哈爾濱工業大學
Harbin Institute of Technology

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	朱宸慷	院系	计算机科学与技术		
班级	2103103	学号	2021110908		
任课教师	聂兰顺	指导教师	聂兰顺		
实验地点	格物 207	实验时间	11.4		
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					

哈尔滨工业大学计算学部
FACULTY OF COMPUTING, HIT

实验目的：

1. 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：

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

选做内容：

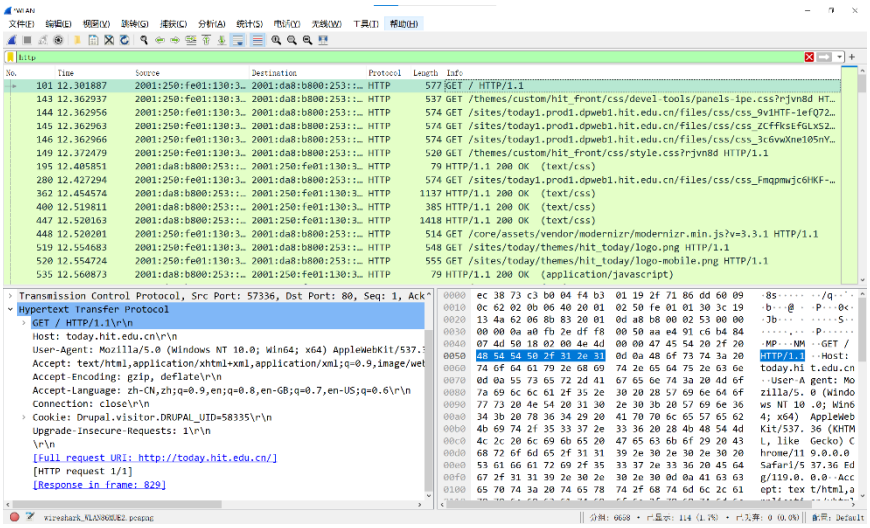
a) 利用 Wireshark 分析 DNS 协议
b) 利用 Wireshark 分析 UDP 协议
c) 利用 Wireshark 分析 ARP 协议

实验过程：

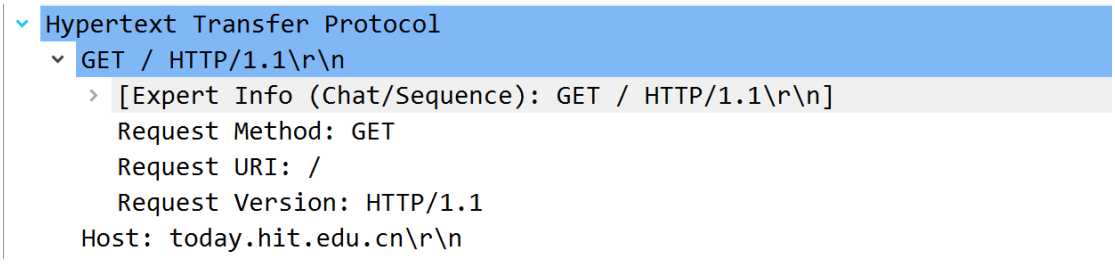
一、 HTTP分析

1、 HTTP GET/response 交互

打开网址http://today.hit.edu.cn/，筛选http报文，在wireshark界面中显示的结果如下图所示：



此时左下角已经展示了报文的部分内容，可以看到本地给出的GET报文指示了浏览器上运行的是HTTP1.1协议



此时再另外寻找一个服务器返回的报文，可以看到服务器上运行的也是HTTP1.1协议，除

此之外，服务器也返回了一个200状态码，意为OK，表示正确接收了请求报文。

```
Hypertext Transfer Protocol
  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
  Content-Type: text/css\r\n
  Transfer-Encoding: chunked\r\n
  Connection: close\r\n
```

此时我们重新返回刚才的浏览器GET报文，可以看到有一段报文内容如下图所示，其中Accept-Language字段的内容是zh-CN，这指示着服务器希望接收zh-CN语言，也就是简体中文内容。

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/wel
Accept-Encoding: gzip, deflate\r\n
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6\r\n
Connection: close\r\n
```

查看对应的IP报文，可以发现使用的是IPv6协议，此时的本机IP地址是2001:250:fe01:130:3c19:134a:6206:8b83，目的IP地址是2001:da8:b800:253::aa0:fb2e

```
Internet Protocol Version 6, Src: 2001:250:fe01:130:3c19:134a:6206:8b83,
  0110 .... = Version: 6
  .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: C
    .... 0000 00.. .... .... .... .... .... = Differentiated Services Co
    .... .... ..00 .... .... .... .... .... = Explicit Congestion Notifi
  .... 1001 0000 1100 0110 0010 = Flow Label: 0x90c62
  Payload Length: 523
  Next Header: TCP (6)
  Hop Limit: 64
  Source Address: 2001:250:fe01:130:3c19:134a:6206:8b83
  Destination Address: 2001:da8:b800:253::aa0:fb2e
```

2、HTTP 条件 GET/response 交互

查看第一个对于 http://today.hit.edu.cn/ 的GET请求报文，可以看到报文中没有IF-MODIFIED-SINCE，这意味着此时浏览器中没有对于该网站的缓存内容，浏览器将会直接向服务器请求网页的内容并且将内容缓存在本地中。

```
GET / HTTP/1.1\r\n
  [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
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/av
Accept-Encoding: gzip, deflate\r\n
Accept-Language: zh-CN,zh;q=0.9\r\n
Connection: close\r\n
Upgrade-Insecure-Requests: 1\r\n
\r\n
```

观察一个最近的服务器响应报文，可以看到服务器指明了将要返回的内文件类型是text/css

```
[4 Reassembled TCP Segments (3253 bytes): #249(1366), #257(1366), #259(51
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    [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
  Content-Type: text/css\r\n
  Transfer-Encoding: chunked\r\n
  Connection: close\r\n
  Date: Wed, 15 Nov 2023 05:04:15 GMT\r\n
  Server: Server\r\n
  X-Content-Type-Options: nosniff\r\n
  X-Frame-Options: SAMEORIGIN\r\n
  Vary: Accept-encoding\r\n
  Last-Modified: Fri, 11 Dec 2020 14:59:56 GMT\r\n
```

并且在服务器响应报文的末尾，指明了文件的大小和部分内容，如下图所示，文件大小为2832字节。

```
File Data: 2832 bytes
Line-based text data: text/css (2 lines)
  [truncated].sidebar-preview{top:auto;bottom:0;left:0;display:block;pac
  [truncated].file{display:inline-block;min-height:16px;padding-left:20
```

在多次重启浏览器后，可以观察到此时出现了一条具有字段IF-MODIFIED-SINCE的报文，内容如下图所示，意为向服务器询问文件是否在 Wed,15 Nov 2023 03:09:24 GMT时间之后修改过。这个时间应该是浏览器缓存该内容时的时间。

```
> Frame 354: 722 bytes on wire (5776 bits), 722 bytes captured (5776 bits) or
> Ethernet II, Src: IntelCor_19:2f:71 (f4:b3:01:19:2f:71), Dst: JuniperN_c3:b
> Internet Protocol Version 6, Src: 2001:250:fe01:130:f1fa:9a72:46c4:fd76, Ds
> Transmission Control Protocol, Src Port: 62549, Dst Port: 80, Seq: 1, Ack:
v Hypertext Transfer Protocol
  > GET /sites/today1.prod1.dpweb1.hit.edu.cn/files/styles/355x220/public/in
    Host: today.hit.edu.cn\r\n
    Connection: keep-alive\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
    Accept: image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8\r\n
    Referer: http://today.hit.edu.cn/\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6\r\n
    If-None-Match: "220a2-60a283af592ff"\r\n
    If-Modified-Since: Wed, 15 Nov 2023 03:09:24 GMT\r\n
    \r\n
    [Full request URI: http://today.hit.edu.cn/sites/today1.prod1.dpweb1.hit
    [HTTP request 1/2]
    [Response in frame: 373]
    [Next request in frame: 544]
```

此时观察服务器的一个响应报文，可以看到服务器给出304 Not Modified状态码并且在Last-Modified字段中给出了与上述请求报文相同的时间，是对于该请求报文的响应。

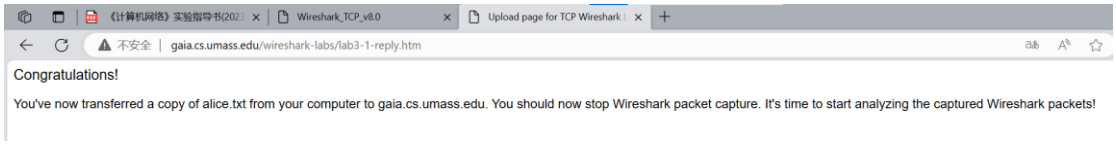
```
v Hypertext Transfer Protocol
  > HTTP/1.1 304 Not Modified\r\n
    Content-Type: image/png\r\n
    Connection: keep-alive\r\n
    Date: Wed, 15 Nov 2023 05:35:17 GMT\r\n
    Server: Server\r\n
    X-Content-Type-Options: nosniff\r\n
    X-Frame-Options: SAMEORIGIN\r\n
    Last-Modified: Wed, 15 Nov 2023 03:09:24 GMT\r\n
```

此时服务器的响应报文没有给出文件的内容，而是在报文的末尾声明了该资源的位置，并且在ETAG字段中指明了对于内容的校验。

```
    ETag: "220a2-60a283af592ff"\r\n
    X-Varnish: 613449894 621674542\r\n
    Age: 36\r\n
    Via: 1.1 varnish-v4\r\n
    X-Varnish-Cache: HIT\r\n
    \r\n
    [HTTP response 1/2]
    [Time since request: 0.012364000 seconds]
    [Request in frame: 354]
    [Next request in frame: 544]
    [Next response in frame: 561]
    [Request URI: http://today.hit.edu.cn/sites/today1.prod1.dpweb1.hit.edu
```

二、TCP分析

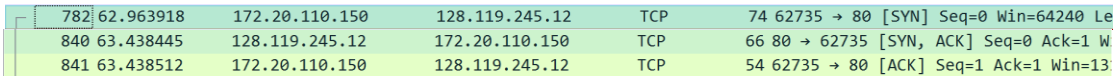
指导书中的网页地址不能正常访问，这里使用教材提供的http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html进行实验。



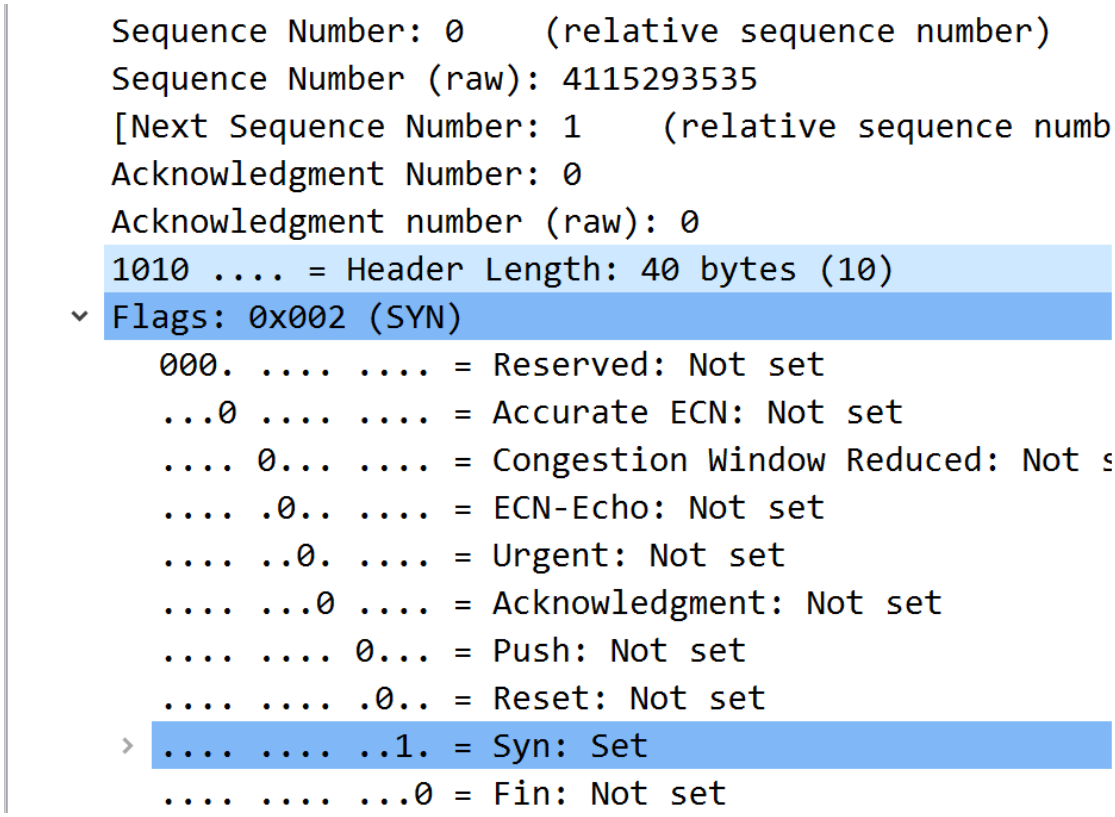
此时上传成功后，停止Wireshark俘获

（1）

寻找到最开始握手的报文，可以看到，TCP请求的源IP地址是172.20.110.150，目的IP地址是128.119.245.12，也就是说，客户端的地址是172.20.110.150,而上传文件的服务器的IP地址是128.119.245.12。除此之外，还可以看到客户端中TCP的端口是62735，而服务器的端口号是80.



客户服务器之间用于初始化TCP连接的TCP SYN报文段的序号是0，为了表示这是一个SYN报文段，它会将Flags中的Syn位设置为1来进行标识。



（2）

在服务器向客户端返回的SYN ACK握手报文中，它需要将报文序号设置为0，在此报文中，Acknowledgement字段的值被设置为1，这是因为服务器期望获得之前报文的后续报文段，这个

值就是先前SYN报文中所指的序号0加1所得到的。因此服务器将ACK和SYN都设置为1来标识这是一个SYN ACK报文。

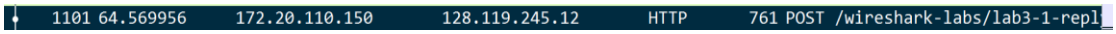
```
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 784851569
[Next Sequence Number: 1    (relative sequence numb
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 4115293536
0110 .... = Header Length: 24 bytes (6)
Flags: 0x012 (SYN, ACK)
   000. .... .... = Reserved: Not set
   ...0 .... .... = Accurate ECN: Not set
   .... 0... .... = Congestion Window Reduced: Not s
   .... .0.. .... = ECN-Echo: Not set
   .... ..0. .... = Urgent: Not set
   .... ...1 .... = Acknowledgment: Set
   .... .... 0... = Push: Not set
   .... .... .0.. = Reset: Not set
   .... .... ..1. = Syn: Set
   .... .... ...0 = Fin: Not set
   [TCP Flags: ·····A··S·]
```

TCP的三次握手即下图中的782，840，841号报文，可以看到这三个报文分别是SYN, ACK SYN和 ACK报文。此时的客户端IP是172.20.110.150，服务端IP是128.119.245.12，端口为62735和80

782	62.963918	172.20.110.150	128.119.245.12	TCP	74 62735 → 80 [SYN] Seq=0 Win=64240 Le
840	63.438445	128.119.245.12	172.20.110.150	TCP	66 80 → 62735 [SYN, ACK] Seq=0 Ack=1 W
841	63.438512	172.20.110.150	128.119.245.12	TCP	54 62735 → 80 [ACK] Seq=1 Ack=1 Win=13

（3）

观察可以找到HTTP POST报文的序号为152358




```
Transmission Control Protocol, Src Port: 62735, Dst Pc
    Source Port: 62735
    Destination Port: 80
    [Stream index: 59]
    [Conversation completeness: Complete, WITH_DATA (31
    [TCP Segment Len: 707]
    Sequence Number: 152358    (relative sequence numbe
    Sequence Number (raw): 3286766827
    [Next Sequence Number: 153065    (relative sequence
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1267948082
```

查看该报文末尾的标记，可以发现共计发送了112个数据帧，而第六个帧的帧序号应该是847。

```
    [Frame: 1096, payload: 145675-147060 (1386 bytes)]
    [Frame: 1097, payload: 147061-148198 (1138 bytes)]
    [Frame: 1098, payload: 148199-149584 (1386 bytes)]
    [Frame: 1099, payload: 149585-150970 (1386 bytes)]
    [Frame: 1100, payload: 150971-152356 (1386 bytes)]
    [Frame: 1101, payload: 152357-153063 (707 bytes)]
    [Segment count: 112]
    [Reassembled TCP length: 153064]
    [Reassembled TCP Data: 504f5354202f7769726573686172
```

```
[112 Reassembled TCP Segments (153064 bytes): #842(743
    [Frame: 842, payload: 0-742 (743 bytes)]
    [Frame: 843, payload: 743-2128 (1386 bytes)]
    [Frame: 844, payload: 2129-3514 (1386 bytes)]
    [Frame: 845, payload: 3515-4900 (1386 bytes)]
    [Frame: 846, payload: 4901-6286 (1386 bytes)]
    [Frame: 847, payload: 6287-7672 (1386 bytes)]
```

寻找到该报文对应的内容，可以发现该报文的TCP序号是6288，这个报文在统计开始的63.438950时间，和其余几个报文一起发送，具体时间不能确定，但是在POST之前。ACK是捎带确认的，是服务器的第六个ACK。

```
Source Port: 62735
Destination Port: 80
[Stream index: 59]
[Conversation completeness: Complete, WITH_DATA (31
[TCP Segment Len: 1386]
Sequence Number: 6288    (relative sequence number)
Sequence Number (raw): 3286620757
[Next Sequence Number: 7674    (relative sequence n
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 1267948082
0101 .... = Header Length: 20 bytes (5)
```

观察报文的信息，可以发现前六个报文的长度分别为743，1386，1386，1386，1386，1386.

```
172.20.110.150   TCP     66 80 → 62735 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1386 SACK_PERM …
128.119.245.12   TCP     54 62735 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=0
128.119.245.12   TCP    797 62735 → 80 [PSH, ACK] Seq=1 Ack=1 Win=131584 Len=743 [TCP segment of …
128.119.245.12   TCP   1440 62735 → 80 [ACK] Seq=744 Ack=1 Win=131584 Len=1386 [TCP segment of a …
128.119.245.12   TCP   1440 62735 → 80 [ACK] Seq=2130 Ack=1 Win=131584 Len=1386 [TCP segment of a…
128.119.245.12   TCP   1440 62735 → 80 [ACK] Seq=3516 Ack=1 Win=131584 Len=1386 [TCP segment of a…
128.119.245.12   TCP   1440 62735 → 80 [ACK] Seq=4902 Ack=1 Win=131584 Len=1386 [TCP segment of a…
128.119.245.12   TCP   1440 62735 → 80 [ACK] Seq=6288 Ack=1 Win=131584 Len=1386 [TCP segment of a…
128.119.245.12   TCP   1440 62735 → 80 [ACK] Seq=7674 Ack=1 Win=131584 Len=1386 [TCP segment of a…
```

在跟踪过程中，可以看到接收端公示的最小的可用缓存空间是29200。在之后的报文信息中可以发现接收端的可用缓存空间一直维持在一个较大的数字131584。

因此在后续接收的时候，接收端会不断增加窗口大小，并维持可用空间，所以接收端缓存其实是够用的。

```
245.12    172.20.110.150   TCP     66 80 → 62735 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1386 SACK_
10.150    128.119.245.12   TCP     54 62735 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=0
10.150    128.119.245.12   TCP    797 62735 → 80 [PSH, ACK] Seq=1 Ack=1 Win=131584 Len=743 [TCP segmer
10.150    128.119.245.12   TCP   1440 62735 → 80 [ACK] Seq=744 Ack=1 Win=131584 Len=1386 [TCP segment
```

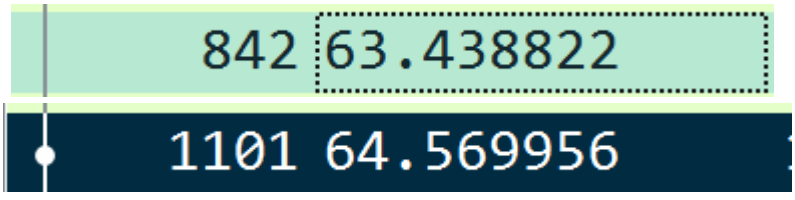
观察发送的报文，可以看到直到出现HTTP POST之前，向服务器发送的报文都没有重复的序号，除此之外，Wireshark中也没有发现被标记出来的TCP Retransmission，因此没有发生重传现象

```
TCP      66 80 → 62735 [SYN, ACK] Seq=0 Ack=1 Win=29200
TCP      54 62735 → 80 [ACK] Seq=1 Ack=1 Win=131584 Len=
TCP     797 62735 → 80 [PSH, ACK] Seq=1 Ack=1 Win=131584
TCP    1440 62735 → 80 [ACK] Seq=744 Ack=1 Win=131584 Le
TCP    1440 62735 → 80 [ACK] Seq=2130 Ack=1 Win=131584 L
TCP    1440 62735 → 80 [ACK] Seq=3516 Ack=1 Win=131584 L
TCP    1440 62735 → 80 [ACK] Seq=4902 Ack=1 Win=131584 L
TCP    1440 62735 → 80 [ACK] Seq=6288 Ack=1 Win=131584 L
TCP    1440 62735 → 80 [ACK] Seq=7674 Ack=1 Win=131584 L
TCP    1440 62735 → 80 [ACK] Seq=9060 Ack=1 Win=131584 L
TCP    1440 62735 → 80 [ACK] Seq=10446 Ack=1 Win=131584
TCP    1440 62735 → 80 [ACK] Seq=11832 Ack=1 Win=131584
```

一个TCP报文头的长度是20字节，由上图可知，共发送了112个数据帧，数据总共153064字节，则总共发送了112*20 + 153064 = 155304字节

```
Sequence Number: 1    (relative sequence number)
Sequence Number (raw): 3286614470
[Next Sequence Number: 1    (relative sequence numb
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 1267948082
0101 .... = Header Length: 20 bytes (5)
```

从开始发送到HTTP POST报文，总共经过64.569956 - 63.438822 = 1.131134秒

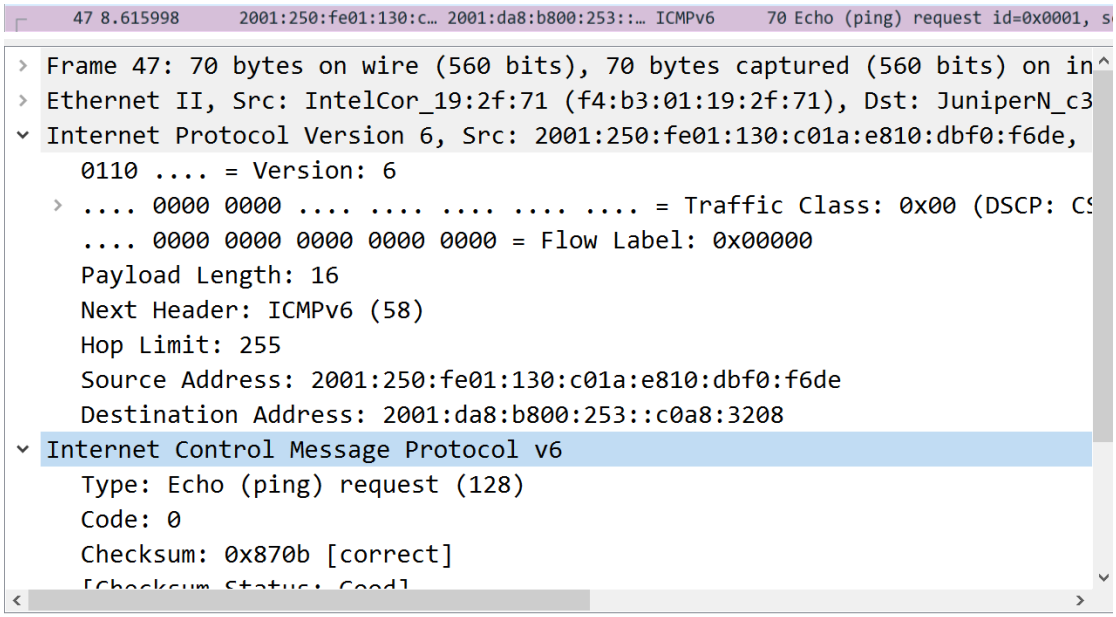


则TCP 连接的 throughput为137,299.38B/S，约134KB/S

三、IP 分析

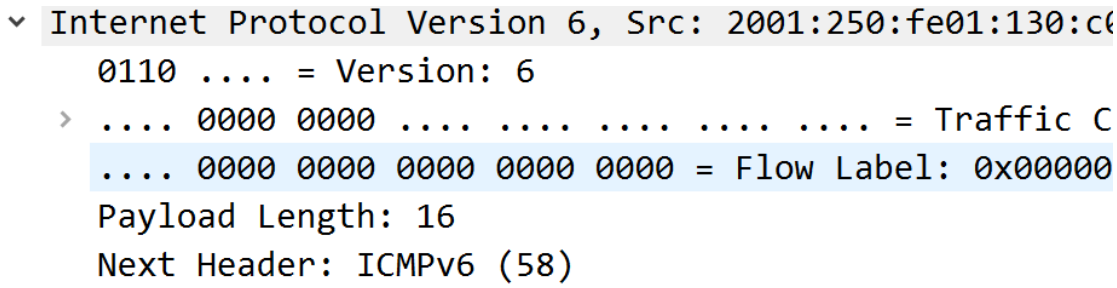
（1）

选择一个ICMP Echo Request消息，观察其Internet Protocol部分报文，内容如下。可以看到源地址是2001:250:fe01:130:c01a:e810:dbf0:f6de，目的地址是2001:da8:b800:253:c0a8:3208



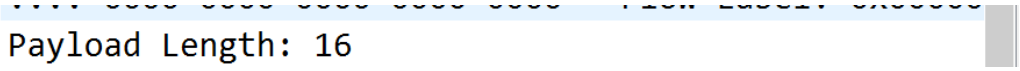
（2）

可以看到，在IP数据包头中，其上层协议字段的值是58，表示ICMPv6



（3）

在IPv6中，IP头长度是固定的40个字节长度，则IP数据包的净载为56-40＝16个字节，与下图中的数据相同。



（4）

IP头部的FragmentHeader部分可以判断是否分片，如果More Fragments被设置，则说明已经被分片。

```
Fragment Header for IPv6
    Next header: ICMPv6 (58)
    Reserved octet: 0x00
    0000 0101 1010 1... = Offset: 181 (1448 bytes)
    .... .... .... .00. = Reserved bits: 0
    .... .... .... ...1 = More Fragments: Yes
```

（5）

主机发出的一系列ICMP消息中IP数据报中有Hoplimit, Identification, Checksum以及Data字段总是发生改变，其余的字段都应该保持常量。对于Hoplimit，随着ICMP报文的转发，数字会逐渐改变；对于每一个数据包，Identification字段都是特别的，每次都需要改变；除此之外，每次所载的Data也可能不同，因此也会导致Checksum的改变。

（6）

对于连续的报文，其包Identification字段值依次递增。

```
    Identification: 0x74d2869d

    Identification: 0x74d2869f
```

（7）

Identification字段和TTL字段的值分别是0xc03d3509和60

```
  Next Header: Fragment Header for IPv6 (44)
  Hop Limit: 60
  Source Address: 2001:da8:b800:253::c0a8:3208
  Destination Address: 2001:250:fe01:130:c01a:e810:db
Fragment Header for IPv6
    Next header: ICMPv6 (58)
    Reserved octet: 0x00
    0000 0000 0000 0... = Offset: 0 (0 bytes)
    .... .... .... .00. = Reserved bits: 0
    .... .... .... ...1 = More Fragments: Yes
    Identification: 0xc03d3509
  [Reassembled IPv6 in frame: 421]
```

（8）

TTL值，也就是Hoplimit字段的值，因为这是第一跳路由器发送的数据报，所以TTL为最大值减一，应该是不变的。而Identification字段用于标识字段，是唯一的，所以会不断改变。

（9）

找到在将包大小改为2000字节后主机发送的第一个ICMP Echo Request消息如下，可以看到，消息被分解为了两个IP数据报。

```
        Next header: ICMPv6 (58)
        Reserved octet: 0x00
        0000 0101 1010 1... = Offset: 181 (1448 bytes)
        .... .... .... .00. = Reserved bits: 0
        .... .... .... ...0 = More Fragments: No
        Identification: 0x74d2868f
    [2 IPv6 Fragments (1960 bytes): #418(1448), #419(51
        [Frame: 418, payload: 0-1447 (1448 bytes)]
        [Frame: 419, payload: 1448-1959 (512 bytes)]
        [Fragment count: 2]
        [Reassembled IPv6 length: 1960]
        [Reassembled IPv6 data: 800084e60001018f202020202
Internet Control Message Protocol v6
```

（10）

观察第一个分片，可以看到More Fragments被设置，这意味着发生了分片。IP头部的Offset信息为0可以确认该数据包是第一个数据包。该分片的长度是1448字节

```
    Source Address: 2001:250:fe01:130:c01a:e810:dbf0:f6
    Destination Address: 2001:da8:b800:253::c0a8:3208
  Fragment Header for IPv6
      Next header: ICMPv6 (58)
      Reserved octet: 0x00
      0000 0000 0000 0... = Offset: 0 (0 bytes)
      .... .... .... .00. = Reserved bits: 0
      .... .... .... ...1 = More Fragments: Yes
      Identification: 0x74d2868f
    [Reassembled IPv6 in frame: 419]
Data (1448 bytes)
    Data: 800084e60001018f2020202020202020202020202020202
    [Length: 1448]
```

（11）

找到在将包大小改为3500字节后主机发送的第一个ICMP Echo Request消息如下。可以看到数据被分为了三片。前两片的Mode Fragments都为yes，最后一片为No，这三片的Offset分别为0，181，362

```
        Next header: ICMPv6 (58)
        Reserved octet: 0x00
        0000 1011 0101 0... = Offset: 362 (2896 bytes)
        .... .... .... .00. = Reserved bits: 0
        .... .... .... ...0 = More Fragments: No
        Identification: 0x74d287bb
    [3 IPv6 Fragments (3460 bytes): #1119(1448), #1120(
        [Frame: 1119, payload: 0-1447 (1448 bytes)]
        [Frame: 1120, payload: 1448-2895 (1448 bytes)]
        [Frame: 1121, payload: 2896-3459 (564 bytes)]
        [Fragment count: 3]
        [Reassembled IPv6 length: 3460]
        [Reassembled IPv6 data: 800060560001022520202020
```

四、分析 Ethernet

（1）

以太网帧的结构如下图所示

以太网帧相关的内容在抓取ARP数据包中详细介绍

五、抓取 ARP 数据包

（1）

使用指令arp -a 获得ARP缓存如下图所示。在ARP缓存中第一列指的是ARP协议的缓存的IP地址，第二列是MAC地址，第三列是类型，即表示是动态类型还是静态类型。

```
Windows PowerShell
接口: 192.168.171.1 --- 0x6
  Internet 地址         物理地址              类型
  192.168.171.254       00-50-56-f3-c4-c6     动态
  192.168.171.255       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
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态

接口: 172.20.110.150 --- 0x7
  Internet 地址         物理地址              类型
  172.20.0.1            44-ec-ce-d2-ff-c2     动态
  172.20.110.26         44-ec-ce-d2-ff-c2     动态
  172.20.110.37         44-ec-ce-d2-ff-c2     动态
  172.20.110.71         44-ec-ce-d2-ff-c2     动态
  172.20.110.99         44-ec-ce-d2-ff-c2     动态
  172.20.110.139        44-ec-ce-d2-ff-c2     动态
  172.20.110.152        44-ec-ce-d2-ff-c2     动态
  172.20.110.169        44-ec-ce-d2-ff-c2     动态
  172.20.110.180        44-ec-ce-d2-ff-c2     动态
  172.20.110.191        44-ec-ce-d2-ff-c2     动态
  172.20.110.195        44-ec-ce-d2-ff-c2     动态
  172.20.110.197        44-ec-ce-d2-ff-c2     动态
  172.20.110.235        44-ec-ce-d2-ff-c2     动态
  172.20.169.76         44-ec-ce-d2-ff-c2     动态
  172.20.220.52         44-ec-ce-d2-ff-c2     动态
  172.20.234.102        44-ec-ce-d2-ff-c2     动态
  172.20.255.255        ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
```

（2）

ARP数据包的格式如下图所示

由九部分构成，分别是硬件类型2字节，协议类型2字节，硬件地址长度1字节，协议地址长度1字节，OP部分2字节，发送端MAC地址6字节，发送端IP地址4字节，目标MAC地址6字节，目标IP地址4字节，

（3）

在ping一个地址之后，可以在Wireshark中找到Arp报文如下，可以通过opcode字段来判断一个ARP数据是请求包还是应答包。

```
Opcode: request (1)
Opcode: reply (2)
```

（4）

当需要查询ARP时，此时不知道目的IP对应的MAC地址，因此需要进行广播查询并寻找；当ARP响应的时候，此时已经找到了源MAC地址，因此ARP响应可以发往明确的局域网地址。

六、抓取 UDP 数据包

（1）

消息是基于UDP的

（2）

多次发送数据后确认，本地ip地址是172.20.110.150，目的地址是39.156.125.82

24	1.239225	172.20.110.150	39.156.125.82	UDP	213 4014 → 8000 Len=171
29	1.530448	39.156.125.82	172.20.110.150	UDP	73 8000 → 4014 Len=31
30	1.637257	172.20.110.150	39.156.125.82	UDP	233 4014 → 8000 Len=191
34	1.847220	39.156.125.82	172.20.110.150	UDP	81 8000 → 4014 Len=39
35	1.848182	172.20.110.150	39.156.125.82	UDP	193 4014 → 8000 Len=151
37	1.929161	39.156.125.82	172.20.110.150	UDP	73 8000 → 4014 Len=31
38	2.288049	172.20.110.150	39.156.125.82	UDP	213 4014 → 8000 Len=171
43	2.613035	39.156.125.82	172.20.110.150	UDP	73 8000 → 4014 Len=31

（3）

观察UDP报文头的信息，可以发现发送消息的端口是4014，服务器端口是8000

```
User Datagram Protocol, Src Port: 4014, Dst Port: 8000
    Source Port: 4014
    Destination Port: 8000
    Length: 179
    Checksum: 0xc05d [unverified]
    [Checksum Status: Unverified]
    [Stream index: 3]
```

（4）

UDP数据报的格式是：源端口号2字节，目的端口号2字节，长度2字节，校验和2字节

（5）

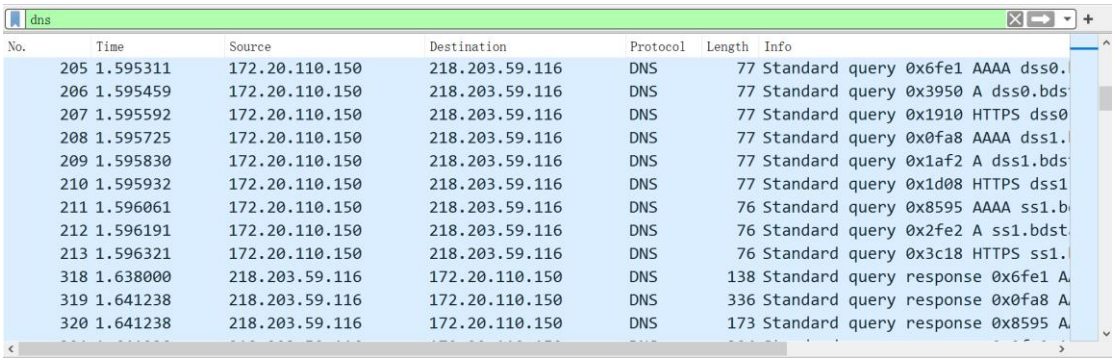
服务器返回ICQ用于确认收到信息。+UDP提供的是不可靠的传输服务，客户端无法确认服务器是否收到信息，因此服务器需要返回一个ICQ数据包，返回接受结果。

可以看出UDP是无连接的。首先UDP没有三次握手来建立连接这个过程。同时UDP首部也没有标志位用于确认传输情况，数据是乱序的。

七、利用 WireShark 进行 DNS 协议分析

（1）

抓包获得的数据如下图所示，可以看到DNS的源地址是172.20.110.150，目的地址是218.203.59.116



（2）

请求报文的内容如下图所示，可以从标志位看出，请求是递归的请求。除此之外，还有DNS请求的在本地的端口号是54502，而在服务器的端口是53.

```
    [Stream index: 17]
  > [Timestamps]
    UDP payload (34 bytes)
v Domain Name System (query)
    Transaction ID: 0x3c18
  v Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a quer
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not t
      .... ...1 .... .... = Recursion desired: Do query
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Una
    Questions: 1
```

```
v User Datagram Protocol, Src Port: 54502, Dst Port: 53
    Source Port: 54502
    Destination Port: 53
    Length: 42
    Checksum: 0x3126 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 17]
```

（3）

在响应报文中，可以看到DNS服务器对于请求www.baidu.com的回答内容。

```
  v Queries
    v sp0.baidu.com: type A, class IN
        Name: sp0.baidu.com
        [Name Length: 13]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  v Answers
    > sp0.baidu.com: type CNAME, class IN, cname www.a.
    > www.a.shifen.com: type A, class IN, addr 39.156.6
    > www.a.shifen.com: type A, class IN, addr 39.156.6
  v Authoritative nameservers
    > a.shifen.com: type NS, class IN, ns ns3.a.shifen.
```

实验结果：
实验结果的内容已经在实验过程中阐述。
问题讨论：
在TCP实验过程中遇到了代理服务器不能正确转发报文的问题，最终尝试更换浏览器、更改设置等方式，在本地最终成功上传了Alice.txt并且成功抓包到了HTTP POST报文，但是由于连接和网络问题，抓包到了大量的无用的信息。
心得体会：
1、 对于各个协议的实现有了深入的了解 2、 对于IPv4和IPv6有了更进一步的了解 3、 学会了Wireshark的基本使用