



# 计算机网络 课程实验报告

实验名称	利用 Wireshark 进行协议分析			
姓名	刘璟烁	院系	计算机科学与技术学院	
班级		学号		
任课教师		指导教师		
实验地点		实验时间		
实验课表现	出勤、表现得分(10)	实验报告得分(40)		实验总分
	操作结果得分(50)			
教师评语				

**实验目的:**

- 1.熟悉并掌握 Wireshark 的基本操作;
- 2.了解网络协议实体间进行交互以及报文交换的情况

**实验内容:**

- 1) 学习 Wireshark 的使用
- 2) 利用 Wireshark 分析 HTTP 协议
- 3) 利用 Wireshark 分析 TCP 协议
- 4) 利用 Wireshark 分析 IP 协议
- 5) 利用 Wireshark 分析 Ethernet 数据帧

**选做:**

- a) 利用 Wireshark 分析 DNS 协议
- b) 利用 Wireshark 分析 UDP 协议
- c) 利用 Wireshark 分析 ARP 协议

**实验过程:****1.实验环境**

- a. Mac OS BigSur;
- b. 与因特网连接的计算机网络系统;
- c. Wireshark。

**2.实验工具掌握****(1).分组嗅探器**

在实验指导书中给出了对分组嗅探工具较为详细的介绍，首先我们想要深入理解网络协议，就需要仔细观察协议实体之间交换的报文序列。为探究协议操作细节，可使协议实体执行某些动作，观察这些动作及其影响。这些任务可以在仿真环境下或在如因特网这样的真实网络环境中完成；

观察正在运行协议实体间交换报文的基本工具被称为分组嗅探器。一个分组嗅探器俘获（嗅探）计算机发送和接收的报文。一般情况下，分组嗅探器将存储和显示出被俘获报文的各协议头部字段的内容。

分组嗅探器是附加计算机普通软件上的，主要有两部分组成。分组俘获库接收计算机发送和接收的每一个链路层帧的拷贝。高层协议（如：HTTP、FTP、TCP、UDP、DNS、IP 等）交换的报文都被封装在链路层帧中，并沿着物理媒体（如以太网的电缆）传输。分组嗅探器的第二个组成部分是分析器。分析器用来显示协议报文所有字段的内容。为此，分析器必须能够理解协议所交换的所有报文的结构。分组分析器理解以太网帧格式，能够识别包含在帧中的 IP 数据报。分组分析器也要理解 IP 数据报的格式，并能从 IP 数据报中提取出 TCP 报文段。然后，它需要理解 TCP 报文段，并能够从中提取出 HTTP 消息。最后，它需要理解 HTTP 消息。

如图为实验指导书中给出分组嗅探工具的结构示意图：

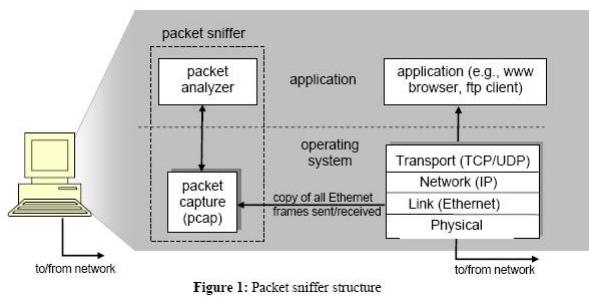


图1.分组嗅探工具结构图

## (2).Wireshark分组分析器

### a.Wireshark分组分析器介绍:

Wireshark 是一种可以运行在 MacOS、Windows、Linux 等操作系统上的分组分析器。网络封包分析器的功能是撷取网络封包，并尽可能显示出最为详细的网络封包资料。它可以直接与网卡进行数据报文交换。Wireshark 的Mac版用户界面如图所示：

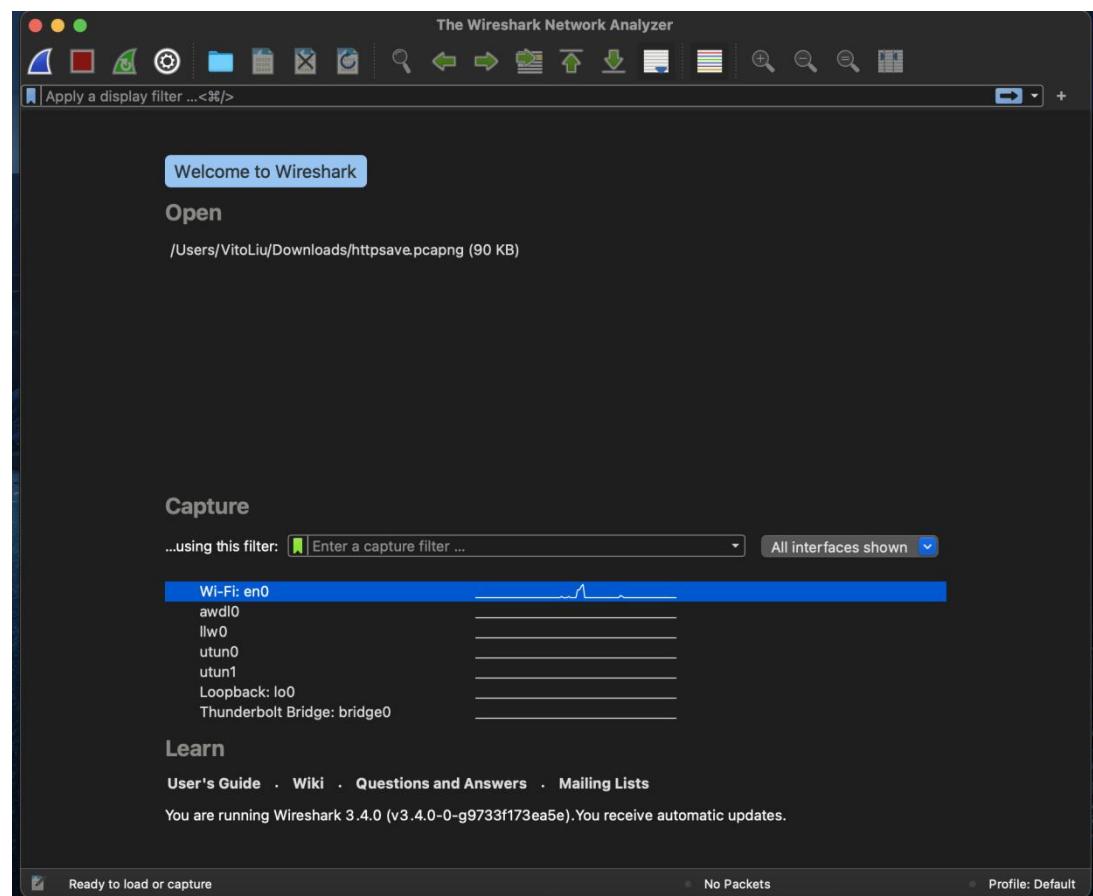
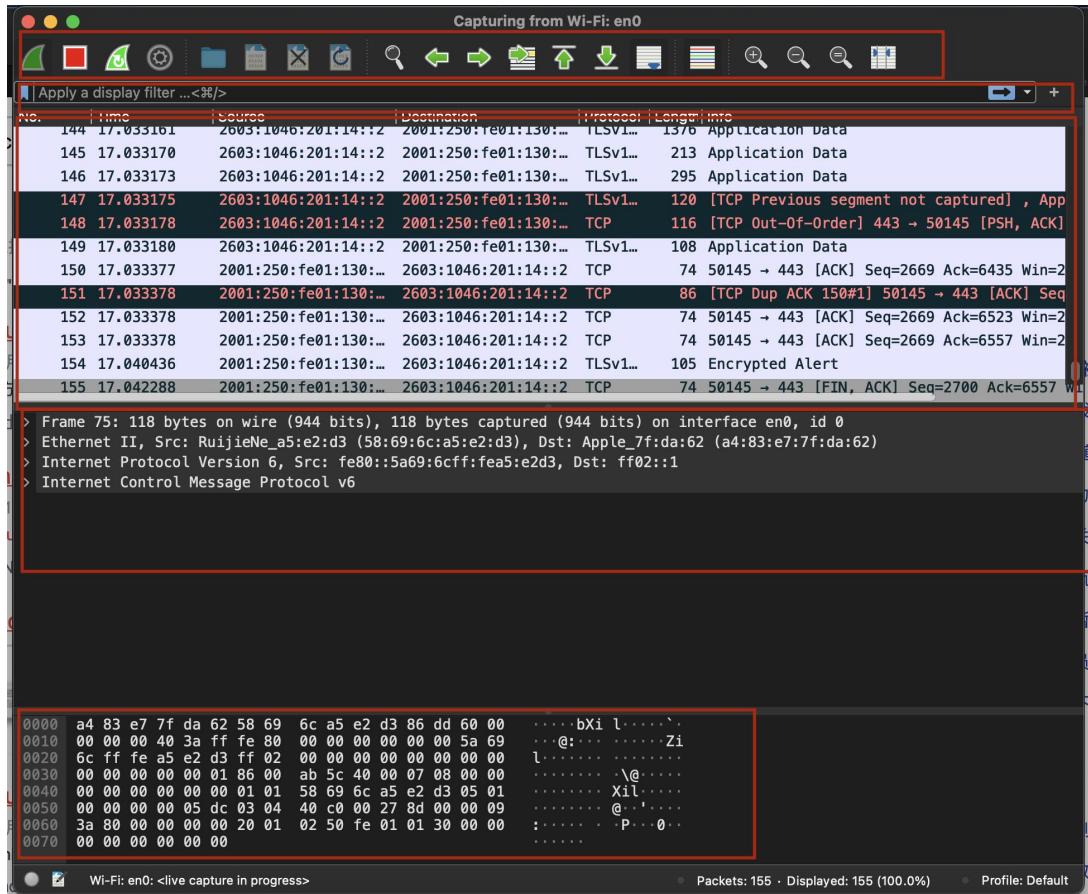


图2.Wireshark窗口

### b.Wireshark的使用

启动Safari浏览器，随后启动 Wireshark。看到如上图所示窗口，选择Capture中的Wi-Fi: end0作为接口，点击start开始嗅探，(这里需要在terminal中写入sudo chmod 777 /dev/bpf\* 指令为其获得权限) 获得权限后开始嗅探网络分组，所有由选定网卡发送和接收的分组都将被俘获，并显示在Wireshark软件中。

开始抓包后，我们随意输入一个网址并打开，可以看到应用界面如下：



从上到下被红色框体抗拒的区域分别是命令菜单、筛选俘获分组、俘获分组列表、分组头部明细、分组内容窗口。

### 3.具体实验任务执行

#### 3.1.HTTP分析

##### 3.1.1.HTTP的GET/response交互

我们首先打开Safari浏览器，然后启动 Wireshark 分组嗅探器。在窗口的显示过滤说明处输入“http”，分组列表子窗口中将只显示所俘获到的HTTP 报文。

尝试打开实验指导书中给定网址<http://hitgs.hit.edu.cn/news>，但访问地址已经无效，如下图：

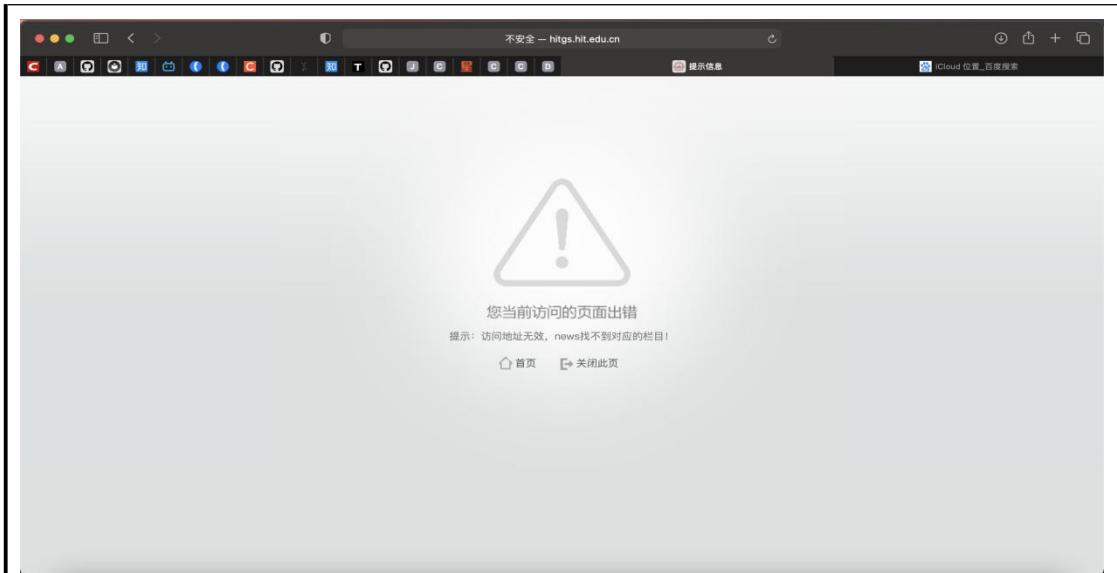


图3.给定网址无效

为此我们重新启动Wireshark，并更换一个网址，比如`http://today.hit.edu.cn`进行分析，可以看到成功打开该网站，如下图：



图4. 打开今日哈工大网站

打开后停止抓包，得到分组列表如下：

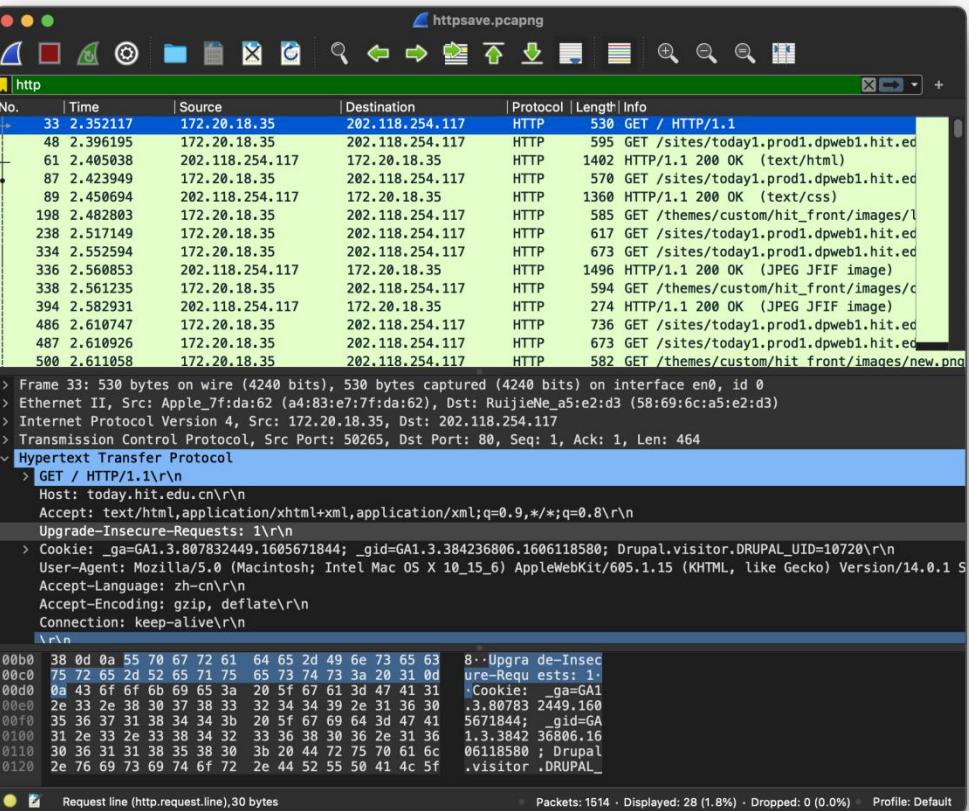
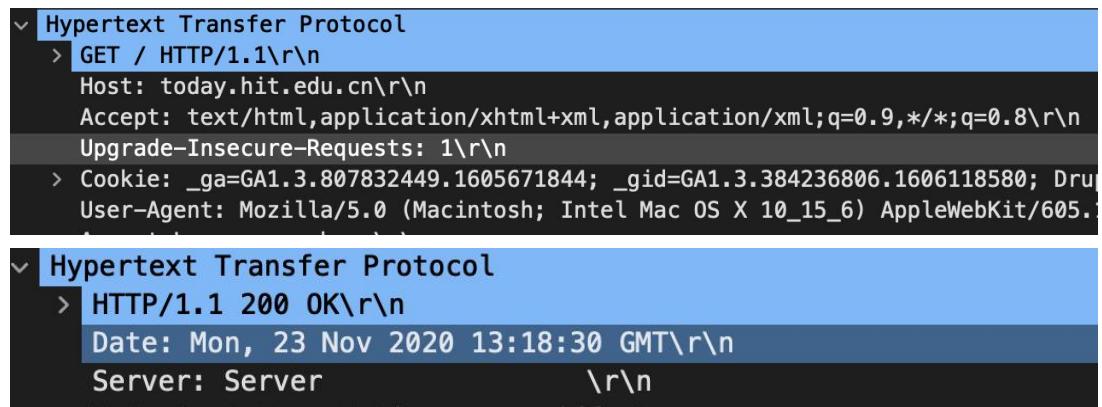


图5.抓包完成后获得的分组列表

我们对该组报文进行分析，首先我们分析最开始的http请求和响应报文可知Safari浏览器和请求的服务器运行的http协议版本号均为HTTP/1.1：



并且由请求报文的Accept-Language部分可知，浏览器请求接收的是中文语言对象：

```

    ▾ Hypertext Transfer Protocol
      > GET / HTTP/1.1\r\n
        Host: today.hit.edu.cn\r\n
        Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
        Upgrade-Insecure-Requests: 1\r\n
      > Cookie: _ga=GA1.3.807832449.1605671844; _gid=GA1.3.384236806.1606118580;
        User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/6
        Accept-Language: zh-cn\r\n
        Accept-Encoding: gzip, deflate\r\n
        Connection: keep-alive\r\n
  
```

分析嗅探到的报文列表可知，本机的ip地址为172.20.18.35，服务器端ip地址为202.118.254.117，服务器响应的状态码为200，即成功找到对象。

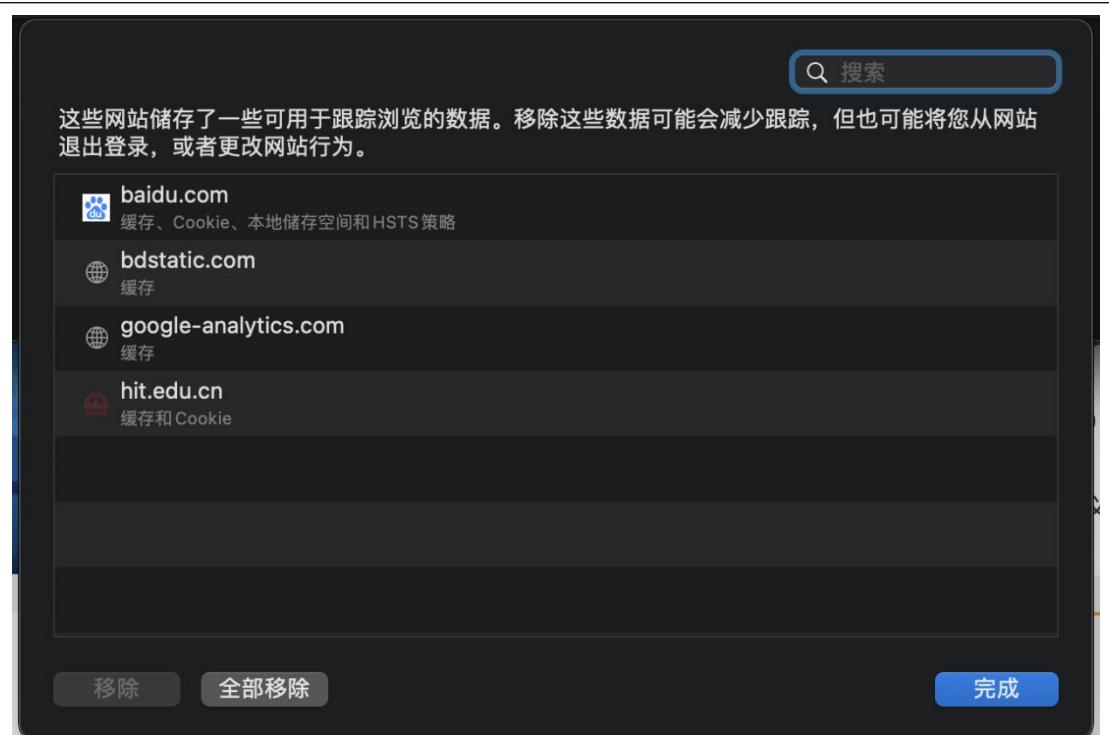
33	2.352117	172.20.18.35	202.118.254.117	HTTP	530	GET / HTTP/1.1
48	2.396195	172.20.18.35	202.118.254.117	HTTP	595	GET /sites/today1.prod1.dpweb1.hit.ed
61	2.405038	202.118.254.117	172.20.18.35	HTTP	1402	HTTP/1.1 200 OK (text/html)
87	2.423949	172.20.18.35	202.118.254.117	HTTP	570	GET /sites/today1.prod1.dpweb1.hit.ed
89	2.450694	202.118.254.117	172.20.18.35	HTTP	1360	HTTP/1.1 200 OK (text/css)

### 3.1.2.HTTP的条件GET/response交互

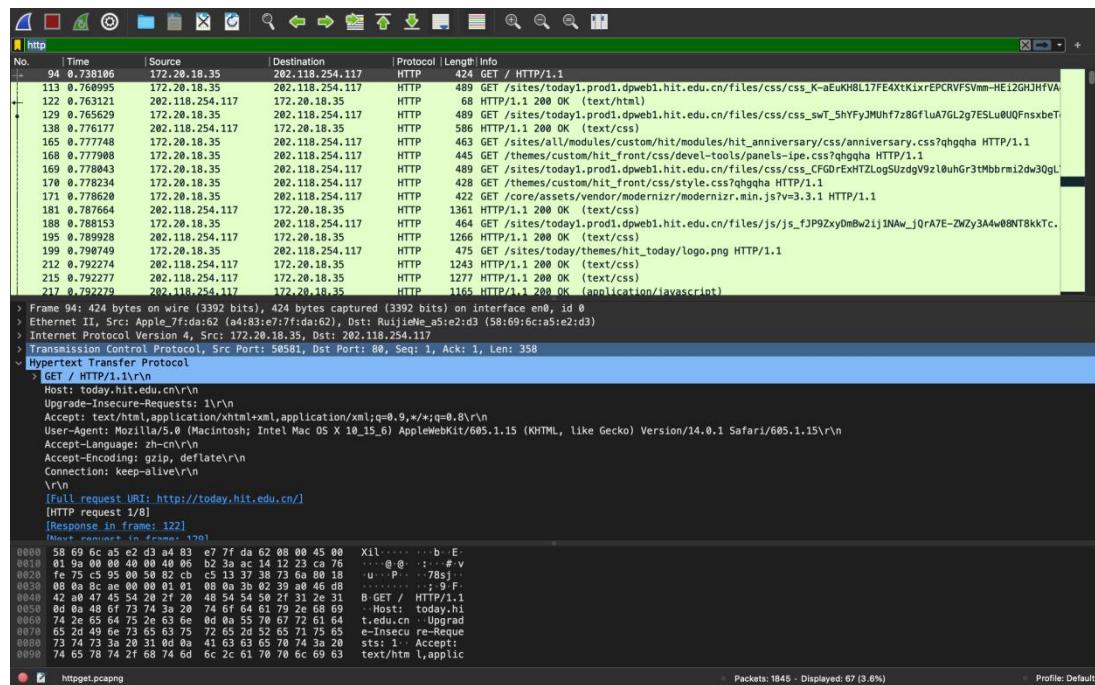
打开浏览器的偏好设置，点击隐私：



点击管理网站数据，随后全部移除所有网站的缓存文件，如下图：

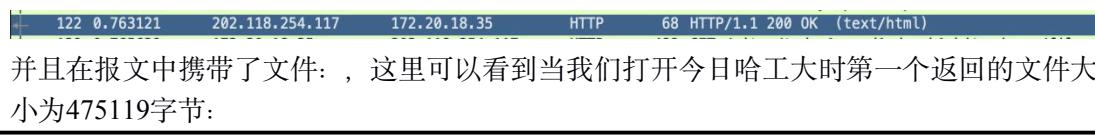


随后启动Wireshark分组嗅探器进行分组嗅探，我们仍然连接今日哈工大，打开网页后停止分组嗅探，得到http分组列表如下：



观察第一条请求报文，可以看到在删除缓存后报文中并没有 IF-MODIFIED-SINCE 语句。

通过观察被请求服务器的响应报文，我们可以看到服务器返回了信息（状态码为200）：



```

[Next response in frame: 158]
[Request URI: http://today.hit.edu.cn/core/assets/vendor/modernizr/modernizr.min.js?v=3.3.1]
Content-encoded entity body (gzip): 25534 bytes -> 475119 bytes
File Data: 475119 bytes
Line-based text data: text/html (13201 lines)

```

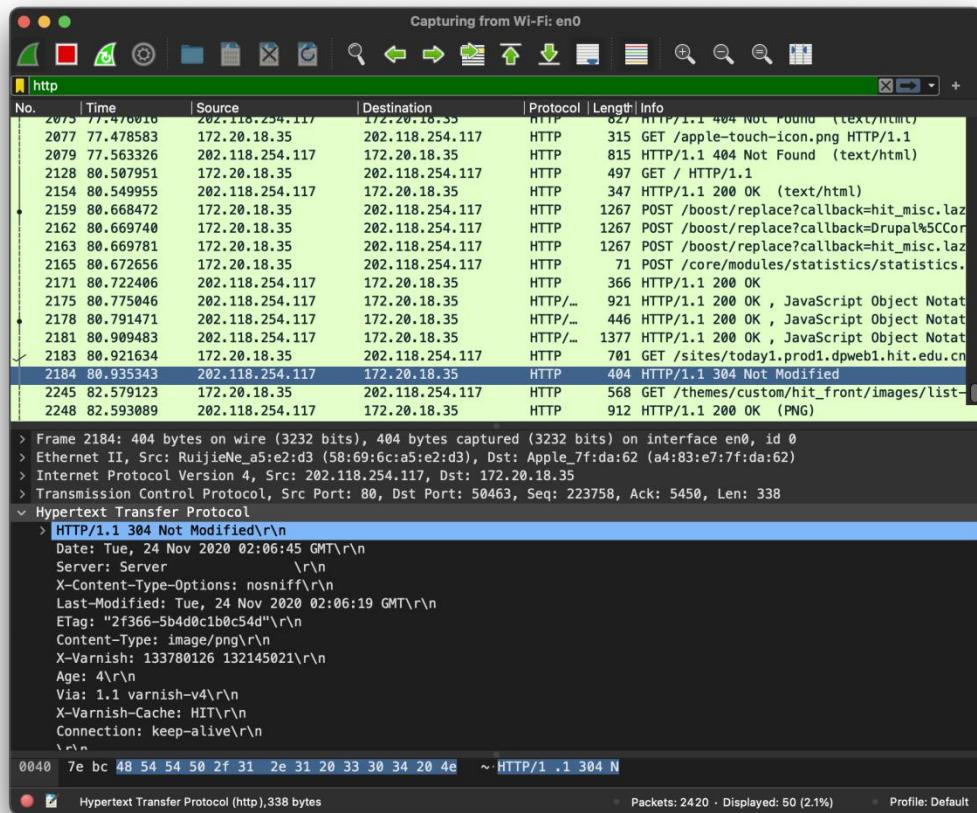
在较晚发送的GET请求信息中含有“If-Modified-Since”行，并且在该行中附有上一次从服务器接受相应内容所处的时间，如下图：

```

Hypertext Transfer Protocol
> GET /sites/today1.prod1.dpweb1.hit.edu.cn/files/styles/355x220/public/inline-images/%5Bdate%3Acustom%3AY%5D/%5Bdate%3Acustom%3Am%5D/%5B
Host: today.hit.edu.cn\r\n
> Cookie: _ga=GA1.3.1634671377.1606139205; _gid=GA1.3.622731749.1606139205\r\n
Connection: keep-alive\r\n
If-None-Match: "18520-5b4c4da205db6"\r\n
Accept: image/webp,image/png,image/svg+xml,image/*;q=0.8,video/*;q=0.8,*/*;q=0.5\r\n
If-Modified-Since: Mon, 23 Nov 2020 11:54:09 GMT\r\n
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0.1 Safari/605.1.15\r\n
Accept-Language: zh-cn\r\n
Referer: http://today.hit.edu.cn/\r\n
Accept-Encoding: gzip, deflate\r\n
\r\n

```

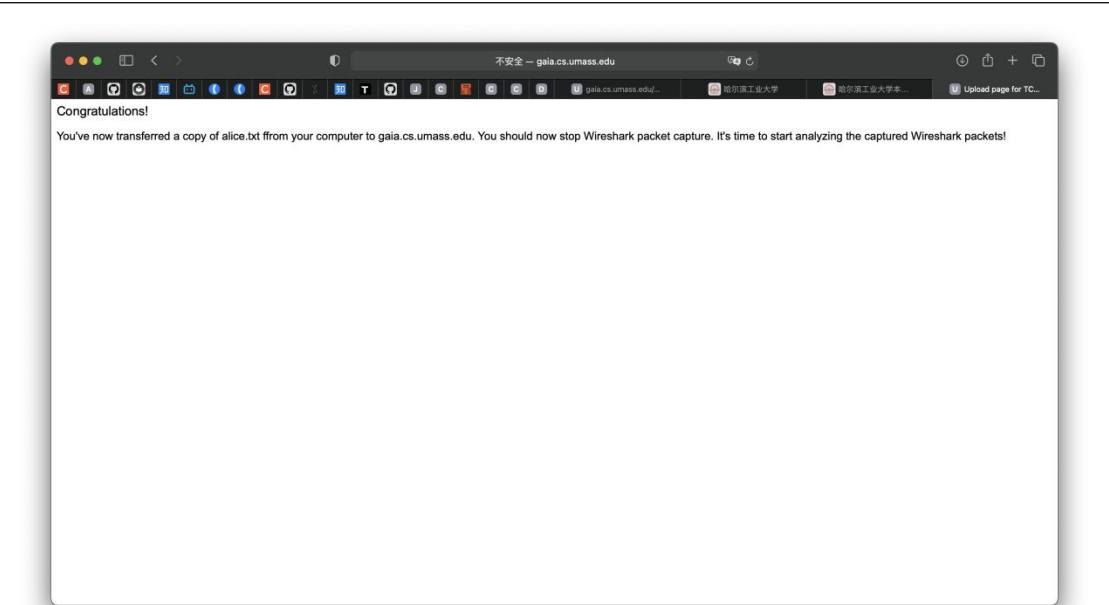
随后可以接收到含有304状态码的报文，表示截止上述事件服务器中对应内容仍未被修改，如下图：



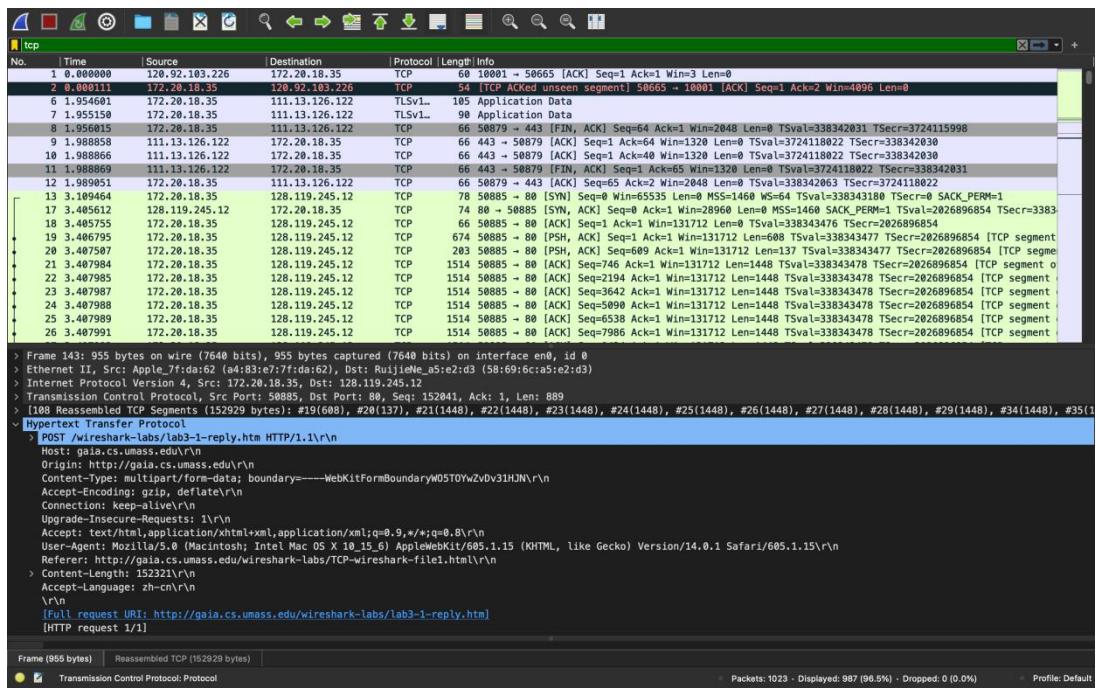
### 3.2.TCP报文嗅探

#### 3.2.1.俘获大量的由本地主机到远程服务器的TCP分组

- 首先下载得到文件alice.txt，并在上传网页中选择文件路径，打开Wireshark嗅探工具后，对文件进行上传：



可以看到上传成功，收到回复，此时也嗅探得到分组，如下图：



问题：通信双方的源和目的ip、端口号？

找到三次握手的对应报文，可以看到源主机和目的主机的ip地址分别为172.20.18.35和128.119.245.12。

客户源端口号和目的端口号分别为50885和80：

```
Transmission Control Protocol, Src Port: 50885, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
Source Port: 50885
Destination Port: 80
```

问题：客户服务器之间用于初始化 TCP 连接的 TCP SYN 报文段的序号 (sequence number) 是多少？在该报文段中，是用什么来标示该报文段是 SYN 报文段的？

可以看到其序号（相对）为0，真实为4162697575，通过在Flags处标志0x002(对应SYN)来标示该报文段为SYN报文段，如下图：

```

    > Transmission Control Protocol, Src Port: 50885, Dst Port: 80, Seq: 0, Len: 0
      Source Port: 50885
      Destination Port: 80
      [Stream index: 2]
      [TCP Segment Len: 0]
      Sequence Number: 0      (relative sequence number)
      Sequence Number (raw): 4162697575
      [Next Sequence Number: 1      (relative sequence number)]
      Acknowledgment Number: 0
      Acknowledgment number (raw): 0
      1011 .... = Header Length: 44 bytes (11)
    > Flags: 0x002 (SYN)
      Window: 65535
  
```

真实序号的十六进制形式：0xf81db967，如下图

```

      Sequence Number (raw): 4162697575
      [Next Sequence Number: 1      (relative sequence number)]
      Acknowledgment Number: 0
      Acknowledgment number (raw): 0
      1011 .... = Header Length: 44 bytes (11)
    > Flags: 0x002 (SYN)

  0000  58 69 6c a5 e2 d3 a4 83  e7 7f da 62 08 00 45 00  Xil.....b..E.
  0010  00 40 00 00 40 00 40 06  06 fd ac 14 12 23 80 77  @...@...@....#.w
  0020  f5 0c c6 c5 00 50 f8 1d  b9 67 00 00 00 00 b0 02  ....P...g.....
  
```

问题：服务器向客户端发送的 SYNACK 报文段序号是多少？该报文段中，Acknowledgement字段的值是多少？Gaia.cs.umass.edu 服务器是如何决定此值的？在该报文段中，是用什么来标示该报文段是SYNACK 报文段的？

可以看到该报文段相对序号为0，真实为0xfbcb3a64f;ACK号相对值为1，真实为0xf81db968；通过对SYN报文段的序号+1得到该值。另外，通过在Flags处标志0x012(对应SYN, ACK)来标示该报文段为SYN报文段：

```

    > Transmission Control Protocol, Src Port: 80, Dst Port: 50885, Seq: 0, Ack: 1, Len: 0
      Source Port: 80
      Destination Port: 50885
      [Stream index: 2]
      [TCP Segment Len: 0]
      Sequence Number: 0      (relative sequence number)
      Sequence Number (raw): 4223903311
      [Next Sequence Number: 1      (relative sequence number)]
      Acknowledgment Number: 1      (relative ack number)
      Acknowledgment number (raw): 4162697576
      1010 .... = Header Length: 40 bytes (10)
    > Flags: 0x012 (SYN, ACK)

  0000  a4 83 e7 7f da 62 58 69  6c a5 e2 d3 08 00 45 74  ....bXi l.....Et
  0010  00 3c 00 00 40 00 25 06  21 8d 80 77 f5 0c ac 14  .<..@% !..w...
  0020  12 23 00 50 c6 c5 fb c3  a6 4f f8 1d b9 68 a0 12  #.P...0..h..
  0030  71 20 48 87 00 00 02 04  05 b4 04 02 08 0a 78 cf  q H.....x.
  0040  fd d6 14 2a b5 0c 01 03  03 07  .....*.....
  
```

问题：寻找三次握手？

如图，列表中该组报文为三次握手的过程：

13 3.189464	172.28.18.35	128.119.245.12	TCP	78 50885 → 80 [SYN] Seq=0 Win=5535 Len=1460 MSS=1460 WS=64 TSval=338343180 TSecr=0 SACK_PERM=1
17 3.405612	128.119.245.12	172.28.18.35	TCP	74 80 → 50885 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=2026896854 TSecr=338343476
18 3.405755	172.28.18.35	128.119.245.12	TCP	66 50885 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=338343476 TSecr=2026896854

分别具有[SYN]、[SYN, ACK]、[ACK]标志，并且分组确认号与前一组的序号有对应关系。

问题：包含POST命令的报文段序号？

如图，我们在载荷部分找到含有POST的内容，对应序列号（相对）为1。

▼ Transmission Control Protocol, Src Port: 50885, Dst Port: 80, Seq: 1, Ack: 1, Len: 608
Source Port: 50885
Destination Port: 80
[Stream index: 2]
[TCP Segment Len: 608]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 4162697576
[Next Sequence Number: 609 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 4223903312
1000 .... = Header Length: 32 bytes (8)
> Flags: 0x018 (PSH, ACK)
0020 f5 0c c6 c5 00 50 f8 1d b9 68 fb c3 a6 50 80 18 .....P...h...P..
0030 08 0a 6e 6b 00 00 01 01 08 0a 14 2a b6 35 78 cf nk.....*5x.
0040 fd d6 50 4f 53 54 20 2f 77 69 72 65 73 68 61 72 ..POST /wirshar
0050 6b 2d 6c 61 62 73 2f 6c 61 62 33 2d 31 2d 72 65 k-labs/lab3-1-re
0060 70 6c 79 2e 68 74 6d 20 48 54 54 50 2f 31 2e 31 ply.htm HTTP/1.1
0070 0d 0a 48 6f 73 74 3a 20 67 61 69 61 2e 63 73 2e .Host: gaia.cs.
0080 75 6d 61 73 73 2e 65 64 75 0d 0a 4f 72 69 67 69 umass.ed u..Orgi
0090 6e 3a 20 68 74 74 70 3a 2f 67 61 69 61 2e 63 n: http://gaia.c
00a0 73 2e 75 6d 61 73 73 2e 65 64 75 0d 0a 43 6f 6e s.umass.edu..Con

问题：如果将包含 HTTP POST 命令的 TCP 报文段看作是 TCP 连接上的第一个报文段，那么该 TCP 连接上的第六个报文段的序号是多少？是何时发送的？该报文段所对应的 ACK 是何时接收的？

可以看到，第六个报文段序号为5090，该报文段于建立连接之后发送的，该报文段对应的 ACK 确认序号（为1）对应的报文是在第三次握手的时候接收的，如下图：

19 3.406795	172.20.18.35	128.119.245.12	TCP	674 50885 → 80 [PSH, ACK] Seq=1 Ack=1 Win=131712 Len=608 TSval=338343477 TSecr=2026896854
20 3.407507	172.20.18.35	128.119.245.12	TCP	203 50885 → 80 [PSH, ACK] Seq=609 Ack=1 Win=131712 Len=137 TSval=338343477 TSecr=2026896854
21 3.407984	172.20.18.35	128.119.245.12	TCP	1514 50885 → 80 [ACK] Seq=746 Ack=1 Win=131712 Len=1448 TSval=338343478 TSecr=2026896854
22 3.407985	172.20.18.35	128.119.245.12	TCP	1514 50885 → 80 [ACK] Seq=2194 Ack=1 Win=131712 Len=1448 TSval=338343478 TSecr=2026896854
23 3.407987	172.20.18.35	128.119.245.12	TCP	1514 50885 → 80 [ACK] Seq=3642 Ack=1 Win=131712 Len=1448 TSval=338343478 TSecr=2026896854
24 3.407988	172.20.18.35	128.119.245.12	TCP	1514 50885 → 80 [ACK] Seq=5090 Ack=1 Win=131712 Len=1448 TSval=338343478 TSecr=2026896854

问题：前六个报文段长度？

根据上图，我们可以得到TCP段长度分别为608、137、1448、1448、1448、1448。

问题：在整个跟踪过程中，接收端公示的最小的可用缓存空间是多少？限制发送端的传输以后，接收端的缓存是否仍然不够用？

可以看到，最小窗口大小为30208字节，且在随后若干报文中窗口大小在增加，并且整个过程未出现减小状况，接收端缓存够用。

30 3.771861	128.119.245.12	172.20.18.35	TCP	66 80 → 50885 [ACK] Seq=1 Ack=609 Win=30208 Len=0 TSval=2026897144 TSecr=338343477
31 3.771870	128.119.245.12	172.20.18.35	TCP	66 80 → 50885 [ACK] Seq=1 Ack=746 Win=31488 Len=0 TSval=2026897145 TSecr=338343477
32 3.771873	128.119.245.12	172.20.18.35	TCP	66 80 → 50885 [ACK] Seq=1 Ack=12330 Win=54656 Len=0 TSval=2026897146 TSecr=338343478
33 3.771875	128.119.245.12	172.20.18.35	TCP	66 80 → 50885 [ACK] Seq=1 Ack=13778 Win=57472 Len=0 TSval=2026897146 TSecr=338343478

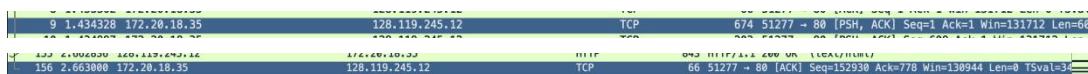
问题：在跟踪文件中是否有重传的报文段？进行判断的依据是什么？

没有重传报文段，在追踪过程中客户端发送的seq一直在增加，因此并未重发，如下图：

序号	时间	源IP	目的IP	协议	序列号	确认号	报文内容	时间戳
19	3.406795	172.20.18.35	128.119.245.12	TCP	674	50885 - 80	[PSH, ACK] Seq=1 Ack=1 Win=131712 Len=608 TStamp=338343477 TSect=2026896854 [TCP segment	
20	3.407507	172.20.18.35	128.119.245.12	TCP	283	50885 - 80	[PSH, ACK] Seq=649 Ack=1 Win=131712 Len=137 TStamp=338343477 TSect=2026896854 [TCP segment	
21	3.407984	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=746 Ack=1 Win=131712 Len=1448 TStamp=338343478 TSect=2026896854 [TCP segment o	
22	3.407985	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=2194 Ack=1 Win=131712 Len=1448 TStamp=338343478 TSect=2026896854 [TCP segment	
23	3.407987	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=3642 Ack=1 Win=131712 Len=1448 TStamp=338343478 TSect=2026896854 [TCP segment	
24	3.407988	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=5098 Ack=1 Win=131712 Len=1448 TStamp=338343478 TSect=2026896854 [TCP segment	
25	3.407989	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=6538 Ack=1 Win=131712 Len=1448 TStamp=338343478 TSect=2026896854 [TCP segment	
26	3.407991	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=7986 Ack=1 Win=131712 Len=1448 TStamp=338343478 TSect=2026896854 [TCP segment	
27	3.407992	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=9434 Ack=1 Win=131712 Len=1448 TStamp=338343478 TSect=2026896854 [TCP segment	
28	3.407993	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=10882 Ack=1 Win=131712 Len=1448 TStamp=338343478 TSect=2026896854 [TCP segment	
29	3.407994	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=12338 Ack=1 Win=131712 Len=1448 TStamp=338343478 TSect=2026896854 [TCP segment	
30	3.771861	128.119.245.12	172.20.18.35	TCP	66	80 - 50885	[ACK] Seq=9886 Ack=1 Win=131712 Len=1448 TStamp=338343478 TSect=2026896854 [TCP segment	
31	3.771870	128.119.245.12	172.20.18.35	TCP	66	80 - 50885	[ACK] Seq=10226 Ack=1 Win=131712 Len=1448 TStamp=338343477 TSect=2026897146	
32	3.771873	128.119.245.12	172.20.18.35	TCP	66	80 - 50885	[ACK] Seq=10674 Ack=1 Win=131712 Len=1448 TStamp=338343478 TSect=2026897146	
33	3.771875	128.119.245.12	172.20.18.35	TCP	66	80 - 50885	[ACK] Seq=11122 Ack=1 Win=131712 Len=1448 TStamp=338343478 TSect=2026897146	
34	3.772055	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=13778 Ack=1 Win=131712 Len=1448 TStamp=338343841 TSect=2026897144 [TCP segment	
35	3.772057	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=15226 Ack=1 Win=131712 Len=1448 TStamp=338343841 TSect=2026897146 [TCP segment	
36	3.772058	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=16674 Ack=1 Win=131712 Len=1448 TStamp=338343841 TSect=2026897146 [TCP segment	
37	3.772060	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=18122 Ack=1 Win=131712 Len=1448 TStamp=338343841 TSect=2026897146 [TCP segment	
38	3.772061	172.20.18.35	128.119.245.12	TCP	1514	50885 - 80	[ACK] Seq=19578 Ack=1 Win=131712 Len=1448 TStamp=338343841 TSect=2026897146 [TCP segment	

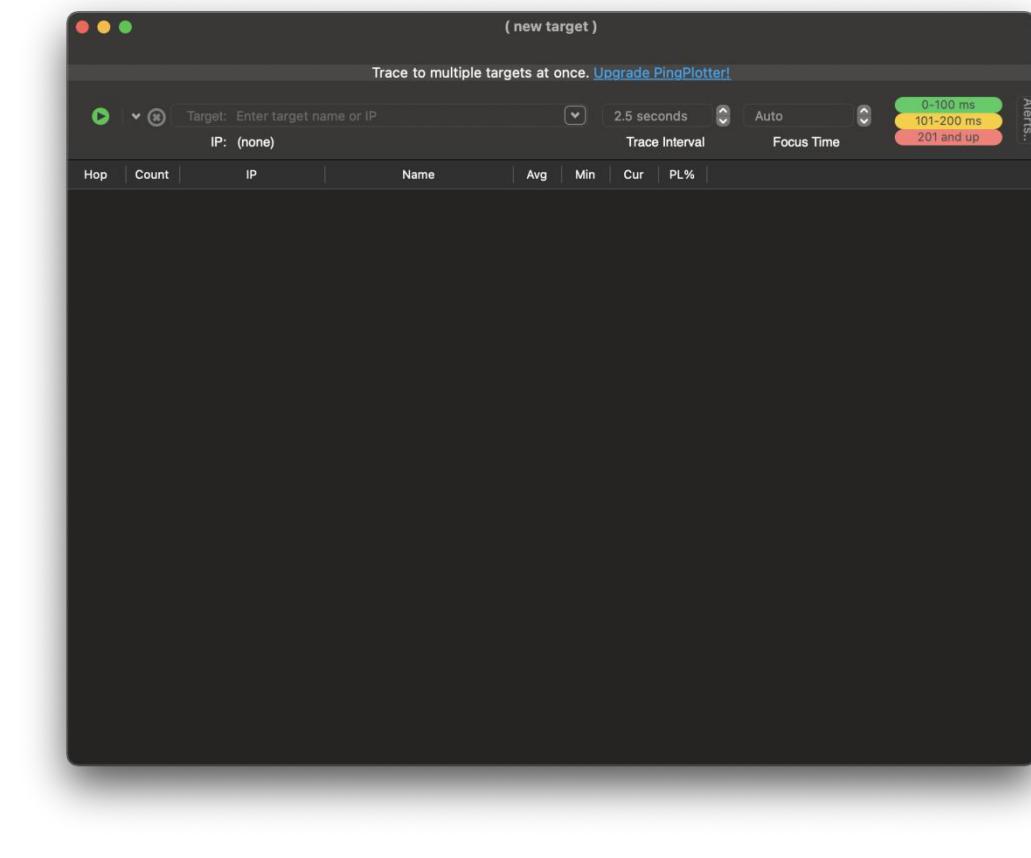
问题：TCP 连接的 throughput (bytes transferred per unit time)是多少？请写出你的计算过程。

由下图可以看到开始时间和结束时间分别为1.434328和2.663000，故throughput为 $152930 / (1.434328 - 2.663000) = 124467.71799146$ (bytes transferred per unit time)

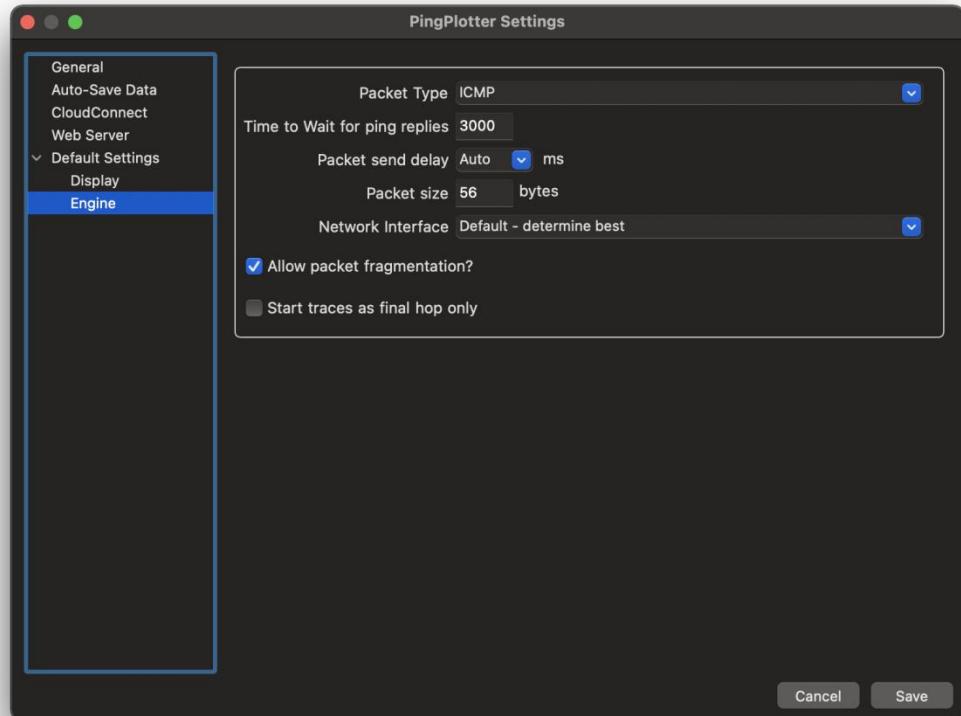


### 3.3. IP报文嗅探

首先下载PingPlotter工具，打开窗口界面如下：



在设置中调整包大小为56byte，如下图：

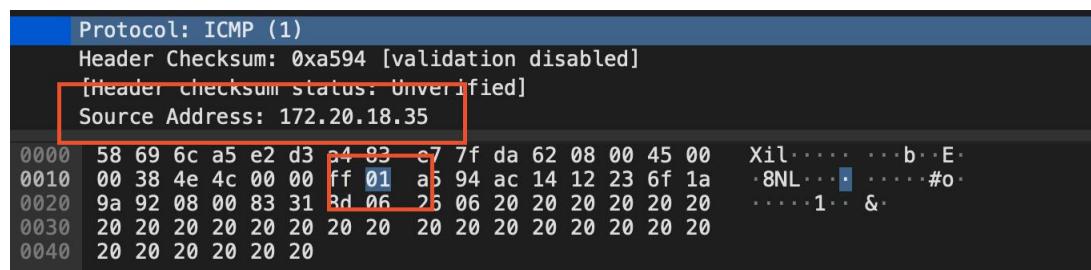


开启wireshark的嗅探功能，在pingplotter中输入www.bilibili.com进行traceroute（注意在新版本的pingplotter中没有设置追踪次数的选项，可以在次数为3时停止追踪）：

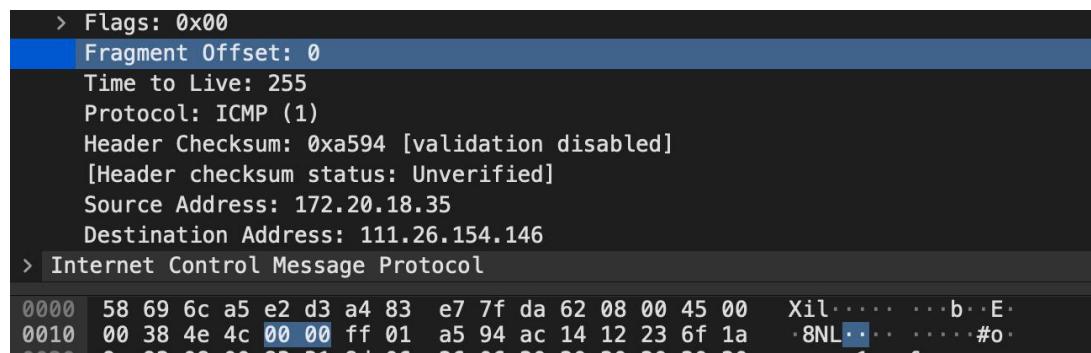


下面对Wireshark嗅探到的分组进行分析：

首先，主机 IP 为 172.20.18.35，IP 数据包头中上层协议字段的值为 1 表示 ICMP 协议，IP 头字节数为 20B，IP 数据包的净载为 36B，确定方式为：IP 分组总长度-IP 首部长度；



该包并未分片，因为标志位全0，下一个发送的包的偏移为0，因此其未分片：



主机发出的 ICMP 报文中中IP数据报一些字段总在发生改变：标识 ID、TTL、首部校验和以及data部分，其他字段不会发生变化。这是因为每个数据报的标识ID唯一，每个数据报有所区别，随之首部校验和也不断改变，另外traceroute会改变每个数据报的TTL，从而达到追踪的目的。而且数据域中封装有 ICMP 的报文，因为 ICMP 的头部信息也在变化，所以 IP 数据报的数据域也随之变化；

部分截图如下，从报文细节部分可以看出TTL、校验和等均发生了变化：

```

Frame 17: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface en0, id 0
Ethernet II, Src: Apple_7:7:da:62 (a4:83:e7:7:7:da:62), Dst: RuijineMe_a5:e2:d3 (58:69:6c:a5:e2:d3)
Internet Protocol Version 4, Src: 172.20.18.35, Dst: 111.26.154.146
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 56
Identification: 0xbccf (48335)
> Flags: 0x00
Fragment Offset: 0
> Time to Live: 1
Protocol: ICMP (1)
Header Checksum: 0x3512 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.20.18.35
ICMP 70 Echo (ping) request id=0x8d06, seq=9999/1575, ttl=1 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=10246/1576, ttl=2 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=10502/1577, ttl=3 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=10758/1578, ttl=4 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=11014/1579, ttl=5 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=11270/1580, ttl=6 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=11526/1581, ttl=7 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=11782/1582, ttl=8 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=12038/1583, ttl=9 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=12294/1584, ttl=10 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=12558/1585, ttl=11 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=12866/1586, ttl=12 (reply)
ICMP 70 Echo (ping) request id=0x8d06, seq=13062/1587, ttl=255 (reply in)

Frame 20: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface en0, id 0
Ethernet II, Src: Apple_7:7:da:62 (a4:83:e7:7:7:da:62), Dst: RuijineMe_a5:e2:d3 (58:69:6c:a5:e2:d3)
Internet Protocol Version 4, Src: 172.20.18.35, Dst: 111.26.154.146
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 56
Identification: 0xb109 (45321)
> Flags: 0x00
Fragment Offset: 0
> Time to Live: 2
Protocol: ICMP (1)
Header Checksum: 0x3fd8 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.20.18.35
ICMP 70 Echo (ping) request id=0x8d06, seq=9734/1574, ttl=255 (reply in)
ICMP 70 Echo (ping) request id=0x8d06, seq=9999/1575, ttl=1 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=10246/1576, ttl=2 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=10502/1577, ttl=3 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=10758/1578, ttl=4 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=11014/1579, ttl=5 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=11270/1580, ttl=6 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=11526/1581, ttl=7 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=11782/1582, ttl=8 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=12038/1583, ttl=9 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=12294/1584, ttl=10 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=12558/1585, ttl=11 (no resp)
ICMP 70 Echo (ping) request id=0x8d06, seq=12866/1586, ttl=12 (reply)
ICMP 70 Echo (ping) request id=0x8d06, seq=13062/1587, ttl=255 (reply in)

0000 58 69 6c a5 e2 d3 a4 83 e7 7f da 62 08 05 45 00 Xl..... .b .E.
0010 00 38 b1 09 00 00 02 01 3f d8 ac 14 12 23 6f 1a :8...: .?...#o...
0020 9a 92 08 00 81 31 8d 06 28 06 20 20 20 20 20 ...:1.. (.
0030 28 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0040 28 20 20 20 20 20 20 20 20 20 20 20 20 20 20

```

identification字段值形式如下，可以看到其为十六进制数，并且线性递增，如下图

```

Internet Protocol Version 4, Src: 172.20.18.35, Dst: 111.26.154.146
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 56
Identification: 0x0ac0 (2752)
> Flags: 0x00

```

当改为 2000 字节后，主机发送的第一个 ICMP 请求消息被分解为2个分组，第一个分组如下，它的长度为1500个字节，数据占1480个字节，标识位M被置为1 (0x20) 说明它不是最后一个分片：

```

▼ Internet Protocol Version 4, Src: 172.20.18.35, Dst: 111.26.154.146
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0x52cf (21199)
  > Flags: 0x20, More fragments
    Fragment Offset: 0
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x7b6d [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.18.35
    Destination Address: 111.26.154.146
    [Reassembled IPv4 in frame: 169]

```

第二个分组则有1480个字节的段偏移，总长520字节：

```

▼ Internet Protocol Version 4, Src: 172.20.18.35, Dst: 111.26.154.146
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 520
    Identification: 0x52cf (21199)
  > Flags: 0x00
    Fragment Offset: 1480
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x9e88 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.18.35
    Destination Address: 111.26.154.146
  ▼ [2 IPv4 Fragments (1980 bytes): #168(1480), #169(500)]

```

当包大小设为3500字节时，分组被分为3片，每片最大1480字节，三片之间片偏移、标志位有所不同。可以看到最后一片的段偏移为2960，第二片为1480，并且第三片标志位为0x01，而前两片标志位为0x20。

```

▼ Internet Protocol Version 4, Src: 172.20.18.35, Dst: 111.26.154.146
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 540
    Identification: 0xb70c (46860)
  > Flags: 0x01
    Fragment Offset: 2960
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x397e [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.18.35
    Destination Address: 111.26.154.146
  ▼ [3 IPv4 Fragments (3480 bytes): #300(1480), #301(1480), #302(520)]

```

### 3.4. 抓取 ARP 数据包

在 terminal 中输入 arp -a 指令，查看 arp 缓冲区的所有条目，得到如下结果，其中第一列括号中的是 ARP 协议的缓存的 IP 地址，at 后是对应的 MAC 地址，on 后对应接口，随后是类型，如下图：

```
(base) AppledeMacBook-Pro:~ VitoLiu$ arp -a
? (172.20.0.1) at 58:69:6c:a5:e2:d3 on en0 ifscope [ethernet]
? (172.20.18.4) at 58:69:6c:a5:e2:d3 on en0 ifscope [ethernet]
? (172.20.18.12) at 58:69:6c:a5:e2:d3 on en0 ifscope [ethernet]
? (224.0.0.251) at 1:0:5e:0:0:fb on en0 ifscope permanent [ethernet]
```

随后，输入指令（需要管理员权限） sudo arp -d -a：

```
(base) AppledeMacBook-Pro:~ VitoLiu$ sudo arp -d -a
172.20.0.1 (172.20.0.1) deleted
172.20.18.4 (172.20.18.4) deleted
172.20.18.12 (172.20.18.12) deleted
172.20.18.213 (172.20.18.213) deleted
224.0.0.251 (224.0.0.251) deleted
(base) AppledeMacBook-Pro:~ VitoLiu$ arp -a
? (172.20.0.1) at 58:69:6c:a5:e2:d3 on en0 ifscope [ethernet]
(base) AppledeMacBook-Pro:~ VitoLiu$
```

可以看到清除了相关缓存，随后我们输入 ping 192.168.1.82，如下图：

```
VitoLiu — ping 192.168.1.82 — 80x24
Last login: Tue Nov 24 22:13:20 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) AppledeMacBook-Pro:~ VitoLiu$ ping 192.168.1.82
PING 192.168.1.82 (192.168.1.82): 56 data bytes
36 bytes from 192.168.121.1: Time to live exceeded
Vr HL TOS Len ID Flg off TTL Pro cks Src Dst
4 5 00 5400 88ef 0 0000 01 01 b088 172.20.18.35 192.168.1.82

Request timeout for icmp_seq 0
36 bytes from 192.168.121.1: Time to live exceeded
Vr HL TOS Len ID Flg off TTL Pro cks Src Dst
4 5 00 5400 ea65 0 0000 01 01 4f12 172.20.18.35 192.168.1.82

Request timeout for icmp_seq 1
36 bytes from 192.168.121.1: Time to live exceeded
Vr HL TOS Len ID Flg off TTL Pro cks Src Dst
4 5 00 5400 23b8 0 0000 01 01 15c0 172.20.18.35 192.168.1.82

Request timeout for icmp_seq 2
36 bytes from 192.168.121.1: Time to live exceeded
Vr HL TOS Len ID Flg off TTL Pro cks Src Dst
```

启动 Wireshark 的嗅探功能，得到报文列表，并对其进行分析。

首先，arp 报文格式示意图如下，其中硬件类型占 2 字节、协议类型占 2 字节、硬件地址长度 1 字节、协议地址长度 1 字节、OP 码 2 字节、源 MAC 地址 6 字节、源 IP 地址 4 字节、目的 MAC 地址字节、目的 IP 地址 4 字节：



可以通过ARP报文的状态码判断它为接收报文还是请求报文，当OP为01时为请求报文，如下：

```

< Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: Apple_7f:da:62 (a4:83:e7:7f:da:62)
  Sender IP address: 172.20.18.35
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 172.20.18.213

```

```

0000 ff ff ff ff ff a4 83 e7 7f da 62 08 06 00 01 ..... b ...
0010 08 00 06 04 00 01 a4 83 e7 7f da 62 ac 14 12 23 ..... b ...
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 # ...
0030

```

当OP为02时是响应报文，如下图：

```

> Ethernet II, Src: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3), Dst: Apple_7f:da:62 (a4:83:e7:7f:da:62)
< Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3)
  Sender IP address: 172.20.18.213
  Target MAC address: Apple_7f:da:62 (a4:83:e7:7f:da:62)
  Target IP address: 172.20.18.35

```

```

0000 a4 83 e7 7f da 62 58 69 6c a5 e2 d3 08 06 00 01 ..... bXi l ...
0010 08 00 06 04 00 02 58 69 6c a5 e2 d3 ac 14 12 d5 ..... bXi l ...
0020 a4 83 e7 7f da 62 ac 14 12 23 00 00 00 00 00 00 ..... b ... #
0030 00 00 00 00 00 00 00 00 9e d2 7c df ..... | ...

```

另外，对于最后一个问题，由于查询 ARP 不知道目的 IP 对应的 MAC 地址，因此需要广播查询，即设置目的 MAC 地址为广播地址（全1）；而相应 ARP 由于在接收到的查询ARP中找到了源MAC地址，因此响应可以直接有一个明确的目的地址。

### 3.5. 抓取 UDP 数据包

首先，早期版本的QQ消息是基于UDP的，但我的环境中（MacOS版QQ）以TCP作为消息传输的底层协议，因此我使用其他软件发送了以UDP为传输层协议的报文并进行了分析，如图：

▼ User Datagram Protocol, Src Port: 55404, Dst Port: 53

Source Port: 55404  
 Destination Port: 53  
 Length: 44  
 Checksum: 0xe93a [unverified]  
 [Checksum Status: Unverified]  
 [Stream index: 0]  
 > [Timestamps]  
 UDP payload (36 bytes)

再由下图，我们可以看到源和目的主机ip分别为172.20.18.35和218.203.59.116。

另外，通过分析旧版本qq的报文可以得知，该应用的源和目的端口号分别为4000和8000：

37 3.838605 172.20.18.35	218.203.59.116
28 2 020728 172 20 18 35	218 203 59 116

UDP数据报的格式如下，即源端口号两字节，目的端口号两字节，UDP 段长度两字节，校验和两字节：

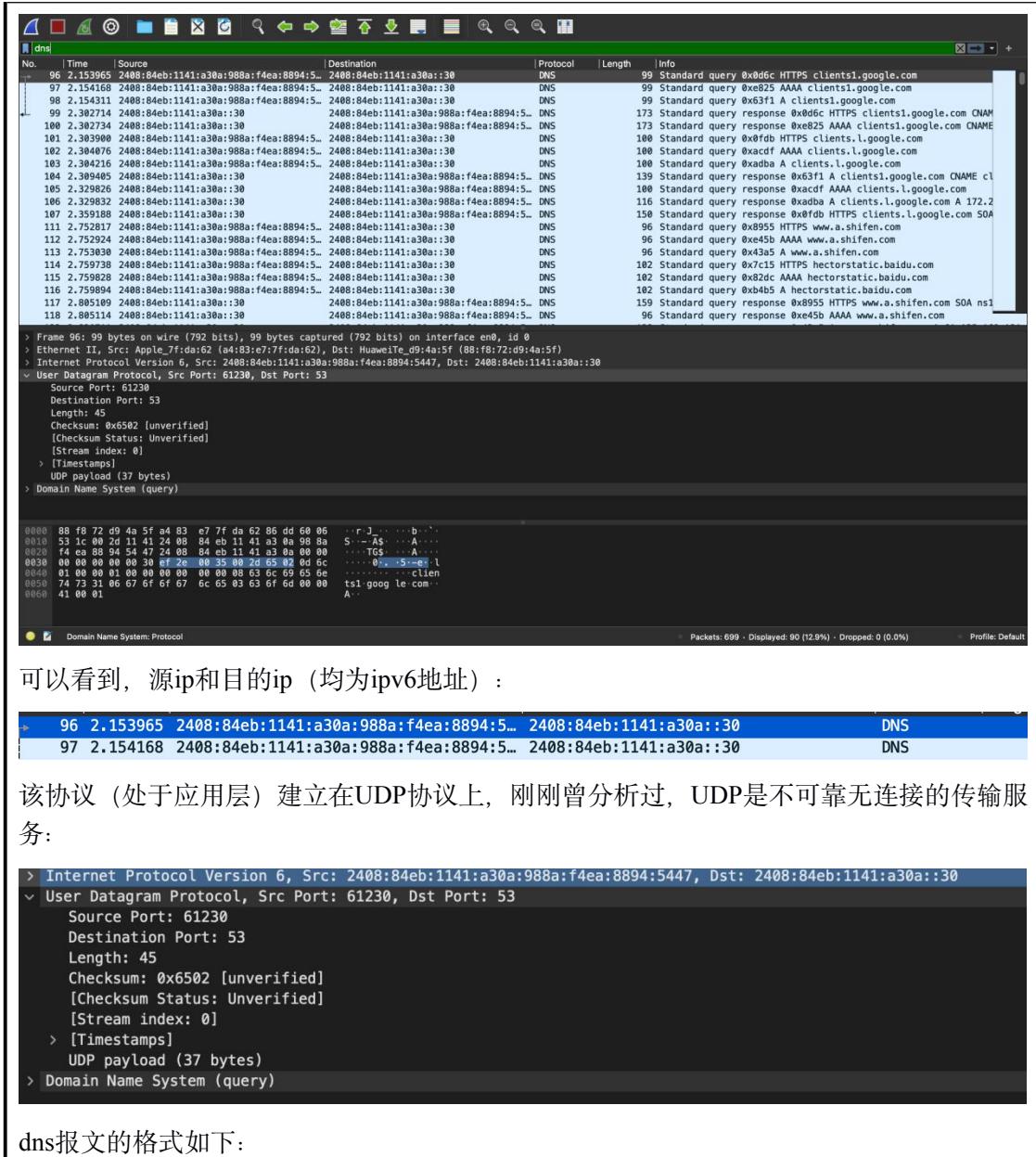
▼ User Datagram Protocol, Src Port: 53, Dst Port: 63736

Source Port: 53  
 Destination Port: 63736  
 Length: 105  
 Checksum: 0xa10a [unverified]  
 [Checksum Status: Unverified]  
 [Stream index: 4]  
 > [Timestamps]  
 UDP payload (97 bytes)

另外，服务器返回 ICQ 为了作为确认。因为 UDP 是不可靠的无连接的传输服务，客户端无法确认服务器已经收到了数据报，所以需要服务器返回 ICQ 报文，可以看出 UDP 是无连接的。因为 TCP 报文需要三次握手建立连接，而且需要 TCP 报文段首部中的标志位，但是 UDP 首部无标志位，UDP 也无序列号。通过抓包分析 UDP 的数据结构可以判断 UDP 是无连接的。

### 3.6.利用 Wireshark 进行 DNS 协议分析

Wireshark开始抓包后，在浏览器中输入www.baidu.com，得到报文列表如下：





下图分别为请求报文消息和响应报文消息，首先是请求(query):

```

▼ Domain Name System (query)
  Transaction ID: 0x7c15
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▼ Queries
    > hectorstatic.baidu.com: type HTTPS, class IN
      [Response In: 123]

```

响应(response):

```

  UDP payload (100 bytes)
▼ Domain Name System (response)
  Transaction ID: 0xcd93
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 3
    Authority RRs: 0
    Additional RRs: 0
  ▼ Queries
    > passport.baidu.com: type A, class IN
  ▼ Answers

```

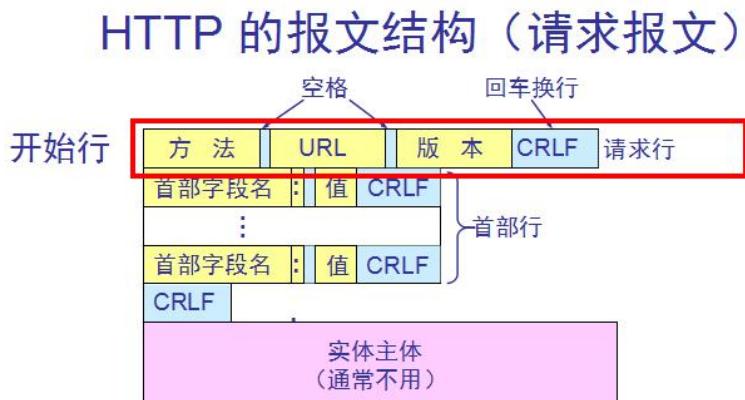
协议总结:

### 1.HTTP协议

HTTP协议有如下特点:

- a.HTTP协议支持客户端/服务器模式。
- b.简单快速: 客户向服务器请求服务时, 只需要传送请求方式和路径, 请求方法常用的有GET、POST、PUT、DELETE.每种方法规定了客户与服务器联系的类型的类型不同。由于HTTP协议简单, 使得HTTP服务器的程序规模小, 因而通信速度很快。
- c.灵活: HTTP协议允许传输任意类型的数据对象, 不同的传输类型由content-Type加以标记。
- d.无连接: 无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求并收到客户的应答后, 即断开连接, 采用这种方式可以节省传输时间。
- e.无状态: HTTP协议是无状态协议, 无状态是指协议对于事务处理没有记忆能力。缺少状态, 意味着如果后续处理需要前面的信息, 则他必须重传, 这样可能导致每次传输的数据量增大。另一方面, 在服务器不需要先前信息时他的应答就很快。

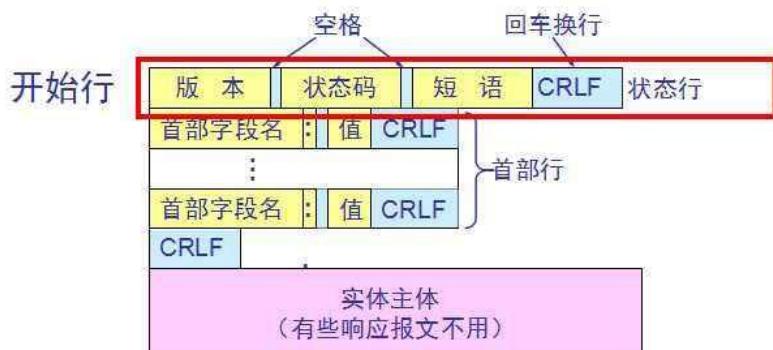
下图分别为HTTP请求报文和响应报文:



报文由三个部分组成, 即**开始行**、**首部行**和**实体主体**。  
在请求报文中, 开始行就是请求行。

图1.请求报文

### HTTP 的报文结构（响应报文）



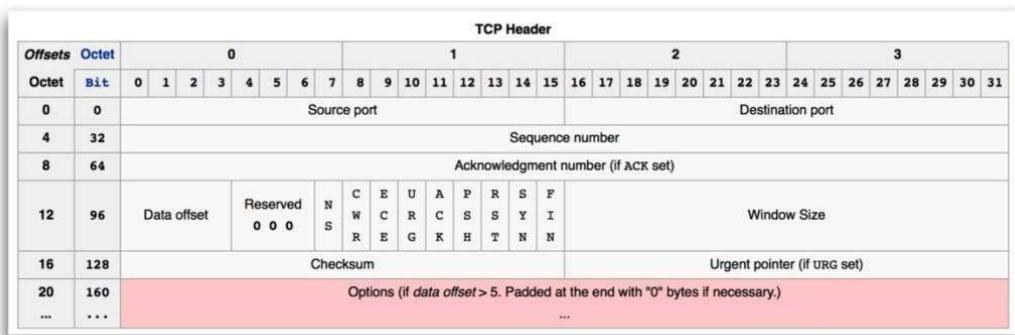
响应报文的开始行是**状态行**。  
状态行包括三项内容, 即**HTTP的版本**, **状态码**,  
以及解释状态码的**简单短语**。

图2.响应报文

## 2.TCP协议

TCP协议是提供面向连接的、可靠的、基于流的数据传输服务，传输单位是报文段；它使用超时重传机制、数据确认等方式确保数据被正确发送至目的地。

如图为TCP报文段格式，含20字节：



## 3.IP协议

IP协议主要有以下几个特点

- a.IP协议是一种无连接、不可靠的分组传送服务的协议。
  - b.IP协议是点-点线路的网络层通信协议。：IP协议是针对原主机-路由器、路由器-路由器、路由器-目的主机之间的数据传输的点-点线路的网络层通信协议。
  - c.IP协议屏蔽了网络在数据链路层、物理层协议与实现技术上的差异。：通过IP协议，网络层向传输层提供的是统一的IP分组，传输层不需要考虑互联网在数据链路层、物理层协议与实现技术上的差异，IP协议使得异构网络的互联变得容易了。
- 另外，ICMP协议主要用来检测网络通信故障和实现链路追踪，最典型的应用就是PING和traceroute。

## 4.ARP协议

地址解析协议，即ARP(Address Resolution Protocol)是根据IP地址获取物理地址的一个TCP/IP协议。主机发送信息时将包含目标IP地址的ARP请求广播到局域网络上的所有主机，并接收返回消息，以此确定目标的物理地址；收到返回消息后将该IP地址和物理地址存入本机ARP缓存中并保留一定时间，下次请求时直接查询ARP缓存以节约资源。

如图为ARP协议格式：



## 5. UDP协议

通过实验及学习，总结UDP有如下特点：

- a. UDP是无连接的，即通信时不需要创建连接（发送数据结束时也没有连接可以释放）所以减小了开销和发送数据前的时延；
- b. UDP采用最大努力交付，不保证可靠交付，因此主机不需要维护复杂的连接状态；
- c. UDP是面向报文的，只在应用层交下来的报文前增加了首部后就向下交付IP层；
- d. UDP是无阻塞控制的，即使网络中存在阻塞，也不会影响发送端的发送频率
- e. UDP支持一对一、一对多、多对一、多对多的互通通信
- f. UDP的首部开销小，只有8个字节，它比TCP的20个字节的首部要短。

如图为UDP报文格式：



## 6. DNS协议

- a. DNS协议就是用来将域名解析到IP地址的一种协议，当然，也可以将IP地址转换为域名的一种协议。
- b. DNS协议基于UDP和TCP协议的，端口号53，用户到服务器采用UDP，DNS服务器通信采用TCP

下图为其实格式：



### DNS报文格式

#### 心得体会:

##### 1.首先在知识理解层面上:

我通过利用Wireshark工具对协议栈各个层次的报文进行抓包并分析，对协议栈中各个层次的协议内容及其交互过程有了更深刻的理解，特别是实地了解了交互过程中各中报文中的内容细节，并且深化了解各协议的报文格式。

##### 2.在技术层面上:

我掌握了Wireshark的主要使用方法，包括对网络报文进行抓包，通过应用显示出的区域进行分析。同时也掌握了pingplotter的traceroute方法，这对于我们日后分析网络状况提供了基本工具。