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	cycleke		院系	计算学部		
班级			学号			
任课教师	刘亚维		指导教师	刘亚维		
实验地点	格物 207		实验时间	2020 年 11 月 21 日		
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						

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：

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

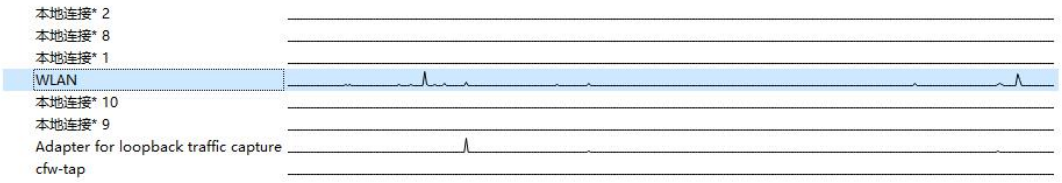
选做内容:

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

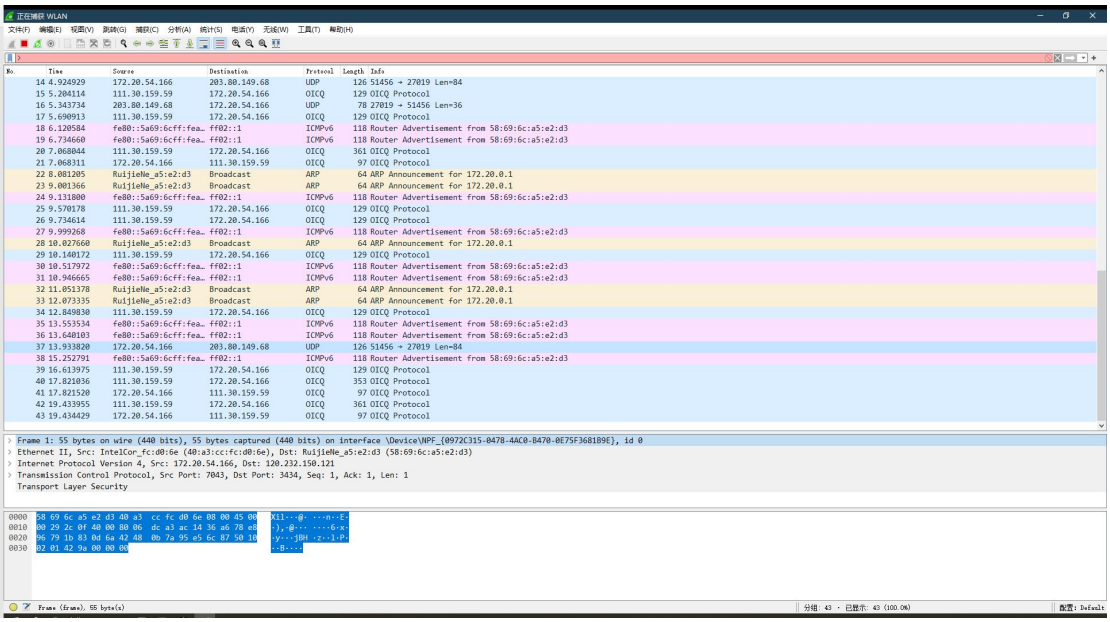
实验过程：

一、Wireshark 的使用

首先在 Wireshark 官网 https://www.wireshark.org/download.html 下载 Wireshark，之后捕获器选择接口进行捕获。由于我本地电脑使用的是无线网卡进行上网，所以我选择捕获 WLAN 接口。



在选定接口后，我可以在界面中看见 Wireshark 的抓包信息。



二、HTTP 分析

1) HTTP GET/response 交互

首先启动浏览器，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。之后开始 Wireshark 分组俘获。

在打开的浏览器中访问 http://today.hit.edu.cn，捕获HTTP报文，之后停止分组俘获。最后将捕获的报文保存到文件中，用于后续分析。

2) HTTP 条件 GET/response 交互

首先启动浏览器，清空浏览器的缓存，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。之后开始 Wireshark 分组俘获。

在打开的浏览器中刷新 http://today.hit.edu.cn 的页面，捕获 HTTP报文，之后停止分组俘获。最后将捕获的报文保存到文件中，用于后续分析。

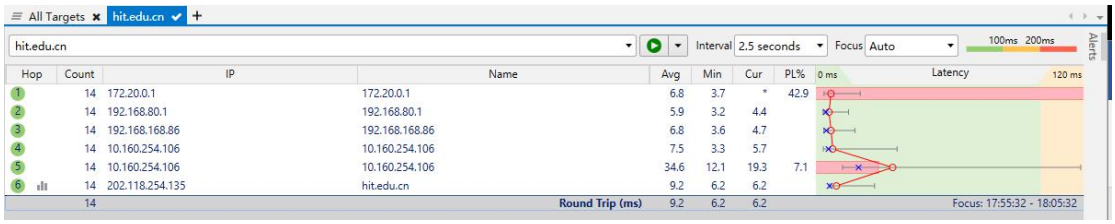
三、TCP 分析

首先下载位于 http://gaia.cs.umass.edu/wireshark-labs/alice.txt 的 alice.txt 文本文件，然后打开 http://gaia.cs.umass.edu/Wireshark-labs/TCP-Wireshark-file1.html ，选中自己下载的文本文件。

此时打开启动Wireshark，开始分组俘获。在浏览器中，单击“Upload alice.txt file”按钮，将文件上传到 gaia.cs.umass.edu 的服务器。在文件上传完毕，一个简短的贺词信息将显示在浏览器窗口中后，停止 Wireshark 的捕获。最后将捕获的报文保存到文件中，用于分析。

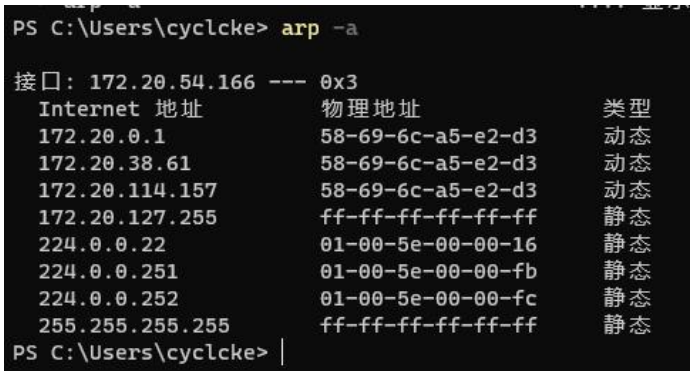
四、IP 分析

打开 Wireshark 进行数据包的捕获，之后使用 pingplotter 依次向 hit.edu.cn 发送大小为 56 字节、2000 字节和 3500 字节的 IP 数据包。最后将捕获的报文保存到文件中，用于后续分析。



五、抓取 ARP 数据包

首先利用 arp 指令查看本地的 ARP 缓存表。



之后开启 Wireshark 的分组捕获，在命令行中 ping 172.20.90.194，ping 通后停止捕获。

六、抓取 UDP 数据包

启动 Wireshark 分组捕获，利用 QQ 给好友发送消息，消息发送结束后，停止分组捕获，之后将这段时间捕获的报文保存到文件中。

七、利用 Wireshark 进行 DNS 协议分析

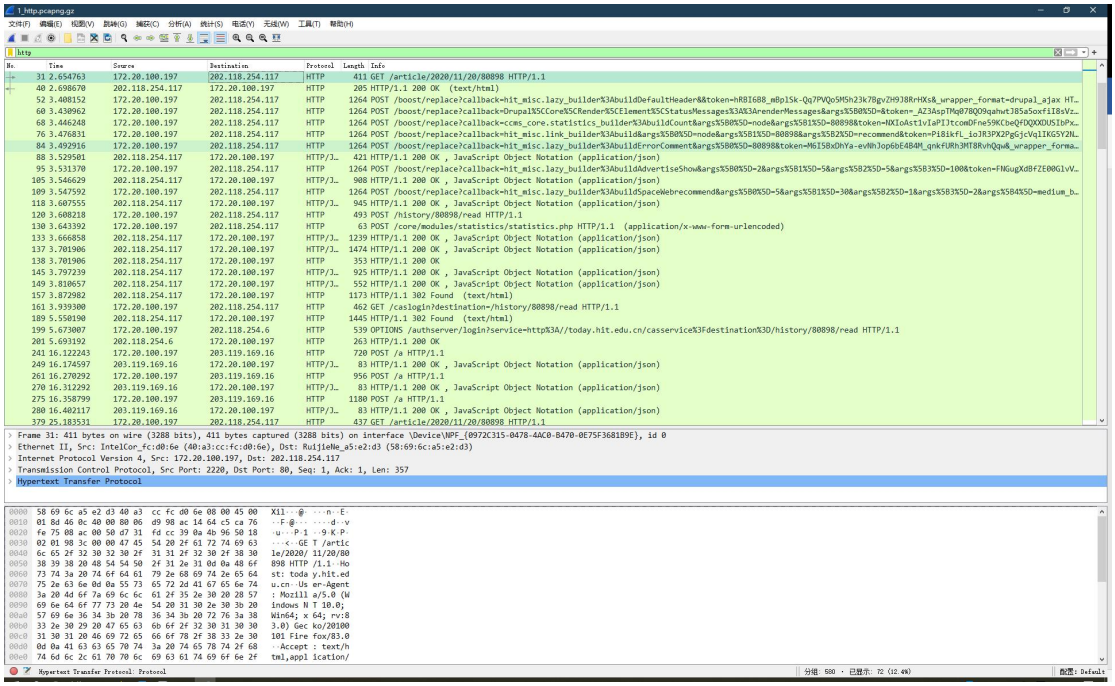
首先打开 Wireshark 进行抓包，在浏览器中访问 www.baidu.com ，完成后停止抓包，保存到文件。

实验结果：

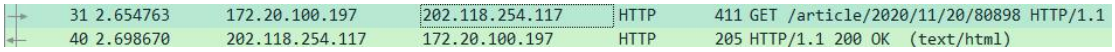
一、HTTP 分析

1) HTTP GET/response 交互

利用 Wireshark，我们可以获得如下的报文，截图如下：



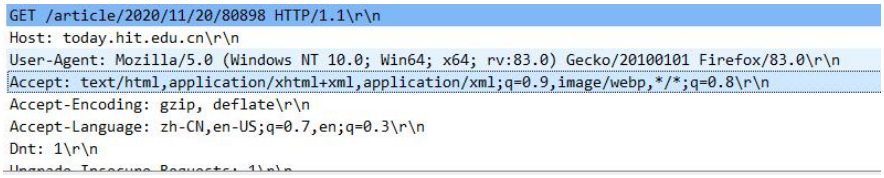
1. 通过截图，可以发现我的浏览器使用的是 HTTP/1.1 协议；访问的服务器使用的也是 HTTP/1.0 协议。



31	2.654763	172.20.100.197	202.118.254.117	HTTP	411 GET /article/2020/11/20/80898 HTTP/1.1
40	2.698670	202.118.254.117	172.20.100.197	HTTP	205 HTTP/1.1 200 OK (text/html)

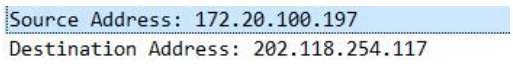
2. 浏览器向服务器指明其可以接收的对象如下，即：

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n



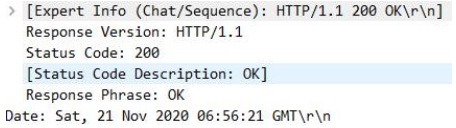
```
GET /article/2020/11/20/80898 HTTP/1.1\r\n
Host: today.hit.edu.cn\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0) Gecko/20100101 Firefox/83.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: zh-CN,en-US;q=0.7,en;q=0.3\r\n
Dnt: 1\r\n
```

3. 我的 IP 地址为 172.20.100.197，服务器 today.hit.edu.cn 的 IP 地址为 202.118.254.117。



```
Source Address: 172.20.100.197
Destination Address: 202.118.254.117
```

4. 服务器向本机浏览器返回的状态码为200。



```
> [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: Sat, 21 Nov 2020 06:56:21 GMT\r\n
```

2) HTTP 条件 GET/response 交互

1. 在清除浏览器缓存之后，访问 today.hit.edu.cn，发出的第一个 HTTP GET 请求如下，可以发现不包含 IF-MODIFIED-SINCE 行：

```
GET /article/2020/11/20/80898 HTTP/1.1\r\n
  [Expert Info (Chat/Sequence): GET /article/2020/11/20/80898 HTTP/1.1\r\n]
  Request Method: GET
  Request URI: /article/2020/11/20/80898
  Request Version: HTTP/1.1
Host: today.hit.edu.cn\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0) Gecko/20100101 Firefox/83.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: zh-CN,en-US;q=0.7,en;q=0.3\r\n
Dnt: 1\r\n
Upgrade-Insecure-Requests: 1\r\n
\r\n
[Full request URI: http://today.hit.edu.cn/article/2020/11/20/80898]
[HTTP request 1/1]
[Response in frame: 40]
```

2. 服务器接下来的服务器响应报文返回了明确的内容（若干 json 文件，通过右键-追踪流-HTTP流可以知道），状态码均为 200。

```
 88 3.529501   202.118.254.117  172.20.100.197   HTTP/J…   421 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
 95 3.531370   172.20.100.197   202.118.254.117  HTTP     1264 POST /boost/replace?callback=hit_misc.lazy_builder%3AbuildAdvertiseShow&args%5B0%5D=2&args%5B1%5D=5&args%5B2%5D=5&args%5B3%5D=100&token=FNGugXdBfZE00G1vV…
105 3.546629   202.118.254.117  172.20.100.197   HTTP/J…   908 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
109 3.547592   172.20.100.197   202.118.254.117  HTTP     1264 POST /boost/replace?callback=hit_misc.lazy_builder%3AbuildSpaceWebrecommend&args%5B0%5D=5&args%5B1%5D=30&args%5B2%5D=1&args%5B3%5D=2&args%5B4%5D=medium_b…
118 3.607555   202.118.254.117  172.20.100.197   HTTP/J…   945 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
```

3. 而在较晚的 HTTP GET 请求中，报文头部中添加了 IF-MODIFIED-SINCE 行，后面接着的是缓存文件上次更改的时间。

```
GET /sites/today1.prod1.dpweb1.hit.edu.cn/files/images/2019/04/10/zqyw.jpg HTTP/1.1\r\n
  [Expert Info (Chat/Sequence): GET /sites/today1.prod1.dpweb1.hit.edu.cn/files/images/2019/04/10/zqyw.jpg HTTP/1.1\r\n]
  Request Method: GET
  Request URI: /sites/today1.prod1.dpweb1.hit.edu.cn/files/images/2019/04/10/zqyw.jpg
  Request Version: HTTP/1.1
Host: today.hit.edu.cn\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0) Gecko/20100101 Firefox/83.0\r\n
Accept: image/webp,*/*\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: zh-CN,en-US;q=0.7,en;q=0.3\r\n
Cache-Control: max-age=0\r\n
Dnt: 1\r\n
If-Modified-Since: Tue, 09 Apr 2019 23:52:22 GMT\r\n
If-None-Match: "19b92-58621a4e57844"\r\n
Referer: http://today.hit.edu.cn/article/2020/11/20/80898\r\n
```

4. 服务器对于较晚的 HTTP GET 请求，其返回的状态码为 304，而且并返回的内容极短（只有392），并没有包含具体的内容。

```
HTTP/1.1 304 Not Modified\r\n
  [Expert Info (Chat/Sequence): HTTP/1.1 304 Not Modified\r\n]
  Response Version: HTTP/1.1
  Status Code: 304
  [Status Code Description: Not Modified]
  Response Phrase: Not Modified
Date: Sat, 21 Nov 2020 06:56:11 GMT\r\n
Server: Server                \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
```

```
546 26.640527   202.118.254.117  172.20.100.197   HTTP     392 HTTP/1.1 304 Not Modified
```

二、TCP 分析

1. 向 gaia.cs.umass.edu 服务器传送文件的客户端（源主机）主机的 IP 地址与 TCP 端口号为 172.20.100.197 和 2415。

2. gaia.cs.umass.edu 服务器 IP 地址为 120.232.150.121，接收端口号为 3434。

```
Internet Protocol Version 4, Src: 172.20.100.197, Dst: 120.232.150.121
Transmission Control Protocol, Src Port: 2415, Dst Port: 3434, Seq: 0, Len: 0
```

3. 客户服务器之间用于初始化 TCP 连接的 TCP SYN 报文段的序号为 0（绝对值为 3301679959）；该报文段将SYN标志位置为 1，表示该报文段为 SYN 段用于 TCP 建立连接。

```
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 3301679959
[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
```

4. 服务器向客户端发送的 SYNACK 报文段序号是 1（绝对值为 3301679960）；Acknowledgement 字段值是 1（绝对值为 2649404903）。服务器将随机指定一个值以决定此值。在该报文段中，在INFO中写入 [SYN,ACK] 来标示的。

```
Sequence Number: 1    (relative sequence number)
Sequence Number (raw): 3301679960
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 2649404903
```

5. 下面的图片展示的就是 TCP 三次握手的过程。首先客户机想服务器端发送 SYN 请求报文；之后服务器向客户机回复 SYN，ACK 报文；最后客户机向服务器回复 ACK 报文段，完成三次握手。

```
6 4.074145   172.20.100.197   120.232.150.121   TCP   66 2415 → 3434 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
7 4.143974   120.232.150.121  172.20.100.197    TCP   66 3434 → 2415 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
8 4.144157   172.20.100.197   120.232.150.121   TCP   54 2415 → 3434 [ACK] Seq=1 Ack=1 Win=131328 Len=0
```

6. 包含 HTTP POST 命令的 TCP 报文的序号是 1（绝对值为 1187582186）。

```
[Stream index: 2]
[TCP Segment Len: 128]
Sequence Number: 1    (relative sequence number)
Sequence Number (raw): 1187582186
[Next Sequence Number: 129    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 798137198
0101 .... = Header Length: 20 bytes (5)
> Flags: 0x018 (PSH, ACK)
Window: 515
[Calculated window size: 515]
[Window size scaling factor: -1 (unknown)]
Checksum: 0x756e [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> [SEQ/ACK analysis]
> [Timestamps]
```

```
0020  a9 e3 32 a6 00 50 46 c9  10 ea 2f 92 9b 6e 50 18   ··2··PF· ··/··nP·
0030  02 03 75 6e 00 00 50 4f  53 54 20 2f 61 20 48 54   ··un··PO ST /a HT
0040  54 50 2f 31 2e 31 0d 0a  43 6f 6e 6e 65 63 74 69   TP/1.1·· Connecti
0050  6f 6e 3a 20 4b 65 65 70  2d 41 6c 69 76 65 0d 0a   on: Keep -Alive··
0060  55 73 65 72 2d 41 67 65  6e 74 3a 20 57 69 6e 48   User-Age nt: WinH
0070  74 74 70 43 6c 69 65 6e  74 0d 0a 43 6f 6e 74 65   ttpClien t··Conte
0080  6e 74 2d 4c 65 6e 67 74  68 3a 20 31 34 33 37 34   nt-Lengt h: 14374
0090  0d 0a 48 6f 73 74 3a 20  70 63 73 2d 73 64 6b 2d   ··Host:  pcs-sdk-
00a0  73 65 72 76 65 72 2e 61  6c 69 62 61 62 61 2e 63   server.a libaba.c
```

7. 第六个报文段的序号为 5889（绝对值为 1187588074）。接受时间为 Nov 26, 2020 10:24:33.814944000 中国标准时间。对应的 ACK 接受时间为 Nov 26, 2020 10:24:33.847832000 中国标准时间。

```
241 9.084496   172.20.54.166   203.119.169.227   TCP    182 12966 → 80 [PSH, ACK] Seq=1 Ack=1 Win=515 Len=128 [TCP segment of a reassembled PDU]
242 9.084663   172.20.54.166   203.119.169.227   TCP   1494 12966 → 80 [ACK] Seq=129 Ack=1 Win=515 Len=1440 [TCP segment of a reassembled PDU]
243 9.084663   172.20.54.166   203.119.169.227   TCP   1494 12966 → 80 [ACK] Seq=1569 Ack=1 Win=515 Len=1440 [TCP segment of a reassembled PDU]
244 9.084663   172.20.54.166   203.119.169.227   TCP   1494 12966 → 80 [ACK] Seq=3009 Ack=1 Win=515 Len=1440 [TCP segment of a reassembled PDU]
245 9.084663   172.20.54.166   203.119.169.227   TCP   1494 12966 → 80 [ACK] Seq=4449 Ack=1 Win=515 Len=1440 [TCP segment of a reassembled PDU]
246 9.084663   172.20.54.166   203.119.169.227   TCP   1494 12966 → 80 [ACK] Seq=5889 Ack=1 Win=515 Len=1440 [TCP segment of a reassembled PDU]
```

```
Sequence Number: 5889    (relative sequence number)
Sequence Number (raw): 1187588074
[Next Sequence Number: 7329    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 798137198
0101 .... = Header Length: 20 bytes (5)
```

```
Encapsulation type: Ethernet (1)
Arrival Time: Nov 26, 2020 10:24:33.814944000 中国标准时间
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1606357473.814944000 seconds
[Time delta from previous captured frame: 0.000000000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 9.084663000 seconds]
```

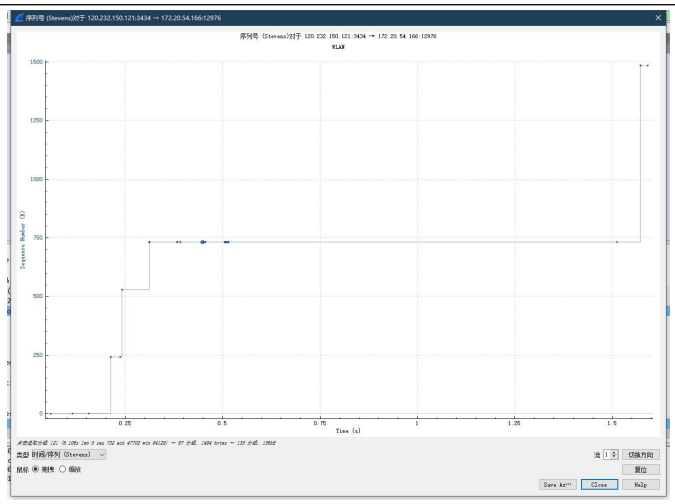
```
254 9.117551   203.119.169.227   172.20.54.166   TCP      60 80 → 12966 [ACK] Seq=1 Ack=7329 Win=30 Len=0
255 9.117551   203.119.169.227   172.20.54.166   TCP      60 80 → 12966 [ACK] Seq=1 Ack=11649 Win=30 Len=0
256 9.117551   203.119.169.227   172.20.54.166   TCP      60 80 → 12966 [ACK] Seq=1 Ack=14503 Win=30 Len=0
257 9.123316   203.119.169.227   172.20.54.166   TCP     324 80 → 12966 [PSH, ACK] Seq=1 Ack=14503 Win=30 Len
258 9.123316   203.119.169.227   172.20.54.166   HTTP/J…  83 HTTP/1.1 200 OK , JavaScript Object Notation (ap
259 9.123396   172.20.54.166     203.119.169.227 TCP      54 12966 → 80 [ACK] Seq=14503 Ack=300 Win=513 Len=0
260 9.157909   172.20.54.166     203.119.169.227 TCP     181 12966 → 80 [PSH, ACK] Seq=14503 Ack=300 Win=513
' Frame 254: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{0972C315-0478-4AC0-B470-0E75F3
  > Interface id: 0 (\Device\NPF_{0972C315-0478-4AC0-B470-0E75F3681B9E})
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov 26, 2020 10:24:33.847832000 中国标准时间
```

8. 前六个 TCP 报文段的长度各是 128、1440、1440、1440、1440 和 1440。

```
241 9.084496   172.20.54.166   203.119.169.227   TCP    182 12966 → 80 [PSH, ACK] Seq=1 Ack=1 Win=515 Len=128 [TCP segment of a reassembled PDU]
242 9.084663   172.20.54.166   203.119.169.227   TCP   1494 12966 → 80 [ACK] Seq=129 Ack=1 Win=515 Len=1440 [TCP segment of a reassembled PDU]
243 9.084663   172.20.54.166   203.119.169.227   TCP   1494 12966 → 80 [ACK] Seq=1569 Ack=1 Win=515 Len=1440 [TCP segment of a reassembled PDU]
244 9.084663   172.20.54.166   203.119.169.227   TCP   1494 12966 → 80 [ACK] Seq=3009 Ack=1 Win=515 Len=1440 [TCP segment of a reassembled PDU]
245 9.084663   172.20.54.166   203.119.169.227   TCP   1494 12966 → 80 [ACK] Seq=4449 Ack=1 Win=515 Len=1440 [TCP segment of a reassembled PDU]
246 9.084663   172.20.54.166   203.119.169.227   TCP   1494 12966 → 80 [ACK] Seq=5889 Ack=1 Win=515 Len=1440 [TCP segment of a reassembled PDU]
```

9. 在整个跟踪过程中，接收端公示的最小的可用缓存空间是 515，限制发送端的传输以后，接收端的缓存是够用。

10. 没有重传的片段。依据为发送端的报文段序号始终在增加，没有出现重复发送某一个序号的报文段的情况，故没有重传的。



11. 最后 ACK 包的序列号为 156318，计时器为 7.249878000 秒，而第一个包的计时器为 5.658626000 秒，所以吞吐量为 $\frac{156318 \times 8bit}{7.249878000s - 5.658626000s} = 0.749\ Mbps$。

三、IP 分析

1. 我的 IP 地址为 172.20.100.197。

```
172.20.100.197    202.118.254.135    ICMP    70 Echo (ping) request  id=0x0001, seq=1/256, ttl=255 (reply in 133)
```

2. 通过分析 IP 数据包，可以分析出上层协议为 ICMP。

```
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 56
Identification: 0x04b0 (1200)
> Flags: 0x00
Fragment Offset: 0
Time to Live: 255
Protocol: ICMP (1)
Header Checksum: 0xdd3c [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.20.100.197
Destination Address: 202.118.254.135
```

3. 通过上面的 Header Length 行和 Total Length 行可以分析得出 IP 头有 20 字节，该IP数据包的净载为 56。

4. 观察 Flags 区，可以发现 More fragments 标志为空，没有其余的帧并且帧的偏移为0，可以推断出没有进行分片。

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

5. 观察 IP 数据包的可以发现 Identification、TTL 和 Checksum 的值总是变化。

```
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 28
Identification: 0xd3ce (54222)
> Flags: 0x40, Don't fragment
Fragment Offset: 0
Time to Live: 63
Protocol: ICMP (1)
Header Checksum: 0x4681 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.20.100.163
Destination Address: 172.20.100.197
```

```
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 56
Identification: 0x04b0 (1200)
Flags: 0x00
Fragment Offset: 0
Time to Live: 255
Protocol: ICMP (1)
Header Checksum: 0xdd3c [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.20.100.197
Destination Address: 202.118.254.135
```

6. 必须保持常量的是版本号、首部长度、Differentiated Services Field 以及协议（始终为ICMP）。必须改变的是 TTL、Checksum 和 Identification，TTL 为生存时间，每次转发必然改变；由于TTL的改变，Checksum 自然也会改变；Identification 则是用于区分不同的 ICMP 报文。

7. Identification 自 1200 开始，接下来的包依次增加 1。

8. Identification 段为 64430，TTL 为 64。

```
Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
Total Length: 84
Identification: 0xfbae (64430)
Flags: 0x00
Fragment Offset: 0
Time to Live: 64
```

9. Identification 段发生变化，这样可以区分不同的 ICMP time-to-live exceeded 消息；但 TTL 保持不变，为 64（均为一次转发）。

10. 该消息被分解为不止一个数据包。

```
Flags: 0x20, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
```

11. IP 头部可以在 Flags 域中，看到 More fragments 被置为 1 且偏移量为 0，表示该分片不为最后一片。该分片的长度为 1500 字节。

```
Total Length: 1500
Identification: 0x04c5 (1221)
Flags: 0x20, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
Fragment Offset: 0
```

12. 原始数据被分成了3片。

```
707 79.668261   172.20.100.197   202.118.254.135   IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=0505) [Reassembled in #709]
708 79.668261   172.20.100.197   202.118.254.135   IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=0505) [Reassembled in #709]
709 79.668261   172.20.100.197   202.118.254.135   ICMP    554 Echo (ping) request  id=0x0001, seq=86/22016, ttl=1 (no response found!)
```

13. 标志位部分、偏移量和 Checksum 部分发生了变化。

```
Flags: 0x20, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
Fragment Offset: 0
```

```
Flags: 0x20, More fragments
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
Fragment Offset: 1480
```

```
Flags: 0x01
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
Fragment Offset: 2960
```

四、抓取 ARP 数据包

1. ARP 表的格式如下。在 ARP 表中，每一项表示一个 IP 地址到物理地址的映射。每一项第一列是IP地址，第二列是物理地址，第三列是类型。

```
PS C:\Users\cyclcke> arp -a

接口: 172.20.54.166 --- 0x3
  Internet 地址         物理地址              类型
  172.20.0.1            58-69-6c-a5-e2-d3     动态
  172.20.38.61          58-69-6c-a5-e2-d3     动态
  172.20.114.157        58-69-6c-a5-e2-d3     动态
  172.20.127.255        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
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态
PS C:\Users\cyclcke>
```

2. ARP数据包的格式如下图所示，共由九部分构成：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（2 字节)，发送端MAC地址（6 字节），发送端IP地址（4 字节），目的MAC地址（6 字节），目的IP地址（4 字节）。

3. 可以通过 Opcode 字段判断，若为 1 则是请求包；若为 2 则是应答包。

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

4. 因为进行 ARP 查询时并不知道目的 IP 地址对应的 MAC 地址，所以需要广播查询；而 ARP 响应报文知道查询主机的 MAC 地址（通过查询主机发出的查询报文获得），且局域网中的其他主机不需要此次查询的结果，因此 ARP 响应要在一个有着明确目的的局域网地址的帧中传送。

五、抓取 UDP 数据包

1. 消息是基于 UDP 的。
2. 本机 IP 地址为 172.20.100.197，目的主机 IP 地址为 111.30.159.62。

```
172.20.100.197      111.30.159.62      OICQ      201 OICQ Protocol
```

3. 主机发送 QQ 消息的端口号为 4022，QQ 服务器的端口号是 8000。

```
User Datagram Protocol, Src Port: 4022, Dst Port: 8000
  Source Port: 4022
  Destination Port: 8000
  Length: 167
  Checksum: 0x4f57 [unverified]
  [Checksum Status: Unverified]
```

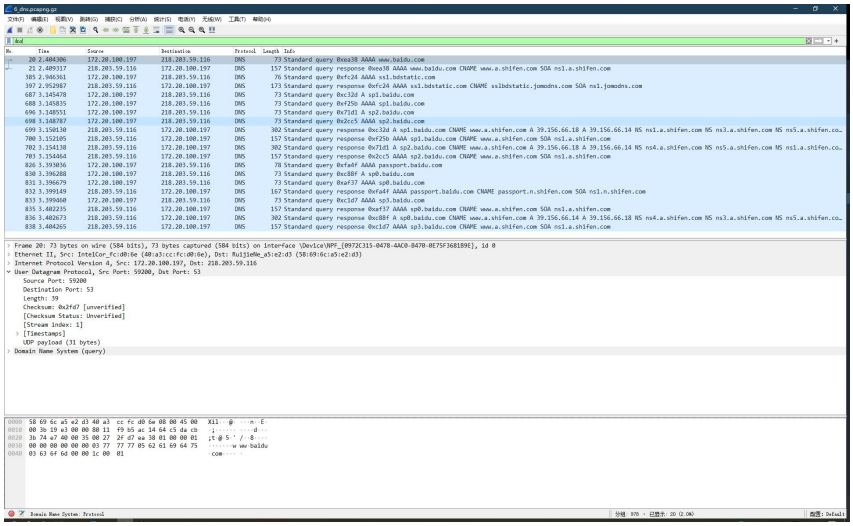
4. UDP 数据报由五部分构成，分别是源端口号（4 字节），目的端口号（4 字节），长度（4 字节），校验和（4 字节）和应用层数据。

```
Source Port: 4022
Destination Port: 8000
Length: 167
Checksum: 0x4f57 [unverified]
[Checksum Status: Unverified]
[Stream index: 0]
> [Timestamps]
UDP payload (159 bytes)
```

5. 因为 UDP 是不可靠的数据传输，需要上层协议来实现可靠数据传输，因此每次发送 ICQ 报文后又回复一个 ICQ 数据包来确认。UDP 是无连接的，因为可以看到发送数据之前没有连接的建立过程（如 TCP 的三次握手），没有序列号，因此为无连接数据传输。

六、利用 Wireshark 进行 DNS 协议分析

利用 Wireshark 进行 DNS 协议抓包的结果如上。



问题讨论：
见实验结果的分析。
心得体会：
通过本次实验，我熟悉并掌握 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。在对于不同的协议报文分析的过程中，我对于各个网络协议的理解更为深入，对于部分细节也有了全新的认识。