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	董琦		院系	数据科学与大数据		
班级	2003501		学号	120L020701		
任课教师	刘亚维		指导教师	刘亚维		
实验地点	G001		实验时间	2022/10/13		
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						

计算机科学与技术学院 SINCE 1956...
School of Computer Science and Technology

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：

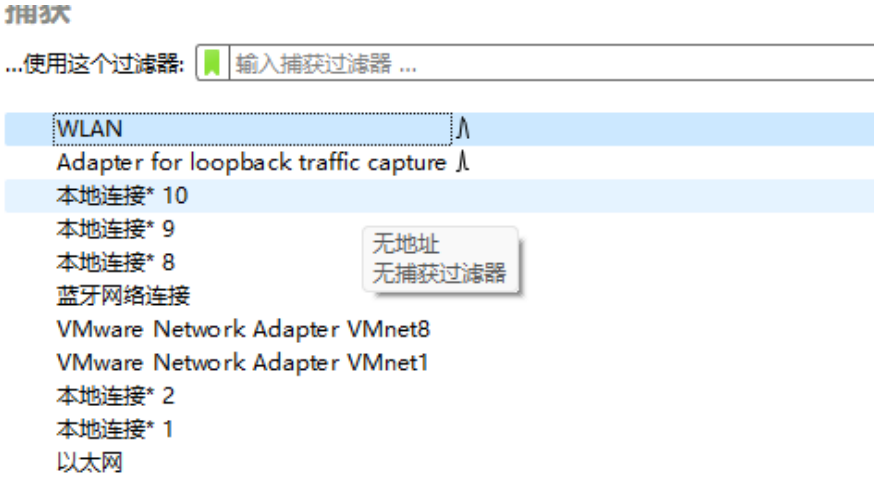
a) 利用Wireshark 分析DNS 协议
b) 利用Wireshark 分析UDP 协议
c) 利用Wireshark 分析ARP 协议

实验过程：

一．Wireshark使用

打开Wireshark，监听WLAN

我们监听WLAN这个端口。

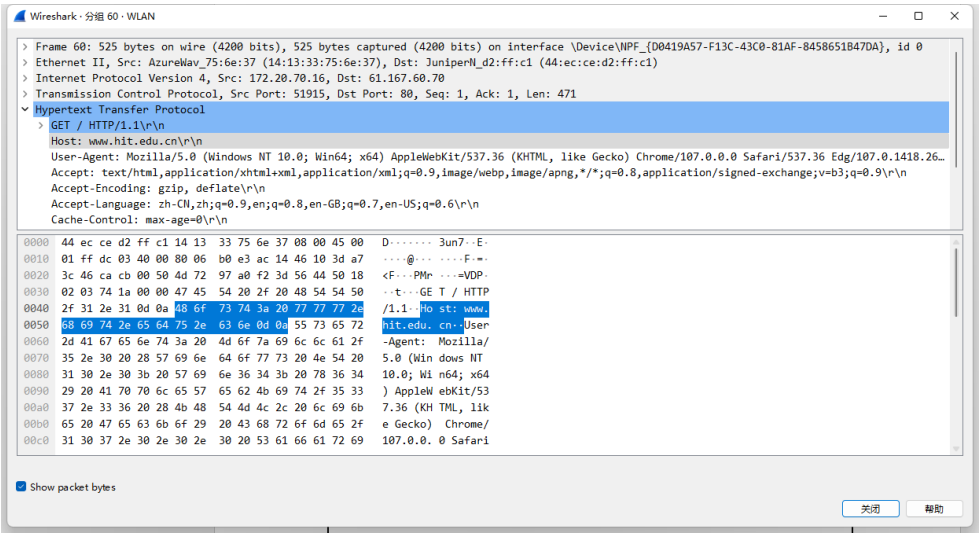


打开浏览器，访问http://www.hit.edu.cn网址。

加载出网页后暂停wireshark抓包。

（学校主页还挺帅的哈）

在wireshark中可以看到发送的http请求：



可以看到链路层、网络层、传输层、应用层的各层协议内容以及报文内容。

二. HTTP分析

1. Get分析

打开浏览器访问http://hitgs.hit.edu.cn/news 开始抓包。

额这个网页挂掉了，换成了主页。

28 1.196110	172.20.70.16	219.217.226.25	HTTP	505 GET /news HTTP/1.1
30 1.201953	219.217.226.25	172.20.70.16	HTTP	523 HTTP/1.1 302 Found (text/html)

所得到的http分组：

31	1.215693	172.20.70.16	219.217.226.25	HTTP	514 GET /news/main.psp HTTP/1.1
36	1.237949	219.217.226.25	172.20.70.16	HTTP	1315 HTTP/1.1 200 200 (text/html)
40	1.289915	172.20.70.16	219.217.226.25	HTTP	486 GET /_css/error/error.css HTTP/1.1
48	1.294445	172.20.70.16	219.217.226.25	HTTP	468 GET /_js/jquery.min.js HTTP/1.1
50	1.294488	219.217.226.25	172.20.70.16	HTTP	870 HTTP/1.1 200 OK (text/css)
54	1.296054	172.20.70.16	219.217.226.25	HTTP	485 GET /_js/themes/icon.css HTTP/1.1
55	1.296588	172.20.70.16	219.217.226.25	HTTP	464 GET /_js/common.js HTTP/1.1
60	1.296758	172.20.70.16	219.217.226.25	HTTP	475 GET /_js/easyui-lang-zh_CN.js HTTP/1.1
61	1.296765	172.20.70.16	219.217.226.25	HTTP	475 GET /_js/jquery.easyui.min.js HTTP/1.1
70	1.300905	219.217.226.25	172.20.70.16	HTTP	1382 HTTP/1.1 200 OK (application/javascript)
73	1.300905	219.217.226.25	172.20.70.16	HTTP	506 HTTP/1.1 200 OK (text/css)
79	1.302460	219.217.226.25	172.20.70.16	HTTP	425 HTTP/1.1 200 OK (application/javascript)
81	1.302960	172.20.70.16	219.217.226.25	HTTP	471 GET /_js/jquery.base64.js HTTP/1.1
96	1.306288	219.217.226.25	172.20.70.16	HTTP	684 HTTP/1.1 200 OK (application/javascript)
116	1.309670	219.217.226.25	172.20.70.16	HTTP	880 HTTP/1.1 200 OK (application/javascript)
136	1.315438	172.20.70.16	219.217.226.25	HTTP	495 GET /_js/themes/default/easyui.css HTTP/1.1
169	1.323759	219.217.226.25	172.20.70.16	HTTP	1211 HTTP/1.1 200 OK (text/css)
191	1.327719	172.20.70.16	219.217.226.25	HTTP	524 GET /_images/error/error.gif HTTP/1.1
202	1.329377	219.217.226.25	172.20.70.16	HTTP	875 HTTP/1.1 200 OK (application/javascript)
206	1.332380	219.217.226.25	172.20.70.16	HTTP	248 HTTP/1.1 200 OK (GIF89a)
208	1.346381	172.20.70.16	219.217.226.25	HTTP	528 GET /_images/error/bg.gif HTTP/1.1
209	1.346622	172.20.70.16	219.217.226.25	HTTP	528 GET /_images/error/bg.jpg HTTP/1.1
210	1.347555	172.20.70.16	219.217.226.25	HTTP	527 GET /_images/error/m.gif HTTP/1.1
211	1.347631	172.20.70.16	219.217.226.25	HTTP	527 GET /_images/error/l.gif HTTP/1.1
213	1.350531	219.217.226.25	172.20.70.16	HTTP	206 HTTP/1.1 200 OK (GIF89a)

可以看到在加载网页时的一系列GET/response交互。

下面具体分析报文回答问题。

请求报文：

```
Hypertext Transfer Protocol
  GET /news/main.psp HTTP/1.1\r\n
  Host: hitgs.hit.edu.cn\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36 Edg/107.0.1418.26\r
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\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
  Upgrade-Insecure-Requests: 1\r\n
  \r\n
  [Full request URI: http://hitgs.hit.edu.cn/news/main.psp]
  [HTTP request 2/12]
  [Prev request in frame: 28]
  [Response in frame: 36]
  [Next request in frame: 40]
```

响应报文：

```
Hypertext Transfer Protocol
  HTTP/1.1 200 200\r\n
  Server:  \r\n
  Date: Sat, 05 Nov 2022 05:20:22 GMT\r\n
  Content-Type: text/html;charset=UTF-8\r\n
  Content-Length: 866\r\n
  Connection: keep-alive\r\n
  X-Frame-Options: SAMEORIGIN\r\n
  Frame-Options: SAMEORIGIN\r\n
  X-Application-Context: application\r\n
  Set-Cookie: JSESSIONID=110DB1085DE31547B6935F8D0C6416D2; Path=/; HttpOnly\r\n
  Vary: Accept-Encoding\r\n
  Content-Encoding: gzip\r\n
  X-Frame-Options: SAMEORIGIN\r\n
```

1) 你的浏览器运行的是HTTP1.0，还是HTTP1.1？你所访问的服务器所运行HTTP 协议的版本号是多少？

都是HTTP1.1。

2) 你的浏览器向服务器指出它能接收何种语言版本的对象？

```
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6\r\n
```

3) 你的计算机的IP 地址是多少？服务器http://hitgs.hit.edu.cn/news的IP 地址是多少？

Source	Destination
172.20.70.16	219.217.226.25
219.217.226.25	172.20.70.16
172.20.70.16	219.217.226.25

我：172.20.70.16 服务器：219.217.226.25

4) 从服务器向你的浏览器返回的状态代码是多少？

200，说明正常响应。

2. 条件GET分析

先清除缓存：

重新访问后结束抓包。

1) 分析你的浏览器向服务器发出的第一个HTTP GET 请求的内容，在该请求报文中，是否有一行是：IF-MODIFIED-SINCE？

```
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
  Host: hitgs.hit.edu.cn\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36 Edg/107.0.1418.26\r
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\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
  Cache-Control: max-age=0\r\n
  Upgrade-Insecure-Requests: 1\r\n
  \r\n
  [Full request URI: http://hitgs.hit.edu.cn/]
  [HTTP request 1/10]
  [Response in frame: 17]
  [Next request in frame: 19]
```

如图，没有。

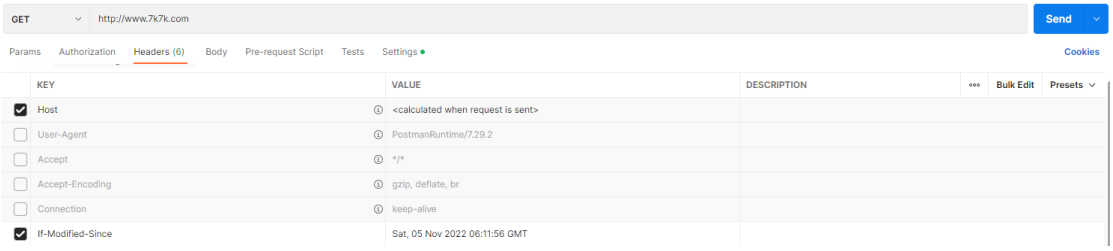
2) 分析服务器响应报文的内容，服务器是否明确返回了文件的内容？如何获知？

是，通过状态码200，报文体中包含了文件内容。

```
> HTTP/1.1 200 OK\r\n
> Line-based text data: text/html (688 lines)
```

3) 分析你的浏览器向服务器发出的较晚的“HTTP GET"请求，在该请求报文中是否有一行是：IF-MODIFIED-SINCE？如果有，在该首部行后面跟着的信息是什么？

有，这里我使用自己构建报文访问www.7k7k.com网站（浏览器访问总是没有）：



抓包到的报文内容：

```
> Frame 15: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface \Device\NPF_{D0419A57-F13C-43C0-81AF-8458651B47DA}, id 0
> Ethernet II, Src: AzureWav_75:6e:37 (14:13:33:75:6e:37), Dst: JuniperN_d2:ff:c1 (44:ec:ce:d2:ff:c1)
> Internet Protocol Version 4, Src: 172.20.70.16, Dst: 111.43.178.112
> Transmission Control Protocol, Src Port: 55502, Dst Port: 80, Seq: 1, Ack: 1, Len: 88
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
  If-Modified-Since: Sat, 05 Nov 2022 06:11:56 GMT\r\n
  Host: www.7k7k.com\r\n
  \r\n
  [Full request URI: http://www.7k7k.com/]
  [HTTP request 1/1]
  [Response in frame: 17]
```

后面跟着的是上一次收到报文的时间（我自己填的）。

4) 服务器对较晚的HTTP GET 请求的响应中的HTTP 状态代码是多少？服务器是否明确返回了文件的内容？请解释。

304，表示没有修改。不明确返回文件内容，这样就使用之前缓存报文内容，节省流量。

三．TCP分析

A． 俘获大量的由本地主机到远程服务器的TCP分组

Jpload page for TCP Wireshark Lab
Computer Networking: A Top Down Approach, 6th edition
Copyright 2012 J.F. Kurose and K.W. Ross, All Rights Reserved

f you have followed the instructions for the TCP Wireshark Lab, you have *already* downloaded an ASCII copy of Ali
packets on your computer.

Click on the Browse button below to select the directory/file name for the copy of alice.txt that is stored on your com

选择文件 alice.txt

Once you have selected the file, click on the "Upload alice.txt file" button below. This will cause your browser to sen
s displayed indicating the the upload is complete. Then stop your Wireshark packet sniffer - you're ready to begin a

Upload alice.txt file

按照指导下载文件并访问网站。

传输完成后停止捕获。

Congratulations!

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

B． 浏览追踪信息

得到如下报文：

tcp and ip.addr==128.119.245.12 and tcp.port==54280

No.	Time	Source	Destination	Protocol	Length	Info
45	2.257434	172.20.70.16	128.119.245.12	TCP	66	54280 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
53	2.536819	128.119.245.12	172.20.70.16	TCP	66	80 → 54280 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1360 SACK_PERM WS=128
54	2.536862	172.20.70.16	128.119.245.12	TCP	54	54280 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=0
72	7.687573	172.20.70.16	128.119.245.12	TCP	839	54280 → 80 [PSH, ACK] Seq=1 Ack=1 Win=131840 Len=785 [TCP segment of a reassembled PDU]
73	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=786 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
74	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=2146 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
75	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=3506 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
76	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=4866 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
77	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=6226 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
78	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=7586 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
79	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=8946 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
80	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=10306 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
81	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=11666 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
82	7.967412	128.119.245.12	172.20.70.16	TCP	56	80 → 54280 [ACK] Seq=1 Ack=786 Win=30848 Len=0
83	7.967452	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=13026 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
84	7.967536	128.119.245.12	172.20.70.16	TCP	56	80 → 54280 [ACK] Seq=1 Ack=7586 Win=44416 Len=0
85	7.967536	128.119.245.12	172.20.70.16	TCP	56	80 → 54280 [ACK] Seq=1 Ack=13026 Win=55296 Len=0
86	7.967572	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=14386 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
87	7.967572	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=15746 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
88	7.967572	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [PSH, ACK] Seq=17106 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
89	7.967572	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=18466 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
90	7.967572	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=19826 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
91	7.967572	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=21186 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
92	7.967572	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=22546 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
93	7.967572	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=23906 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
94	7.967572	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=25266 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
95	7.967572	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=26626 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]

1） 向gaia.cs.umass.edu 服务器传送文件的客户端主机的IP 地址和TCP 端口号是多少？

172.20.70.16，54280

2） Gaia.cs.umass.edu 服务器的IP 地址是多少？对这一连接，它用来发送和接收TCP 报文的端口号是多少？

128.119.245.12，80.

C． TCP 基础

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

客户：

```
Transmission Control Protocol, Src Port: 54280, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 54280
    Destination Port: 80
    [Stream index: 7]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 840377001
    [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 .... .... = Accurate ECN: Not set
      .... 0... .... = Congestion Window Reduced: 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
      .... .... ...0 = Fin: Not set
      [TCP Flags: ··········S·]
```

序号为840377001。将标记SYN置为1.

服务器：

```
Transmission Control Protocol, Src Port: 80, Dst Port: 54280, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 54280
    [Stream index: 7]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 3755220972
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 840377002
    1000 .... = Header Length: 32 bytes (8)
  Flags: 0x012 (SYN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Accurate ECN: Not set
      .... 0... .... = Congestion Window Reduced: Not set
      .... .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·]
    Window: 29200
```

序号为3755220972。同样将标记SYN置为1.

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

如1）问第二图，即为SYNACK报文段。序号为3755220972，acknowlegement=1，表示收到SYN报文。Acknowlegement number=840377002，为客户发送的报文序号加1.将ACK和SYN标记全置为1来表示为SYNACK报文。

3) 你能从捕获的数据包中分析出tcp 三次握手过程吗？

可以，这三个报文即三次握手：

Source	Destination	Protocol	Length	Info
172.20.70.16	128.119.245.12	TCP	66	54280 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
128.119.245.12	172.20.70.16	TCP	66	80 → 54280 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1360 SACK_PERM WS=128
172.20.70.16	128.119.245.12	TCP	54	54280 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=0

第三个报文中，可以看到发送方和接收方均确认了报文序号和窗口大小。

```
Transmission Control Protocol, Src Port: 54280, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
    Source Port: 54280
    Destination Port: 80
    [Stream index: 7]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 840377002
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 3755220973
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Accurate ECN: Not set
      .... 0... .... = Congestion Window Reduced: Not set
```

4) 包含HTTP POST 命令的TCP 报文段的序号是多少？

如图，该报文为包含POST的报文段。

```
Transmission Control Protocol, Src Port: 54280, Dst Port: 80, Seq: 1, Ack: 1, Len: 785
    Source Port: 54280
    Destination Port: 80
    [Stream index: 7]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 785]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 840377002
    [Next Sequence Number: 786    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 3755220973
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x018 (PSH, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Accurate ECN: Not set
      .... 0... .... = Congestion Window Reduced: Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
```

```
0030  02 03 e3 f1 00 00 50 4f  53 54 20 2f 77 69 72 65   ......PO ST /wire
0040  73 68 61 72 6b 2d 6c 61  62 73 2f 6c 61 62 33 2d   shark-la bs/lab3-
0050  31 2d 72 65 70 6c 79 2e  68 74 6d 20 48 54 54 50   1-reply. htm HTTP
0060  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 67 61 69 61   /1.1··Ho st: gaia
0070  2e 63 73 2e 75 6d 61 73  73 2e 65 64 75 0d 0a 43   .cs.umas s.edu··C
0080  6f 6e 6e 65 63 74 69 6f  6e 3a 20 6b 65 65 70 2d   onnectio n: keep-
0090  61 6c 69 76 65 0d 0a 43  6f 6e 74 65 6e 74 2d 4c   alive··C ontent-L
00a0  65 6e 67 74 68 3a 20 31  35 32 33 32 31 0d 0a 43   ength: 1 52321··C
00b0  61 63 68 65 2d 43 6f 6e  74 72 6f 6c 3a 20 6d 61   ache-Con trol: ma
00c0  78 2d 61 67 65 3d 30 0d  0a 55 70 67 72 61 64 65   x-age=0· ·Upgrade
00d0  2d 49 6e 73 65 63 75 72  65 2d 52 65 71 75 65 73   -Insecur e-Reques
00e0  74 73 3a 20 31 0d 0a 4f  72 69 67 69 6e 3a 20 68   ts: 1··O rigin: h
00f0  74 74 70 3a 2f 2f 67 61  69 61 2e 63 73 2e 75 6d   ttp://ga ia.cs.um
0100  61 73 73 2e 65 64 75 0d  0a 43 6f 6e 74 65 6e 74   ass.edu· ·Content
```

序号为840377002

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

如图，为此报文段。

```
> Internet Protocol Version 4, Src: 172.20.70.16, Dst: 128.119.245.12
v Transmission Control Protocol, Src Port: 54280, Dst Port: 80, Seq: 6226, Ack: 1, Len: 1360
    Source Port: 54280
    Destination Port: 80
    [Stream index: 7]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 1360]
    Sequence Number: 6226    (relative sequence number)
    Sequence Number (raw): 840383227
    [Next Sequence Number: 7586    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 3755220973
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 515
    [Calculated window size: 131840]
    [Window size scaling factor: 256]
    Checksum: 0x722b [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  v [Timestamps]
      [Time since first frame in this TCP stream: 5.430216000 seconds]
      [Time since previous frame in this TCP stream: 0.000000000 seconds]
  > [SEQ/ACK analysis]
    TCP payload (1360 bytes)
    [Reassembled PDU in frame: 220]
    TCP segment data (1360 bytes)
```

序号为840383227，发送时间如timestamps，为第一个TCP帧的5.430216秒后。

```
v Transmission Control Protocol, Src Port: 80, Dst Port: 54280, Seq: 1, Ack: 7586, Len: 0
    Source Port: 80
    Destination Port: 54280
    [Stream index: 7]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 3755220973
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 7586    (relative ack number)
    Acknowledgment number (raw): 840384587
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 347
    [Calculated window size: 44416]
    [Window size scaling factor: 128]
    Checksum: 0x0054 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  v [Timestamps]
      [Time since first frame in this TCP stream: 5.710102000 seconds]
      [Time since previous frame in this TCP stream: 0.000084000 seconds]
  > [SEQ/ACK analysis]
```

上图为对应的ACK报文，这里可以看到是采用滑动窗口协议。

接收时间戳如箭头所指。

6) 前六个TCP 报文段的长度各是多少？

72	7.687573	172.20.70.16	128.119.245.12	TCP	839	54280 → 80 [PSH, ACK] Seq=1 Ack=1 Win=131840 Len=785 [TCP segment of a reassembled PDU]
73	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=786 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
74	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=2146 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
75	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=3506 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
76	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=4866 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
77	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=6226 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
78	7.687650	172.20.70.16	128.119.245.12	TCP	1414	54280 → 80 [ACK] Seq=7586 Ack=1 Win=131840 Len=1360 [TCP segment of a reassembled PDU]

分别为785、1360、1360、1360、1360、1360.

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

53	2.536819	128.119.245.12	172.20.70.16	TCP	66	80 → 54280 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1360 SACK_PERM WS=128

如图，最小时窗口大小为29200.发送报文数量有限，不存在不够用的情况。

8) 在跟踪文件中是否有重传的报文段？进行判断的依据是什么？

没有，客户发送的报文序号严格单调递增，没有重复的，

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

开始发送文件时间为：7.687573s

72 7.687573	172.20.70.16	128.119.245.12	TCP	839 54280 → 80 [PSH, ACK] Seq=1 Ack=1 Win=131840 Len=785 [TCP segment of a reassembled PDU]

收到最后一个ACK时间为：8.809907s

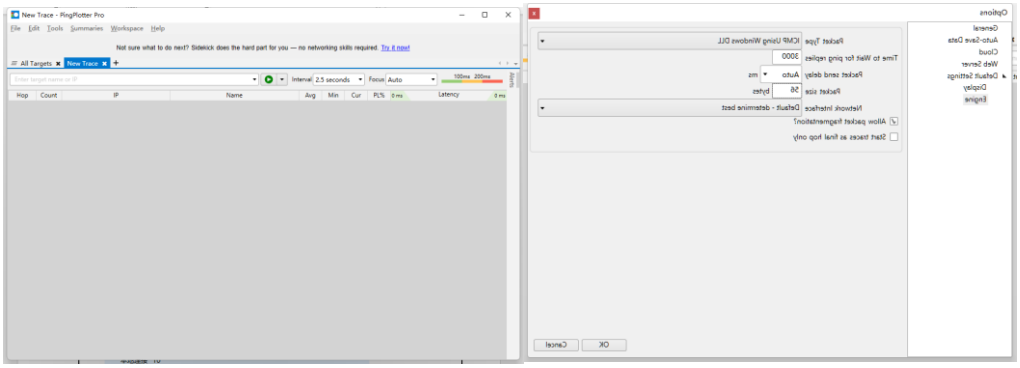
259 8.809907	128.119.245.12	172.20.70.16	TCP	56 80 → 54280 [ACK] Seq=1 Ack=153107 Win=288768 Len=0

共传输153107-1=153106字节，耗时1.122334s。

故吞吐量为：136.418KBPS.

四. IP分析

下载pingPlotter程序并设置packet：



根据指导完成操作后终止抓包。

ip and ip.addr==69.63.184.14

No.	Time	Source	Destination	Protocol	Length	Info
2006	21.425289	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=56ea) [Reassembled in #2008]
2007	21.425289	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=56ea) [Reassembled in #2008]
2008	21.425289	172.20.70.16	69.63.184.14	ICMP	554	Echo (ping) request id=0x0001, seq=1105/20740, ttl=255 (no response found!)
2009	21.459971	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=56eb) [Reassembled in #2011]
2010	21.459971	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=56eb) [Reassembled in #2011]
2011	21.459971	172.20.70.16	69.63.184.14	ICMP	554	Echo (ping) request id=0x0001, seq=1106/20996, ttl=1 (no response found!)
2012	21.465545	10.0.3.0	172.20.70.16	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2013	21.496037	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=56ec) [Reassembled in #2015]
2014	21.496037	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=56ec) [Reassembled in #2015]
2015	21.496037	172.20.70.16	69.63.184.14	ICMP	554	Echo (ping) request id=0x0001, seq=1107/21252, ttl=2 (no response found!)
2016	21.500041	192.168.82.1	172.20.70.16	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2020	21.531040	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=56ed) [Reassembled in #2022]
2021	21.531040	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=56ed) [Reassembled in #2022]
2022	21.531040	172.20.70.16	69.63.184.14	ICMP	554	Echo (ping) request id=0x0001, seq=1108/21508, ttl=3 (no response found!)
2023	21.535296	10.1.0.2	172.20.70.16	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2027	21.566160	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=56ee) [Reassembled in #2029]
2028	21.566160	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=56ee) [Reassembled in #2029]
2029	21.566160	172.20.70.16	69.63.184.14	ICMP	554	Echo (ping) request id=0x0001, seq=1109/21764, ttl=4 (no response found!)
2030	21.576042	111.40.55.129	172.20.70.16	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2034	21.601270	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=56ef) [Reassembled in #2036]
2035	21.601270	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=56ef) [Reassembled in #2036]
2036	21.601270	172.20.70.16	69.63.184.14	ICMP	554	Echo (ping) request id=0x0001, seq=1110/22020, ttl=5 (no response found!)
2037	21.608023	111.41.85.5	172.20.70.16	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2039	21.637561	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=56f0) [Reassembled in #2041]
2040	21.637561	172.20.70.16	69.63.184.14	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=56f0) [Reassembled in #2041]
2041	21.637561	172.20.70.16	69.63.184.14	ICMP	554	Echo (ping) request id=0x0001, seq=1111/22276, ttl=6 (no response found!)
2042	21.644035	218.203.72.1	172.20.70.16	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)

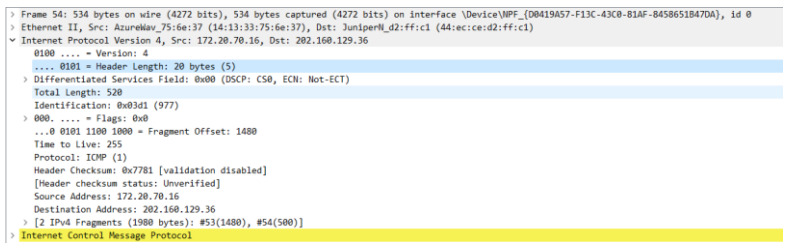
这里访问的是www.google.com

1) 你主机的IP地址是什么？

172.20.70.16

2) 在IP数据包头中，上层协议（upper layer）字段的值是什么？

是IPV4协议。



3) IP头有多少字节？该IP数据包的净载为多少字节？并解释你是怎样确定该IP数据包的净载大小的？

IP头有20个字节。净载为500字节，由total length=520，520-20=500可得。

4) 该IP数据包分片了吗？解释你是如何确定该P数据包是否进行了分片

分片了，由标志位可得，而且前面还有俩个同一分组的包。

```
▾ 000. .... = Flags: 0x0
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
   ...0 1011 1001 0000 = Fragment Offset: 2960
```

1) 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？

ID、TTL、Header checksum这三个字段总在变化。

2) 哪些字段必须保持常量？哪些字段必须改变？为什么？

ID、TTL、Header checksum这三个字段总在变化。ID用来标识不同的数据，TTL每次路过一个路由器会减1，由于前两个发生改变，头部校验和Header checksum也会发生改变。MF，数据包长度，源和目的IP地址可能会变。除以上的三个字段其他正常情况下，不发生改变。

3) 描述你看到的IP数据包Identification字段值的形式。

单调递增。

```
▾ Internet Protocol Version 4, Src: 192.168.82.1, Dst: 172.20.70.16
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  › Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x0000 (0)
  › 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 254
    Protocol: ICMP (1)
    Header Checksum: 0xb7f6 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.82.1
    Destination Address: 172.20.70.16
```

1) Identification字段和TTL字段的值是什么？

0，255.

2) 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？

保持不变。原因：IP是无连接服务，相同的标识是为了分段后组装成同一段，给同一个主机返回的标识不代表序号，因此Identification字段不变；又因为 是第一跳路由器发回的数据报，所以TTL字段是最大值-1（254）

```
▾ Internet Protocol Version 4, Src: 172.20.70.16, Dst: 202.160.129.36
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  › Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 520
    Identification: 0x03d1 (977)
  › 000. .... = Flags: 0x0
    ...0 0101 1100 1000 = Fragment Offset: 1480
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x7781 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.70.16
    Destination Address: 202.160.129.36
  ▾ [2 IPv4 Fragments (1980 bytes): #53(1480), #54(500)]
      [Frame: 53, payload: 0-1479 (1480 bytes)]
      [Frame: 54, payload: 1480-1979 (500 bytes)]
      [Fragment count: 2]
      [Reassembled IPv4 length: 1980]
      [Reassembled IPv4 data: 080035bb0001068820202020202020202020202020202020202020202020202020202020…]
▾ Internet Control Message Protocol
```

1) 该消息是否被分解成不止一个IP数据报？
是，分成俩个。

2) 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？

```
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
...0 0000 0000 0000 = Fragment Offset: 0
```

Don’t fragments为0表示被进行分片，More fragments为1表示不是最后一个分片。长度为1480B

```
Internet Protocol Version 4, Src: 172.20.70.16, Dst: 69.63.184.14
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 540
    Identification: 0x56ea (22250)
  v 000. .... = Flags: 0x0
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
    ...0 1011 1001 0000 = Fragment Offset: 2960
    Time to Live: 255
    Protocol: ICMP (1)
    Header Checksum: 0x7212 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.20.70.16
    Destination Address: 69.63.184.14
  v [3 IPv4 Fragments (3480 bytes): #2006(1480), #2007(1480), #2008(520)]
      [Frame: 2006, payload: 0-1479 (1480 bytes)]
      [Frame: 2007, payload: 1480-2959 (1480 bytes)]
      [Frame: 2008, payload: 2960-3479 (520 bytes)]
      [Fragment count: 3]
      [Reassembled IPv4 length: 3480]
      [Reassembled IPv4 data: 080019d400010451202020202020202020202020202020202020202020202
> Internet Control Message Protocol
```

1) 原始数据包被分成了多少片？
3片。

2) 这些分片中IP数据报头部哪些字段发生了变化？
Fragment offset,还有more fragments标志位。

五. 抓取ARP数据包

1. 查看ARP缓存（部分）

```
PS C:\Users\DQ\Desktop> arp -a

接口: 192.168.158.1 --- 0x3
  Internet 地址         物理地址              类型
  192.168.158.255       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

接口: 192.168.109.1 --- 0xb
  Internet 地址         物理地址              类型
  192.168.109.255       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
```

2. Ping我的手机：

```
C:\Windows\system32>ping 172.20.79.146

正在 Ping 172.20.79.146 具有 32 字节的数据:
来自 172.20.79.146 的回复: 字节=32 时间=100ms TTL=63
来自 172.20.79.146 的回复: 字节=32 时间=8ms TTL=63
来自 172.20.79.146 的回复: 字节=32 时间=24ms TTL=63
来自 172.20.79.146 的回复: 字节=32 时间=32ms TTL=63

172.20.79.146 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 8ms, 最长 = 100ms, 平均 = 41ms
```

3. 抓取ARP分组：

1	0.000000	AzureWav_75:6e:37	Broadcast	ARP	42 Who has 172.20.70.15? Tell 172.20.70.16
2	0.003084	JuniperN_d2:ff:c1	AzureWav_75:6e:37	ARP	60 172.20.70.15 is at 44:ec:ce:d2:ff:c1
3	24.865381	AzureWav_75:6e:37	Broadcast	ARP	42 Who has 172.20.70.60? Tell 172.20.70.16
4	24.867239	JuniperN_d2:ff:c1	AzureWav_75:6e:37	ARP	60 172.20.70.60 is at 44:ec:ce:d2:ff:c1
5	53.374719	AzureWav_75:6e:37	Broadcast	ARP	42 Who has 172.20.79.146? Tell 172.20.70.16
6	53.377455	JuniperN_d2:ff:c1	AzureWav_75:6e:37	ARP	60 172.20.79.146 is at 44:ec:ce:d2:ff:c1
7	56.169865	AzureWav_75:6e:37	JuniperN_d2:ff:c1	ARP	42 Who has 172.20.0.1? Tell 172.20.70.16
8	56.178535	JuniperN_d2:ff:c1	AzureWav_75:6e:37	ARP	60 172.20.0.1 is at 44:ec:ce:d2:ff:c1

（1） 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?

是每一个IP地址与MAC物理地址的转换映射条目。

（2） 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,回答下面的问题：

1) 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？

以太网的 ARP 请求和应答的分组格式，如图 6-11 所示。

图 6-11 ARP 请求和应答的分组格式

如上图所示。

2) 如何判断一个ARP数据是请求包还是应答包？

查看OP字段。OP为1为查询，为2为响应。

```
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: AzureWav_75:6e:37 (14:13:33:75:6e:37)
    Sender IP address: 172.20.70.16
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 172.20.79.146
```

```
0000  ff ff ff ff ff ff 14 13  33 75 6e 37 08 06 00 01   ........ 3un7....
0010  08 00 06 04 00 01 14 13  33 75 6e 37 ac 14 46 10   ........ 3un7..F.
0020  00 00 00 00 00 00 ac 14  4f 92                      ........ O.
```

3) 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？

因为你询问的时候不清楚哪个节点有对应IP的条目，所以需要广播。而响应时知道是哪个节点发出了询问，故可以明确目的地址。

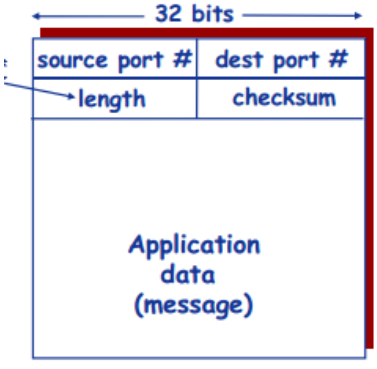
六．抓取UDP数据包

抓取到数据包如下：（查询可知QQ使用端口为4000）

udp and udp.port==8000

o.	Time	Source	Destination	Protocol	Length	Info
3	0.642300	39.156.132.122	172.20.70.16	OICQ	129	OICQ Protocol
4	0.840017	39.156.132.122	172.20.70.16	OICQ	129	OICQ Protocol
5	1.196106	39.156.132.122	172.20.70.16	OICQ	129	OICQ Protocol
7	3.852057	39.156.132.122	172.20.70.16	OICQ	129	OICQ Protocol
8	4.528322	39.156.132.122	172.20.70.16	OICQ	129	OICQ Protocol
9	5.096200	39.156.132.122	172.20.70.16	OICQ	129	OICQ Protocol
10	5.649451	172.20.70.16	39.156.132.122	OICQ	89	OICQ Protocol
12	5.658923	172.20.70.16	39.156.132.122	UDP	193	54476 → 8000 Len=151
13	5.667485	172.20.70.16	39.156.132.122	UDP	145	54476 → 8000 Len=103
14	5.667621	172.20.70.16	39.156.132.122	UDP	81	54476 → 8000 Len=39
15	5.667681	172.20.70.16	39.156.132.122	OICQ	81	OICQ Protocol
19	5.689029	172.20.70.16	39.156.132.122	OICQ	145	OICQ Protocol
20	5.689137	172.20.70.16	39.156.132.122	UDP	865	54476 → 8000 Len=823
25	5.702570	39.156.132.122	172.20.70.16	OICQ	89	OICQ Protocol
30	5.710258	39.156.132.122	172.20.70.16	OICQ	73	OICQ Protocol
31	5.722212	39.156.132.122	172.20.70.16	OICQ	433	OICQ Protocol
32	5.723989	39.156.132.122	172.20.70.16	UDP	801	8000 → 54476 Len=759
34	5.729947	39.156.132.122	172.20.70.16	UDP	249	8000 → 54476 Len=207
36	5.745317	172.20.70.16	39.156.132.122	UDP	121	54476 → 8000 Len=79
37	5.745437	172.20.70.16	39.156.132.122	UDP	121	54476 → 8000 Len=79
38	5.745510	172.20.70.16	39.156.132.122	UDP	121	54476 → 8000 Len=79
39	5.745587	172.20.70.16	39.156.132.122	UDP	121	54476 → 8000 Len=79
43	5.764581	39.156.132.122	172.20.70.16	UDP	89	8000 → 54476 Len=47
46	5.812036	39.156.132.122	172.20.70.16	UDP	257	8000 → 54476 Len=215
47	5.814831	39.156.132.122	172.20.70.16	UDP	273	8000 → 54476 Len=231
48	5.815298	39.156.132.122	172.20.70.16	UDP	273	8000 → 54476 Len=231
49	5.815298	39.156.132.122	172.20.70.16	UDP	257	8000 → 54476 Len=215

1) 消息是基于UDP的还是TCP的？
   UDP。。。。。。
2) 你的主机ip地址是什么？目的主机ip地址是什么？
   我的：172.20.70.16 目的：39.156.132.122
3) 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？
   我的：54476 目的：8000
4) 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？
   格式如图：俩字节的源端口，俩字节的目的端口，俩字节的长度，俩字节的首部检验和，余下的为内容。

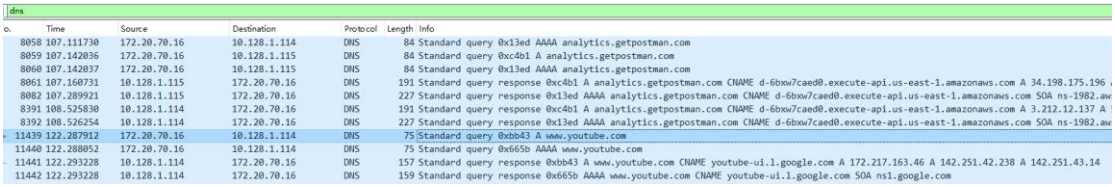



```
User Datagram Protocol, Src Port: 54476, Dst Port: 8000
    Source Port: 54476
    Destination Port: 8000
    Length: 159
    Checksum: 0x9e71 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  > [Timestamps]
    UDP payload (151 bytes)
> Data (151 bytes)
0000  44 ec ce d2 ff c1 14 13  33 75 6e 37 08 00 45 00   D···  3un7··E·
0010  00 b3 b2 bd 00 00 80 11  e9 41 ac 14 46 10 27 9c   ····  ·A··F·'·
0020  84 7a d4 cc 1f 40 00 9f  9e 71 02 3b 1d 03 60 5f   ·z···@··q·;··`_
```

5) 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？
   因为服务器需返回接收的结果给客户端。因为服务器只提供了一次返回的ACK，所以不保证数据一定送达。可以看出UDP是无连接的。UDP数据包没有序列号，因此不能像TCP协议那样先握手再发送数据，因为每次只发送一个数据报，然后等待服务器响应。

七． DNS分析

抓到包如下：（这里访问的是www.youtube.com）



打开看一下：

```
Domain Name System (query)
    Transaction ID: 0xbb43
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v www.youtube.com: type A, class IN
        Name: www.youtube.com
        [Name Length: 15]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
    [Response In: 11441]
```

```
0000  44 ec ce d2 ff c1 14 13  33 75 6e 37 08 00 45 00   D······· 3un7··E·
0010  00 3d 0e c9 00 00 80 11  2d d1 ac 14 46 10 0a 80   ·=····· -···F···
0020  01 72 cb 85 00 35 00 29  3b fe bb 43 01 00 00 01   ·r···5·) ;··C····
0030  00 00 00 00 00 00 03 77  77 77 07 79 6f 75 74 75   ·······w ww·youtu
0040  62 65 03 63 6f 6d 00 00  01 00 01                  be·com·· ···
```

这个是返回的报文：

```
Domain Name System (response)
    Transaction ID: 0xbb43
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 4
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v www.youtube.com: type A, class IN
        Name: www.youtube.com
        [Name Length: 15]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  v Answers
    > www.youtube.com: type CNAME, class IN, cname youtube-ui.l.google.com
    > youtube-ui.l.google.com: type A, class IN, addr 172.217.163.46
    > youtube-ui.l.google.com: type A, class IN, addr 142.251.42.238
    > youtube-ui.l.google.com: type A, class IN, addr 142.251.43.14
    [Request In: 11439]
    [Time: 0.005316000 seconds]
```

```
0000  14 13 33 75 6e 37 44 ec  ce d2 ff c1 08 00 45 00   ··3un7D· ·····E·
0010  00 8f 05 b9 00 00 3c 11  7a 8f 0a 80 01 72 ac 14   ·····<· z····r··
0020  46 10 00 35 cb 85 00 7b  d7 b4 bb 43 81 80 00 01   F··5···{ ···C····
0030  00 04 00 00 00 00 03 77  77 77 07 79 6f 75 74 75   ·······w ww·youtu
0040  62 65 03 63 6f 6d 00 00  01 00 01 c0 0c 00 05 00   be·com·· ···· ····
0050  01 00 00 03 3c 00 16 0a  79 6f 75 74 75 62 65 2d   ····<··· youtube-
0060  75 69 01 6c 06 67 6f 6f  67 6c 65 c0 18 c0 2d 00   ui·l·goo gle····-·
0070  01 00 01 00 00 00 c4 00  04 ac d9 a3 2e c0 2d 00   ········ ····.·-·
0080  01 00 01 00 00 00 c4 00  04 8e fb 2a ee c0 2d 00   ········ ···*··-·
0090  01 00 01 00 00 00 c4 00  04 8e fb 2b 0e            ········ ···+·
```

可以看到返回了四个条目，其中一个为CNAME类型，为规范主机名，三个为A类型，为规范主机名的IP地址。

实验结果：

这次实验成功完成了实验内容的七个任务，并且针对实验中问题做了思考和解答。

问题讨论：

协议总结

1. HTTP协议

1) HTTP 协议支持客户/服务器模式；

2) 简单快速：客户向服务器请求服务时，只需传送请求方法和路径；请求方法常用 GET、HEAD、POST 等，每种方法规定了客户与服务器联系的不同类型；HTTP 协议简单，服务器程序规模小，通信速度较快；

3) 灵活性：HTTP 允许传输任意类型数据对象；正在传输数据类型由Content-Type 标记；

4) 无连接：无连接是指每次连接只处理一个请求；服务器处理完客户请求，并收到客户应答后，即断开连接，节省传输时间；

5) 无状态：指协议对于事务处理无记忆能力；应答快，但传输数据量较大。

2. TCP协议

1) 提供面向连接的、可靠的、基于流的数据传输服务，传输单位是报文段；

2) 使用超时重发、数据确认等方式确保数据被正确发送至目的地。

3. IP 协议

1) 任务: 负责对数据包进行路由选择和存储转发；

2) 负责为分组交换网上的不同主机提供通信服务。在发送数据时，网络层把运输层产生的报文段和用户数据报封装成分组（IP 数据报）或包进行传送；

3) IP 协议:逐跳发送模式；根据数据包的目的地 IP 地址决定数据如何发送；如果数据包不能直接发送至目的地，IP 协议负责寻找一个合适的下一跳路由器，并将数据包交付给该路由器转发；

4) ICMP 协议：因特网控制报文协议，用于检测网络连接

4. ARP协议

地址解析协议 ARP（Address Resolution Protocol），负责完成逻辑地址向物理地址的动态映射，将 32 位逻辑地址（IP 地址）转换为 48 位的物理地址（MAC地址）

5. UDP协议

1) 为了在给定的主机上能识别多个目的地址，同时允许多个应用程序在同一台主机上工作并能独立地进行数据包的发送和接收，设计用户数据报协议 UDP。

2) UDP 使用底层的互联网协议来传送报文，同 IP 一样提供不可靠的无连接数据包传输服务。它不提供报文到达确认、排序、及流量控制等功能。

6. DNS协议

1) DNS 是一种可以将域名和 IP 地址相互映射的以层次结构分布的数据库系统。

2) DNS 系统采用递归查询请求的方式来响应用户的查询，为互联网的运行提供关键性的基础服务。目前绝大多数的防火墙和网络都会开放 DNS 服务，DNS 数据包不会被拦截，因此可以基于 DNS 协议建立隐蔽信道，从而顺利穿过防火墙，在客户端和服务器之间传输数据。

心得体会：

这次实验首先是学会了抓包软件wireshark的使用，然后在七个不同的任务中对于互联网的应用层、运输层、网络层、链路层的各层协议有了深入的认识，更加深了对于计算机网络体系结构的认识，收益匪浅。