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	孙月晴		院系	计算机科学与技术		
班级	1603104		学号	1160300901		
任课教师	刘亚维		指导教师	刘亚维		
实验地点	格物 213		实验时间	2018 年 10 月 31 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：

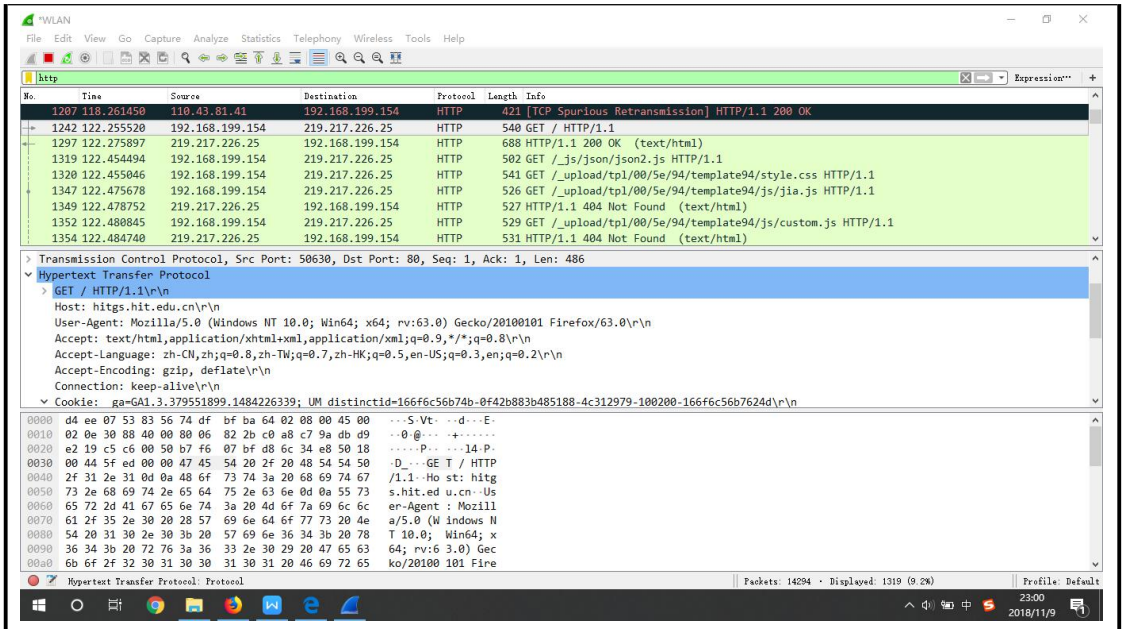
a) 利用 Wireshark 分析 DNS 协议
b) 利用 Wireshark 分析 UDP 协议
c) 利用 Wireshark 分析 ARP 协议

实验过程及结果：

(一) HTTP 分析

1）HTTP GET/response 交互

✧ 启动 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。
✧ 开始 Wireshark 分组俘获。
✧ 在打开的 Web browser窗口中输入以下地址：http://hitgs.hit.edu.cn停止分组俘获。



✧ 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？

```
> Internet Protocol Version 4, Src: 192.168.199.154, Dst: 219.217.226.25
> Transmission Control Protocol, Src Port: 50630, Dst Port: 80, Seq: 1, Ack: 1, Len: 486
v 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; rv:63.0) Gecko/20100101 Firefc
```

浏览器运行的是HTTP1.1,所访问的服务器所运行HTTP协议的版本号是1.1

✧ 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=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2\r\n
```

中文和简体中文，优先支持简体中文,中文（台湾）,中文（香港特区）,英语（美国），

✧ 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？

	Time	Source	Destination	Protocol	Length	Info
1242	122.255520	192.168.199.154	219.217.226.25	HTTP	540	GET / HTTP/1.1

本地计算机 IP：192.168.199.154

服务器 IP：219.217.226.25

✧ 从服务器向你的浏览器返回的状态代码是多少？

Time	Source	Destination	Protocol	Length	Info
122.255520	192.168.199.154	219.217.226.25	HTTP	540	GET / HTTP/1.1
122.275897	219.217.226.25	192.168.199.154	HTTP	688	HTTP/1.1 200 OK

200 OK

2）HTTP 条件 GET/response 交互

✧ 启动浏览器，清空浏览器的缓存（在浏览器中，选择“工具”菜单中的“Internet 选项”命令，在出现的对话框中，选择“删除文件”）。

✧ 启动 Wireshark 分组俘获器。开始 Wireshark 分组俘获。

✧ 在浏览器的地址栏中输入以下 URL: http://hitgs.hit.edu.cn,在你的浏览器中重新输入相同的 URL 或单击浏览器中的“刷新”按钮。

✧ 停止 Wireshark 分组俘获，在显示过滤筛选说明处输入“http”,分组列表子窗口中将只显示所俘获到的 HTTP 报文。

根据俘获窗口内容，思考以下问题：

✧ 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
  ▾ [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: hitgs.hit.edu.cn\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:63.0) Gecko/201
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\
  Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q
  Accept-Encoding: gzip, deflate\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
```

没有

✧ 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？

```
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
  Date: Sun, 11 Nov 2018 13:16:00 GMT\r\n
  Server: Apache/2.2.22 (Unix) DAV/2 mod_jk/1.2.23\r\n
  Accept-Ranges: bytes\r\n
```

明确返回了文件的内容,观察服务器返回的信息可以看到状态码

HTTP Status Code 为 304 时不返回文件

HTTP Status Code 为 200 时返回文件

✧ 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？

```
If-Modified-Since: Tue, 23 May 2017 08:18:34 GMT\r\n
```

有,后面代表时间，即文件最后的修改时间

✧ 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。

状态码是 304, 不会返回明确文件，浏览器使用没有过期的缓存文件

(二) TCP 分析

A. 俘获大量的由本地主机到远程服务器的 TCP 分组

（1） 启动浏览器，打开 http://gaia.cs.umass.edu/Wireshark-labs/alice.txt 网页，得到 ALICE'S ADVENTURES IN WONDERLAND 文本，将该文件保存到你的主机上。

（2） 打开 http://gaia.cs.umass.edu/Wireshark-labs/TCP-Wireshark-file1.html，在 Browse 按钮旁的文本框中输入保存在你的主机上的文件 ALICE'S ADVENTURES IN WONDERLAND 的全名（含路径），此时不要按“Upload alice.txt file”按钮。

（3）启动 Wireshark，开始分组俘获。

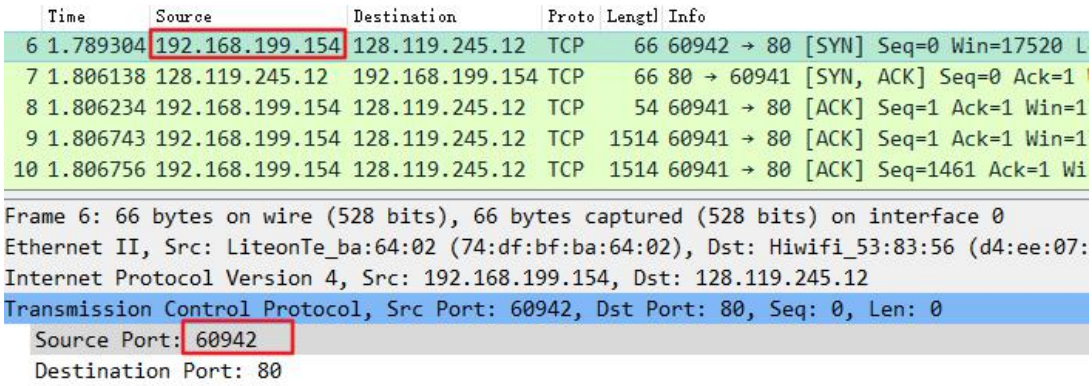
（4）在浏览器中，单击“Upload alice.txt file”按钮，将文件上传到gaia.cs.umass.edu 服务器，一旦文件上传完毕，一个简短的贺词信息将显示在你的浏览器窗口中。

（5）停止俘获。

B. 浏览追踪信息

在显示筛选规则中输入“tcp”,可以看到在本地主机和服务器之间传输的一系列 tcp 和 http 报文，你应该能看到包含 SYN 报文的三次握手。也可以看到有主机向服务器发送的一个 HTTP POST 报文和一系列的 “http continuation”报文。根据操作思考以下问题：

- 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？



客户端主机的 IP 地址是 192.168.199.154

TCP 端口号是 60942

- 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？

服务器的 IP 地址是128.119.254.12

接收 TCP 报文的端口号是80

C. TCP 基础

根据操作思考以下问题：

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

```
Sequence number: 0    (relative sequence number)
[Next sequence number: 0    (relative sequence number)]
Acknowledgment number: 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
 v .... .... ..1. = Syn: Set
```

用于初始化 TCP 连接的 TCP SYN 报文段的序号为0,SYN被设置为1说明是SYN报文段。

- 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？

```
Sequence number: 0    (relative sequence number)
[Next sequence number: 0    (relative sequence number)]
Acknowledgment number: 1    (relative ack number)
1000 .... = Header Length: 32 bytes (8)
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
```

SYNACK 报文段的序号是0，ACKnowledgement 序号是 1

ACK的值是SYN消息中的Sequence number加1所得

Acknowledgement和SYN都设置为1说明该报文段是 SYNACK 报文段

- 你能从捕获的数据包中分析出 tcp 三次握手过程吗？

Source	Destination	Prot	Lengtl	Info
192.168.199.154	128.119.245.12	TCP	66	60941 → 80 [SYN] Seq=0 Win=17520 Len=
192.168.199.154	128.119.245.12	TCP	66	60942 → 80 [SYN] Seq=0 Win=17520 Len=
128.119.245.12	192.168.199.154	TCP	66	80 → 60941 [SYN, ACK] Seq=0 Ack=1 Wir
192.168.199.154	128.119.245.12	TCP	54	60941 → 80 [ACK] Seq=1 Ack=1 Win=1740

首先客户端向服务器发送 seq=0 的建立连接的请求

然后服务器向客户端返回 seq=0,ack=1 的响应，允许建立连接

客户端收到响应，返回 seq=1,ack=1 的确认报文，连接建立

- 包含 HTTP POST 命令的 TCP 报文段的序号是多少？

…	192.168.199.154	128.119.245.12	HTTP	1131	POST /ethereal-labs/lab3-1-reply.htm
…	128.119.245.12	192.168.199.154	TCP	54	80 → 59849 [ACK] Seq=1 Ack=74461 Wir
…	128.119.245.12	192.168.199.154	TCP	54	80 → 59849 [ACK] Seq=1 Ack=75921 Wir

```
[Stream index: 8]
[TCP Segment Len: 1077]
Sequence number: 151841    (relative sequence number)
[Next sequence number: 152918    (relative sequence number)]
Acknowledgment number: 1    (relative ack number)
0101 .... = Header Length: 20 bytes (5)
Flags: 0x018 (PSH, 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
   .... .... 1... = Push: Set
   .... .... .0.. = Reset: Not set
   .... .... ..0. = Syn: Not set
   .... .... ...0 = Fin: Not set
```

序号是151841

- 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？

```
[SEQ/ACK analysis]
   [This is an ACK to the segment in frame: 235]
   [The RTT to ACK the segment was: 0.989236000 seconds]
   [iRTT: 0.269566000 seconds]
```

第6个报文段的序号是1

发送时间:Nov 11, 2018 23:14:47.154950000 中国标准时间

对应的 ACK 接收时间:对应的 ack 经过 RTT 延迟后接受，时间为 Nov 11, 2018 23:14:47.285061000 中国标准时间

- 前六个 TCP 报文段的长度各是多少？

Source	Destination	Prot	Lengtl	Info
192.168.199.154	128.119.245.12	TCP	66	60941 → 80 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 WS
192.168.199.154	128.119.245.12	TCP	66	60942 → 80 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 WS
128.119.245.12	192.168.199.154	TCP	66	80 → 60941 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0
192.168.199.154	128.119.245.12	TCP	54	60941 → 80 [ACK] Seq=1 Ack=1 Win=17408 Len=0
192.168.199.154	128.119.245.12	TCP	1514	60941 → 80 [ACK] Seq=1 Ack=1 Win=17408 Len=1460 [T
192.168.199.154	128.119.245.12	TCP	1514	60941 → 80 [ACK] Seq=1461 Ack=1 Win=17408 Len=1460
192.168.199.154	128.119.245.12	TCP	1514	60941 → 80 [ACK] Seq=2921 Ack=1 Win=17408 Len=1460
192.168.199.154	128.119.245.12	TCP	1514	60941 → 80 [ACK] Seq=4381 Ack=1 Win=17408 Len=1460
192.168.199.154	128.119.245.12	TCP	1514	60941 → 80 [ACK] Seq=5841 Ack=1 Win=17408 Len=1460
192.168.199.154	128.119.245.12	TCP	1514	60941 → 80 [ACK] Seq=7301 Ack=1 Win=17408 Len=1460

都是1460

- 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？

```
> Flags: 0x012 (SYN, ACK)
  Window size value: 29200
  [Calculated window size: 29200]
  Checksum: 0xf8c7 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
```

接受方通知给发送方的最低窗口大小为29200，即在服务器端传回的第一个ACKz中的窗口大小。够用。

- 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，从表中可以看出从源端发往目的地的序号逐渐增加，如果这其中有重传的报文段，则其序号中应该有小于其临近的分组序号的分组，图中未看到这样的分组，故没有重发片段。

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

传输的数据总量为TCP 段第一个序列号和最后的序列号的ACK之间的差值。因此，总数据是 153259 字节。整个传输时间是第一个 TCP 段（即 9 号段 1.806743 秒）的时间和最后的 ACK（即第 262 段 11.005017 秒) 时间的差值。因此，总传输时间是 11.005017-1.806743= 1.791293 秒。因此，TCP 连接的吞吐量为 153259/9.198274=16.661 KByte/s。

（三）IP 分析

A. 通过执行 traceroute 执行捕获数据包

实验步骤：

（1） 启动 Wireshark 并开始数据包捕获

（2） 启动 pingplotter 并“Address to Trace Window”域中输入目的地址。

在“# of times to Trace”域中输入“3”，这样就不过采集过多的数据。Edit->Options->Packet，将 Packet Size(in bytes,default=56)域设为 56，这样将发送一系列大小为 56 字节的包。然后按下“Trace”按钮。

（3） Edit->Options->Packet，然后将 Packet Size(in bytes,default=56) 域改为 2000，这样将发送一系列大小为 2000 字节的包。然后按下 “Resume”按钮。

（4） 最后，将 Packet Size(in bytes,default=56)域改为 3500，发送一 系列大小为 3500 字节的包。然后按下“Resume”按钮。

（5） 停止 Wireshark 的分组捕获。

B. 对捕获的数据包进行分析

（1）选择第一个你的主机发出的 ICMP Echo Request 消息，在 packet details 窗口展开数据包的 Internet Protocol 部分

思考下列问题：

- 你主机的 IP 地址是什么？

	Time	Source	Destination	Protoc	Lengtl	Info
16	1.208522	192.168.199.154	61.167.60.70	ICMP	70	Echo (ping) request id=0x0001
17	1.210647	61.167.60.70	192.168.199.154	ICMP	70	Echo (ping) reply id=0x0001

我的主机的 IP 地址是 192.168.199.154

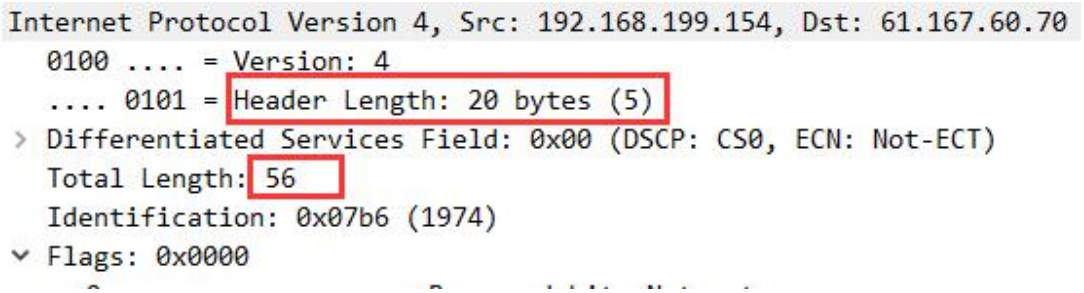
- 在 IP 数据包头中，上层协议（upper layer）字段的值是什么？

```
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x07b6 (1974)
v Flags: 0x0000
    0... .... .... .... = Reserved bit: Not set
    .0.. .... .... .... = Don't fragment: Not set
    ..0. .... .... .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment offset: 0
  Time to live: 255
  Protocol: ICMP (1)
  Header checksum: 0xb1de [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.199.154
  Destination: 61.167.60.70
```

ICMP(1)

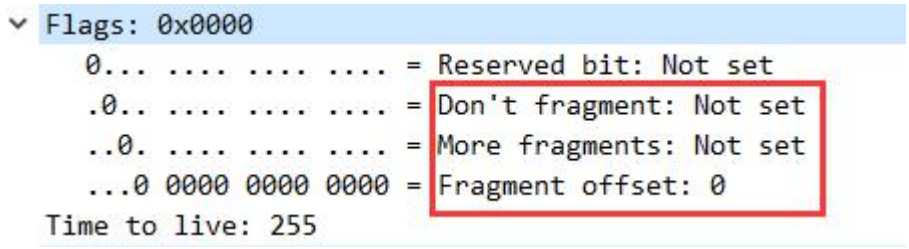
- IP 头有多少字节？该 IP 数据包的净载为多少字节？并解释你是怎样确定

该 IP 数据包的净载大小的？



IP 头有 20 字节,该 IP 数据包的净载为 56-20=36 字节

- 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



没有，分片位移为 0，More fragments 为 0 表示后面无分片。

（2）单击 Source 列按钮，这样将对捕获的数据包按源 IP 地址排序。选择第一个你的主机发出的 ICMP Echo Request 消息，在 packet details 窗口展开数据包的 Internet Protocol 部分。

思考下列问题：

- 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？

checksum,identification,time to live 字段总是发生改变

- 哪些字段必须保持常量？哪些字段必须改变？为什么？

必须保持不变的：version(版本)、header length(头部长度)、differentiated services field(区分服务)、 protocol（协议）、Source（源地址）

必须改变的：time to live、identification（标识）、header chechsum(头部检验和)

标识是源主机赋予 IP 数据报的标识符、头部检验和用于保证 IP 数据报报头的完整性。每当数据经过一台路由器时,ttl 减 1,重新计算校验和

- 描述你看到的 IP 数据包 Identification 字段值的形式。

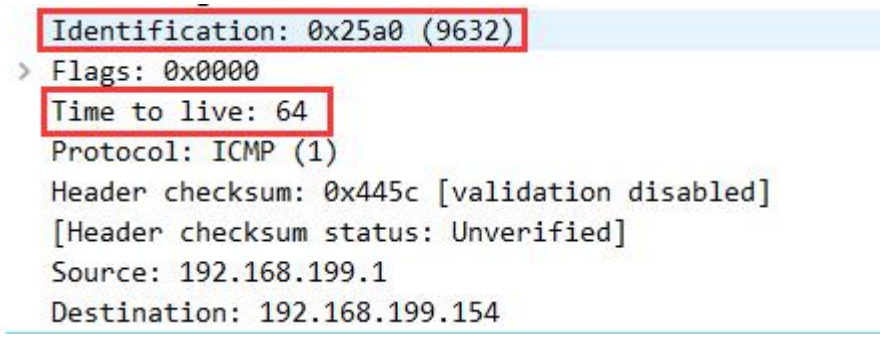
每一个 IP 数据报头部的标识号域都不一样，每次加 1

（3）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。

思考下列问题：

- Identification 字段和 TTL 字段的值是什么？



Identification 字段为 9632,TTL 字段的值是 64

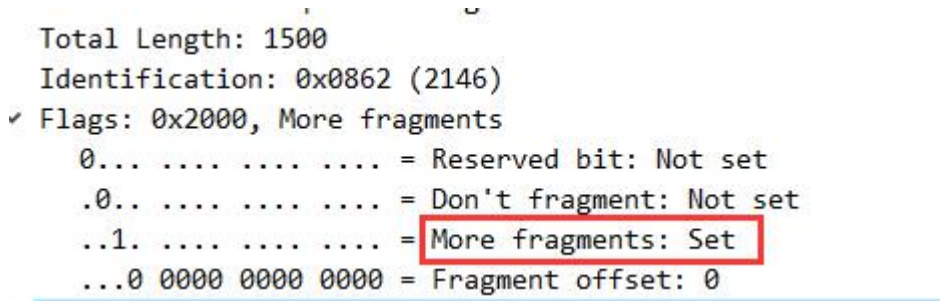
- 最近的路由器（第一跳）返回给你主机的 ICMP Time-to-live exceeded 消息中这些值是否保持不变？为什么？

每一个固定的路由器都有一个固定的 TTL 值，所以最近的那个路由器回复的所有的 ICMP TTL-exceeded 的 TTL 的值都不会改变。

（4）单击 Time 列按钮，这样将对捕获的数据包按时间排序。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。

思考下列问题：

- 该消息是否被分解成不止一个 IP 数据报？



是的

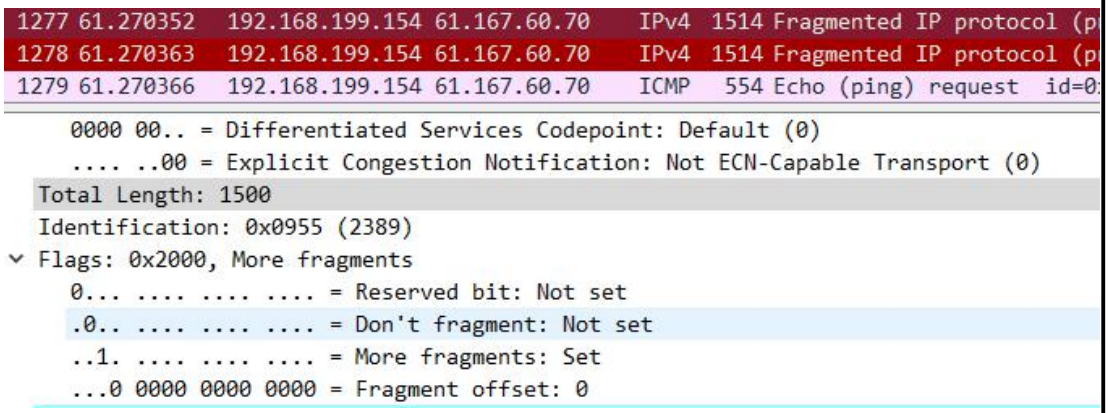
- 观察第一个 IP 分片，IP 头部的哪些信息表明数据包被进行了分片？IP 头部的哪些信息表明数据包是第一个而不是最后一个分片？该分片的长度是多少

Flags 中 More fragments 位为 1 说明数据报进行了分片;而 fragment offset 为 0 说明它是第一个分片,More fragments 位为 1 说明它不是最后一个分片;该分片的长度是 1480 字节

C. 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。

思考下列问题：

- 原始数据包被分成了多少片？



被分成了 3 片,identification 相同说明是同一组的

- 这些分片中 IP 数据报头部哪些字段发生了变化？

Total Length,Flags 中的 More fragment 和 Fragment offset,Header checksum 字段发生了变化

（四）抓取 ARP 数据包

（1）利用 MS-DOS 命令：arp 或 c:\windows\system32\arp 查看主机上 ARP 缓存的内容。

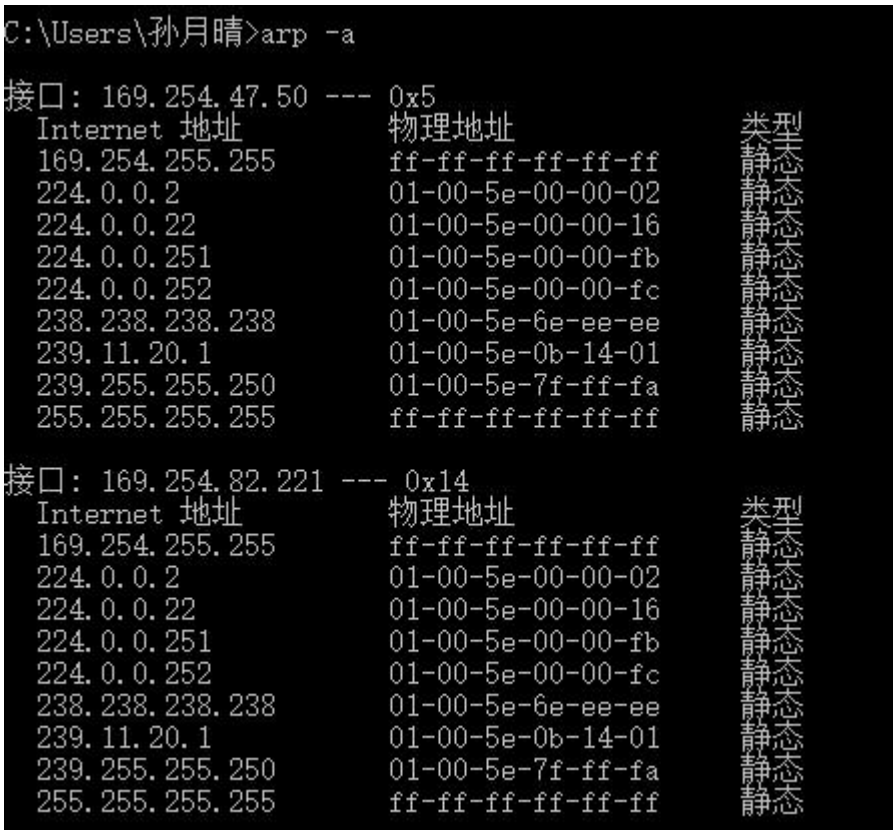
（2）在命令行模式下输入：ping 192.168.1.82（或其他 IP 地址）

（3）启动 Wireshark，开始分组俘获。

思考下面问题：

（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？

输入 apr –a 查看主机上 ARP 缓存的内容，结果如下图所示：



```
C:\Users\孙月晴>arp -a

接口: 169.254.47.50 --- 0x5
  Internet 地址         物理地址              类型
  169.254.255.255       ff-ff-ff-ff-ff-ff     静态
  224.0.0.2             01-00-5e-00-00-02     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
  238.238.238.238       01-00-5e-6e-ee-ee     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: 169.254.82.221 --- 0x14
  Internet 地址         物理地址              类型
  169.254.255.255       ff-ff-ff-ff-ff-ff     静态
  224.0.0.2             01-00-5e-00-00-02     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
  238.238.238.238       01-00-5e-6e-ee-ee     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
```

ARP 缓存中的每一列分别表示 IP 地址所对应的物理地址和类型（动态配置或静态配置）

（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：

- 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？

ARP 数据包由 9 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（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） 。

截取的一个 ARP 数据包如下：

```
Address Resolution Protocol (request)
   Hardware type: Ethernet (1)
   Protocol type: IPv4 (0x0800)
   Hardware size: 6
   Protocol size: 4
   Opcode: request (1)
   Sender MAC address: Hiwifi_53:83:56 (d4:ee:07:53:83:56)
   Sender IP address: 192.168.199.1
   Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
   Target IP address: 192.168.199.154
```

➢ 如何判断一个 ARP 数据是请求包还是应答包？

```
    Time      Source            Destination       Protoc Lengtl Info
  831 23.960352   Hiwifi_53:83:56 LiteonTe_ba:6…  ARP      42 Who has 192.168.199.154?
  514 15.100497   Hiwifi_53:83:56 LiteonTe_ba:6…  ARP      42 192.168.199.1 is at d4:ee
Frame 831: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
Ethernet II, Src: Hiwifi_53:83:56 (d4:ee:07:53:83:56), Dst: LiteonTe_ba:64:02 (74:df
> Destination: LiteonTe_ba:64:02 (74:df:bf:ba:64:02)
> Source: Hiwifi_53:83:56 (d4:ee:07:53:83:56)
   Type: ARP (0x0806)
Address Resolution Protocol (request)
   Hardware type: Ethernet (1)
   Protocol type: IPv4 (0x0800)
   Hardware size: 6
   Protocol size: 4
   Opcode: request (1)
   Sender MAC address: Hiwifi_53:83:56 (d4:ee:07:53:83:56)
   Sender IP address: 192.168.199.1
   Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
   Target IP address: 192.168.199.154
```

```
    Time      Source            Destination       Protoc Lengtl Info
  831 23.960352   Hiwifi_53:83:56 LiteonTe_ba:6…  ARP      42 Who has 192.168.199.154?
  514 15.100497   Hiwifi_53:83:56 LiteonTe_ba:6…  ARP      42 192.168.199.1 is at d4:ee
Frame 514: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
Ethernet II, Src: Hiwifi_53:83:56 (d4:ee:07:53:83:56), Dst: LiteonTe_ba:64:02 (74:df
> Destination: LiteonTe_ba:64:02 (74:df:bf:ba:64:02)
> Source: Hiwifi_53:83:56 (d4:ee:07:53:83:56)
   Type: ARP (0x0806)
Address Resolution Protocol (reply)
   Hardware type: Ethernet (1)
   Protocol type: IPv4 (0x0800)
   Hardware size: 6
   Protocol size: 4
   Opcode: reply (2)
   Sender MAC address: Hiwifi_53:83:56 (d4:ee:07:53:83:56)
   Sender IP address: 192.168.199.1
   Target MAC address: LiteonTe_ba:64:02 (74:df:bf:ba:64:02)
   Target IP address: 192.168.199.154
```

通过 OP 字段。当 OP 字段值为 0x0001 时是请求包，当 OP 字段值为

0x0002 时是应答包。

- 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局域网地址的帧中传送。

（五）抓取 UDP 数据包

（1）启动 Wireshark，开始分组捕获；
（2）发送 QQ 消息给你的好友；
（3）停止 Wireshark 组捕获；
（4）在显示筛选规则中输入“udp”并展开数据包的细节

分析 QQ 通讯中捕获到的 UDP 数据包。根据操作思考以下问题：

- 消息是基于 UDP 的还是 TCP 的？

	Time	Source	Destination	Protoc	Lengtl	Info
3	2.311434	192.168.199.154	182.254.110.91	UDP	193	4011 → 8000 Len=151
4	2.457622	182.254.110.91	192.168.199.154	UDP	73	8000 → 4011 Len=31

消息是基于 UDP

- 你的主机 ip 地址是什么？目的主机 ip 地址是什么？

我的主机 ip 是 192.168.199.154
目的主机 ip 是 182.254.110.91

- 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: 4011, Dst Port: 8000
    Source Port: 4011
    Destination Port: 8000
```

我的主机发送 QQ 消息的端口号是 4011,QQ 服务器的端口号 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？

UDP 数据报格式由 5 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。

抓取的一个 UDP 数据报如下所示：

```
User Datagram Protocol, Src Port: 4011, Dst Port: 8000
    Source Port: 4011
    Destination Port: 8000
    Length: 159
    Checksum: 0xcca3 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
Data (151 bytes)
```

- 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 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。

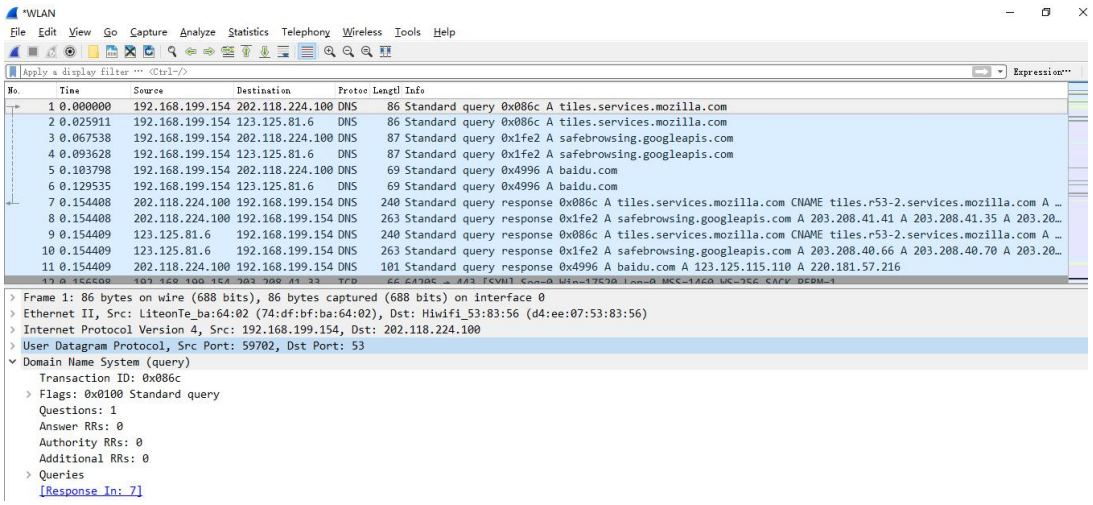
（六）利用 WireShark 进行 DNS 协议分析

（1）打开浏览器键入:www.baidu.com

（2）打开 Wireshark,启动抓包.

（3）在控制台回车执行完毕后停止抓包.

① 打开浏览器，输入 www.baidu.com,DNS 查询消息如下图：



② 我的电脑 IP 地址：192.168.199.154，本地域名服务器 IP 地址：202.118.224.100.如图：

```
Header checksum: 0x954a [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.199.154
Destination: 202.118.224.100
```

③ UDP 报文的源端口号 59702，目的端口号 53

```
User Datagram Protocol, Src Port: 59702, Dst Port: 53
    Source Port: 59702
    Destination Port: 53
    Length: 52
    Checksum: 0xc4c4 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
```

④ DNS 查询报文内容如下图，表示查询主机域名为 www.baidu.com 的主机的 IP 地址

```
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 72
  Identification: 0x723c (29244)
v Flags: 0x0000
    0... .... .... .... = Reserved bit: Not set
    .0.. .... .... .... = Don't fragment: Not set
    ..0. .... .... .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment offset: 0
  Time to live: 128
  Protocol: UDP (17)
  Header checksum: 0x954a [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.199.154
  Destination: 202.118.224.100
```

⑤ DNS 回复信息：

```
5 0.103798    192.168.199.154 202.118.224.100 DNS     69 Standard query 0x4996 A
6 0.129535    192.168.199.154 123.125.81.6    DNS     69 Standard query 0x4996 A
7 0.154408    202.118.224.100 192.168.199.154 DNS    240 Standard query response
8 0.154408    202.118.224.100 192.168.199.154 DNS    263 Standard query response
```

主机域名为 www.baidu.com 的主机 IP 地址为：202.118.224.100

```
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 226
  Identification: 0x5c67 (23655)
v Flags: 0x0000
    0... .... .... .... = Reserved bit: Not set
    .0.. .... .... .... = Don't fragment: Not set
    ..0. .... .... .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment offset: 0
  Time to live: 57
  Protocol: UDP (17)
  Header checksum: 0xf185 [validation disabled]
  [Header checksum status: Unverified]
  Source: 202.118.224.100
  Destination: 192.168.199.154
```

心得体会：

通过本次实验，我熟悉了Wireshark 的基本操作，了解对HTTP, TCP, IP, UDP, ARP, DNS各种协议的分析，验证了课程中所教授的课程知识，对协议中的报文的格式有了更深的认识，对其各个部分的作用有了一定的了解。对网络中，各个协议的工作方式，能够通过 wirdshark 进行分析，并得到各种报文的格式。深入理解了报文在网络中发送接收及转发的过程，此次实验综合所学的协议，加深了对协议的认识，也修正了之前学习中的错误理解，收获颇多。