哈爾濱工業大學

Harbin Institute of Technology

计算机网络
课程实验报告

<table>
<tr><td>实验名称</td><td colspan="6">利用 Wireshark 进行协议分析</td></tr>
<tr><td>姓名</td><td colspan="2">梅智敏</td><td>院系</td><td colspan="3">计算机院软件工程</td></tr>
<tr><td>班级</td><td colspan="2">1837101</td><td>学号</td><td colspan="3">1183710118</td></tr>
<tr><td>任课教师</td><td colspan="2">李全龙</td><td>指导教师</td><td colspan="3">李全龙</td></tr>
<tr><td>实验地点</td><td colspan="2">格物 213</td><td>实验时间</td><td colspan="3">2020.11.21</td></tr>
<tr><td rowspan="2">实验课表现</td><td>出勤、表现得分(10)</td><td></td><td rowspan="2">实验报告
得分(40)</td><td rowspan="2"></td><td rowspan="2">实验总分</td><td rowspan="2"></td></tr>
<tr><td>操作结果得分(50)</td><td></td></tr>
<tr><td colspan="7">教师评语</td></tr>
<tr><td colspan="7"></td></tr>
</table>

计算学部

实验目的：

熟悉并掌握 Wireshark 的基本操作，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：

- 学习 Wireshark 的使用
- 利用 Wireshark 分析 HTTP 协议
- 利用 Wireshark 分析 TCP 协议
- 利用 Wireshark 分析 IP 协议
- 利用 Wireshark 分析 Ethernet 数据帧

选做内容：

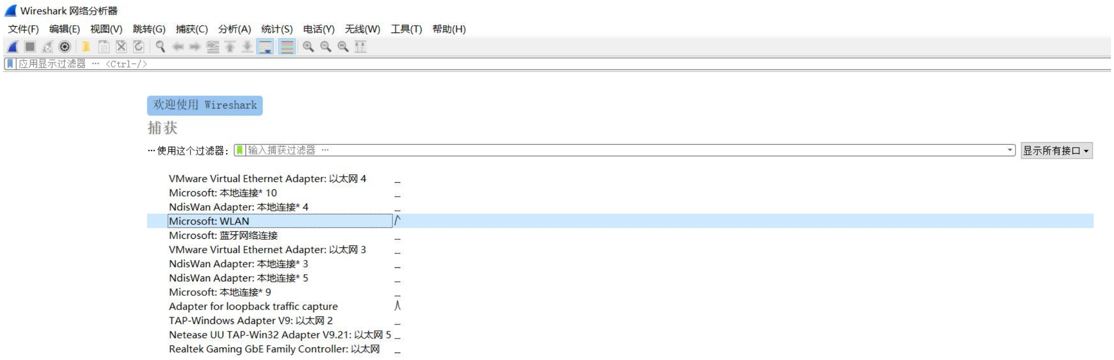
- 利用 Wireshark 分析 DNS 协议
- 利用 Wireshark 分析 UDP 协议
- 利用 Wireshark 分析 ARP 协议

实验过程及结果：

以文字描述、实验结果截图等形式阐述实验过程，必要时可附相应的代码截图或以附件形式提交。

(一) Wireshark的使用

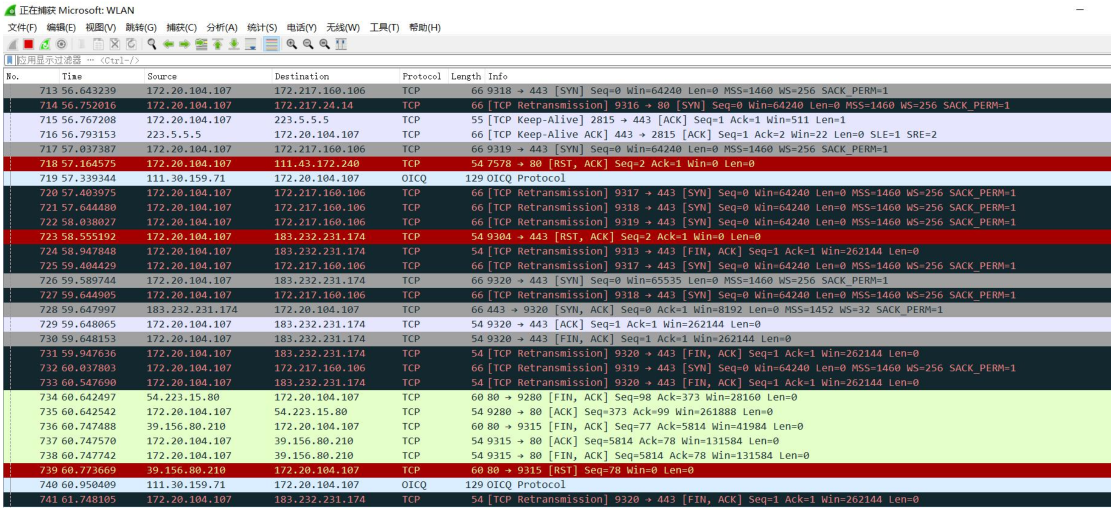
从官网下载安装之后选择网络接口为WLAN（因为我用的是校园网）


Wireshark 网络分析器
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
应用显示过滤器 … <Ctrl-/>
欢迎使用 Wireshark
捕获
显示所有接口 ▾

…使用这个过滤器：输入捕获过滤器 …

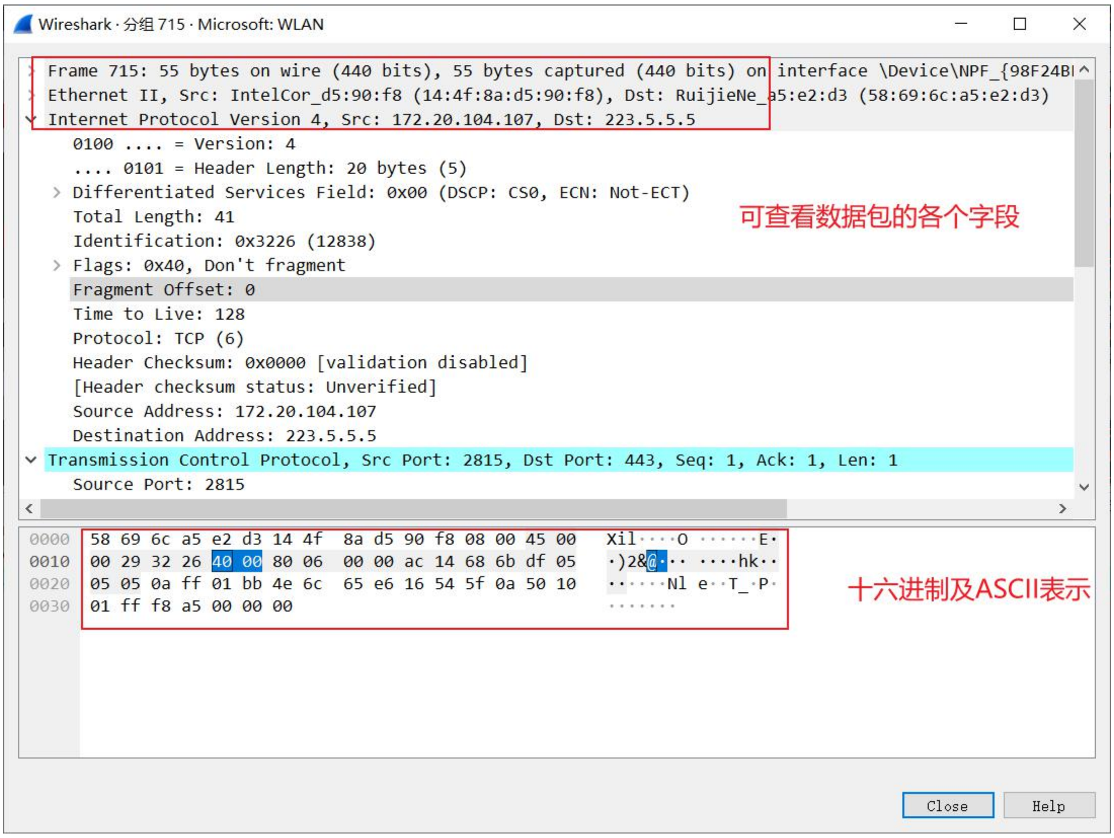
```
VMware Virtual Ethernet Adapter: 以太网 4
Microsoft: 本地连接* 10
NdisWan Adapter: 本地连接* 4
Microsoft: WLAN
Microsoft: 蓝牙网络连接
VMware Virtual Ethernet Adapter: 以太网 3
NdisWan Adapter: 本地连接* 3
NdisWan Adapter: 本地连接* 5
Microsoft: 本地连接* 9
Adapter for loopback traffic capture
TAP-Windows Adapter V9: 以太网 2
Netease UU TAP-Win32 Adapter V9.21: 以太网 5
Realtek Gaming GbE Family Controller: 以太网
```

选择之后便开始自动抓取数据包，我们可以从过滤器中筛选数据包类别


正在捕获 Microsoft: WLAN
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
应用显示过滤器 … <Ctrl-/>

```
No.   Time        Source            Destination       Protocol Length Info
  713 56.643239   172.20.104.107    172.217.160.106   TCP      66 9318 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  714 56.752016   172.20.104.107    172.217.24.14     TCP      66 [TCP Retransmission] 9316 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  715 56.767208   172.20.104.107    223.5.5.5         TCP      55 [TCP Keep-Alive] 2815 → 443 [ACK] Seq=1 Ack=1 Win=511 Len=1
  716 56.793153   223.5.5.5         172.20.104.107    TCP      66 [TCP Keep-Alive ACK] 443 → 2815 [ACK] Seq=1 Ack=2 Win=22 Len=0 SLE=1 SRE=2
  717 57.037387   172.20.104.107    172.217.160.106   TCP      66 9319 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  718 57.164575   172.20.104.107    111.43.172.240    TCP      54 7578 → 80 [RST, ACK] Seq=2 Ack=1 Win=0 Len=0
  719 57.339344   111.30.159.71     172.20.104.107    OICQ    129 OICQ Protocol
  720 57.403975   172.20.104.107    172.217.160.106   TCP      66 [TCP Retransmission] 9317 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  721 57.644480   172.20.104.107    172.217.160.106   TCP      66 [TCP Retransmission] 9318 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  722 58.038027   172.20.104.107    172.217.160.106   TCP      66 [TCP Retransmission] 9319 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  723 58.555192   172.20.104.107    183.232.231.174   TCP      54 9304 → 443 [RST, ACK] Seq=2 Ack=1 Win=0 Len=0
  724 58.947848   172.20.104.107    183.232.231.174   TCP      54 [TCP Retransmission] 9313 → 443 [FIN, ACK] Seq=1 Ack=1 Win=262144 Len=0
  725 59.404429   172.20.104.107    172.217.160.106   TCP      66 [TCP Retransmission] 9317 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  726 59.589744   172.20.104.107    183.232.231.174   TCP      66 9320 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
  727 59.644905   172.20.104.107    172.217.160.106   TCP      66 [TCP Retransmission] 9318 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  728 59.647997   183.232.231.174   172.20.104.107    TCP      66 443 → 9320 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1452 WS=32 SACK_PERM=1
  729 59.648065   172.20.104.107    183.232.231.174   TCP      54 9320 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0
  730 59.648153   172.20.104.107    183.232.231.174   TCP      54 9320 → 443 [FIN, ACK] Seq=1 Ack=1 Win=262144 Len=0
  731 59.947636   172.20.104.107    183.232.231.174   TCP      54 [TCP Retransmission] 9320 → 443 [FIN, ACK] Seq=1 Ack=1 Win=262144 Len=0
  732 60.037803   172.20.104.107    172.217.160.106   TCP      66 [TCP Retransmission] 9319 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  733 60.547690   172.20.104.107    183.232.231.174   TCP      54 [TCP Retransmission] 9320 → 443 [FIN, ACK] Seq=1 Ack=1 Win=262144 Len=0
  734 60.642497   54.223.15.80      172.20.104.107    TCP      60 80 → 9280 [FIN, ACK] Seq=98 Ack=373 Win=28160 Len=0
  735 60.642542   172.20.104.107    54.223.15.80      TCP      54 9280 → 80 [ACK] Seq=373 Ack=99 Win=261888 Len=0
  736 60.747488   39.156.80.210     172.20.104.107    TCP      60 80 → 9315 [FIN, ACK] Seq=77 Ack=5814 Win=41984 Len=0
  737 60.747570   172.20.104.107    39.156.80.210     TCP      54 9315 → 80 [ACK] Seq=5814 Ack=78 Win=131584 Len=0
  738 60.747742   172.20.104.107    39.156.80.210     TCP      54 9315 → 80 [FIN, ACK] Seq=5814 Ack=78 Win=131584 Len=0
  739 60.773669   39.156.80.210     172.20.104.107    TCP      60 80 → 9315 [RST] Seq=78 Win=0 Len=0
  740 60.950409   111.30.159.71     172.20.104.107    OICQ    129 OICQ Protocol
  741 61.748105   172.20.104.107    183.232.231.174   TCP      54 [TCP Retransmission] 9320 → 443 [FIN, ACK] Seq=1 Ack=1 Win=262144 Len=0
```

利用Wireshark可以对数据报的各个字段进行查看分析



```
Frame 715: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface \Device\NPF_{98F24BI
Ethernet II, Src: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8), Dst: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3)
Internet Protocol Version 4, Src: 172.20.104.107, Dst: 223.5.5.5
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 41
    Identification: 0x3226 (12838)
  > Flags: 0x40, Don't fragment
    Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.104.107
    Destination Address: 223.5.5.5
Transmission Control Protocol, Src Port: 2815, Dst Port: 443, Seq: 1, Ack: 1, Len: 1
    Source Port: 2815

0000  58 69 6c a5 e2 d3 14 4f  8a d5 90 f8 08 00 45 00   Xil····O ······E·
0010  00 29 32 26 40 00 80 06  00 00 ac 14 68 6b df 05   ·)2&@··  ····hk··
0020  05 05 0a ff 01 bb 4e 6c  65 e6 16 54 5f 0a 50 10   ······Nl e··T_·P·
0030  01 ff f8 a5 00 00 00                               ·······
```

可查看数据包的各个字段

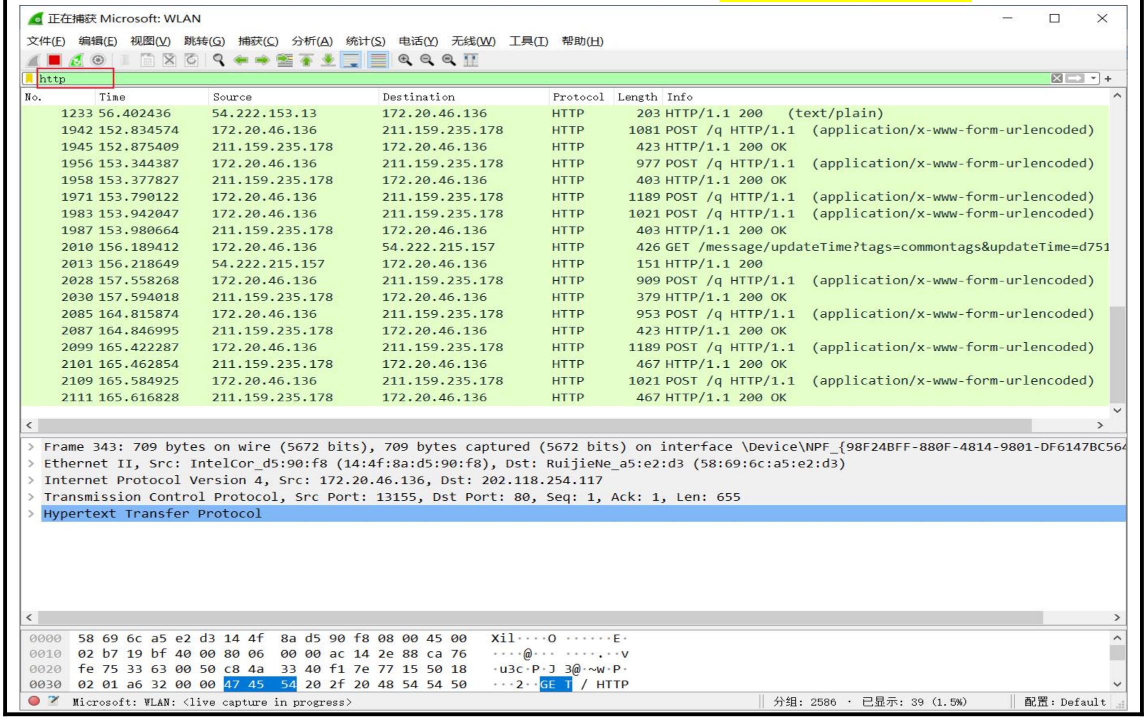
十六进制及ASCII表示

(二)HTTP分析

1)   HTTP GET/response 交互

首先启动谷歌Web browser，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。

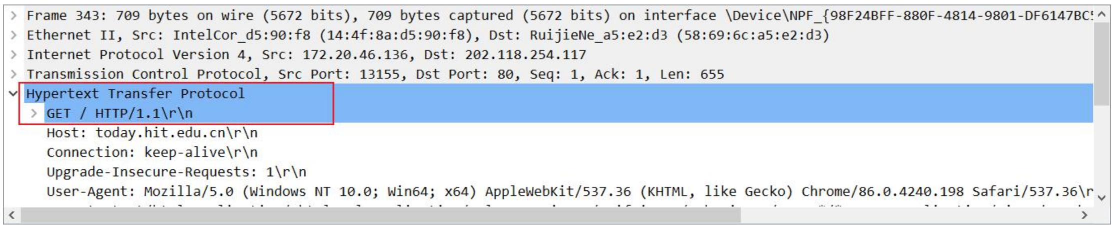
然后在打开的谷歌浏览器地址栏输入今日哈工大的网址：http://today.hit.edu.cn/即可



正在捕获 Microsoft: WLAN
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

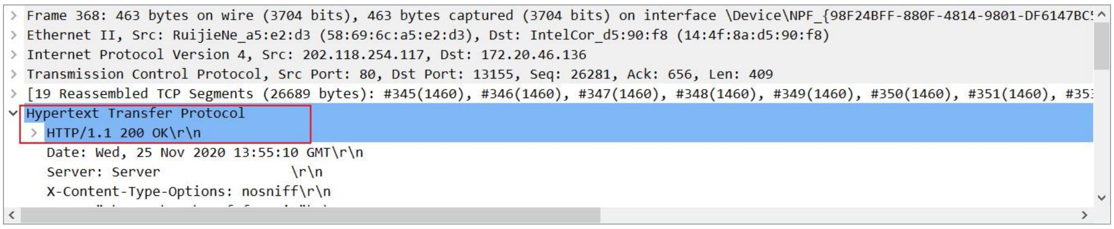
```
http

No.   Time        Source           Destination      Protocol Length Info
 1233 56.402436   54.222.153.13    172.20.46.136    HTTP      203 HTTP/1.1 200   (text/plain)
 1942 152.834574  172.20.46.136    211.159.235.178  HTTP     1081 POST /q HTTP/1.1  (application/x-www-form-urlencoded)
 1945 152.875409  211.159.235.178  172.20.46.136    HTTP      423 HTTP/1.1 200 OK
 1956 153.344387  172.20.46.136    211.159.235.178  HTTP      977 POST /q HTTP/1.1  (application/x-www-form-urlencoded)
 1958 153.377827  211.159.235.178  172.20.46.136    HTTP      403 HTTP/1.1 200 OK
 1971 153.790122  172.20.46.136    211.159.235.178  HTTP     1189 POST /q HTTP/1.1  (application/x-www-form-urlencoded)
 1983 153.942047  172.20.46.136    211.159.235.178  HTTP     1021 POST /q HTTP/1.1  (application/x-www-form-urlencoded)
 1987 153.980664  211.159.235.178  172.20.46.136    HTTP      403 HTTP/1.1 200 OK
 2010 156.189412  172.20.46.136    54.222.215.157   HTTP      426 GET /message/updateTime?tags=commontags&updateTime=d751
 2013 156.218649  54.222.215.157   172.20.46.136    HTTP      151 HTTP/1.1 200
 2028 157.558268  172.20.46.136    211.159.235.178  HTTP      909 POST /q HTTP/1.1  (application/x-www-form-urlencoded)
 2030 157.594018  211.159.235.178  172.20.46.136    HTTP      379 HTTP/1.1 200 OK
 2085 164.815874  172.20.46.136    211.159.235.178  HTTP      953 POST /q HTTP/1.1  (application/x-www-form-urlencoded)
 2087 164.846995  211.159.235.178  172.20.46.136    HTTP      423 HTTP/1.1 200 OK
 2099 165.422287  172.20.46.136    211.159.235.178  HTTP     1189 POST /q HTTP/1.1  (application/x-www-form-urlencoded)
 2101 165.462854  211.159.235.178  172.20.46.136    HTTP      467 HTTP/1.1 200 OK
 2109 165.584925  172.20.46.136    211.159.235.178  HTTP     1021 POST /q HTTP/1.1  (application/x-www-form-urlencoded)
 2111 165.616828  211.159.235.178  172.20.46.136    HTTP      467 HTTP/1.1 200 OK

> Frame 343: 709 bytes on wire (5672 bits), 709 bytes captured (5672 bits) on interface \Device\NPF_{98F24BFF-880F-4814-9801-DF6147BC564
> Ethernet II, Src: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8), Dst: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3)
> Internet Protocol Version 4, Src: 172.20.46.136, Dst: 202.118.254.117
> Transmission Control Protocol, Src Port: 13155, Dst Port: 80, Seq: 1, Ack: 1, Len: 655
> Hypertext Transfer Protocol

0000  58 69 6c a5 e2 d3 14 4f  8a d5 90 f8 08 00 45 00   Xil····O ······E·
0010  02 b7 19 bf 40 00 80 06  00 00 ac 14 2e 88 ca 76   ····@··· ·····.·v
0020  fe 75 33 63 00 50 c8 4a  33 40 f1 7e 77 15 50 18   ·u3c·P·J 3@·~w·P·
0030  02 01 a6 32 00 00 47 45  54 20 2f 20 48 54 54 50   ···2··GE T / HTTP

Microsoft: WLAN: <live capture in progress>      分组: 2586 · 已显示: 39 (1.5%)      配置: Default
```

- 浏览器运行的协议版本为HTTP 1.1



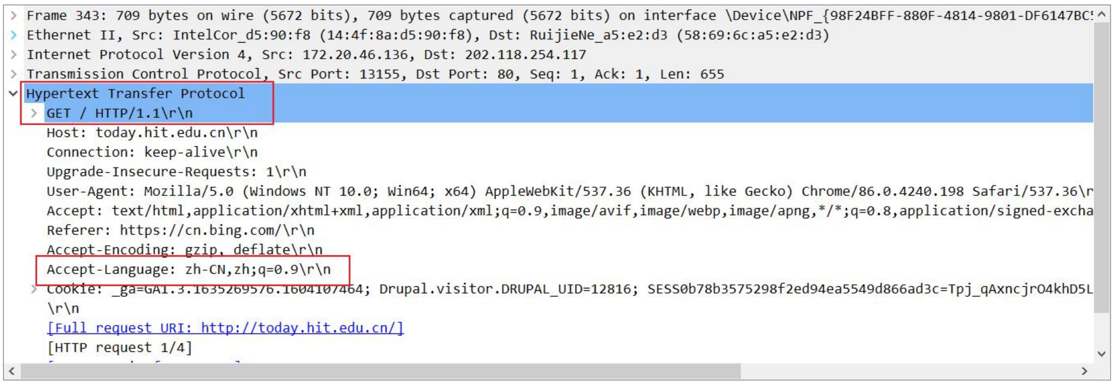
```
> Frame 343: 709 bytes on wire (5672 bits), 709 bytes captured (5672 bits) on interface \Device\NPF_{98F24BFF-880F-4814-9801-DF6147BC5
> Ethernet II, Src: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8), Dst: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3)
> Internet Protocol Version 4, Src: 172.20.46.136, Dst: 202.118.254.117
> Transmission Control Protocol, Src Port: 13155, Dst Port: 80, Seq: 1, Ack: 1, Len: 655
v Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: today.hit.edu.cn\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36\r
```

- 所访问的服务器运行的协议版本是HTTP 1.1



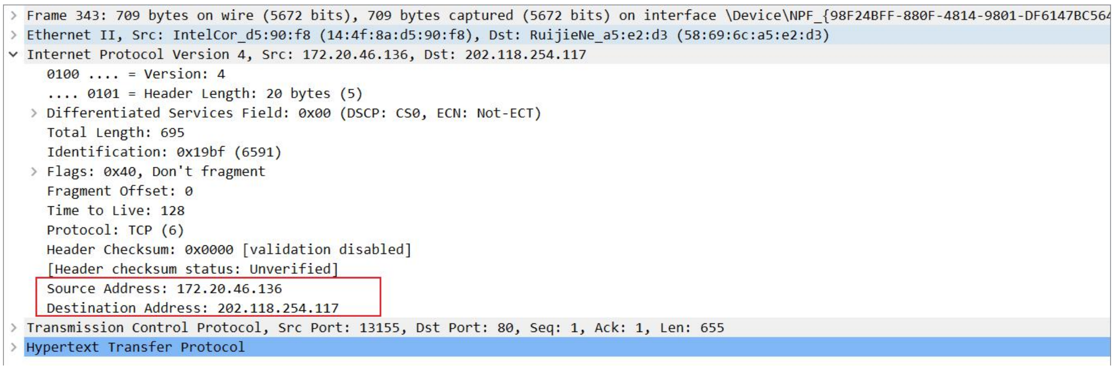
```
> Frame 368: 463 bytes on wire (3704 bits), 463 bytes captured (3704 bits) on interface \Device\NPF_{98F24BFF-880F-4814-9801-DF6147BC5
> Ethernet II, Src: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3), Dst: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8)
> Internet Protocol Version 4, Src: 202.118.254.117, Dst: 172.20.46.136
> Transmission Control Protocol, Src Port: 80, Dst Port: 13155, Seq: 26281, Ack: 656, Len: 409
> [19 Reassembled TCP Segments (26689 bytes): #345(1460), #346(1460), #347(1460), #348(1460), #349(1460), #350(1460), #351(1460), #35
v Hypertext Transfer Protocol
  > HTTP/1.1 200 OK\r\n
    Date: Wed, 25 Nov 2020 13:55:10 GMT\r\n
    Server: Server                \r\n
    X-Content-Type-Options: nosniff\r\n
```

- 浏览器向服务器指出它能接收的语言版本为：zh-CN



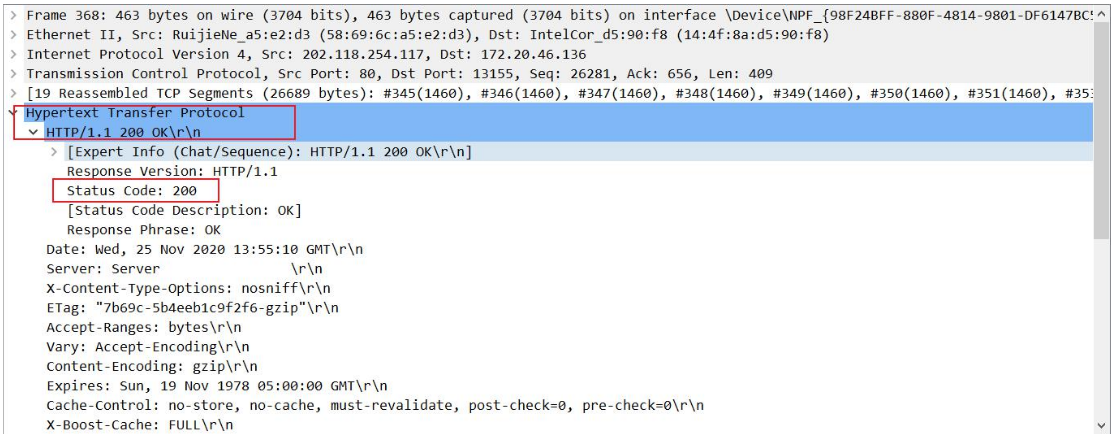
```
> Frame 343: 709 bytes on wire (5672 bits), 709 bytes captured (5672 bits) on interface \Device\NPF_{98F24BFF-880F-4814-9801-DF6147BC5
> Ethernet II, Src: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8), Dst: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3)
> Internet Protocol Version 4, Src: 172.20.46.136, Dst: 202.118.254.117
> Transmission Control Protocol, Src Port: 13155, Dst Port: 80, Seq: 1, Ack: 1, Len: 655
v Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: today.hit.edu.cn\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36\r
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-excha
    Referer: https://cn.bing.com/\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: zh-CN,zh;q=0.9\r\n
  > Cookie: _ga=GA1.3.1635269576.1604107464; Drupal.visitor.DRUPAL_UID=12816; SESS0b78b3575298f2ed94ea5549d866ad3c=Tpj_qAxncjrO4khD5L
    \r\n
    [Full request URI: http://today.hit.edu.cn/]
    [HTTP request 1/4]
```

- 本机IP地址为：172.20.46.136
  服务器IP地址为：202.118.254.117



```
> Frame 343: 709 bytes on wire (5672 bits), 709 bytes captured (5672 bits) on interface \Device\NPF_{98F24BFF-880F-4814-9801-DF6147BC56
> Ethernet II, Src: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8), Dst: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3)
v Internet Protocol Version 4, Src: 172.20.46.136, Dst: 202.118.254.117
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 695
    Identification: 0x19bf (6591)
  > Flags: 0x40, Don't fragment
    Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.46.136
    Destination Address: 202.118.254.117
> Transmission Control Protocol, Src Port: 13155, Dst Port: 80, Seq: 1, Ack: 1, Len: 655
> Hypertext Transfer Protocol
```

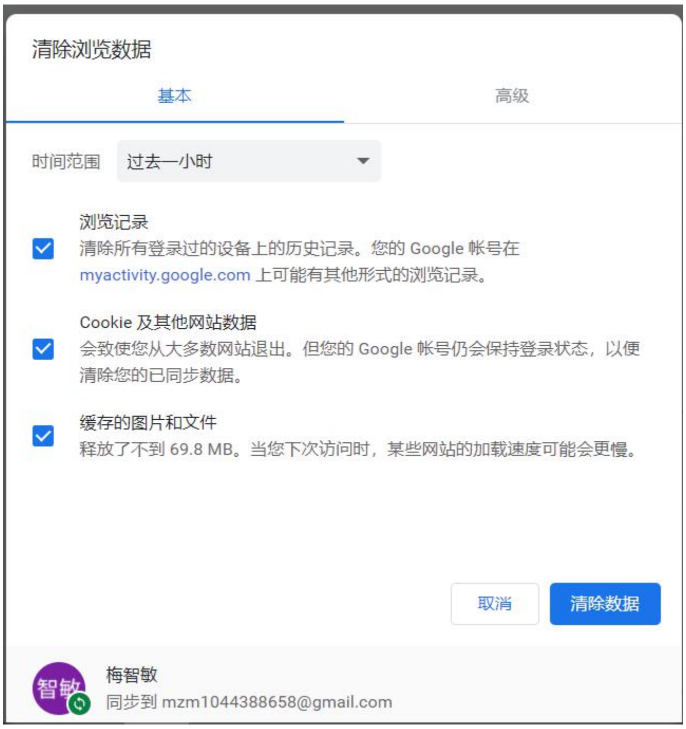
- 服务器向你的浏览器返回的状态代码为：200



```
> Frame 368: 463 bytes on wire (3704 bits), 463 bytes captured (3704 bits) on interface \Device\NPF_{98F24BFF-880F-4814-9801-DF6147BC
> Ethernet II, Src: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3), Dst: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8)
> Internet Protocol Version 4, Src: 202.118.254.117, Dst: 172.20.46.136
> Transmission Control Protocol, Src Port: 80, Dst Port: 13155, Seq: 26281, Ack: 656, Len: 409
> [19 Reassembled TCP Segments (26689 bytes): #345(1460), #346(1460), #347(1460), #348(1460), #349(1460), #350(1460), #351(1460), #35
v Hypertext Transfer Protocol
  v HTTP/1.1 200 OK\r\n
    > [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
    Date: Wed, 25 Nov 2020 13:55:10 GMT\r\n
    Server: Server            \r\n
    X-Content-Type-Options: nosniff\r\n
    ETag: "7b69c-5b4eeb1c9f2f6-gzip"\r\n
    Accept-Ranges: bytes\r\n
    Vary: Accept-Encoding\r\n
    Content-Encoding: gzip\r\n
    Expires: Sun, 19 Nov 1978 05:00:00 GMT\r\n
    Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0\r\n
    X-Boost-Cache: FULL\r\n
```

2)　HTTP 条件 GET/response 交互

首先清空谷歌浏览器的缓存数据



清除浏览数据

时间范围　过去一小时

☑ 浏览记录
清除所有登录过的设备上的历史记录。您的 Google 帐号在 myactivity.google.com 上可能有其他形式的浏览记录。

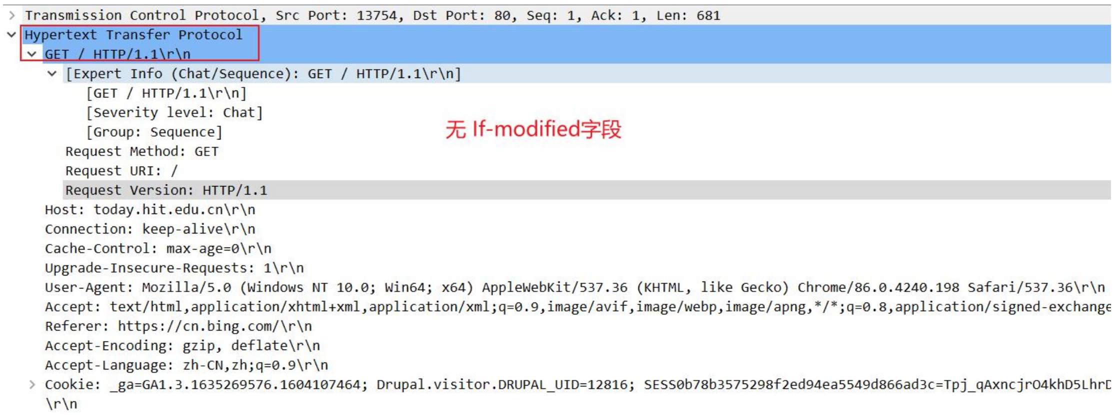
☑ Cookie 及其他网站数据
会致使您从大多数网站退出。但您的 Google 帐号仍会保持登录状态，以便清除您的已同步数据。

☑ 缓存的图片和文件
释放了不到 69.8 MB。当您下次访问时，某些网站的加载速度可能会更慢。

梅智敏
同步到 mzm1044388658@gmail.com

然后再连续访问2次http://today.hit.edu.cn/即可

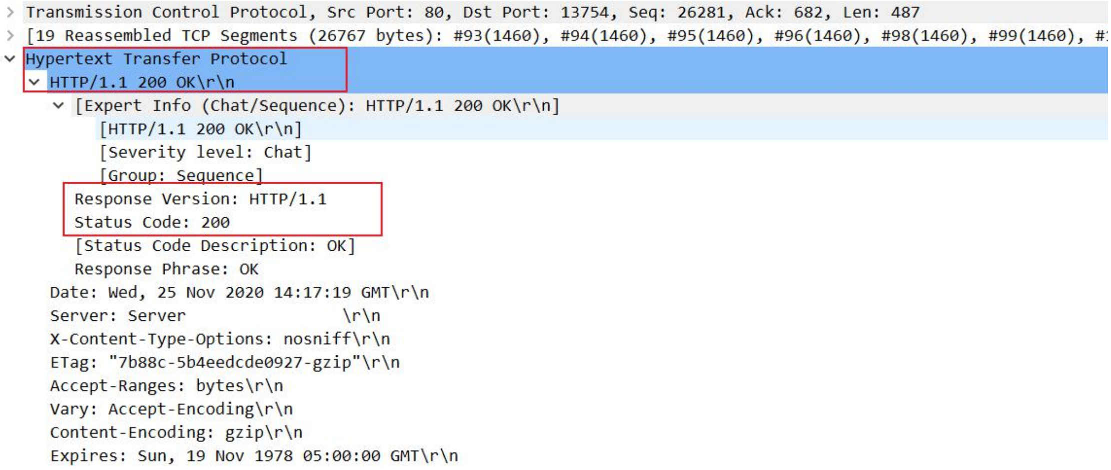
- 浏览器向服务器发出的第一个 HTTP GET 请求的内容中，并无IF-MODIFIED-SINCE



```
> Transmission Control Protocol, Src Port: 13754, Dst Port: 80, Seq: 1, Ack: 1, Len: 681
v Hypertext Transfer Protocol
  v GET / HTTP/1.1\r\n
    v [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
        [GET / HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: GET
      Request URI: /
      Request Version: HTTP/1.1
    Host: today.hit.edu.cn\r\n
    Connection: keep-alive\r\n
    Cache-Control: max-age=0\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
    Referer: https://cn.bing.com/\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: zh-CN,zh;q=0.9\r\n
  > Cookie: _ga=GA1.3.1635269576.1604107464; Drupal.visitor.DRUPAL_UID=12816; SESS0b78b3575298f2ed94ea5549d866ad3c=Tpj_qAxncjrO4khD5LhrD
    \r\n
```

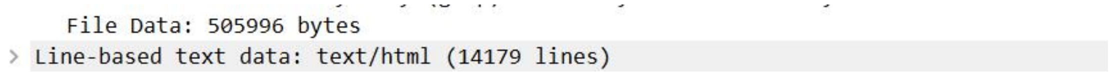
无 If-modified字段

- 分析服务器响应报文的内容，服务器明确返回了文件的内容，通过状态码获知。



```
> Transmission Control Protocol, Src Port: 80, Dst Port: 13754, Seq: 26281, Ack: 682, Len: 487
> [19 Reassembled TCP Segments (26767 bytes): #93(1460), #94(1460), #95(1460), #96(1460), #98(1460), #99(1460), #
v Hypertext Transfer Protocol
  v HTTP/1.1 200 OK\r\n
    v [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
        [HTTP/1.1 200 OK\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
    Date: Wed, 25 Nov 2020 14:17:19 GMT\r\n
    Server: Server                \r\n
    X-Content-Type-Options: nosniff\r\n
    ETag: "7b88c-5b4eedcde0927-gzip"\r\n
    Accept-Ranges: bytes\r\n
    Vary: Accept-Encoding\r\n
    Content-Encoding: gzip\r\n
    Expires: Sun, 19 Nov 1978 05:00:00 GMT\r\n
```

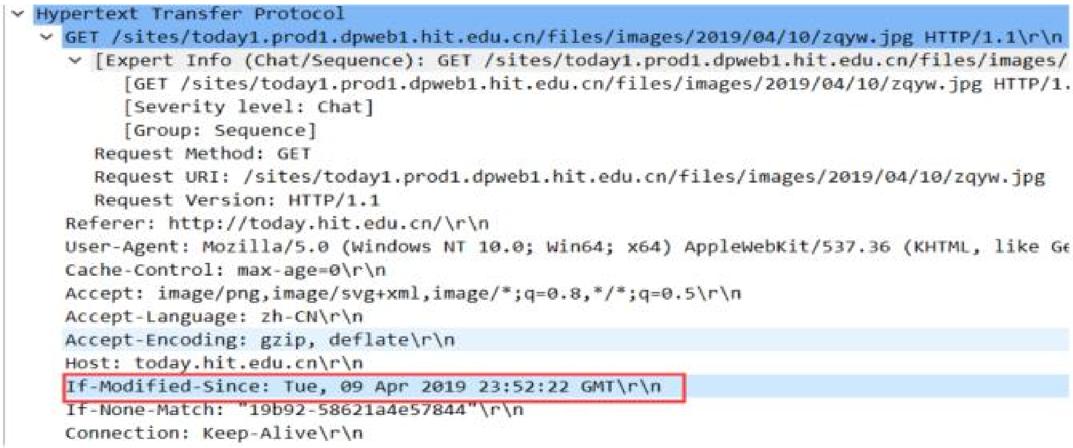
通过状态码为200可以获知明确返回了文件内容



```
    File Data: 505996 bytes
> Line-based text data: text/html (14179 lines)
```

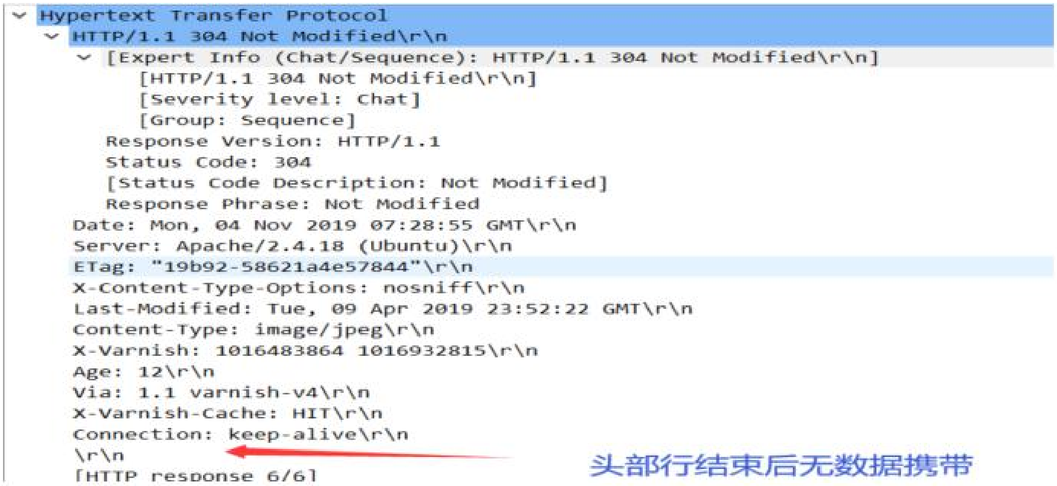
而且相应报文中也确实携带了数据

- 浏览器向服务器发出的较晚的“HTTP GET”请求，报文中有IF-MODIFIED-SINCE字段，且在该首部行后面跟着的信息是本地缓存文件的最新版本（用时间来表示）



```
v Hypertext Transfer Protocol
  v GET /sites/today1.prod1.dpweb1.hit.edu.cn/files/images/2019/04/10/zqyw.jpg HTTP/1.1\r\n
    v [Expert Info (Chat/Sequence): GET /sites/today1.prod1.dpweb1.hit.edu.cn/files/images/
        [GET /sites/today1.prod1.dpweb1.hit.edu.cn/files/images/2019/04/10/zqyw.jpg HTTP/1.
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: GET
      Request URI: /sites/today1.prod1.dpweb1.hit.edu.cn/files/images/2019/04/10/zqyw.jpg
      Request Version: HTTP/1.1
    Referer: http://today.hit.edu.cn/\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
    Cache-Control: max-age=0\r\n
    Accept: image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5\r\n
    Accept-Language: zh-CN\r\n
    Accept-Encoding: gzip, deflate\r\n
    Host: today.hit.edu.cn\r\n
    If-Modified-Since: Tue, 09 Apr 2019 23:52:22 GMT\r\n
    If-None-Match: "19b92-58621a4e57844"\r\n
    Connection: Keep-Alive\r\n
```

- 服务器对较晚的 HTTP GET 请求的响应中的 HTTP 状态代码是304，服务器未返回文件的内容。解释如下：若本地有目的URL的缓存文件，客户端会构造请求报文来确认该报文是否是最新版本；若是，则返回状态码304，并且不携带数据，客户端直接从本地缓存中获取文件内容即可。

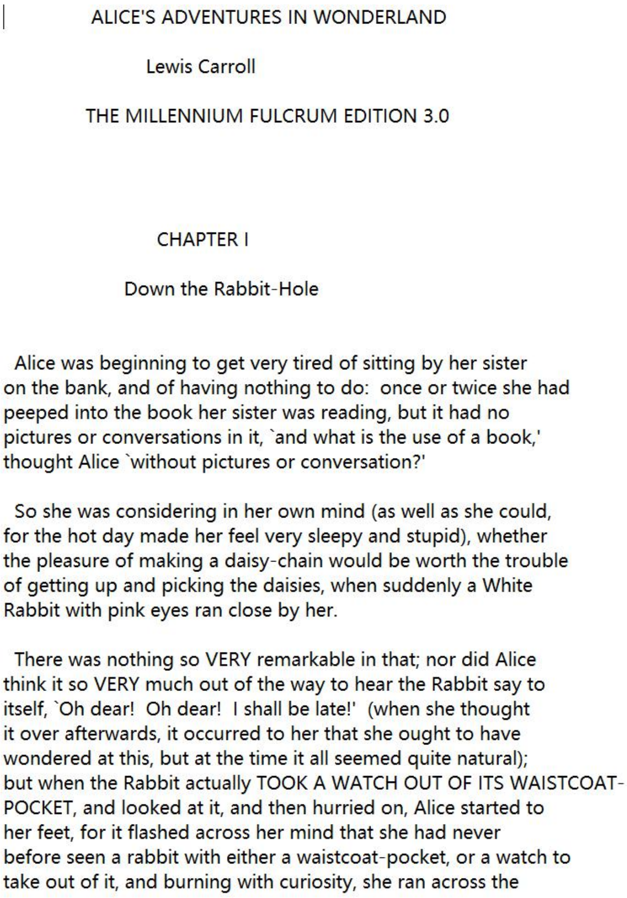


```
v Hypertext Transfer Protocol
  v HTTP/1.1 304 Not Modified\r\n
    v [Expert Info (Chat/Sequence): HTTP/1.1 304 Not Modified\r\n]
        [HTTP/1.1 304 Not Modified\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Response Version: HTTP/1.1
      Status Code: 304
      [Status Code Description: Not Modified]
      Response Phrase: Not Modified
    Date: Mon, 04 Nov 2019 07:28:55 GMT\r\n
    Server: Apache/2.4.18 (Ubuntu)\r\n
    ETag: "19b92-58621a4e57844"\r\n
    X-Content-Type-Options: nosniff\r\n
    Last-Modified: Tue, 09 Apr 2019 23:52:22 GMT\r\n
    Content-Type: image/jpeg\r\n
    X-Varnish: 1016483864 1016932815\r\n
    Age: 12\r\n
    Via: 1.1 varnish-v4\r\n
    X-Varnish-Cache: HIT\r\n
    Connection: keep-alive\r\n
    \r\n
    [HTTP response 6/6]
```

头部行结束后无数据携带

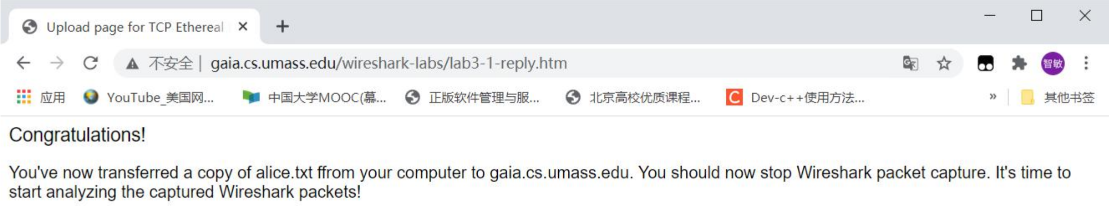
(三)TCP分析

首先启动浏览器，进入指导书中的URL，得到文本文件如下



```
            ALICE'S ADVENTURES IN WONDERLAND

                          Lewis Carroll

               THE MILLENNIUM FULCRUM EDITION 3.0




                            CHAPTER I

                      Down the Rabbit-Hole


  Alice was beginning to get very tired of sitting by her sister
on the bank, and of having nothing to do:  once or twice she had
peeped into the book her sister was reading, but it had no
pictures or conversations in it, `and what is the use of a book,'
thought Alice `without pictures or conversation?'

  So she was considering in her own mind (as well as she could,
for the hot day made her feel very sleepy and stupid), whether
the pleasure of making a daisy-chain would be worth the trouble
of getting up and picking the daisies, when suddenly a White
Rabbit with pink eyes ran close by her.

  There was nothing so VERY remarkable in that; nor did Alice
think it so VERY much out of the way to hear the Rabbit say to
itself, `Oh dear!  Oh dear!  I shall be late!'  (when she thought
it over afterwards, it occurred to her that she ought to have
wondered at this, but at the time it all seemed quite natural);
but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-
POCKET, and looked at it, and then hurried on, Alice started to
her feet, for it flashed across her mind that she had never
before seen a rabbit with either a waistcoat-pocket, or a watch to
take out of it, and burning with curiosity, she ran across the
```

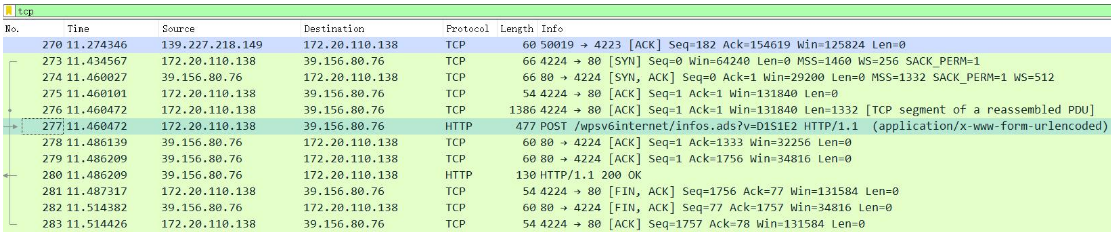
然后在给定网站中进行上传该文件即可



Congratulations!

You've now transferred a copy of alice.txt ffrom your computer to gaia.cs.umass.edu. You should now stop Wireshark packet capture. It's time to start analyzing the captured Wireshark packets!

下面对我们的追踪信息进行分析



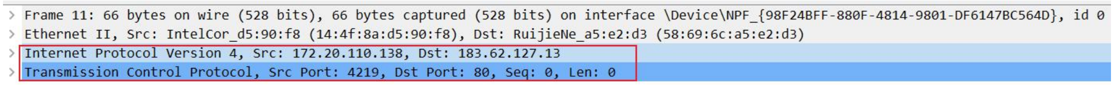
```
tcp
No.    Time       Source           Destination      Protocol Length Info
   270 11.274346  139.227.218.149  172.20.110.138   TCP        60 50019 → 4223 [ACK] Seq=182 Ack=154619 Win=125824 Len=0
   273 11.434567  172.20.110.138   39.156.80.76     TCP        66 4224 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   274 11.460027  39.156.80.76     172.20.110.138   TCP        66 80 → 4224 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1332 SACK_PERM=1 WS=512
   275 11.460101  172.20.110.138   39.156.80.76     TCP        54 4224 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=0
   276 11.460472  172.20.110.138   39.156.80.76     TCP      1386 4224 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=1332 [TCP segment of a reassembled PDU]
   277 11.460472  172.20.110.138   39.156.80.76     HTTP      477 POST /wpsv6internet/infos.ads?v=D1S1E2 HTTP/1.1  (application/x-www-form-urlencoded)
   278 11.486139  39.156.80.76     172.20.110.138   TCP        60 80 → 4224 [ACK] Seq=1 Ack=1333 Win=32256 Len=0
   279 11.486209  39.156.80.76     172.20.110.138   TCP        60 80 → 4224 [ACK] Seq=1 Ack=1756 Win=34816 Len=0
   280 11.486209  39.156.80.76     172.20.110.138   HTTP      130 HTTP/1.1 200 OK
   281 11.487317  172.20.110.138   39.156.80.76     TCP        54 4224 → 80 [FIN, ACK] Seq=1756 Ack=77 Win=131584 Len=0
   282 11.514382  39.156.80.76     172.20.110.138   TCP        60 80 → 4224 [FIN, ACK] Seq=77 Ack=1757 Win=34816 Len=0
   283 11.514426  172.20.110.138   39.156.80.76     TCP        54 4224 → 80 [ACK] Seq=1757 Ack=78 Win=131584 Len=0
```

- 向 gaia.cs.umass.edu 服务器传送文件的客户端主机的 IP 地址为：172.20.110.138

TCP 端口号是4219

Gaia.cs.umass.edu 服务器的 IP 地址是183.62.127.13

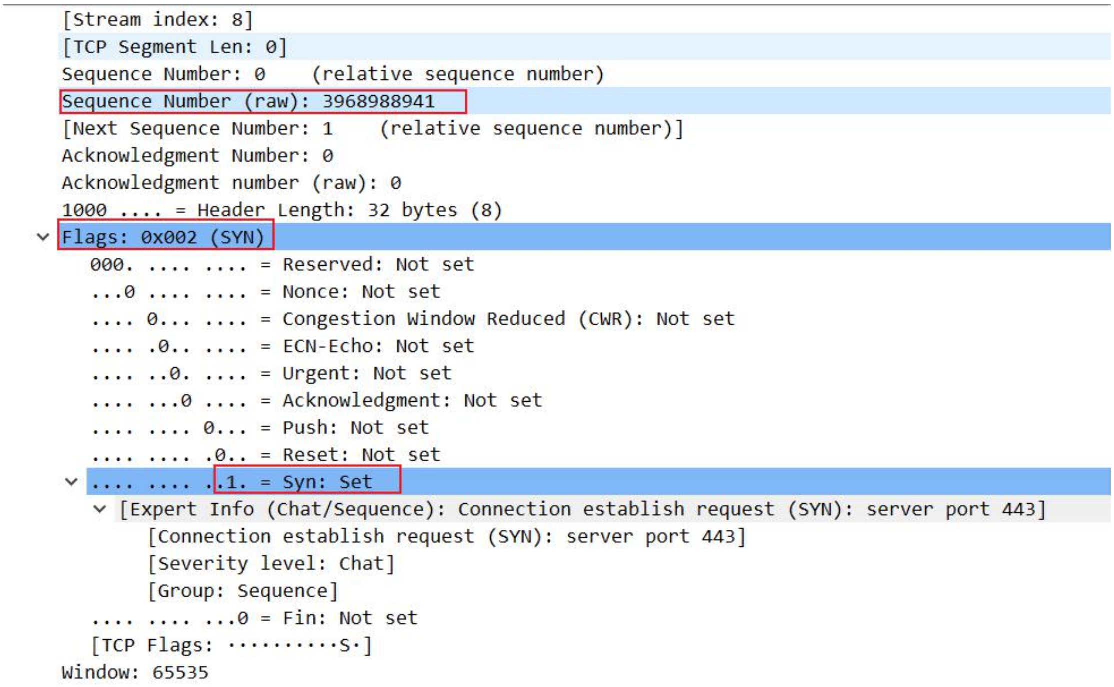
对这一连接，它用来发送和接收 TCP 报文的端口号是80



```
> Frame 11: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{98F24BFF-880F-4814-9801-DF6147BC564D}, id 0
> Ethernet II, Src: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8), Dst: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3)
> Internet Protocol Version 4, Src: 172.20.110.138, Dst: 183.62.127.13
> Transmission Control Protocol, Src Port: 4219, Dst Port: 80, Seq: 0, Len: 0
```

- 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:3968988941

在该报文段中，用SYN标志位为1来标示该报文段是 SYN 报文段



```
    [Stream index: 8]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 3968988941
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  ˅ Flags: 0x002 (SYN)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    ˅ .... .... ..1. = Syn: Set
      ˅ [Expert Info (Chat/Sequence): Connection establish request (SYN): server port 443]
          [Connection establish request (SYN): server port 443]
          [Severity level: Chat]
          [Group: Sequence]
      .... .... ...0 = Fin: Not set
      [TCP Flags: ··········S·]
    Window: 65535
```

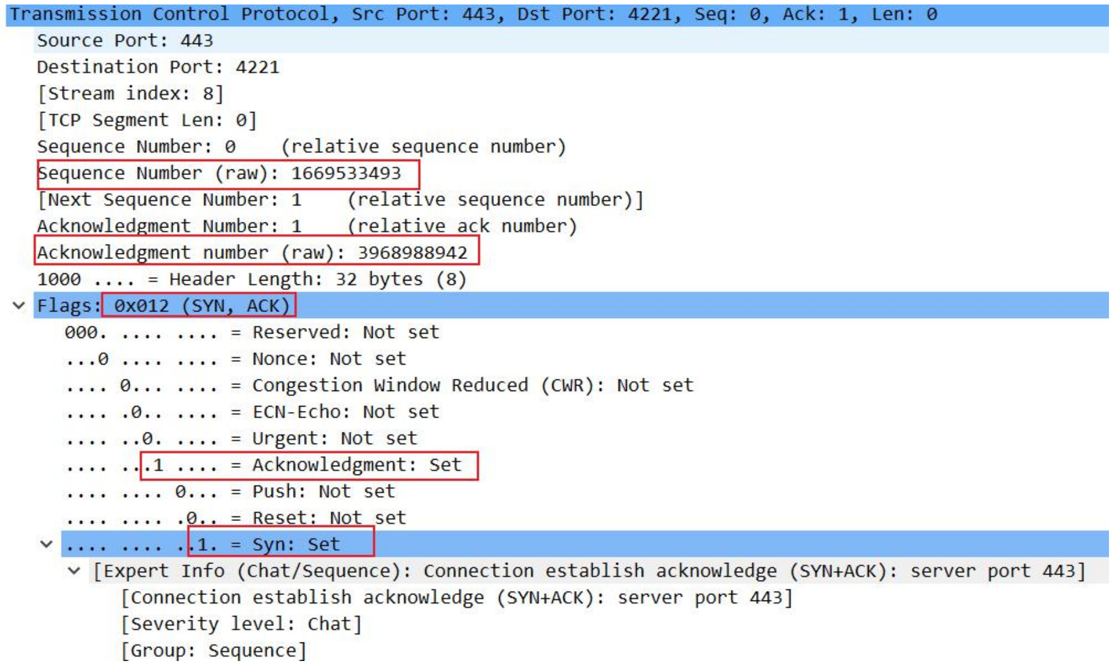
上图中，各个Flags只有Syn为1，其余皆为0

- 服务器向客户端发送的 SYNACK 报文段序号是：1669533493

该报文段中，Acknowledgement字段的值是：3968988942

Gaia.cs.umass.edu 服务器是如何决定此值的：其值就是客户端的SYN报文段的序列号加一

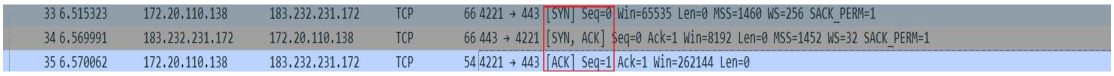
在该报文段中，是用什么来标示该报文段是SYNACK 报文段的：将ACK和SYN标志位同时置1，其余标志位为0



```
Transmission Control Protocol, Src Port: 443, Dst Port: 4221, Seq: 0, Ack: 1, Len: 0
    Source Port: 443
    Destination Port: 4221
    [Stream index: 8]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1669533493
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 3968988942
    1000 .... = Header Length: 32 bytes (8)
  v Flags: 0x012 (SYN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    v .... .... ..1. = Syn: Set
      v [Expert Info (Chat/Sequence): Connection establish acknowledge (SYN+ACK): server port 443]
          [Connection establish acknowledge (SYN+ACK): server port 443]
          [Severity level: Chat]
          [Group: Sequence]
```

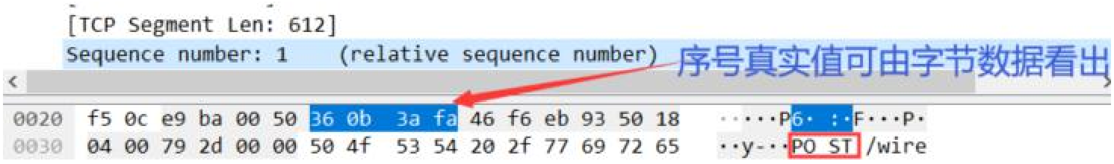
- 可以从捕获的数据包中分析出TCP的三次握手

可以从它们的标志位明显看出来



```
33 6.515323   172.20.110.138   183.232.231.172   TCP   66 4221 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
34 6.569991   183.232.231.172  172.20.110.138    TCP   66 443 → 4221 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1452 WS=32 SACK_PERM=1
35 6.570062   172.20.110.138   183.232.231.172   TCP   54 4221 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0
```

a) 包含 HTTP POST 命令的 TCP 报文段的序号是0x360b3afa，即第一个SYN报文段序号+1



```
[TCP Segment Len: 612]
Sequence number: 1    (relative sequence number)
```

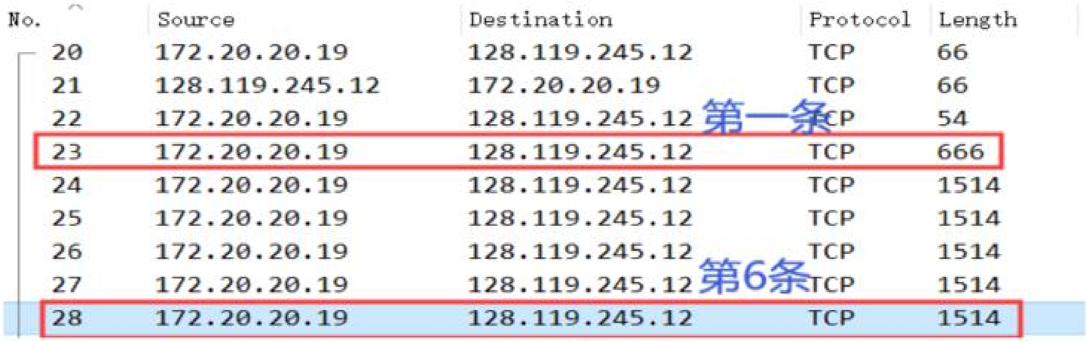
序号真实值可由字节数据看出

```
0020  f5 0c e9 ba 00 50 36 0b  3a fa 46 f6 eb 93 50 18   ·····P6· :·F···P·
0030  04 00 79 2d 00 00 50 4f  53 54 20 2f 77 69 72 65   ··y-··PO ST /wire
```

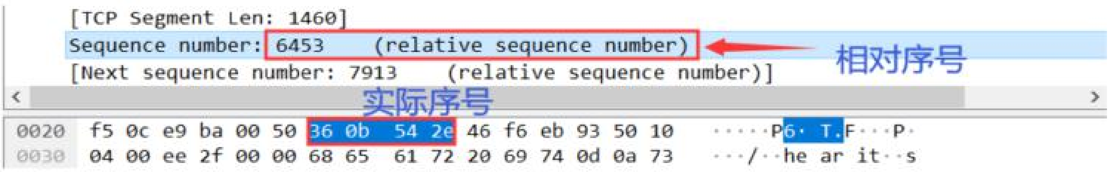
b) 如果将包含 HTTP POST 命令的 TCP 报文段看作是 TCP 连接上的第一个报文段，那么该 TCP 连接上的第六个报文段的序号是0x360b542e

发送时间为：三次握手建立TCP连接之后，四次握手断开连接之前

该报文段所对应的 ACK 确认号所对应的报文是：第三次握手时接收的

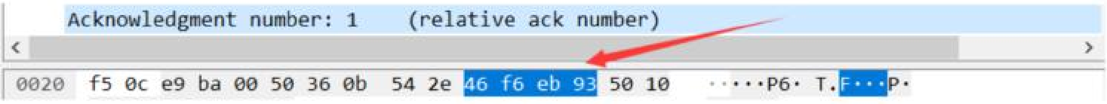


No.	Source	Destination	Protocol	Length
20	172.20.20.19	128.119.245.12	TCP	66
21	128.119.245.12	172.20.20.19	TCP	66
22	172.20.20.19	128.119.245.12	TCP	54
23	172.20.20.19	128.119.245.12	TCP	666
24	172.20.20.19	128.119.245.12	TCP	1514
25	172.20.20.19	128.119.245.12	TCP	1514
26	172.20.20.19	128.119.245.12	TCP	1514
27	172.20.20.19	128.119.245.12	TCP	1514
28	172.20.20.19	128.119.245.12	TCP	1514



```
    [TCP Segment Len: 1460]
    Sequence number: 6453    (relative sequence number)
    [Next sequence number: 7913    (relative sequence number)]

0020  f5 0c e9 ba 00 50 36 0b  54 2e 46 f6 eb 93 50 10   ·····P6· T.F···P·
0030  04 00 ee 2f 00 00 68 65  61 72 20 69 74 0d 0a 73   ···/··he ar it··s
```

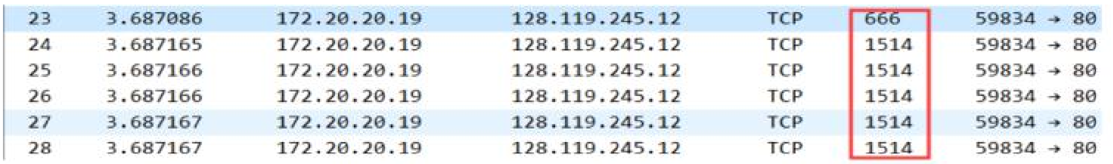
该报文段相对序号为 6453，实际为 0x36 0b 54 2e



```
    Acknowledgment number: 1    (relative ack number)

0020  f5 0c e9 ba 00 50 36 0b  54 2e 46 f6 eb 93 50 10   ·····P6· T.F···P·
```

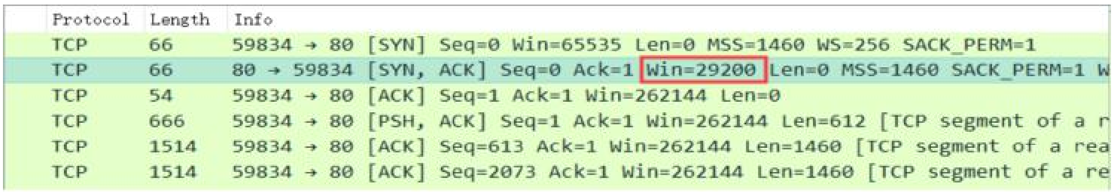
该报文段对应的 ACK 序号所标识的报文是在第三次握手时接收

c) 前六个 TCP 报文段的长度各是多少？



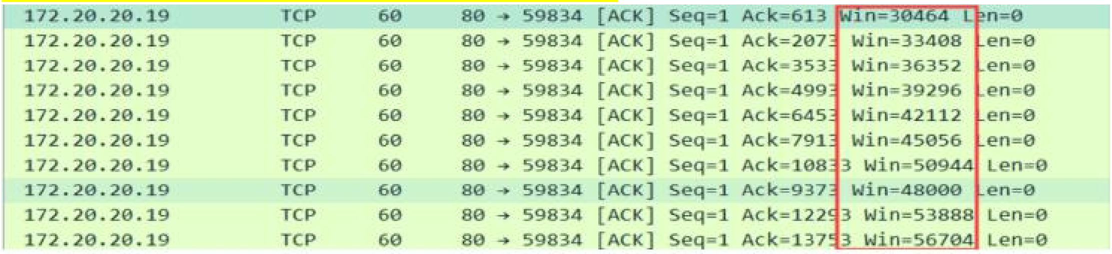
```
23   3.687086   172.20.20.19   128.119.245.12   TCP   666    59834 → 80
24   3.687165   172.20.20.19   128.119.245.12   TCP   1514   59834 → 80
25   3.687166   172.20.20.19   128.119.245.12   TCP   1514   59834 → 80
26   3.687166   172.20.20.19   128.119.245.12   TCP   1514   59834 → 80
27   3.687167   172.20.20.19   128.119.245.12   TCP   1514   59834 → 80
28   3.687167   172.20.20.19   128.119.245.12   TCP   1514   59834 → 80
```

d) 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：29200



```
Protocol  Length  Info
TCP       66      59834 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
TCP       66      80 → 59834 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 W
TCP       54      59834 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
TCP       666     59834 → 80 [PSH, ACK] Seq=1 Ack=1 Win=262144 Len=612 [TCP segment of a r
TCP       1514    59834 → 80 [ACK] Seq=613 Ack=1 Win=262144 Len=1460 [TCP segment of a rea
TCP       1514    59834 → 80 [ACK] Seq=2073 Ack=1 Win=262144 Len=1460 [TCP segment of a re
```

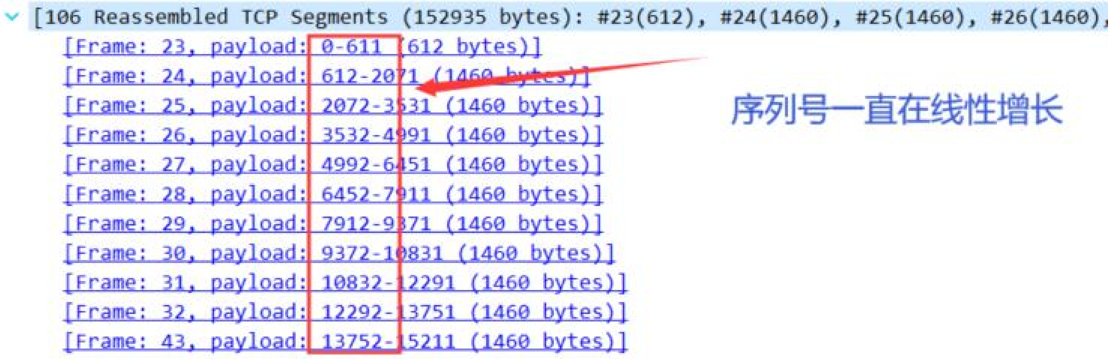
限制发送端的传输以后，接收端的缓存是否仍然不够用？：在限制发送方的传输之后，接收端的缓存空间大小开始不断增大，说明空间够用。



```
172.20.20.19   TCP   60   80 → 59834 [ACK] Seq=1 Ack=613 Win=30464 Len=0
172.20.20.19   TCP   60   80 → 59834 [ACK] Seq=1 Ack=2073 Win=33408 Len=0
172.20.20.19   TCP   60   80 → 59834 [ACK] Seq=1 Ack=3533 Win=36352 Len=0
172.20.20.19   TCP   60   80 → 59834 [ACK] Seq=1 Ack=4993 Win=39296 Len=0
172.20.20.19   TCP   60   80 → 59834 [ACK] Seq=1 Ack=6453 Win=42112 Len=0
172.20.20.19   TCP   60   80 → 59834 [ACK] Seq=1 Ack=7913 Win=45056 Len=0
172.20.20.19   TCP   60   80 → 59834 [ACK] Seq=1 Ack=10833 Win=50944 Len=0
172.20.20.19   TCP   60   80 → 59834 [ACK] Seq=1 Ack=9373 Win=48000 Len=0
172.20.20.19   TCP   60   80 → 59834 [ACK] Seq=1 Ack=12293 Win=53888 Len=0
172.20.20.19   TCP   60   80 → 59834 [ACK] Seq=1 Ack=13753 Win=56704 Len=0
```

e) 在跟踪文件中是否有重传的报文段？：不存在

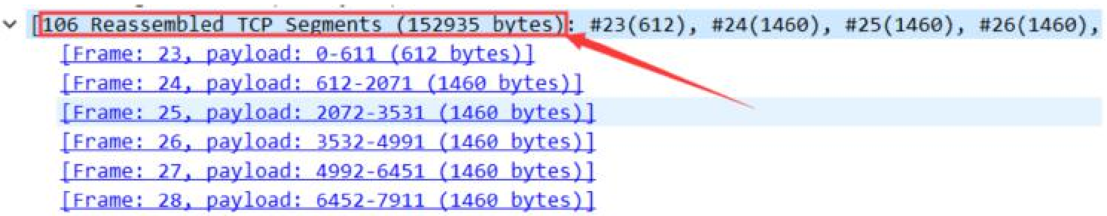
进行判断的依据是什么？：序列号一直在严格递增



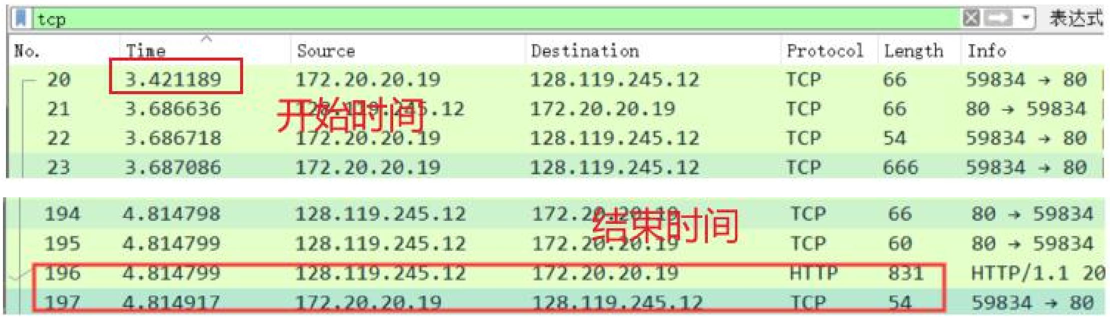
```
∨ [106 Reassembled TCP Segments (152935 bytes): #23(612), #24(1460), #25(1460), #26(1460),
      [Frame: 23, payload: 0-611 (612 bytes)]
      [Frame: 24, payload: 612-2071 (1460 bytes)]
      [Frame: 25, payload: 2072-3531 (1460 bytes)]
      [Frame: 26, payload: 3532-4991 (1460 bytes)]
      [Frame: 27, payload: 4992-6451 (1460 bytes)]
      [Frame: 28, payload: 6452-7911 (1460 bytes)]
      [Frame: 29, payload: 7912-9371 (1460 bytes)]
      [Frame: 30, payload: 9372-10831 (1460 bytes)]
      [Frame: 31, payload: 10832-12291 (1460 bytes)]
      [Frame: 32, payload: 12292-13751 (1460 bytes)]
      [Frame: 43, payload: 13752-15211 (1460 bytes)]
```

f) TCP 连接的 throughput (bytes transferred per unit time)是多少？：109730.9Bps

请写出你的计算过程：总字节数/总时间=152935/（4.814917-3.421189）=109730.9Bps



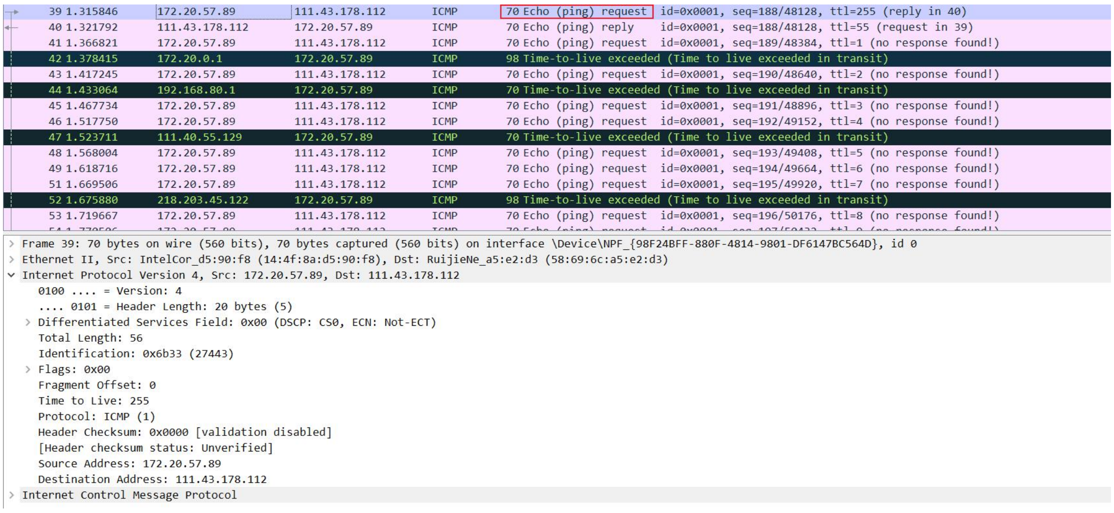
```
∨ [106 Reassembled TCP Segments (152935 bytes): #23(612), #24(1460), #25(1460), #26(1460),
      [Frame: 23, payload: 0-611 (612 bytes)]
      [Frame: 24, payload: 612-2071 (1460 bytes)]
      [Frame: 25, payload: 2072-3531 (1460 bytes)]
      [Frame: 26, payload: 3532-4991 (1460 bytes)]
      [Frame: 27, payload: 4992-6451 (1460 bytes)]
      [Frame: 28, payload: 6452-7911 (1460 bytes)]
```



```
tcp
No.    Time        Source            Destination       Protocol  Length  Info
 20    3.421189    172.20.20.19      128.119.245.12    TCP       66      59834 → 80
 21    3.686636    128.119.245.12    172.20.20.19      TCP       66      80 → 59834
 22    3.686718    172.20.20.19      128.119.245.12    TCP       54      59834 → 80
 23    3.687086    172.20.20.19      128.119.245.12    TCP       666     59834 → 80

194    4.814798    128.119.245.12    172.20.20.19      TCP       66      80 → 59834
195    4.814799    128.119.245.12    172.20.20.19      TCP       60      80 → 59834
196    4.814799    128.119.245.12    172.20.20.19      HTTP      831     HTTP/1.1 20
197    4.814917    172.20.20.19      128.119.245.12    TCP       54      59834 → 80
```

(四)IP分析

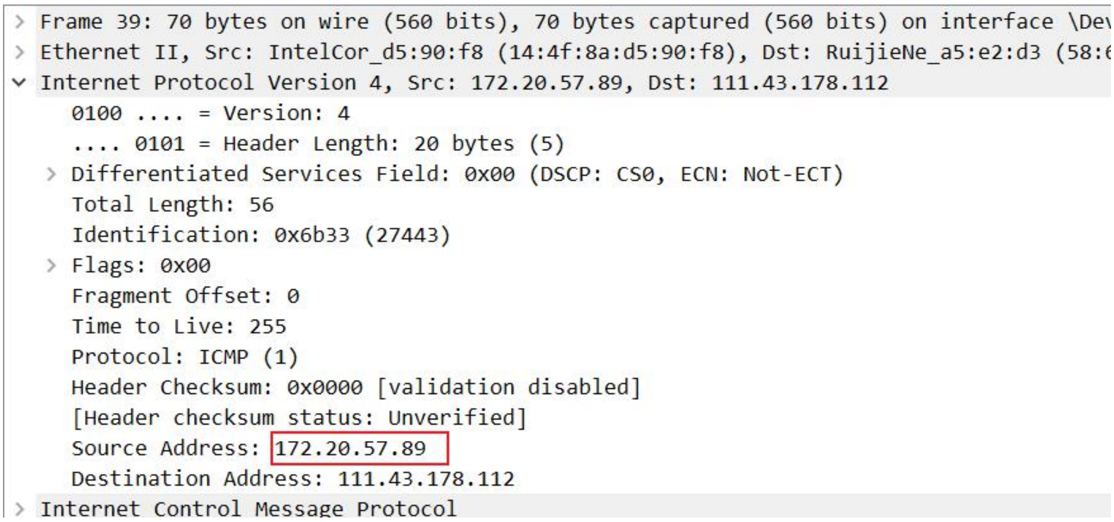
- 选择第一个主机发出的ICMP Echo Request消息



```
39 1.315846  172.20.57.89    111.43.178.112  ICMP  70 Echo (ping) request  id=0x0001, seq=188/48128, ttl=255 (reply in 40)
40 1.321792  111.43.178.112  172.20.57.89    ICMP  70 Echo (ping) reply    id=0x0001, seq=188/48128, ttl=55 (request in 39)
41 1.366821  172.20.57.89    111.43.178.112  ICMP  70 Echo (ping) request  id=0x0001, seq=189/48384, ttl=1 (no response found!)
42 1.378415  172.20.0.1      172.20.57.89    ICMP  98 Time-to-live exceeded (Time to live exceeded in transit)
43 1.417245  172.20.57.89    111.43.178.112  ICMP  70 Echo (ping) request  id=0x0001, seq=190/48640, ttl=2 (no response found!)
44 1.433064  192.168.80.1    172.20.57.89    ICMP  70 Time-to-live exceeded (Time to live exceeded in transit)
45 1.467734  172.20.57.89    111.43.178.112  ICMP  70 Echo (ping) request  id=0x0001, seq=191/48896, ttl=3 (no response found!)
46 1.517750  172.20.57.89    111.43.178.112  ICMP  70 Echo (ping) request  id=0x0001, seq=192/49152, ttl=4 (no response found!)
47 1.523711  111.40.55.129   172.20.57.89    ICMP  70 Time-to-live exceeded (Time to live exceeded in transit)
48 1.568004  172.20.57.89    111.43.178.112  ICMP  70 Echo (ping) request  id=0x0001, seq=193/49408, ttl=5 (no response found!)
49 1.618716  172.20.57.89    111.43.178.112  ICMP  70 Echo (ping) request  id=0x0001, seq=194/49664, ttl=6 (no response found!)
51 1.669506  172.20.57.89    111.43.178.112  ICMP  70 Echo (ping) request  id=0x0001, seq=195/49920, ttl=7 (no response found!)
52 1.675880  218.203.45.122  172.20.57.89    ICMP  98 Time-to-live exceeded (Time to live exceeded in transit)
53 1.719667  172.20.57.89    111.43.178.112  ICMP  70 Echo (ping) request  id=0x0001, seq=196/50176, ttl=8 (no response found!)
```

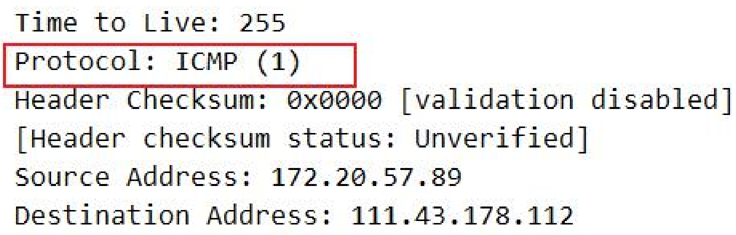
```
> Frame 39: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{98F24BFF-880F-4814-9801-DF6147BC564D}, id 0
> Ethernet II, Src: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8), Dst: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3)
∨ Internet Protocol Version 4, Src: 172.20.57.89, Dst: 111.43.178.112
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x6b33 (27443)
  > Flags: 0x00
    Fragment Offset: 0
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.57.89
    Destination Address: 111.43.178.112
> Internet Control Message Protocol
```

你主机的IP地址是什么：172.20.57.89



```
> Frame 39: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Dev
> Ethernet II, Src: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8), Dst: RuijieNe_a5:e2:d3 (58:6
∨ Internet Protocol Version 4, Src: 172.20.57.89, Dst: 111.43.178.112
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x6b33 (27443)
  > Flags: 0x00
    Fragment Offset: 0
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.57.89
    Destination Address: 111.43.178.112
> Internet Control Message Protocol
```

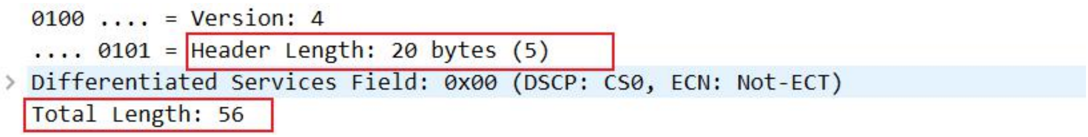
在IP数据包头中，上层协议（upper layer）字段的值是什么：1



```
Time to Live: 255
Protocol: ICMP (1)
Header Checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.20.57.89
Destination Address: 111.43.178.112
```

IP头有多少字节：20 该IP数据包的净载为多少字节：36

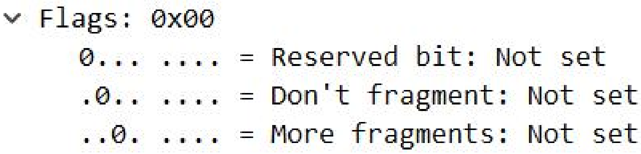
并解释你是怎样确定该IP数据包的净载大小的：总长度-头部长度



```
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
```

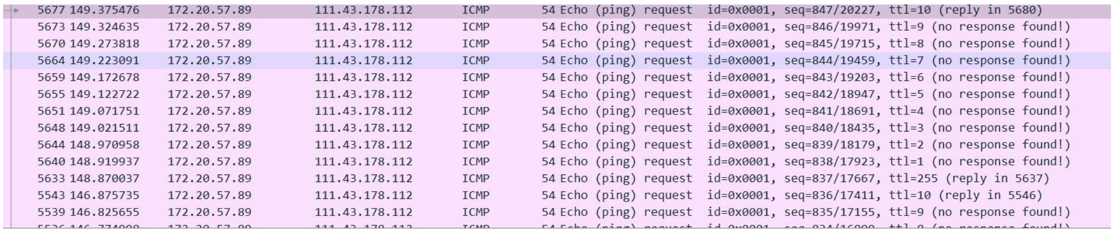
该IP数据包分片了吗：未

解释你是如何确定该P数据包是否进行了分片：标志位全0



```
v Flags: 0x00
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
```

- 分析源主机发出的一系列报文



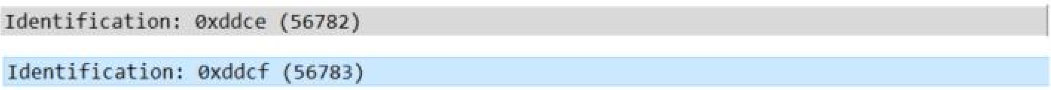
```
5677 149.375476   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=847/20227, ttl=10 (reply in 5680)
5673 149.324635   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=846/19971, ttl=9 (no response found!)
5670 149.273818   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=845/19715, ttl=8 (no response found!)
5664 149.223091   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=844/19459, ttl=7 (no response found!)
5659 149.172678   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=843/19203, ttl=6 (no response found!)
5655 149.122722   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=842/18947, ttl=5 (no response found!)
5651 149.071751   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=841/18691, ttl=4 (no response found!)
5648 149.021511   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=840/18435, ttl=3 (no response found!)
5644 148.970958   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=839/18179, ttl=2 (no response found!)
5640 148.919937   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=838/17923, ttl=1 (no response found!)
5633 148.870037   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=837/17667, ttl=255 (reply in 5637)
5543 146.875735   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=836/17411, ttl=10 (reply in 5546)
5539 146.825655   172.20.57.89   111.43.178.112   ICMP   54 Echo (ping) request  id=0x0001, seq=835/17155, ttl=9 (no response found!)
```

你主机发出的一系列ICMP消息中IP数据报中哪些字段总是发生改变：标志ID、TTL、checksum、数据域

哪些字段必须保持常量？哪些字段必须改变？为什么？

答：除了上述4个字段之外，其他字段必须保持常量。原因是：表示ID唯一、随之checksum也不同、TTL在变大（由于ICMP的ping检测），而且数据域中封装有ICMP的报文，故数据域也在变化

描述你看到的IP数据包Identification字段值的形式。

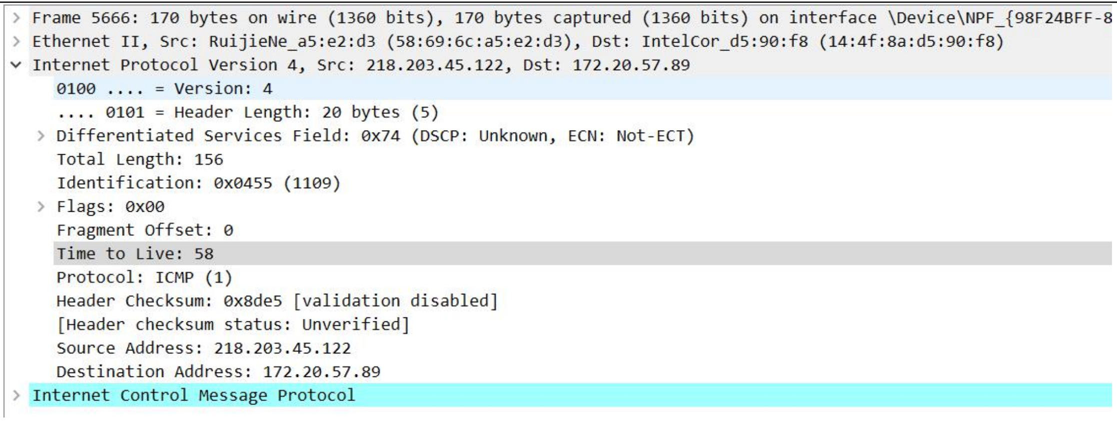


```
Identification: 0xddce (56782)

Identification: 0xddcf (56783)
```

答：16b数据，线性递增

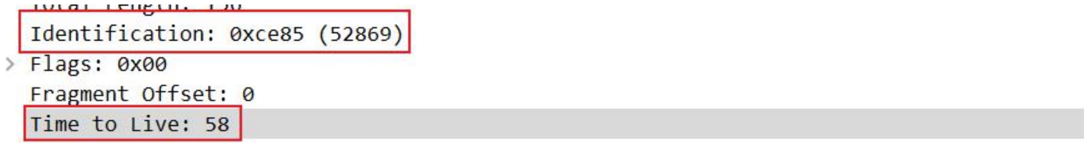
- 找到由最近的路由器（第一跳 ）返回给你主机的 ICMP Time-to-live exceeded消息。

Identification字段：1109 TTL字段的值是：58



```
> Frame 5666: 170 bytes on wire (1360 bits), 170 bytes captured (1360 bits) on interface \Device\NPF_{98F24BFF-8
> Ethernet II, Src: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3), Dst: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8)
v Internet Protocol Version 4, Src: 218.203.45.122, Dst: 172.20.57.89
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x74 (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 156
    Identification: 0x0455 (1109)
  > Flags: 0x00
    Fragment Offset: 0
    Time to Live: 58
    Protocol: ICMP (1)
    Header Checksum: 0x8de5 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 218.203.45.122
    Destination Address: 172.20.57.89
> Internet Control Message Protocol
```

最近的路由器（第一跳 ）返回给你主机的ICMP Time-to-live exceeded消息中这些值是否保持不变？为什么？



```
    Identification: 0xce85 (52869)
  > Flags: 0x00
    Fragment Offset: 0
    Time to Live: 58
```

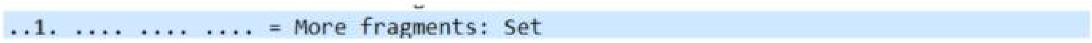
ID变化，但是TTL不变。因为：第一跳路由器设置TTL字段为REC指定的值，故不变；但是ID唯一标识一个IP数据报，故变化。

- 找到在将包大小改为2000字节后你的主机发送的第一个ICMP Echo Request消息。

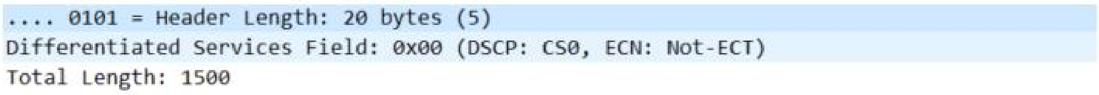
该消息是否被分解成不止一个IP数据报：被分解为2片

观察第一个IP分片， IP头部的哪些信息表明数据包被进行了分片？ IP头部的哪些信息表明数据包是第一个而不是最后一个分片？该分片的长度是多少

答：第一个分片中的标志位MF被置为1，说明后面还有分片该分片数据长度为1480B，IP总长度为1500B



```
..1. .... .... .... = More fragments: Set
```



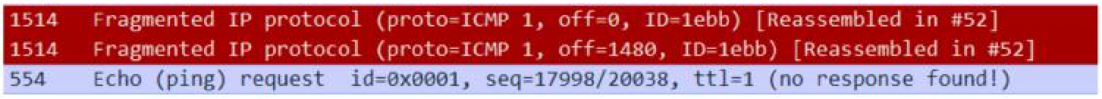
```
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1500
```

- 找到在将包大小改为3500字节后你的主机发送的第一个ICMP Echo Request消息。

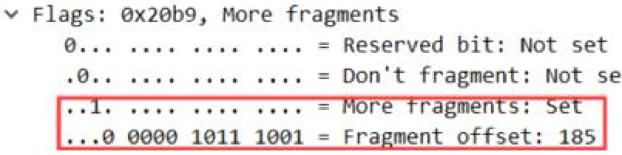
原始数据包被分成了多少片：3片

这些分片中IP数据报头部哪些字段发生了变化？

答：MF变化，片偏移变化，第一个和第二个分片的标志位MF=1，第一个分片的片偏移为0，第二个是185，第三个是370



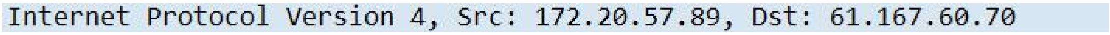
```
1514   Fragmented IP protocol (proto=ICMP 1, off=0, ID=1ebb) [Reassembled in #52]
1514   Fragmented IP protocol (proto=ICMP 1, off=1480, ID=1ebb) [Reassembled in #52]
554    Echo (ping) request  id=0x0001, seq=17998/20038, ttl=1 (no response found!)
```



```
v Flags: 0x20b9, More fragments
    0... .... .... .... = Reserved bit: Not set
    .0.. .... .... .... = Don't fragment: Not set
    ..1. .... .... .... = More fragments: Set
    ...0 0000 1011 1001 = Fragment offset: 185
```

(五)Ethernet数据帧分析

你的主机IP和目的主机IP是什么？

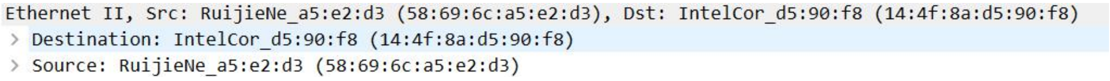


```
Internet Protocol Version 4, Src: 172.20.57.89, Dst: 61.167.60.70
```

我的主机发送的第一跳的HTTP请求报文的帧结构是什么？封装了上层哪些数据？

答：以太网帧封装了上层的IP数据报，IP封装了上层的TCP数据报，TCP数据包封装了上层的HTTP数据。

我的主机mac地址为和目的主机mac地址如下：



```
Ethernet II, Src: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3), Dst: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8)
 > Destination: IntelCor_d5:90:f8 (14:4f:8a:d5:90:f8)
 > Source: RuijieNe_a5:e2:d3 (58:69:6c:a5:e2:d3)
```

发送数据域长度范围：46B-1500B

计算过程：

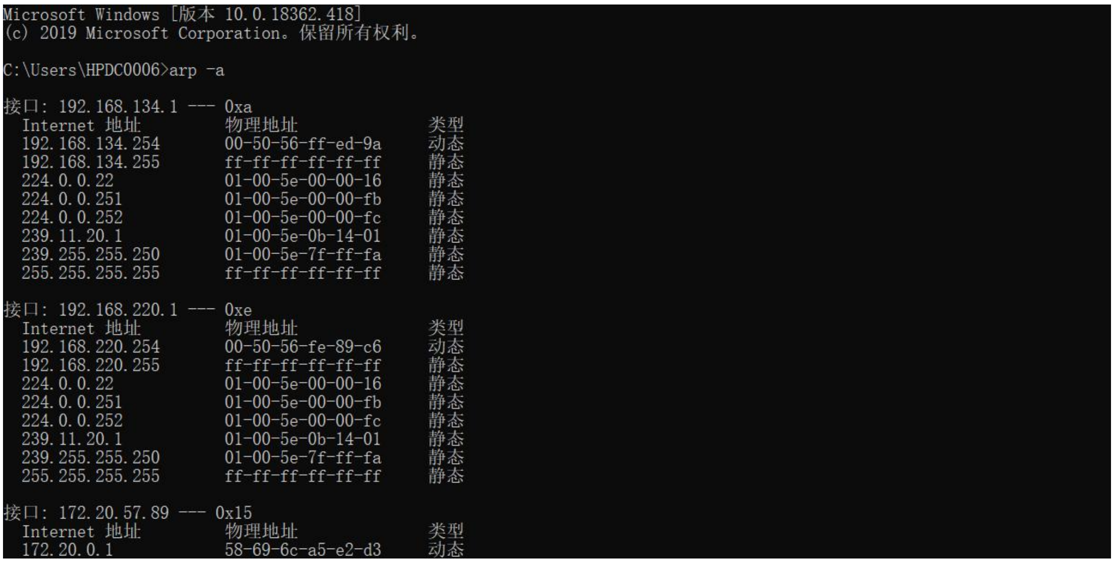
MTU=1500B，故数据域最长为1500B

R=10Mbps，RTTmax=512μs，Lmin / R = RTTmax

Lmin=512bits=64B，Datamin=Lmin-18=46B

(六)抓取ARP数据报

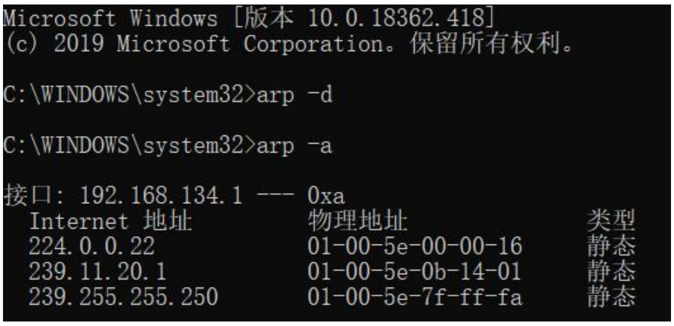
- 利用 MS-DOS 命令： arp 或 c:\windows\system32\arp 查看主机上 ARP 缓存的内容。说明 ARP 缓存中每一列的含义是什么?



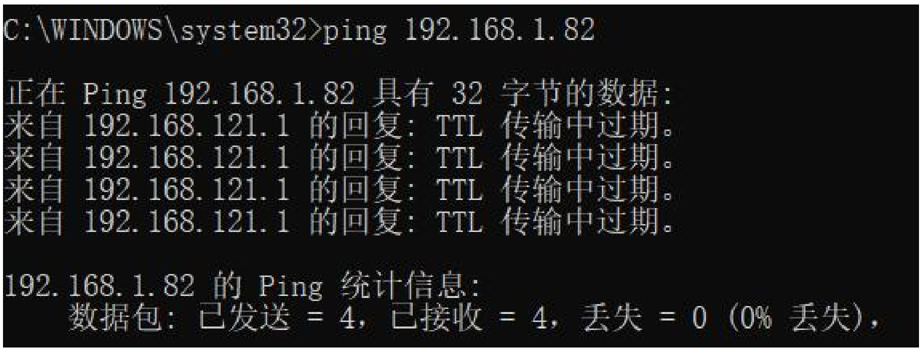
```
Microsoft Windows [版本 10.0.18362.418]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\HPDC0006>arp -a

接口: 192.168.134.1 --- 0xa
  Internet 地址         物理地址              类型
  192.168.134.254       00-50-56-ff-ed-9a     动态
  192.168.134.255       ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  239.11.20.1           01-00-5e-0b-14-01     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态

接口: 192.168.220.1 --- 0xe
  Internet 地址         物理地址              类型
  192.168.220.254       00-50-56-fe-89-c6     动态
  192.168.220.255       ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  239.11.20.1           01-00-5e-0b-14-01     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态

接口: 172.20.57.89 --- 0x15
  Internet 地址         物理地址              类型
  172.20.0.1            58-69-6c-a5-e2-d3     动态
```

第一列是ARP协议缓存的IP地址，第二列是对应的MAC地址，第三列是类型

- 清除主机上 ARP 缓存的内容,抓取 ping 命令时的数据包。分析数据包



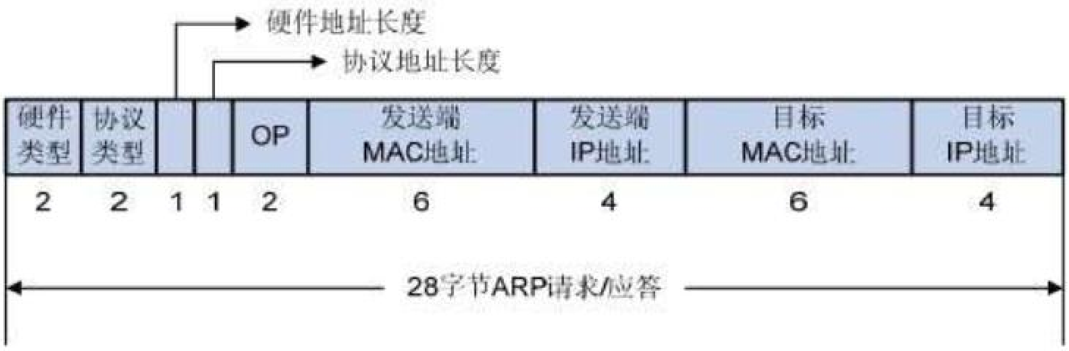
```
Microsoft Windows [版本 10.0.18362.418]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\WINDOWS\system32>arp -d

C:\WINDOWS\system32>arp -a

接口: 192.168.134.1 --- 0xa
  Internet 地址         物理地址              类型
  224.0.0.22            01-00-5e-00-00-16     静态
  239.11.20.1           01-00-5e-0b-14-01     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
```



```
C:\WINDOWS\system32>ping 192.168.1.82

正在 Ping 192.168.1.82 具有 32 字节的数据:
来自 192.168.121.1 的回复: TTL 传输中过期。
来自 192.168.121.1 的回复: TTL 传输中过期。
来自 192.168.121.1 的回复: TTL 传输中过期。
来自 192.168.121.1 的回复: TTL 传输中过期。

192.168.1.82 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
```

ARP数据包的格式是怎样的？由几部分构成，各个部分所占的字节数是多少？

如下图所示：



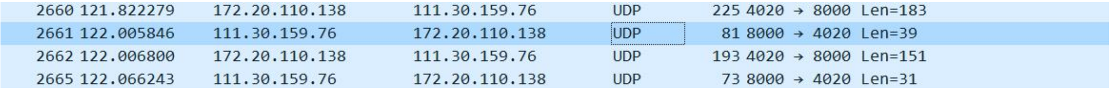
如何判断一个ARP数据是请求包还是应答包？

通过“OP”字段的值来判断：当其值为 0×0001 时是请求，为 0×0002 时是应答。

为什么ARP查询要在广播帧中传送，而ARP响应要在一个有着明确目的局域网地址的帧中传送？

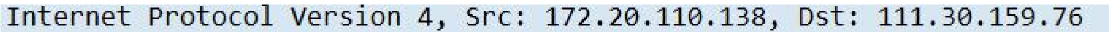
因为ARP查询不知道目的MAC地址，故需要广播；ARP在接收到的ARP查询中知道了源MAC地址，故在有明确目的局域网地址的帧中传输。

(七)抓取UDP数据报



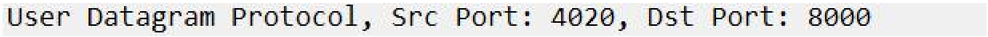
```
2660 121.822279   172.20.110.138   111.30.159.76    UDP    225 4020 → 8000 Len=183
2661 122.005846   111.30.159.76    172.20.110.138   UDP     81 8000 → 4020 Len=39
2662 122.006800   172.20.110.138   111.30.159.76    UDP    193 4020 → 8000 Len=151
2665 122.066243   111.30.159.76    172.20.110.138   UDP     73 8000 → 4020 Len=31
```

- 消息是基于UDP的还是TCP的：UDP
- 你的主机ip地址是什么？目的主机ip地址是什么？



```
Internet Protocol Version 4, Src: 172.20.110.138, Dst: 111.30.159.76
```

- 你的主机发送QQ消息的端口号和QQ服务器的端口号分别是多少？



```
User Datagram Protocol, Src Port: 4020, Dst Port: 8000
```

- 数据报的格式是什么样的？都包含哪些字段，分别占多少字节？

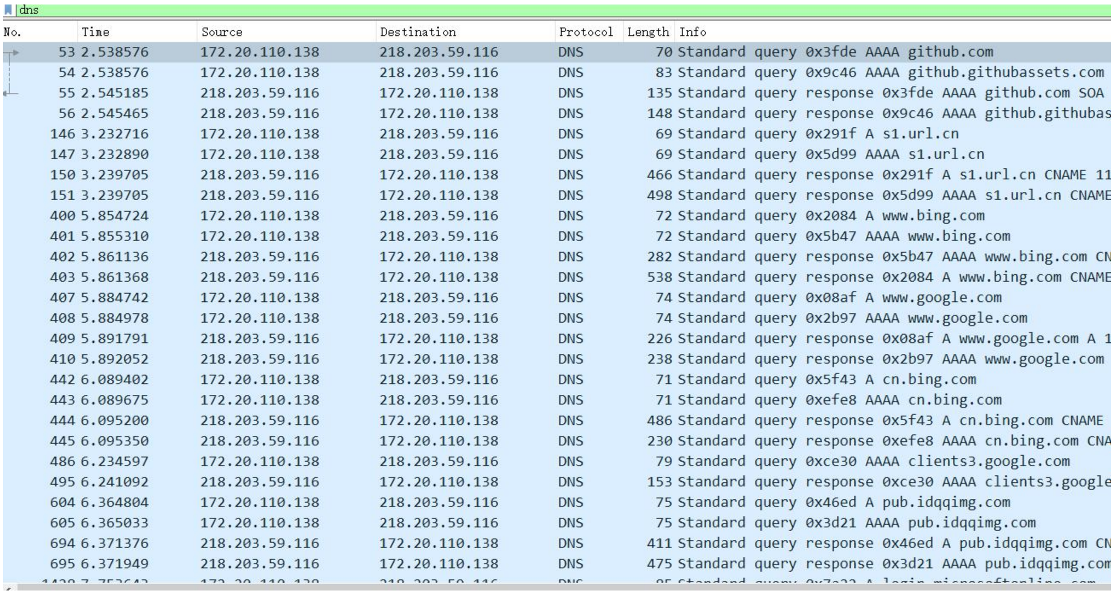
字段	源端口号	目的端口号	UDP 长度	UDP 校验和
长度	2B	2B	2B	2B

- 为什么你发送一个ICQ数据包后，服务器又返回给你的主机一个ICQ数据包？这UDP的不可靠数据传输有什么联系？对比前面的TCP协议分析，你能看出UDP是无连接的吗？

答：服务器返回ICQ数据包是为了确认收到，因为UDP是不可靠无连接传输，客户端无法确认服务器是否已经收到了数据，所以需要服务器返回ICQ报文

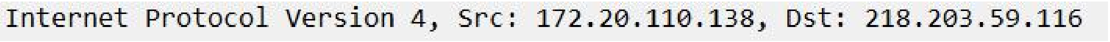
可以看出UDP是无连接的，因为UDP首部无标志位，也无序列号

(八)DNS协议分析



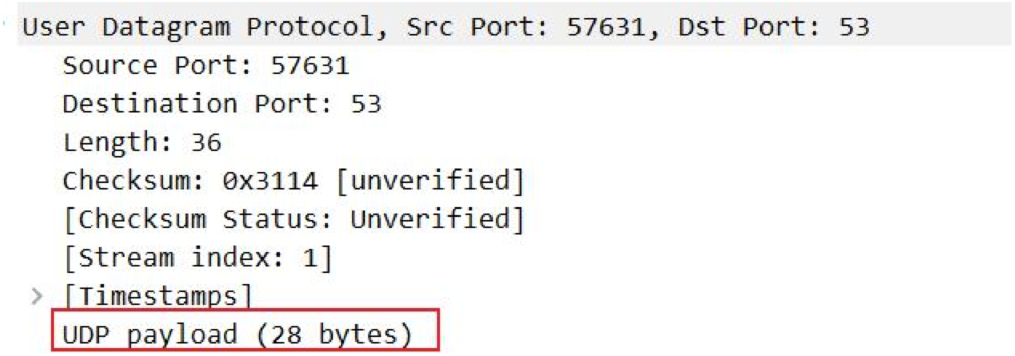
```
dns
No.      Time        Source           Destination      Protocol  Length Info
      53 2.538576    172.20.110.138   218.203.59.116   DNS          70 Standard query 0x3fde AAAA github.com
      54 2.538576    172.20.110.138   218.203.59.116   DNS          83 Standard query 0x9c46 AAAA github.githubassets.com
      55 2.545185    218.203.59.116   172.20.110.138   DNS         135 Standard query response 0x3fde AAAA github.com SOA
      56 2.545465    218.203.59.116   172.20.110.138   DNS         148 Standard query response 0x9c46 AAAA github.githubas
     146 3.232716    172.20.110.138   218.203.59.116   DNS          69 Standard query 0x291f A s1.url.cn
     147 3.232890    172.20.110.138   218.203.59.116   DNS          69 Standard query 0x5d99 AAAA s1.url.cn
     150 3.239705    218.203.59.116   172.20.110.138   DNS         466 Standard query response 0x291f A s1.url.cn CNAME 11
     151 3.239705    218.203.59.116   172.20.110.138   DNS         498 Standard query response 0x5d99 AAAA s1.url.cn CNAME
     400 5.854724    172.20.110.138   218.203.59.116   DNS          72 Standard query 0x2084 A www.bing.com
     401 5.855310    172.20.110.138   218.203.59.116   DNS          72 Standard query 0x5b47 AAAA www.bing.com
     402 5.861136    218.203.59.116   172.20.110.138   DNS         282 Standard query response 0x5b47 AAAA www.bing.com CN
     403 5.861368    218.203.59.116   172.20.110.138   DNS         538 Standard query response 0x2084 A www.bing.com CNAME
     407 5.884742    172.20.110.138   218.203.59.116   DNS          74 Standard query 0x08af A www.google.com
     408 5.884978    172.20.110.138   218.203.59.116   DNS          74 Standard query 0x2b97 AAAA www.google.com
     409 5.891791    218.203.59.116   172.20.110.138   DNS         226 Standard query response 0x08af A www.google.com A 1
     410 5.892052    218.203.59.116   172.20.110.138   DNS         238 Standard query response 0x2b97 AAAA www.google.com
     442 6.089402    172.20.110.138   218.203.59.116   DNS          71 Standard query 0x5f43 A cn.bing.com
     443 6.089675    172.20.110.138   218.203.59.116   DNS          71 Standard query 0xefe8 AAAA cn.bing.com
     444 6.095200    218.203.59.116   172.20.110.138   DNS         486 Standard query response 0x5f43 A cn.bing.com CNAME
     445 6.095350    218.203.59.116   172.20.110.138   DNS         230 Standard query response 0xefe8 AAAA cn.bing.com CNA
     486 6.234597    172.20.110.138   218.203.59.116   DNS          79 Standard query 0xce30 AAAA clients3.google.com
     495 6.241092    218.203.59.116   172.20.110.138   DNS         153 Standard query response 0xce30 AAAA clients3.google
     604 6.364804    172.20.110.138   218.203.59.116   DNS          75 Standard query 0x46ed A pub.idqqimg.com
     605 6.365033    172.20.110.138   218.203.59.116   DNS          75 Standard query 0x3d21 AAAA pub.idqqimg.com
     694 6.371376    218.203.59.116   172.20.110.138   DNS         411 Standard query response 0x46ed A pub.idqqimg.com CN
     695 6.371949    218.203.59.116   172.20.110.138   DNS         475 Standard query response 0x3d21 AAAA pub.idqqimg.com
```

- 主机IP和目的IP



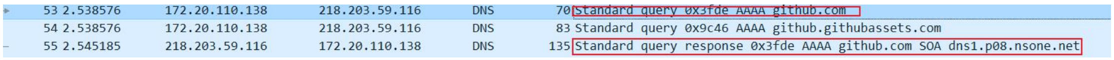
```
Internet Protocol Version 4, Src: 172.20.110.138, Dst: 218.203.59.116
```

- DNS下层协议为：UDP

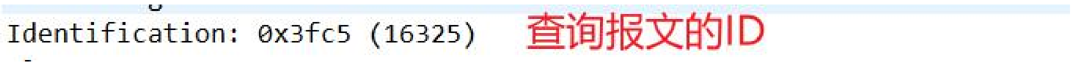


```
User Datagram Protocol, Src Port: 57631, Dst Port: 53
    Source Port: 57631
    Destination Port: 53
    Length: 36
    Checksum: 0x3114 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
  > [Timestamps]
    UDP payload (28 bytes)
```

- DNS的一对查询与响应报文的ID相同

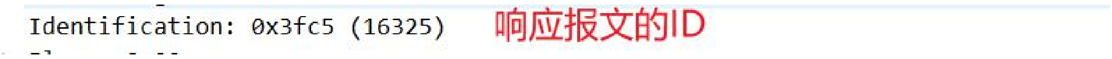


```
53 2.538576   172.20.110.138   218.203.59.116   DNS    70 Standard query 0x3fde AAAA github.com
54 2.538576   172.20.110.138   218.203.59.116   DNS    83 Standard query 0x9c46 AAAA github.githubassets.com
55 2.545185   218.203.59.116   172.20.110.138   DNS   135 Standard query response 0x3fde AAAA github.com SOA dns1.p08.nsone.net
```



```
Identification: 0x3fc5 (16325)
```

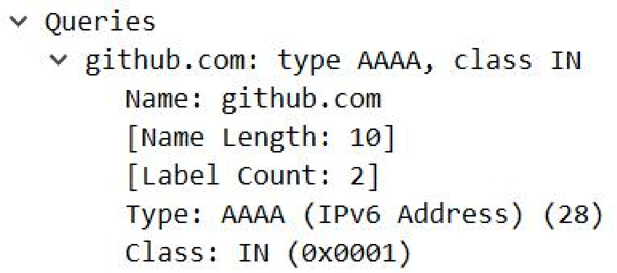
查询报文的ID



```
Identification: 0x3fc5 (16325)
```

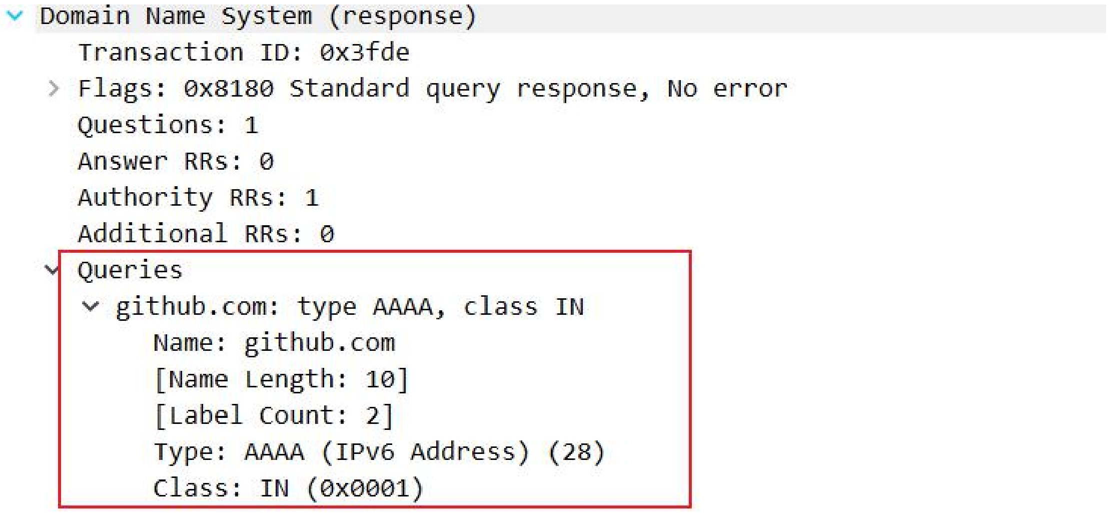
响应报文的ID

- 请求内容如下，包含“Name”“Type”“Class”等字段



```
Queries
    github.com: type AAAA, class IN
        Name: github.com
        [Name Length: 10]
        [Label Count: 2]
        Type: AAAA (IPv6 Address) (28)
        Class: IN (0x0001)
```

- 响应报文同样包含上述字段



```
Domain Name System (response)
    Transaction ID: 0x3fde
  Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 0
    Authority RRs: 1
    Additional RRs: 0
  Queries
    github.com: type AAAA, class IN
        Name: github.com
        [Name Length: 10]
        [Label Count: 2]
        Type: AAAA (IPv6 Address) (28)
        Class: IN (0x0001)
```

问题讨论：

全部的思考问题已经在前面的实验过程中给出答案，故此处不再赘述。

心得体会：

1. 掌握了Wireshark的基本使用方法，知道如何利用它来进行抓包分析
2. 通过对各层数据报文的结构分析，我对计算机网络的分层协议有了更深入的理解

其实就是通过各层协议将上层的数据分组加上本协议的头部字段和尾部字段，从而转入下一层进行传输，当接受方收到相应的数据报时，按照协议进行解析即可。

- 应用层：支持各种网络应用
- 传输层：TCP、UDP等进程间通信协议
- 网络层：通过路由存储转发将数据分组从源主机送到目的主机，如IP协议
- 链路层：进行相邻网络设备之间的数据传输，如：Ethernet