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	实验 3：利用 Wireshark 进行协议分析					
姓名			院系			
班级			学号			
任课教师			指导教师			
实验地点			实验时间			
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：

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：

1. Wireshark 是一种可以运行在 Windows, UNIX, Linux 等操作系统上的分组分析器。Wireshark是免费的，可以从https://www.wireshark.org/download.html 得到。

2. 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 窗口中输入一下地址：http://hitgs.hit.edu.cn/news
- 停止分组俘获。

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/zhxw/list.htm,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。

3. 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，如图6-6所示，窗口如下图所示。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”。

（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”。

C. TCP基础

4. IP分析

通过分析执行 traceroute 程序发送和接收到的 IP 数据包，我们将研究 IP 数据包的各个字段，并详细研究 IP 分片。

A. 通过执行 traceroute 执行捕获数据包为了产生一系列 IP 数据报，我们利用 traceroute 程序发送具有不同大小的数据包给目的主机 X。

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。

B. 对捕获的数据包进行分析

（1）在你的捕获窗口中，应该能看到由你的主机发出的一系列ICMP Echo Request包和中间路由器返回的一系列ICMP TTL-exceeded消息。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，

（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。在“listing of captured packets”窗口，你会看到许多后续的ICMP消息（或许还有你主机上运行的其他协议的数据包）

（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

（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

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。

5. 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。

6. 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；

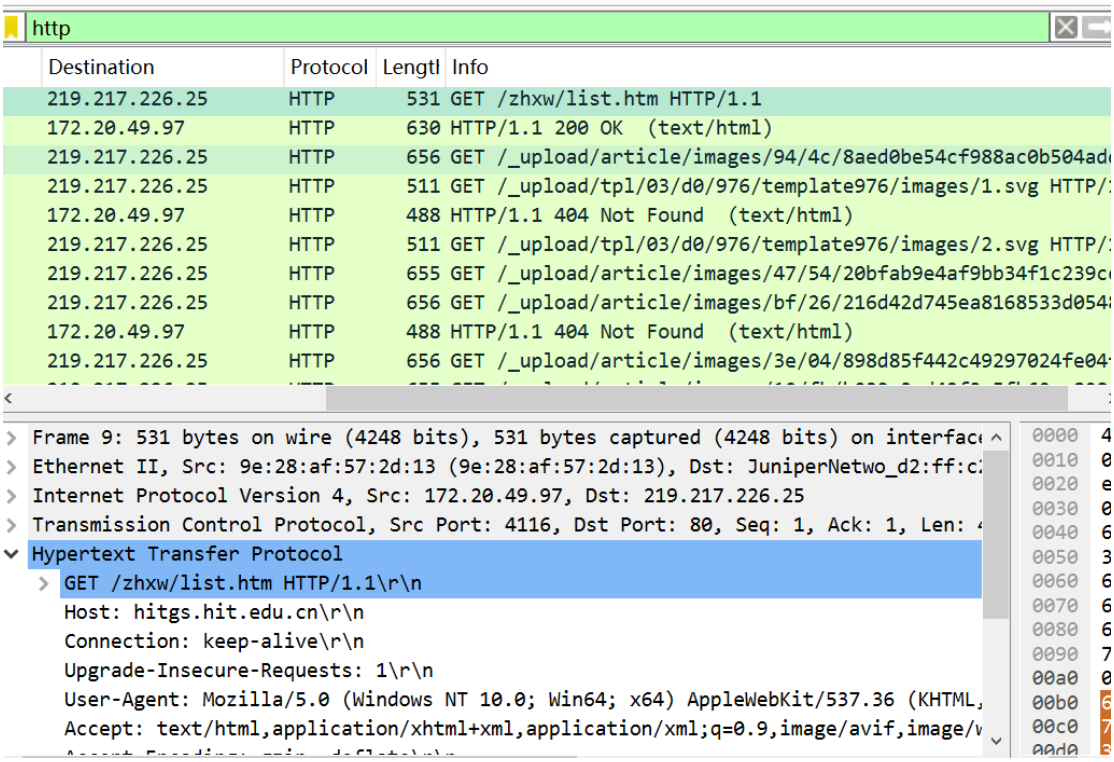
（3）停止 Wireshark 组捕获；
（4）在显示筛选规则中输入“udp”并展开数据包的细节。
7.利用 WireShark 进行 DNS 协议分析
（1）打开浏览器键入:www.baidu.com
（2）打开 Wireshark,启动抓包
（3）在控制台回车执行完毕后停止抓包。

实验结果：

1. HTTP 分析

1）HTTP GET/response 交互

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



- **你的浏览器向服务器指出它能接收何种语言版本的对象？**简体中文、任意中文

```
Host: hitgs.hit.edu.cn\r\n
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTM
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
Accept-Encoding: gzip, deflate\r\n
Accept-Language: zh-CN,zh;q=0.9\r\n
\r\n
[Full request URI: http://hitgs.hit.edu.cn/zhxw/list.htm]
[HTTP request 1/2]
[Response in frame: 16]
```

- 你的计算机的 IP 地址是多少？172.20.49.97
- 服务器http://hitgs.hit.edu.cn/zhxw/list.htm的IP 地址是多少？
  219.217.226.25

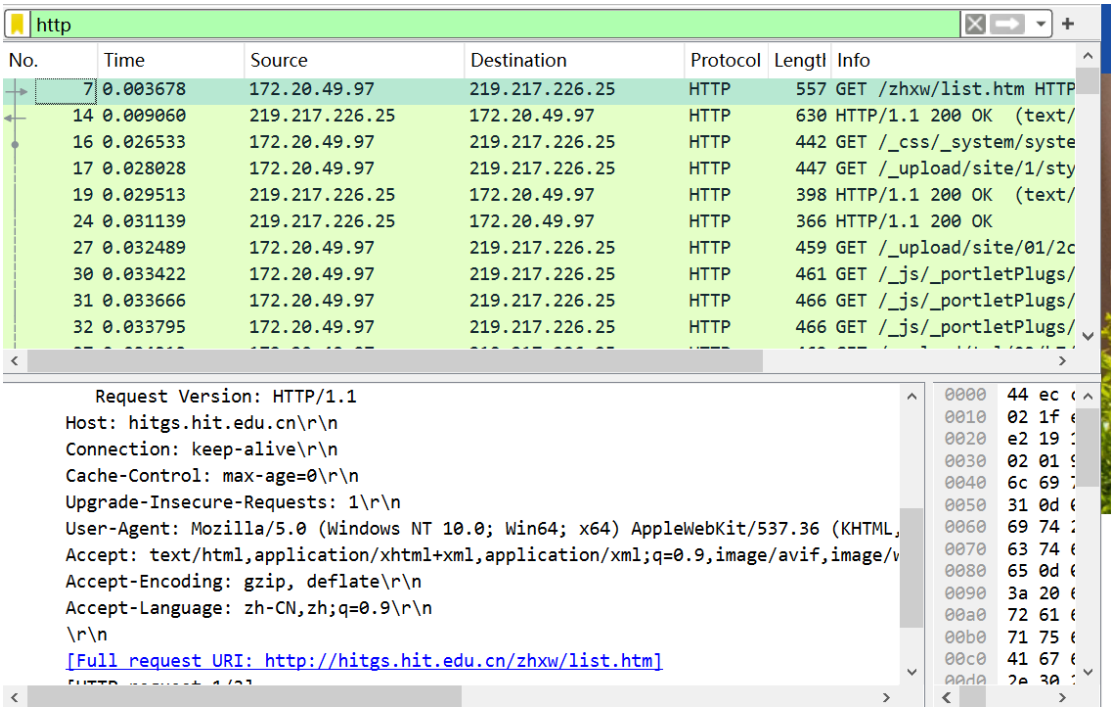
Source	Destination
172.20.49.97	219.217.226.25
219.217.226.25	172.20.49.97
172.20.49.97	219.217.226.25
172.20.49.97	219.217.226.25
219.217.226.25	172.20.49.97
172.20.49.97	219.217.226.25
172.20.49.97	219.217.226.25
172.20.49.97	219.217.226.25

- 从服务器向你的浏览器返回的状态代码是多少？ 200、404

l	Lengtl	Info
	531	GET /zhxw/list.htm HTTP/1.1
	630	HTTP/1.1 200 OK (text/html)
	656	GET /_upload/article/images/94/4c/8aed0be54cf988
	511	GET /_upload/tpl/03/d0/976/template976/images/1
	488	HTTP/1.1 404 Not Found (text/html)
	511	GET /_upload/tpl/03/d0/976/template976/images/2
	655	GET /_upload/article/images/47/54/20bfab9e4af9bb
	656	GET /_upload/article/images/bf/26/216d42d745ea81
	488	HTTP/1.1 404 Not Found (text/html)

2）HTTP 条件 GET/response 交互

- **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？** 没有，缓存文件已经被删除



- **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？**
已经明确的返回了。①返回了表示请求成功的状态码200；②返回了传输文件的大小和类型，在Content-Type和Content-Length字段中体现；③返回的数据内容就是文件的内容

```
X-Frame-Options: SAMEORIGIN\r\n
\r\n
[HTTP response 1/6]
[Time since request: 0.005382000 seconds]
[Request in frame: 7]
[Next request in frame: 16]
[Next response in frame: 19]
[Request URI: http://hitgs.hit.edu.cn/zhxw/list.htm]
Content-encoded entity body (gzip): 7584 bytes -> 43431 bytes
File Data: 43431 bytes
Line-based text data: text/html (652 lines)
```

- **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，在该部行后面跟着的信息是什么？**
存在，该行后跟着的信息是浏览器上次从服务器获取资源的时间

```
Referer: http://hitgs.hit.edu.cn/zhxw/list.htm\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: "25136-6174e9205b951-gzip"\r\n
If-Modified-Since: Tue, 30 Apr 2024 11:21:25 GMT\r\n
\r\n
[Full request URI: http://hitgs.hit.edu.cn/_upload/article/images/3e/04/898d85f4
[HTTP request 3/4]
[Prev request in frame: 360]
[Response in frame: 580]
[Next request in frame: 585]
```

- **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**

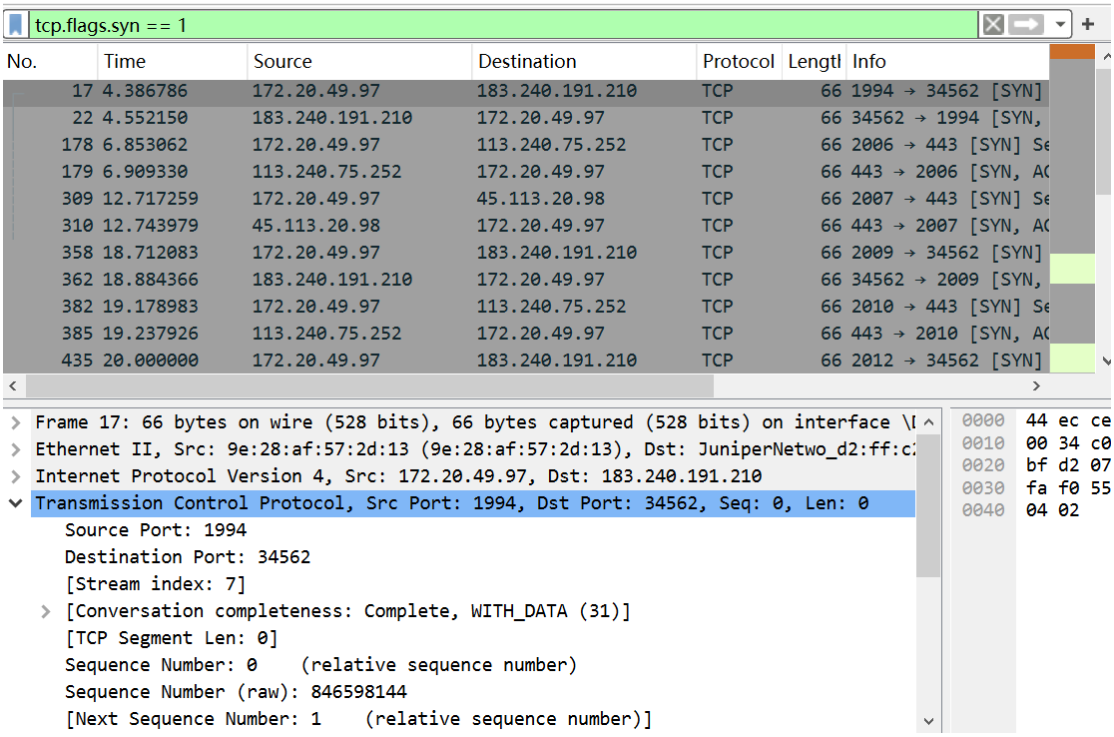
  应该为304，表示"Not Modified"，服务器不返回文件的内容，而是让客户端通过本地缓存访问，所以服务器没有明确返回文件内容。

2. TCP 分析

Congratulations!

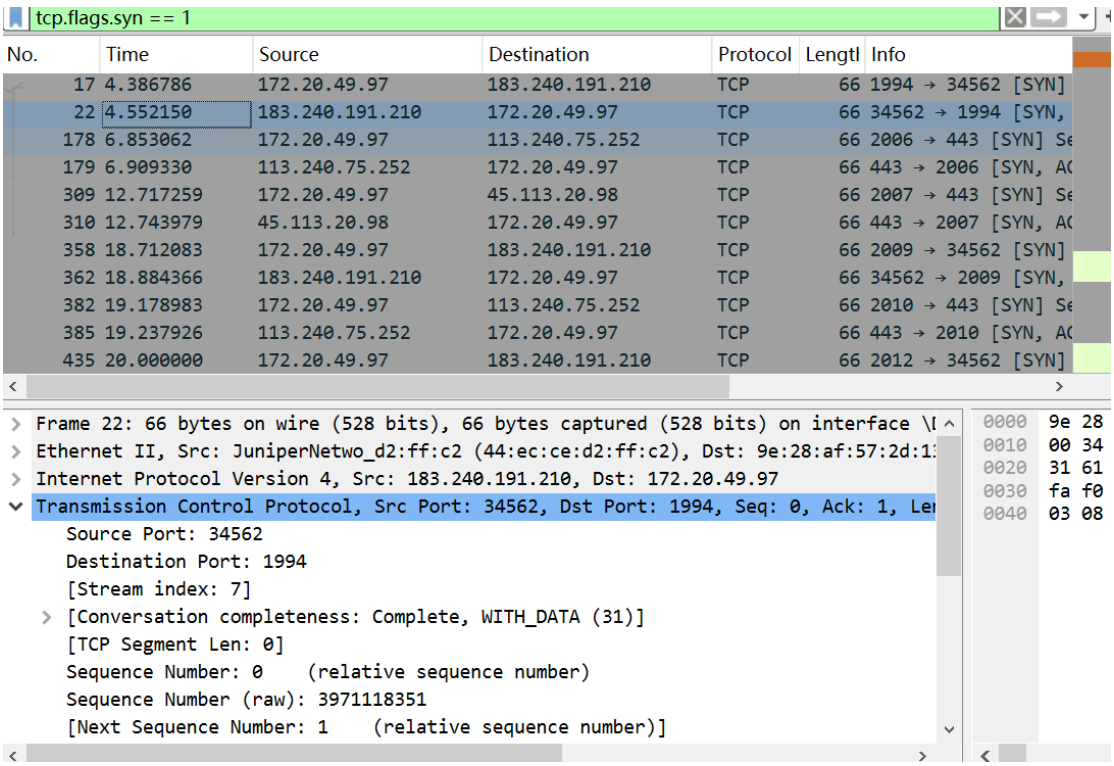
You've now transferred a copy of alice.txt from your computer to gaia.cs.umass.edu. You should now stop Wireshark packet capture. It's time to start analyzing the captured Wireshark packets!

- **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？**



主机ip:172.20.49.97,端口号：1994

- **Gaia.cs.umass.edu 服务器的 IP 地址是多少？** 183.240.191.210

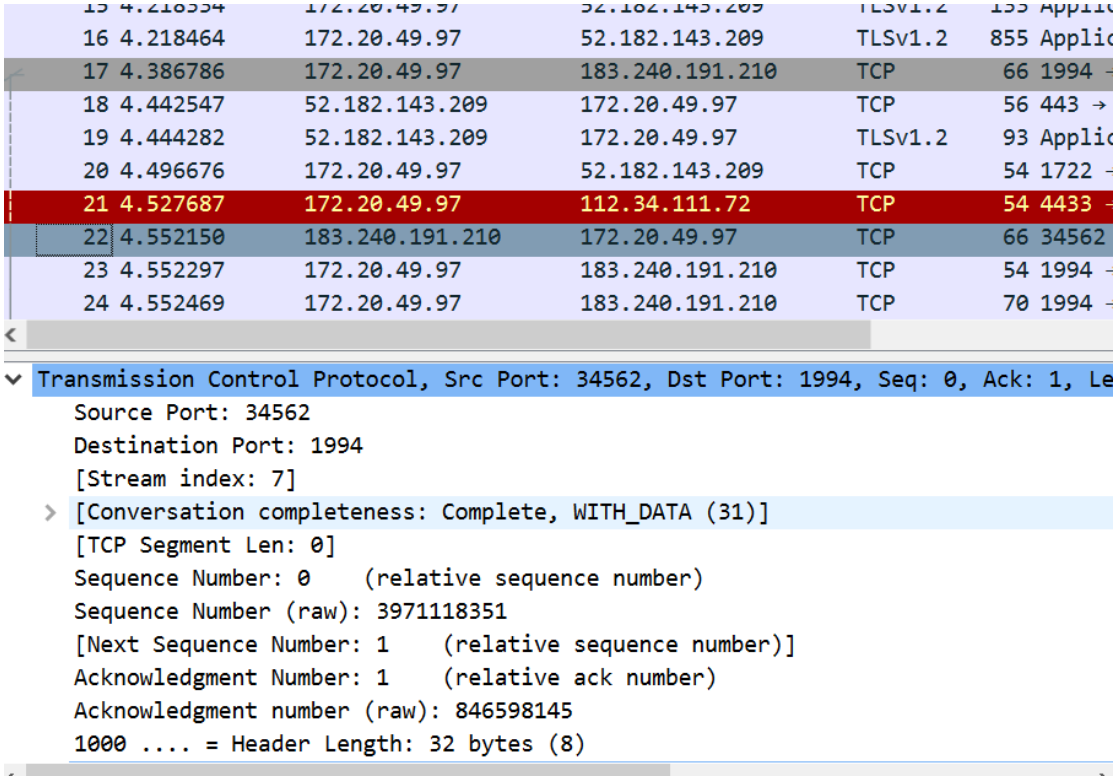


- 对这一连接，它用来发送和接收 TCP 报文的端口号是多少？

34562

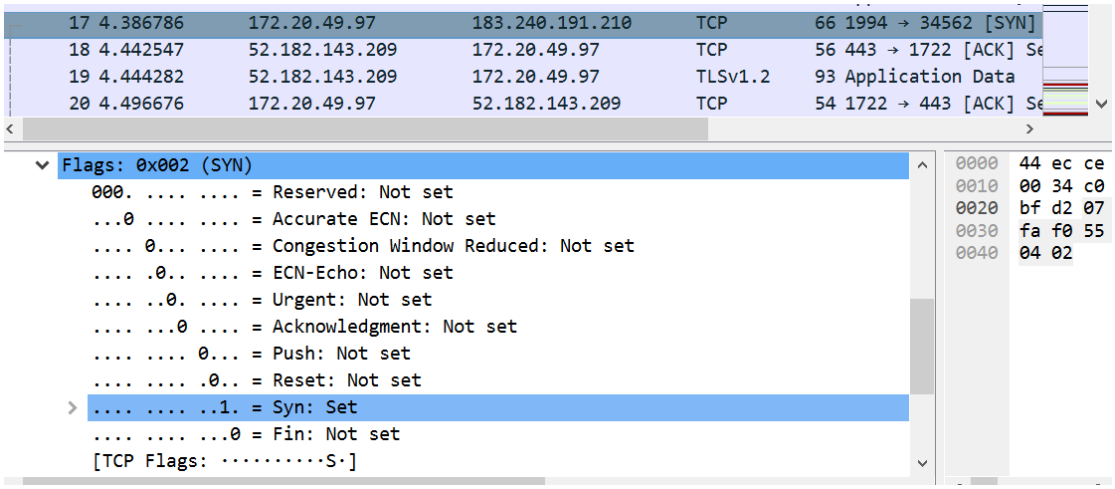
- 客户服务器之间用于初始化 TCP 连接的 TCP SYN 报文段的序号（sequence number）是多少？

3971118351



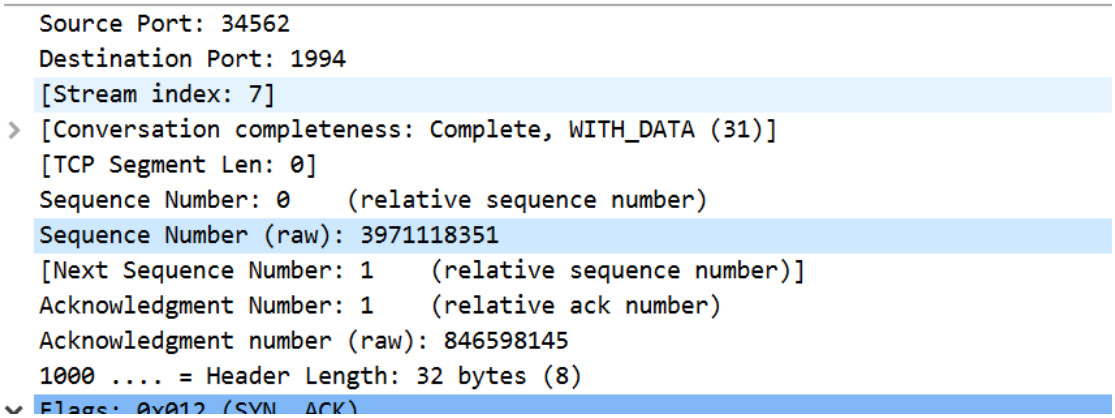
- 在该报文段中，是用什么来标示该报文段是 SYN 报文段的？

通过标志位flags的SYN字段，该字段置为1时标示该报文段是 SYN 报文段的



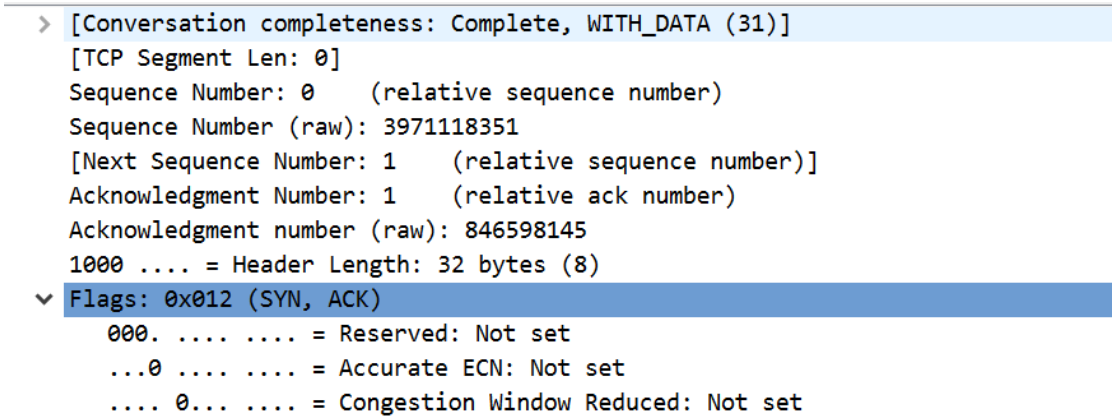
- **服务器向客户端发送的 SYNACK 报文段序号是多少？**

3971118351



- **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？**

846598145。Gaia.cs.umass.edu 服务器是通过发送方的序列号加上数据的长度来计算的



- **在该报文段中，是用什么来标示该报文段是SYNACK 报文段的？**

SYN和ACK标志位用于标识SYN-ACK报文段。如果一个报文段是SYN-ACK报文段，那么它的SYN标志位和ACK标志位都会被设置为1。

```
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·]
```

- **你能从捕获的数据包中分析出 tcp 三次握手过程吗？**

通过过滤器`tcp.flags.syn == 1`筛选带有SYN标志的TCP数据包来找到表示文件传输开始的数据包。

tcp.flags.syn == 1

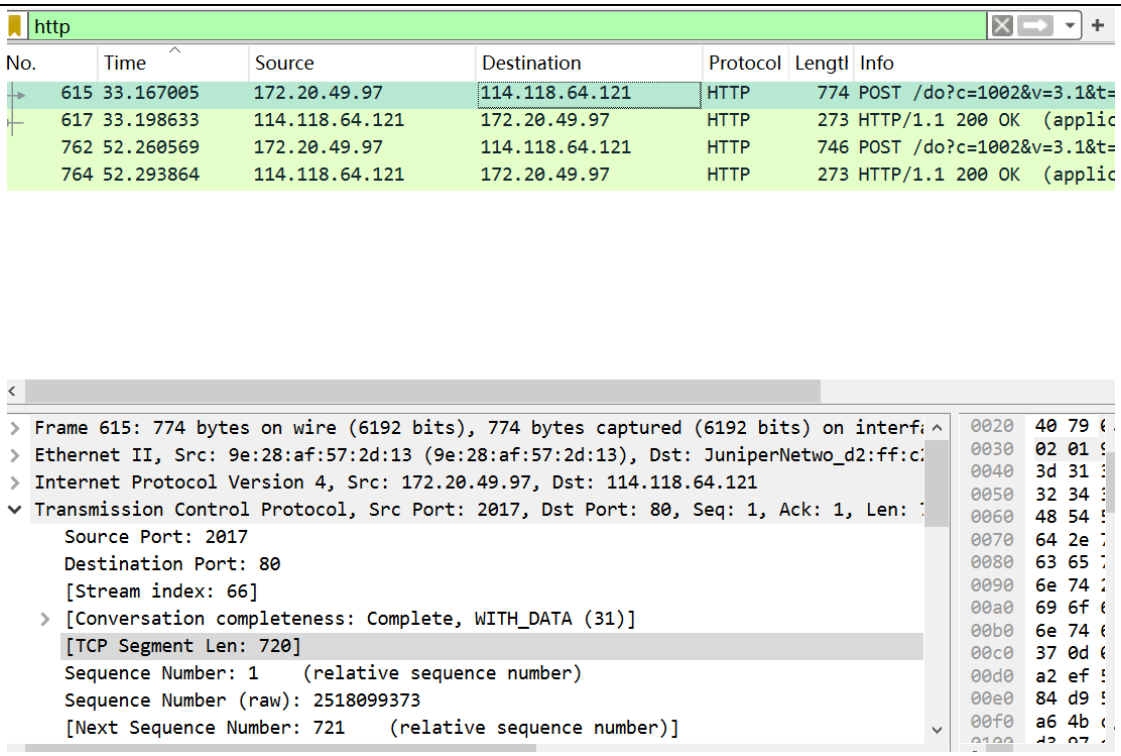
No.	Time	Source	Destination	Protocol	Lengtl	Info
17	4.386786	172.20.49.97	183.240.191.210	TCP	66	1994 → 34562 [SYN]
22	4.552150	183.240.191.210	172.20.49.97	TCP	66	34562 → 1994 [SYN,
178	6.853062	172.20.49.97	113.240.75.252	TCP	66	2006 → 443 [SYN] Se
179	6.909330	113.240.75.252	172.20.49.97	TCP	66	443 → 2006 [SYN, AC
309	12.717259	172.20.49.97	45.113.20.98	TCP	66	2007 → 443 [SYN] Se
310	12.743979	45.113.20.98	172.20.49.97	TCP	66	443 → 2007 [SYN, AC
358	18.712083	172.20.49.97	183.240.191.210	TCP	66	2009 → 34562 [SYN]
362	18.884366	183.240.191.210	172.20.49.97	TCP	66	34562 → 2009 [SYN,
382	19.178983	172.20.49.97	113.240.75.252	TCP	66	2010 → 443 [SYN] Se
385	19.237926	113.240.75.252	172.20.49.97	TCP	66	443 → 2010 [SYN, AC
435	20.000000	172.20.49.97	183.240.191.210	TCP	66	2012 → 34562 [SYN]

22	4.552150	183.240.191.210	172.20.49.97	TCP	66	34562 → 1994 [SYN, AC
23	4.552297	172.20.49.97	183.240.191.210	TCP	54	1994 → 34562 [ACK] Se

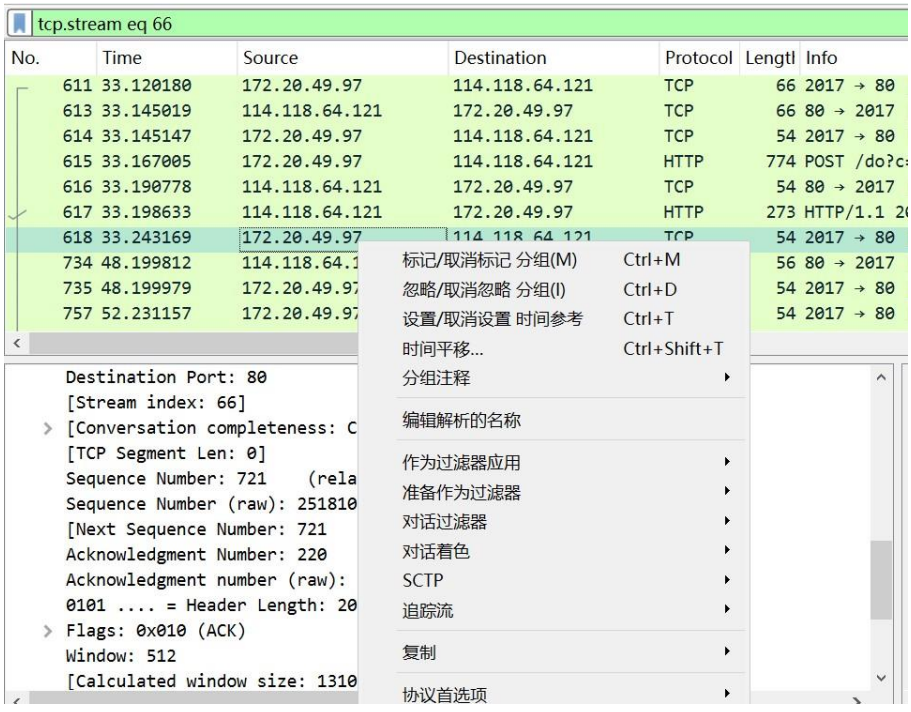
分析可知，在序号17的数据报客户端向服务器发生连接请求，在序号22的数据报服务器回复一个带有SYN和ACK标志的数据包到客户端，在序号23的数据报客户端再回复一个带有ACK标志的数据包到服务器，完成了三次握手，接下来开始进行tcp传输。

- **包含 HTTP POST 命令的 TCP 报文段的序号是多少？**

2518099373



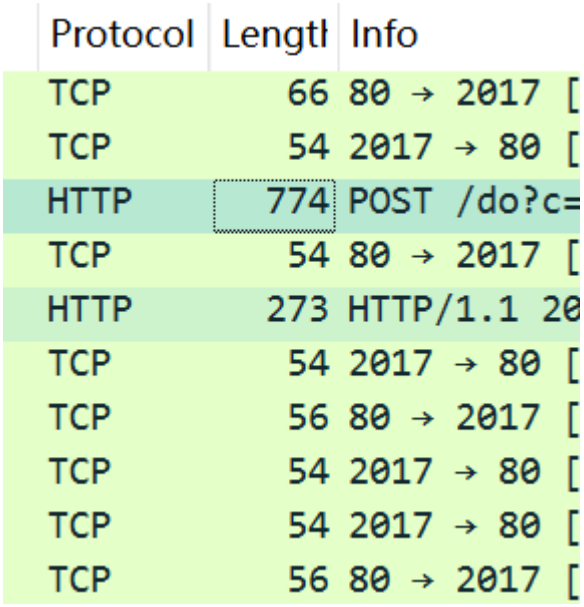
- **如果将包含 HTTP POST 命令的 TCP 报文段看作是 TCP 连接上的第一个报文段，那么该 TCP 连接上的第六个报文段的序号是多少？是何时发送的？该报文段所对应的 ACK 是何时接收的？**



使用wireshark的“追踪流”功能追踪这些报文段，发现TCP 连接上的第六个报文段的序号是2518100093，是在48.199979时间戳时发送，该报文对应的ack是在52.260337的时间戳被接收的

- **前六个 TCP 报文段的长度各是多少？**

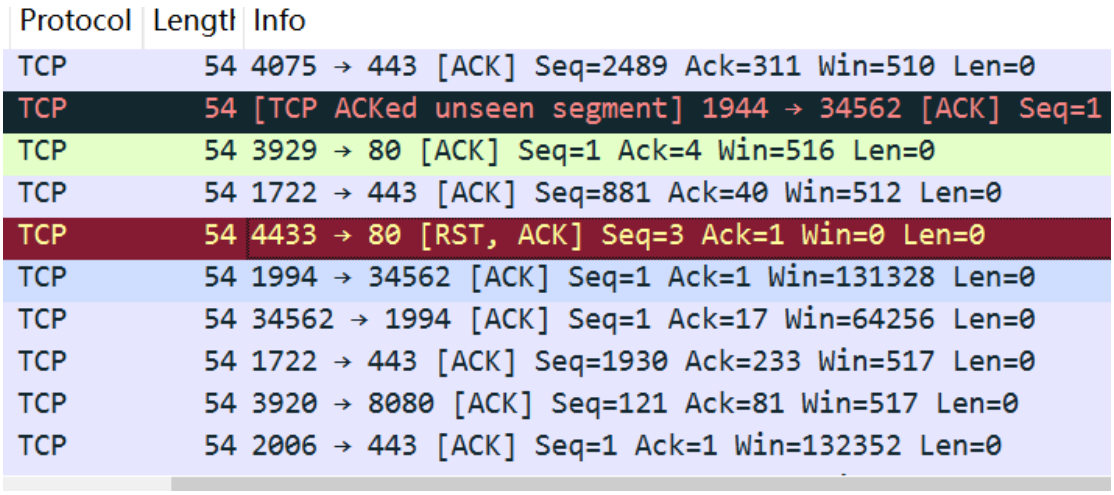
774、54、273、54、56、54



Protocol	Lengtl	Info
TCP	66	80 → 2017 [
TCP	54	2017 → 80 [
HTTP	774	POST /do?c=
TCP	54	80 → 2017 [
HTTP	273	HTTP/1.1 20
TCP	54	2017 → 80 [
TCP	56	80 → 2017 [
TCP	54	2017 → 80 [
TCP	54	2017 → 80 [
TCP	56	80 → 2017 [

- **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？**

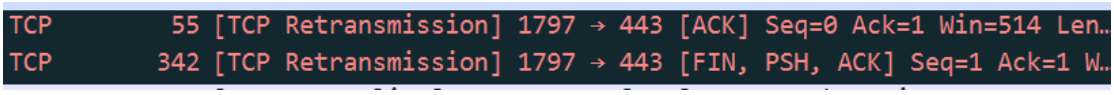
54。仍然不够用，判断依据是摘要信息中win字段变为了0，表明接收端的缓存仍然不够用。在此之后接收端缓存窗口就扩大了。



Protocol	Lengtl	Info
TCP	54	4075 → 443 [ACK] Seq=2489 Ack=311 Win=510 Len=0
TCP	54	[TCP ACKed unseen segment] 1944 → 34562 [ACK] Seq=1
TCP	54	3929 → 80 [ACK] Seq=1 Ack=4 Win=516 Len=0
TCP	54	1722 → 443 [ACK] Seq=881 Ack=40 Win=512 Len=0
TCP	54	4433 → 80 [RST, ACK] Seq=3 Ack=1 Win=0 Len=0
TCP	54	1994 → 34562 [ACK] Seq=1 Ack=1 Win=131328 Len=0
TCP	54	34562 → 1994 [ACK] Seq=1 Ack=17 Win=64256 Len=0
TCP	54	1722 → 443 [ACK] Seq=1930 Ack=233 Win=517 Len=0
TCP	54	3920 → 8080 [ACK] Seq=121 Ack=81 Win=517 Len=0
TCP	54	2006 → 443 [ACK] Seq=1 Ack=1 Win=132352 Len=0

- **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？**

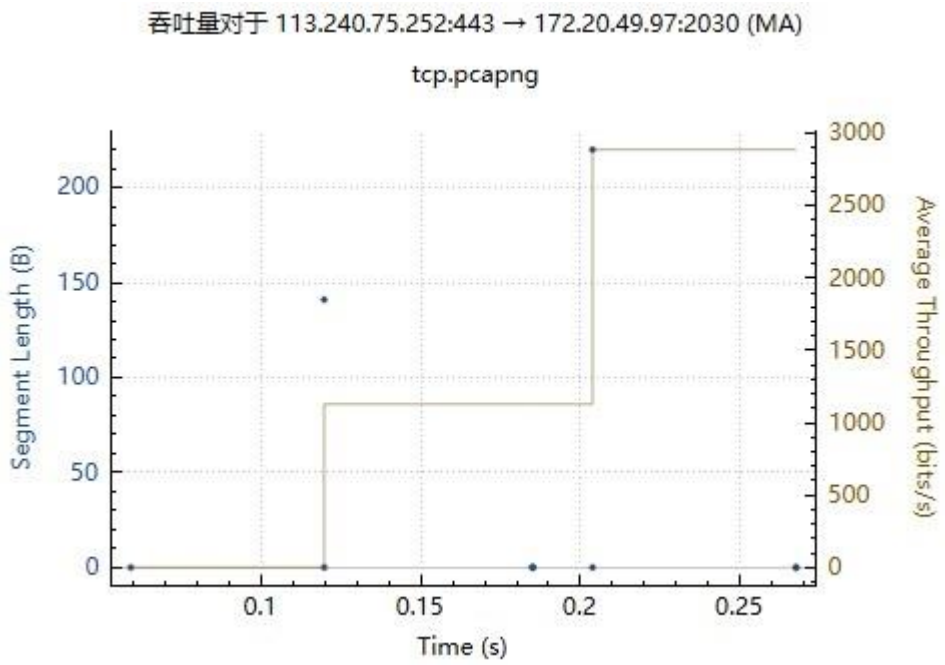
是，在跟踪文件中存在序列号、确认号完全相同的两份数据报，判断是重传的数据报，在wireshark中被标记“TCP Retransmission”



TCP	55	[TCP Retransmission] 1797 → 443 [ACK] Seq=0 Ack=1 Win=514 Len…
TCP	342	[TCP Retransmission] 1797 → 443 [FIN, PSH, ACK] Seq=1 Ack=1 W…

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

吞吐量 = 传输的字节数/单位时间。这里取一次tcp流的时间统计。



在这个tcp流中，在约0.2s时间内传输了约150字节，吞吐量约150/0.2≈750 bytes/s，与图中给出的以位/秒为单位的数据基本一致。

3. IP 分析

（1）在你的捕获窗口中，应该能看到由你的主机发出的一系列ICMP Echo Request包和中间路由器返回的一系列ICMP TTL-exceeded消息。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。

- **你主机的IP地址是什么？**

o.	Time	Source	Destination	Protocol	Lengtl	Info
1	0.000000	172.20.49.97	61.167.60.70	ICMP	70	Echo
2	0.002797	61.167.60.70	172.20.49.97	ICMP	70	Echo
3	0.050064	172.20.49.97	61.167.60.70	ICMP	70	Echo
4	0.053038	10.0.3.0	172.20.49.97	ICMP	70	Time-
5	0.100037	172.20.49.97	61.167.60.70	ICMP	70	Echo
6	0.103561	192.168.82.1	172.20.49.97	ICMP	70	Time-
7	0.150597	172.20.49.97	61.167.60.70	ICMP	70	Echo
8	0.153167	10.160.254.106	172.20.49.97	ICMP	70	Time-
9	0.200575	172.20.49.97	61.167.60.70	ICMP	70	Echo
10	0.233077	10.160.254.106	172.20.49.97	ICMP	70	Time-
11	0.251585	172.20.49.97	61.167.60.70	ICMP	70	Echo

主机IP：172.20.49.97

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

```
Identification: 0x5f9a (24474)
> 000. .... = Flags: 0x0
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 255
Protocol: ICMP (1)
Header Checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.20.49.97
Destination Address: 61.167.60.70
v Internet Control Message Protocol
```

值为1，表示ICMP协议

- **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**

IP头有20个字节

```
v Internet Protocol Version 4, Src: 172.20.49.97, Dst: 61.167.60.70
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
```

IP数据包的净载为56字节 - 20字节 = 36字节，因为整个报文长度是56字节

```
v Internet Protocol Version 4, Src: 172.20.49.97, Dst: 61.167.60.70
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x5f9a (24474)
```

- **该IP数据包的净载大小的？**

与前文一致，IP数据包的净载为56字节 - 20字节 = 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。可以通过IP头中的"Flags"和"Fragment Offset"字段来确定是否进行了分片。这两个值都为0表明没有分片

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

（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。在“listing of captured packets”窗口，你会看到许多后续的ICMP消息。

- **你主机发出的一系列ICMP消息中IP数据报中哪些字段总是发生改变？**

  标识符（Identification）、TTL

```
Internet Protocol Version 4, Src: 172.20.49.97, Dst: 61.167.60.70
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x5fa4 (24484)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
  > Time to Live: 4
    Protocol: ICMP (1)
```

- **哪些字段必须保持常量？哪些字段必须改变？为什么？**

  必须保持常量：版本号、头部长度、总长度、协议、原地址和目标地址。在同一系列的ICMP消息中，这些字段一般不应该发生变化，在发送之初就已经确定

  必须改变：标识符、TTL。标识符在每个新的数据报中都会增加，用于唯一标识从一个特定主机发出的IP数据报。TTL在每次数据报通过一个路由器时减1，当它到达0时，数据报会被丢弃。因此，即使数据报的源和目的地是相同的，TTL的值也可能因为路由路径的变化而变化。

- **描述你看到的IP数据包Identification字段值的形式**

  一个十六进制数，在每个新的数据报中都会加一

```
Total Length: 56
Identification: 0x5fad (24493)
> 000. .... = Flags: 0x0
```

（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

- **Identification字段和TTL字段的值是什么？**

  使用语句“icmp.type==11”筛选ICMP Time-to-live exceeded消息，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。

```
Internet Protocol Version 4, Src: 10.0.3.0, Dst: 172.20.49.97
    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: 255
    Protocol: ICMP (1)
```

  Identification字段：0

  TTL：255

- **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**

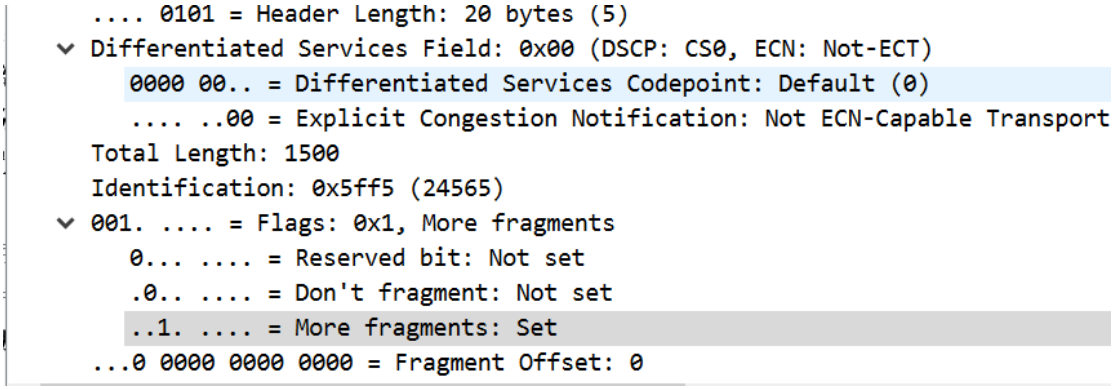
是否保持不变？为什么？

保持不变。这是因为ICMP错误消息（包括TTL超时）会在消息体中包含原始触发错误的IP包的首部和部分数据，这其中包括当时的Identification字段和TTL值。

（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。

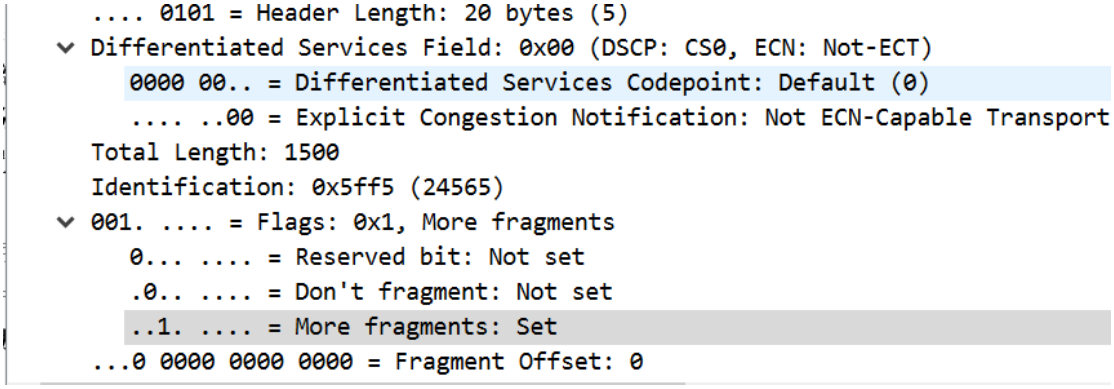
- **该消息是否被分解成不止一个IP数据报？**

  是的，more Fragment字段被设置为set表明该数据报被切片



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

  more Fragment字段被设置为set表明该数据报进行了切片，IP头部的fragment offset为0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1500字节（包含头部20字节）



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

- **原始数据包被分成了多少片？**

  3片，可以通过最后一个分片的"More fragments"标志没有被设置、而fragment offset不为0找到。

```
Identification: 0x5778 (24568)
000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
...0 0000 1011 1001 = Fragment Offset: 1480
```

- **这些分片中IP数据报头部哪些字段发生了变化？**
  Fragment Offset片偏移、more fragment（最后一个分片为0）、长度（最后一个分片可能没有填满）

4. 抓取 ARP 数据包

- **说明 ARP 缓存中每一列的含义是什么？**
  IP地址、物理地址、类型，用于在网络上查找一个IP地址对应的MAC地址的协议

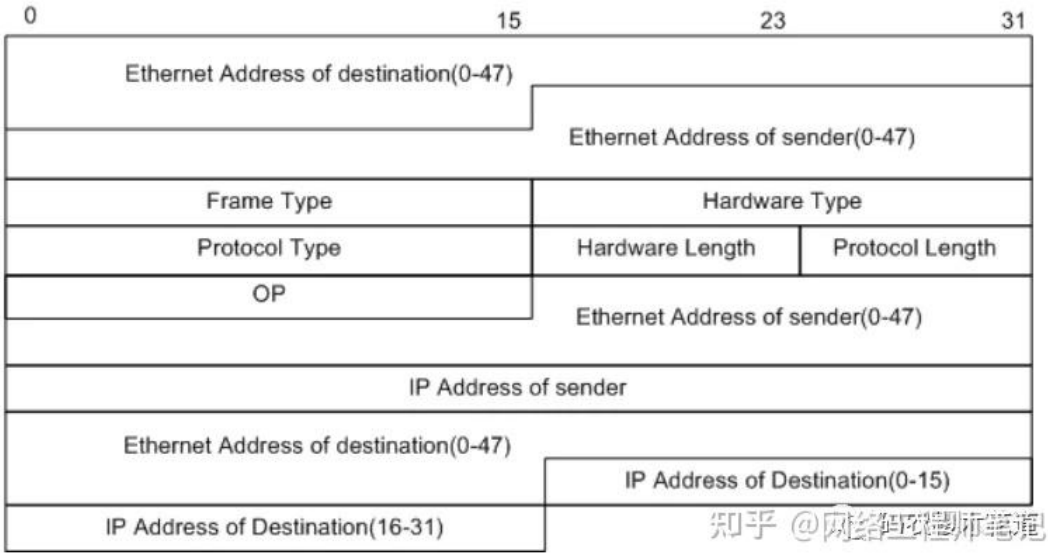
```
C:\Users\86150>arp -a

接口: 192.168.174.1 --- 0xe
  Internet 地址         物理地址              类型
  192.168.174.254       00-50-56-ee-1f-5d     动态
  192.168.174.255       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
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态

接口: 192.168.126.1 --- 0x14
  Internet 地址         物理地址              类型
  192.168.126.254       00-50-56-e4-d7-a6     动态
  192.168.126.255       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
  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: 172.20.49.97 --- 0x15
  Internet 地址         物理地址              类型
  172.20.0.1            44-ec-ce-d2-ff-c2     动态
  172.20.255.255        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
```

- **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？**

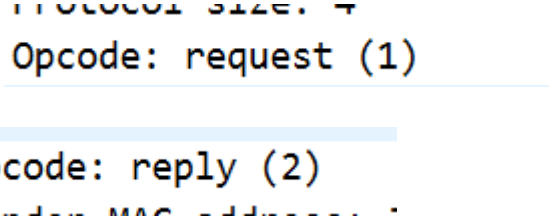



```
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: 9e:28:af:57:2d:13 (9e:28:af:57:2d:13)
    Sender IP address: 172.20.49.97
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 172.20.0.1
```

- 硬件类型（Hardware Type）：这部分占2个字节。它定义了网络的硬件类型，例如以太网或无线局域网。
- 协议类型（Protocol Type）：这部分占2个字节。它定义了网络层协议类型，通常是IP。
- 硬件地址长度（Hardware Address Length）：这部分占1个字节。它定义了硬件地址（如MAC地址）的长度。
- 协议地址长度（Protocol Address Length）：这部分占1个字节。它定义了协议地址（如IP地址）的长度。
- 操作码（Operation）：这部分占2个字节。它定义了ARP数据包的类型，例如请求（1）或响应（2）。
- 发送方硬件地址（Sender Hardware Address）：这部分的长度由硬件地址长度字段定义，对于以太网，通常是6个字节。它定义了发送ARP数据包的设备的硬件地址。
- 发送方协议地址（Sender Protocol Address）：这部分的长度由协议地址长度字段定义，对于IPv4，通常是4个字节。它定义了发送ARP数据包的设备的协议地址。
- 目标硬件地址（Target Hardware Address）：这部分的长度由硬件地址长度字段定义，对于以太网，通常是6个字节。它定义了目标设备的硬件地址。
- 目标协议地址（Target Protocol Address）：这部分的长度由协议地址长度字段定义，对于IPv4，通常是4个字节。它定义了目标设备的协议地址。

- **如何判断一个ARP数据是请求包还是应答包？**

判断一个 ARP 分组是 ARP 请求还是应答的字段是“OP”，当其值为 0×0001 时是请求，为 0×0002 时是应答。



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

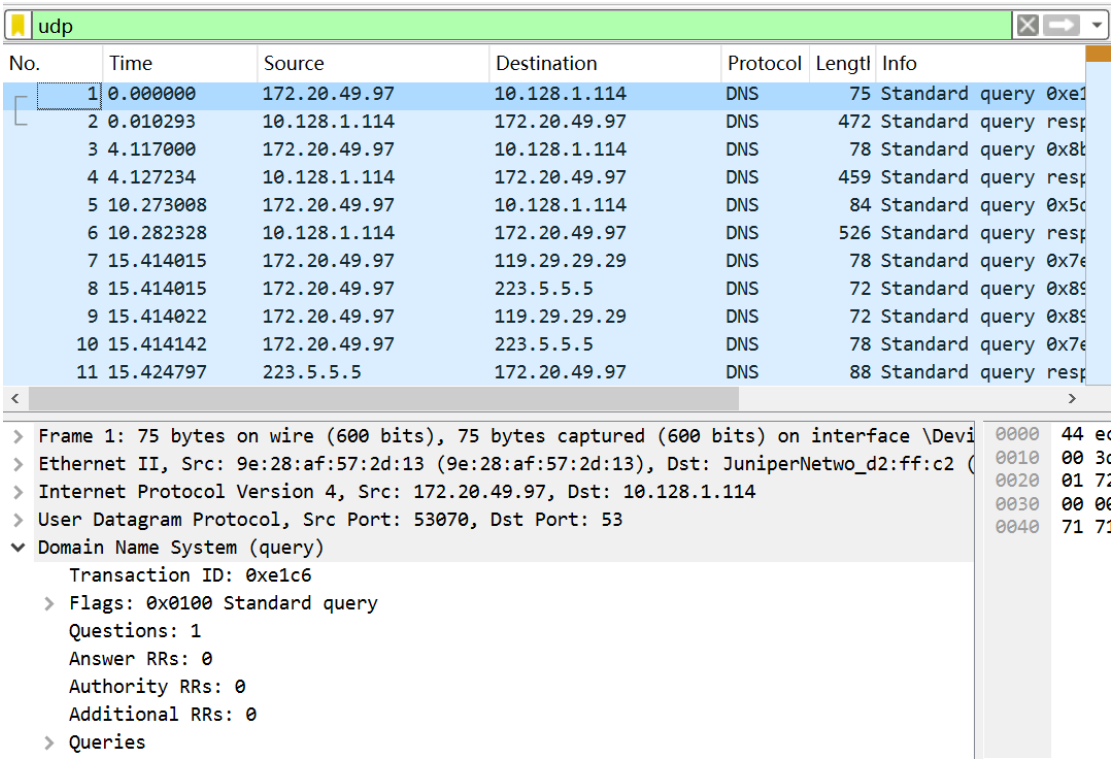
ARP查询时，发送设备不知道目标IP地址对应的MAC地址，所以它需要询问所有的设备，向广播地址FF:FF:FF:FF:FF:FF发送。

ARP响应时，发送设备已经知道请求设备的地址，可以进行单播，没有必要将这个信息发送给其他的设备。

5. 抓取 UDP 数据包

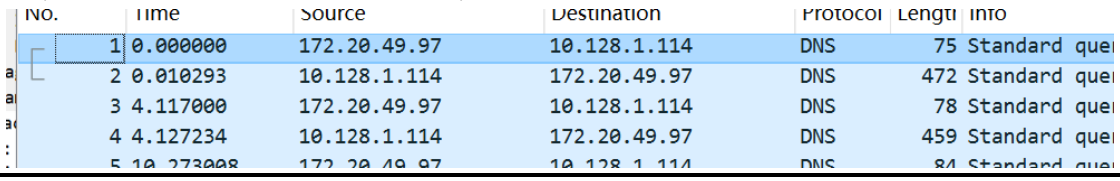
- **消息是基于UDP的还是TCP的？**

基于UDP



- **你的主机ip地址是什么？目的主机ip地址是什么？**

主机IP：172.20.49.97，目的主机IP：10.128.1.114



- **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？**
  主机端口号：53070；服务器端口号：53

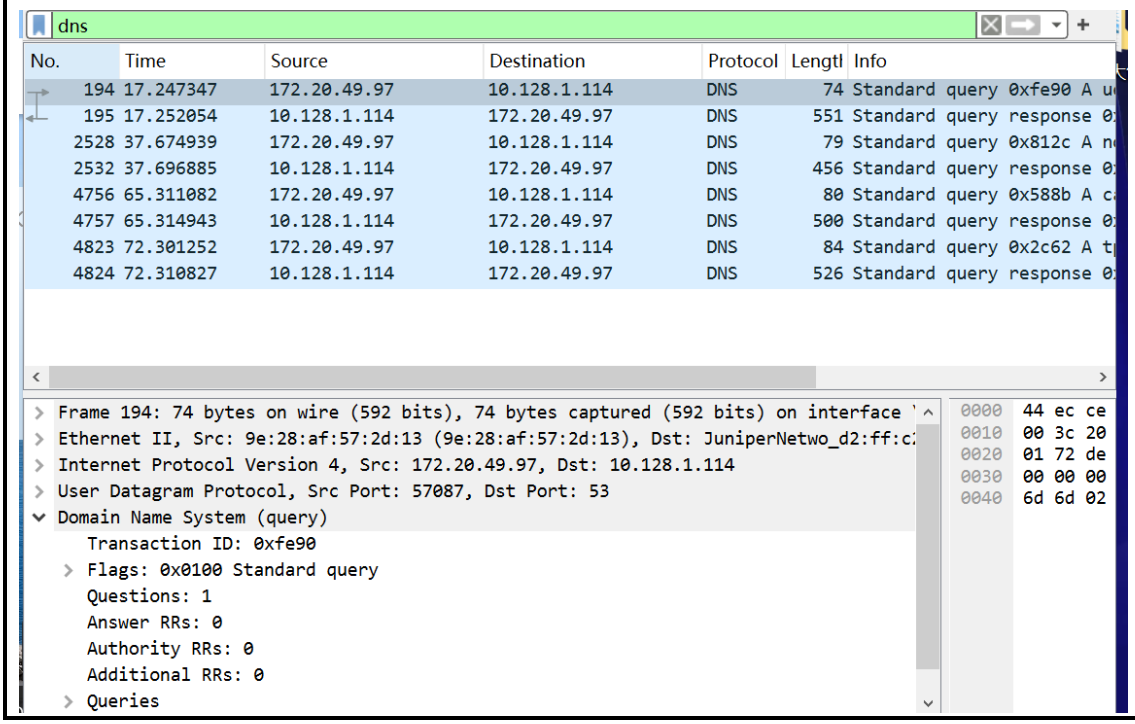
```
User Datagram Protocol, Src Port: 53070, Dst Port: 53
    Source Port: 53070
    Destination Port: 53
    Length: 41
    Checksum: 0xe9a1 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
```

- **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？**
  包含四个字段：源端口、目的端口、长度、校验和，每个字段都是2个字节

```
User Datagram Protocol, Src Port: 53070, Dst Port: 53
    Source Port: 53070
    Destination Port: 53
    Length: 41
    Checksum: 0xe9a1 [unverified]
```

- **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？这和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？**
  为的是在不可靠的UDP连接上在应用层层面上保障可靠性，这和tcp发送ack有相似之处。TCP在数据传输前需要进行三次握手连接，而UDP第一个数据报就开始传输数据，不需要预先建立连接。这就是为什么UDP被称为无连接的协议。

6. 利用 WireShark 进行 DNS 协议分析



- 客户端IP：172.20.49.97
- 服务器IP：10.128.1.114
- 标志位

```
Domain Name System (query)
    Transaction ID: 0xfe90
  Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
```

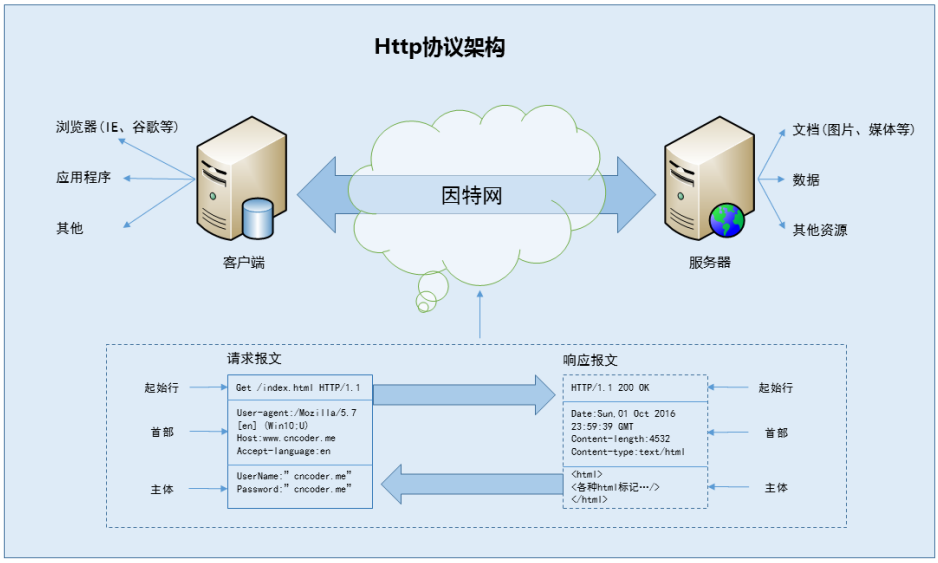
  - QR（Query/Response）：这个位标识消息是查询（0）还是响应（1）。
  - Opcode：一个4位的字段，用于指定查询或响应的类型。标准查询（0）、反向查询（1）或状态查询（2）。
  - AA（Authoritative Answer）：在响应中设置，表示响应是来自域的授权源。
  - TC（Truncation）：如果响应无法在单个UDP数据包中传输，这个位会被设置。
  - RD（Recursion Desired）：在查询中设置，请求服务器进行递归查询。
  - RA（Recursion Available）：在响应中设置，表示服务器可以进行递归查询。
  - RCode（Response code）：一个4位的字段，表示响应的状态，没有错误（0）、格式错误（1）、服务器失败（2）或名称错误（3）。

问题讨论：

总结一下本实验涉及到的各个协议。

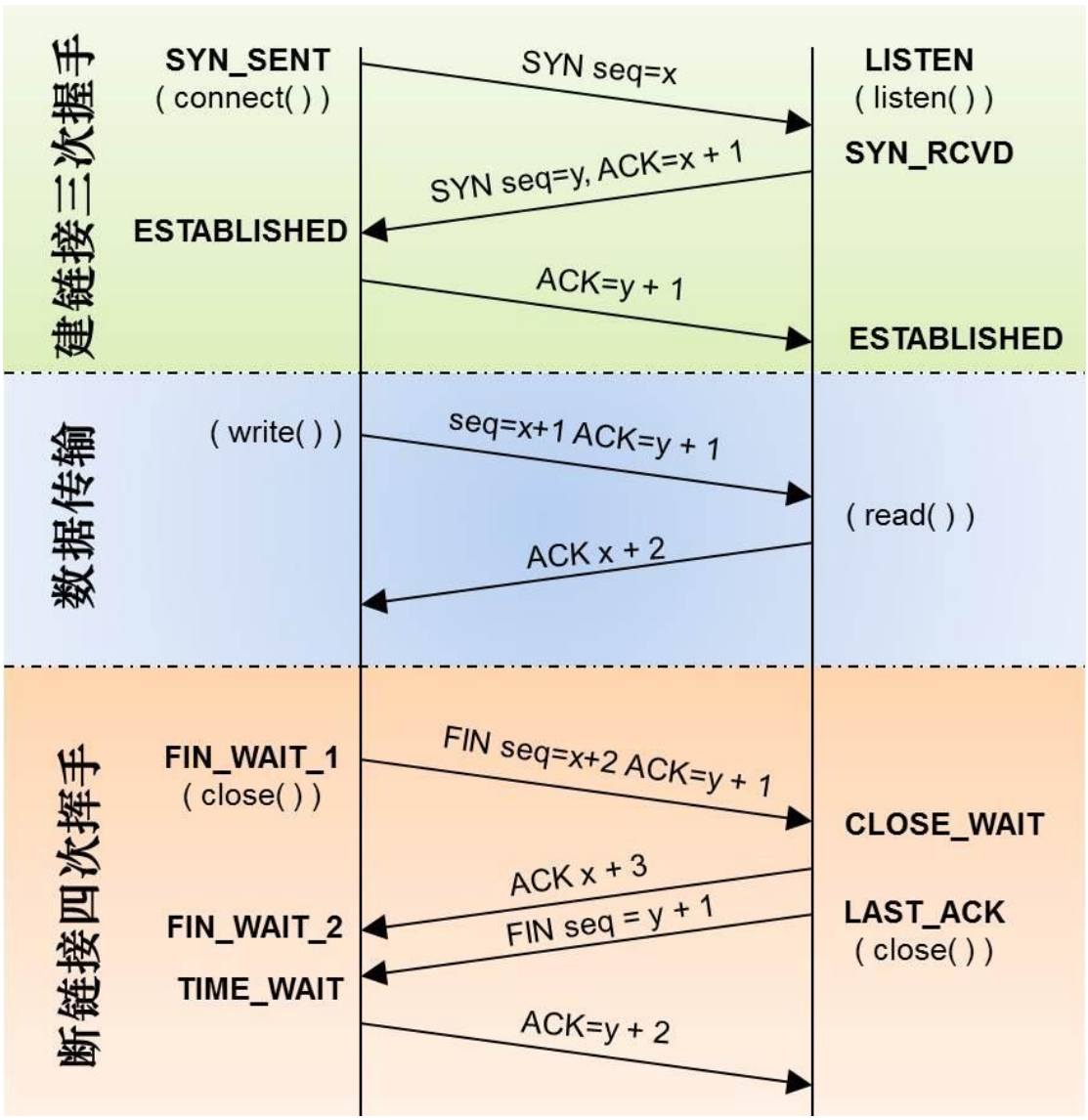
1.HTTP协议-应用层协议

- 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。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。
- 灵活:HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。
- 无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。
- 无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。



2.TCP协议-传输层协议

- 面向连接：三次握手、四次挥手



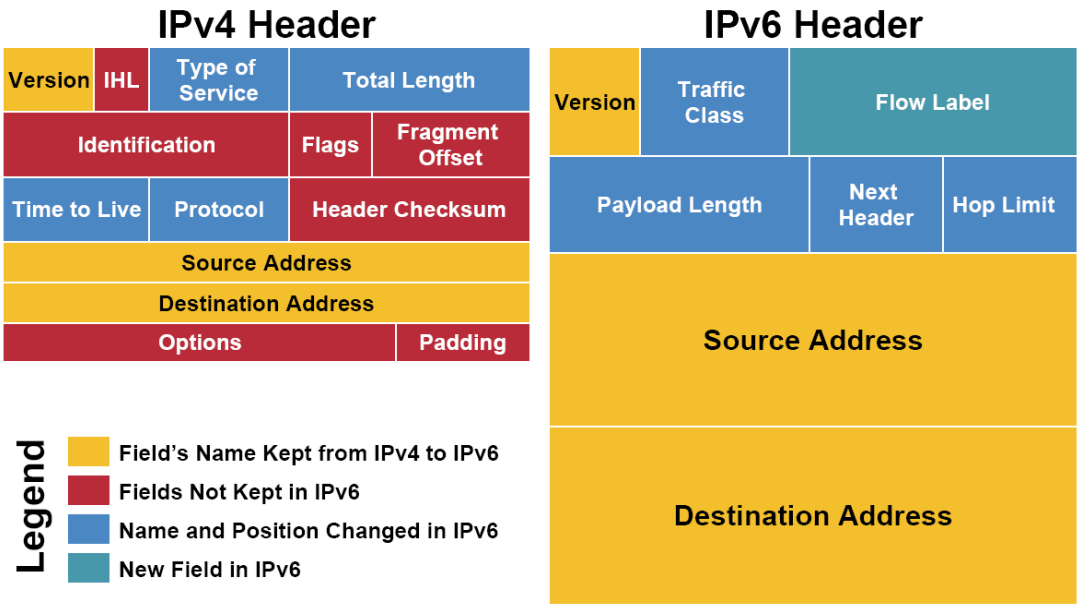
- 可靠性：序列号、ACK确认、重传、校验和
- 流量控制：滑动窗口
- 拥塞控制：慢启动、拥塞避免
- 全双工通信
- 多路复用

TCP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																Urgent pointer (if URG set)															
20 ...	160 ...	Options (if Data Offset > 5, padded at the end with "0" bytes if necessary) ...																															

3. IP协议-网络层协议

- 无连接服务：尽力而为
- IP 地址：每个连接到网络的设备都被分配一个唯一的 IP 地址，用于标识设备在网络中的位置。IPv4：是当前广泛使用的版本，使用 32 位地址，提供大约 43 亿个唯一的地址。IPv6：是 IP 协议的下一代版本，使用 128 位地址，极大地扩展了可分配的地址数量。
- 分段和重组：由于不同网络的最大传输单元（MTU）可能不同，IP 允许数据在传输过程中被分段。接收端的 IP 层负责将这些分段重新组装成原始数据。
- 路由：IP 协议使用路由表来决定如何将数据包从一个网络转发到另一个网络，直至到达目的地。
- NAT（网络地址转换）：NAT 允许多个设备共享一个公共 IP 地址来访问互联网，这在 IPv4 地址耗尽的情况下非常有用。
- 子网划分：子网划分允许将一个较大的网络划分为多个较小的子网，以提高网络的效率和安全性。

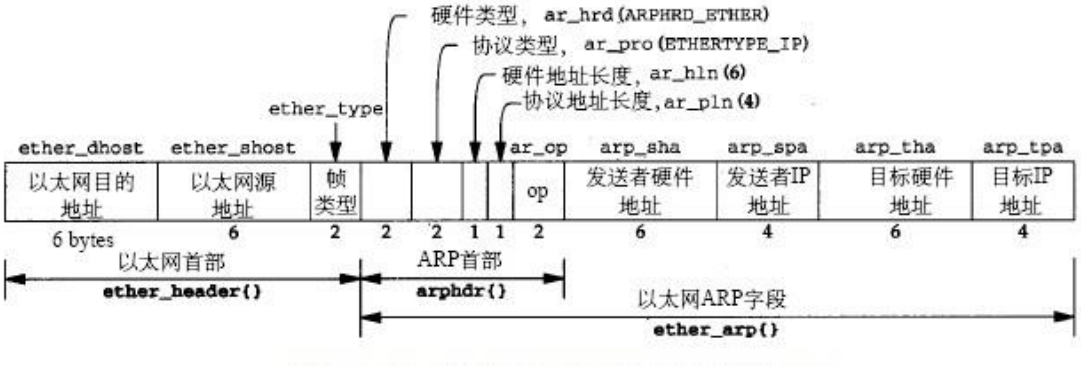


4. ARP协议-网络层/链路层协议

- 请求MAC地址：当设备A想要向设备B发送数据，但只知道B的IP地址时，设备A会使用ARP来获取B的MAC地址。
- 广播ARP请求：设备A向局域网内的所有设备发送一个ARP请求，询问拥有目标IP地址

的设备的MAC地址。

- 接收ARP请求：局域网内的所有设备都会接收到这个ARP请求，但只有拥有目标IP地址的设备B会响应。
- 发送ARP响应：设备B回复一个ARP响应，提供它的MAC地址。
- 缓存ARP条目：设备A收到响应后，会将其缓存在ARP表中，以备将来使用。



5.UDP协议-传输层协议

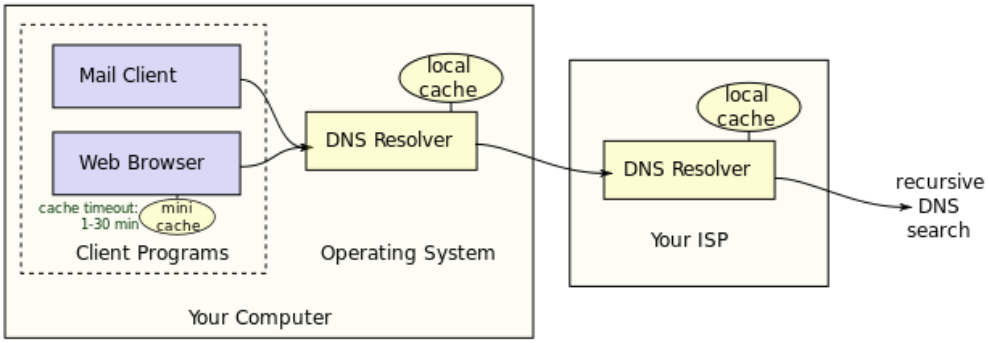
- 无连接
- 简单快速
- 不可靠：尽力而为
- 支持单播、多播和广播

UDP Header

Offsets Octet		0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port														
4	32	Length																Checksum														

6. DNS 协议-应用层协议

- 本地缓存检查：当用户输入一个域名时，DNS 客户端首先检查本地缓存，看是否有该域名的IP地址。
- 递归查询：如果本地缓存中没有，客户端会向配置的 DNS 服务器发送查询请求。
- 迭代查询：DNS 服务器会尝试在本地缓存或权威服务器中查找域名的IP地址。如果没有找到，它会向其他 DNS 服务器发送查询请求，直到找到答案或确定域名不存在。
- 响应返回：找到答案后，DNS 服务器将响应返回给客户端，客户端使用这个 IP 地址来建立连接。



心得体会：

1.熟悉了wireshark等嗅探工具的使用
2.加深了对实验中涉及到的各个协议的理解，特别是对tcp三次握手的过程
3.深化了对各个协议报文形式的理解，以便从中得到对应的标识位