [ACK] Seq=699 Ack=1395 Win=90624 Len=0
10.132.249.183	117.156.18.55	TCP	54 12565 → 80 [ACK] Seq=699 Ack=2789 Win=93184 Len=0
10.132.249.183	117.156.18.55	TCP	54 12565 → 80 [ACK] Seq=699 Ack=2976 Win=96256 Len=0

本地服务器向目标服务器回传确认信息，以表明实际已接收到之前所请求的内容。

117.156.18.55	10.132.249.183	TCP	60 80 → 12565 [FIN, ACK] Seq=2976 Ack=699 Win=16000 Len=0
---------------	----------------	-----	---

其后跟的 FIN 即代表 connection finish，说明 server 端本次传输已完成，其发起请求，试图请求本次连接

10.132.249.183	117.156.18.55	TCP	54 12565 → 80 [ACK] Seq=699 Ack=2977 Win=96256 Len=0
10.132.249.183	117.156.18.55	HTTP	752 GET /gchatpic_new/19210480373D4D74523C18905EE2462CF6E85A9F5
10.132.249.183	117.156.18.55	TCP	54 12565 → 80 [FIN, ACK] Seq=1397 Ack=2977 Win=96256 Len=0

分析其后三条记录，均为本地服务器向目标服务器发送的内容，分别为 ack，即告诉对方目前本地请求还未结束，请等待，后接 http get 请求即完成其需要的后续动作，最后向目标服务器发送 fin，ack 告知对方，自己的操作已经完成，这边可以进行关闭连接。

但实际情况目标服务器应该返回一个携带 ack 标志的信息以进行正常回应。

109375	-156.446899	117.156.18.55	10.132.249.183	TCP	60 80 → 12565 [RST] Seq=2977 Win=0 Len=0
109376	-156.446898	117.156.18.55	10.132.249.183	TCP	60 80 → 12565 [RST] Seq=2977 Win=0 Len=0

但本次实际操作得到的两条记录未上图所示，其中 flag 为 RST

因为还未学到 TCP 其中的标志位信息，为了更好的理解本次实验，下又进行了资料查阅工作

查阅资料后发现 rst 表示复位，用来异常的关闭连接。而发送 RST 包关闭连接时，不必等待缓冲区的包都发出去（这里不像 FIN 包），直接丢弃缓存区的包，发送 RST 包。而接收端收到 RST 包后，也不必发送 ACK 包来进行确认。经常出现的两种情况如下：

- 1) a 向 b 发起连接，但 b 上未监听相应的端口，这时 b 上的 tcp 处理程序会发 rst 包
- 2) ab 已经建立正常连接，在通讯时，a 向 b 发送 fin 包要求关闭连接，b 在发出 ack 后，网断开了，即 a 并未实际收到。a 通过若干原因放弃了这个连接，网通后，B 又开始发送数据包，A 因为逻辑上已经断开与 B 的连接，此时不知道该数据包的来源，就发 RST 数据包再把该连接给关闭掉

注意到一个点，再持续传输 http 请求得到的对象时，三次发送返回得到了三个包，客户端确认了三次。实际上通过修改请求的机制，利用滑动窗口的方式可以加快数据传输。接收端可以等收到许多包后至发送一个 ack，这样发送端在每次发送完一个数据包后不用等待其 ack。当然如果利用流水线机制或者并发机制则可忽略这个点。一般使用的都是 http 持久化连接，且流水线模式工作。

因为 wireshark 主要是对 http 等的抓包，对现在主流的 https 不适用，如果想进行尝试抓 https 包的时候，就需要用到 fiddler 来进行操作了。

如图，在浏览器中输入 google.com 后，利用 fiddler 进行抓包尝试

65	200	HTTPS	www.google.com	/xjs/_/js/k=xjs.s.en_US.smoPhilCC1U....	143,228	public, ...	text/javascript; charset=UTF-8
66	200	HTTPS	www.gstatic.com	/og/_/js/k=og.og2.en_US.qJyM-dkx6...	51,861	public, ...	text/javascript; charset=UTF-8
67	204	HTTPS	www.google.com	/gen_204?s=webhp&t=aft&atyp=csi&...	0		text/html; charset=UTF-8
68	200	HTTPS	www.google.com	/xjs/_/js/k=xjs.s.en_US.smoPhilCC1U....	41,004	public, ...	text/javascript; charset=UTF-8

```
Session Properties (63) www.google.com/images/hpp/first_responder_day_2018.gif
SESSION STATE: Done.
The request was forwarded to the gateway.
Response Entity Size: 38028 bytes.

== FLAG ==
BitFlags: [IsHTTPS, ClientPipeReused, ServerPipeReused, SentToGateway] 0x019
U2-BACKCOLOR: Yellow
U2-OLDCOLOR: Gray
X-CLIENTIP: 1ffff:127.0.0.1
X-CLIENTPORT: 49187
X-EGRESSPORT: 49188
X-PROCESSTIME: chrome:4936
X-RESPONSEBODYTRANSFERLENGTH: 38,028
X-SERVERSOCKET: REUSE ServerPipe#6

== TIMING INFO ==
ClientConnected: 21:08:55.244
ClientBeginRequest: 21:08:55.603
GotRequestHeaders: 21:08:58.603
ClientDoneRequest: 21:08:58.603
Determine Gateway: 0ms
DNS Lookup: 0ms
TCP/IP Connect: 0ms
HTTPS Handshake: 0ms
ServerConnected: 21:08:55.245
FidderBeginRequest: 21:08:55.603
ServerGotRequest: 21:08:58.603
ServerBeginResponse: 21:08:59.218
GotResponseHeaders: 21:08:59.218
ServerDoneResponse: 21:08:59.767
ClientBeginResponse: 21:08:59.767
ClientDoneResponse: 21:08:59.767

Overall Elapsed: 0:00:03.164

The response was buffered before delivery to the client.

== WININET CACHE INFO ==
This URL is not present in the WININET cache. [Code: 2]
= Note: Data above shows WININET's current cache state, not the state at the time of the request.
= Note: Data above shows WININET's Medium Integrity (Non-Protected Mode) cache only.

<
Hit ESC to close, F5 to refresh, ALT+Up/Down to switch Session.
```

而这个即为一个抓到的图片包的 session，虽然具体的报文结构不了解，但是其中的 clientid 和 port 与 server 的 port 都可以认出。

<http://nhwztsfbmxrkcdvl.neverssl.com/online>

这里发现了一个纯的 http 网站

The image shows a Wireshark window titled "Wireshark · 追踪 HTTP 流 (tcp.stream eq 121) · WLAN". The main pane displays the details of a selected packet (packet 4126). The packet is a GET request for "/online" from the host "nhwztsfbmxrkcdvl.neverssl.com". The response is an HTTP/1.1 304 Not Modified status. The response headers include "Connection: keep-alive", "Date: Mon, 28 Oct 2019 12:46:14 GMT", "Server: AmazonS3", "Vary: Accept-Encoding", "X-Cache: Hit from cloudfront", "Via: 1.1 0490c35d7749fc1c1479f84160370a4b.cloudfront.net (CloudFront)", "X-Amz-Cf-Pop: HKG60-C1", "X-Amz-Cf-Id: UHk3cuHLEamuyKZKB0KAGk8anuA2sQ1k9Bb-fFoPs8tBavk_Sadrww==", and "Age: 41820". The bottom pane shows the packet bytes in ASCII format.

```
GET /online HTTP/1.1
Host: nhwztsfbmxrkcdvl.neverssl.com
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/77.0.3865.120 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
If-Modified-Since: Fri, 16 Feb 2018 21:32:40 GMT

HTTP/1.1 304 Not Modified
Connection: keep-alive
Date: Mon, 28 Oct 2019 12:46:14 GMT
Server: AmazonS3
Vary: Accept-Encoding
X-Cache: Hit from cloudfront
Via: 1.1 0490c35d7749fc1c1479f84160370a4b.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: HKG60-C1
X-Amz-Cf-Id: UHk3cuHLEamuyKZKB0KAGk8anuA2sQ1k9Bb-fFoPs8tBavk_Sadrww==
Age: 41820
```

分组 4126。1 客户端 分组，1 服务器 分组，1 turn(s)。点击选择。

整个对话 (852 bytes)

显示和保存数据为 ASCII

查找: 查找下一个 (N)

滤掉此流 打印 Save as... 返回 Close Help

该图即为追踪 http 流后得到的结果，可以看到本地发出的 get 请求与得到的 http 结果。且是经过软件预处理排版之后的

4117	105.910273	10.132.1.103	13.225.100.71	TCP	66 60186 → 80 [SYN] Seq=0 Win=64240 Len=0
4122	106.126045	13.225.100.71	10.132.1.103	TCP	66 80 → 60186 [SYN, ACK] Seq=0 Ack=1 Win=
4123	106.126393	10.132.1.103	13.225.100.71	TCP	54 60186 → 80 [ACK] Seq=1 Ack=1 Win=13235
4126	106.126832	10.132.1.103	13.225.100.71	HTTP	567 GET /online HTTP/1.1
4131	106.352170	13.225.100.71	10.132.1.103	TCP	60 80 → 60186 [ACK] Seq=1 Ack=514 Win=304
4132	106.352170	13.225.100.71	10.132.1.103	HTTP	393 HTTP/1.1 304 Not Modified
4133	106.392223	10.132.1.103	13.225.100.71	TCP	54 60186 → 80 [ACK] Seq=514 Ack=340 Win=1
5306	151.353016	10.132.1.103	13.225.100.71	TCP	55 [TCP Keep-Alive] 60186 → 80 [ACK] Seq=
5317	151.569907	13.225.100.71	10.132.1.103	TCP	66 [TCP Keep-Alive ACK] 80 → 60186 [ACK]
8365	196.571348	10.132.1.103	13.225.100.71	TCP	55 [TCP Keep-Alive] 60186 → 80 [ACK] Seq=
8379	196.801326	13.225.100.71	10.132.1.103	TCP	66 [TCP Keep-Alive ACK] 80 → 60186 [ACK]
9490	241.801373	10.132.1.103	13.225.100.71	TCP	55 [TCP Keep-Alive] 60186 → 80 [ACK] Seq=
9501	242.045794	13.225.100.71	10.132.1.103	TCP	66 [TCP Keep-Alive ACK] 80 → 60186 [ACK]
10632	287.046830	10.132.1.103	13.225.100.71	TCP	55 [TCP Keep-Alive] 60186 → 80 [ACK] Seq=
10641	287.235552	13.225.100.71	10.132.1.103	TCP	66 [TCP Keep-Alive ACK] 80 → 60186 [ACK]

结果。而且

可以发现该网站后端配置的连接方式为持久化连接，只要保持页面打开状态，客户端与服务

器会不断的收发信息。

Time	Source	Destination	Protoc	Length	Info
8 804.835318	10.132.1.103	202.115.32.39	DNS	89	Standard query 0x5092 A nhwztsfbmxrkcdvl.neverssl.com
13 805.036890	202.115.32.39	10.132.1.103	DNS	466	Standard query response 0x5092 A nhwztsfbmxrkcdvl.neverssl.com A 13.225.100.139 A 13.

访问过该域名，所以再次 ping 该地址后，抓包显示出以上两条信息，即没有经过其他 dns 服务器的查询，直接通过本地 DNS 服务器的反馈即找到了 ip 地址。因为之前

小结

1. 实验中给出了 dns 清楚缓存的命令
2. 实验中给出了 windows 条件下 hosts 文件的修改方法
3. 实验中给出了 windows 上首选 dns 服务器的配置位置
4. 学习使用了命令行中的 ping 与 nslookup 两个命令，配合 wireshark 工具对网络中传输的数据进行了初步的抓包尝试，并且了解到了一些网络信息的过滤方法。
5. 实验过程中进一步加强了自己对环境的调试与配置能力,通过观察抓包得到记录的详细信息,对 dns, icmp, http 的报文结构有了直观的认识。强化了自己的对网络连接中三次握手与四次挥手的直观认识。
6. 计算机网络的知识涉及面很广，需要多下手才能有最鲜明的认识

指导
老师
评
议

成绩评定： 指导教师签名：