浙江理工大学

《计算机网络》

实验报告

20 23 ~ 20 24 学年第 1 学期

学　院	计算机科学与技术学院
班　级	计算机科学与技术 21(4)班
姓　名	陈昊天
学　号	2021329600006
任课教师	黄海、杨东鹤 (第 13 周 三 345 节)

计算机科学与技术专业

20 23 年 12 月

实验 3　Wireshark 软件使用与协议分析

3.1-----ARP 协议分析

一． 实验目的

学习 Wireshark 的基本操作,抓取和分析有线局域网的数据包;掌握以太网 MAC 帧的基本结构，掌握 ARP 协议的特点及工作过程。

二． 实验内容

使用 Wireshark 抓取局域网的数据包并进行分析：

1. 学习 **Wireshark** 基本操作：重点掌握捕获过滤器和显示过滤器。

2. 观察 **MAC** 地址：了解 MAC 地址的组成，辨识 MAC 地址类型。

3. 分析以太网帧结构：观察以太网帧的首部和尾部，了解数据封装成帧的原理。

4. 分析 **ARP** 协议：抓取 ARP 请求和应答报文，分析其工作过程。

三． 实验原理

3.1 Wireshark 简介

Wireshark 软件是目前全球使用最广泛的开源网络数据包分析工具（前身为 Ethereal），由 Gerald Combs 编写并于 1988 年获开源许可发布。网络数据包分析是指进入网络通信系统、捕获和解码网络上实时传输数据以搜集统计信息的过程。通过 Wireshark 对网络数据进行分析，我们能够了解网络是如何运行的、数据包是如何被转发的、应用是如何被访问的；能够分析各层网络协议的性能、掌握通信主体的运行情况，确认带宽分配和时延大小、查看应用的快慢并改进优化，识别网络中存在的攻击或恶意行为、解决网络异常和故障。Wireshark 可以在 Windows、Linux 和 macOS 操作系统中运行，具备好的图形界面、丰富的统计图表分析功能

3.2 以太网 **MAC** 帧格式

本实验基于使用最广泛的有线局域网（以太网 Ethernet II）。其中，MAC 地址（Media Access Control Address，媒体存取控制位址）或称物理地址（Physical Address），用于在网络中标识网卡。MAC 地址的长度为 48 位（6 个字节），通常表示为 12 个 16 进制数，如：00–16–EA–AE–3C–40。其中前 3 个字节的 16 进制数 00–16–EA 代表网络硬件制造商的编号、即组织唯一标志符（OUI），它由 IEEE 分配；而后 3 个字节的 16 进制数 AE–3C–40 代表该制造商所生产的某个网络产品（如网卡)的系列号。

3.3 ARP 协议及数据报格式

地址解析协议（Address Resolution Protocol，ARP），主要作用是将 IP 地址解析为 MAC 地址。当某主机或网络设备要发送数据给目标主机时，必须知道对方的网络层地址（即 IP 地址），而且在数据链路层封装成帧时，还必须有目标主机（或下一跳路由器）的 MAC 地址。

ARP 报文总长度为 28 字节，MAC 地址长度为 6 字 节，IP 地址长度为 4 字节。每个字段的含义如下：

- **硬件类型**：指明了发送方想知道的硬件接口类型，以太网的值为 1。
- **协议类型**：表示要映射的协议地址类型。IP 地址的类型值为 0x0800。
- **硬件地址长度和协议地址长度**：分别指出硬件地址和协议地址的长度，以字节为单位。在以太网中，它们的值分别为 6 和 4。
- **操作码（op）**：用来表示这个报文的类型，ARP 请求为 1，ARP 响应为 2，RARP 请求为 3，RARP 响应为 4。

四． 实验条件

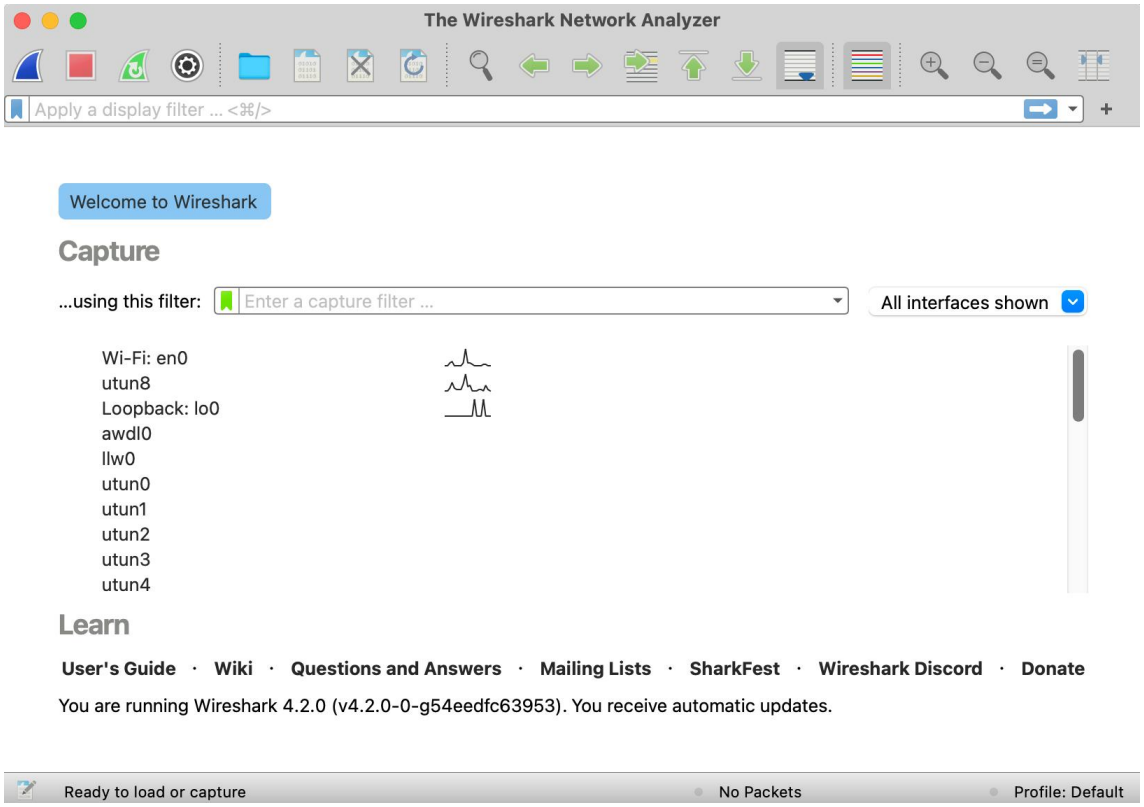
- PC 机一台，连入局域网
- Wireshark 软件

五． 实验步骤

5.1 WireShark 基本使用

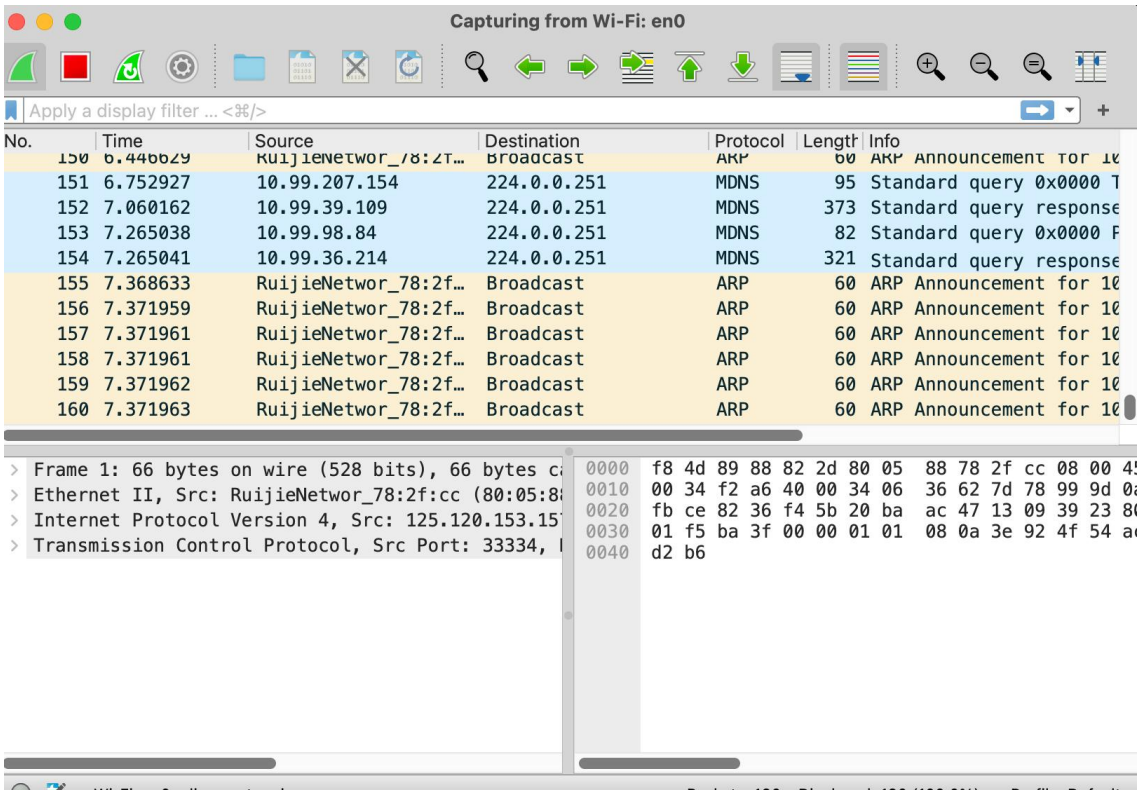
1. 通过 Wireshark 官网下载最新版软件，按默认选项安装。

2. 运行 Wireshark 软件，程序界面会显示当前的网络接口列表，点击要观察的网络接口，开始捕捉数据包，Wireshark 软件选择网络接口的界面如图所示。

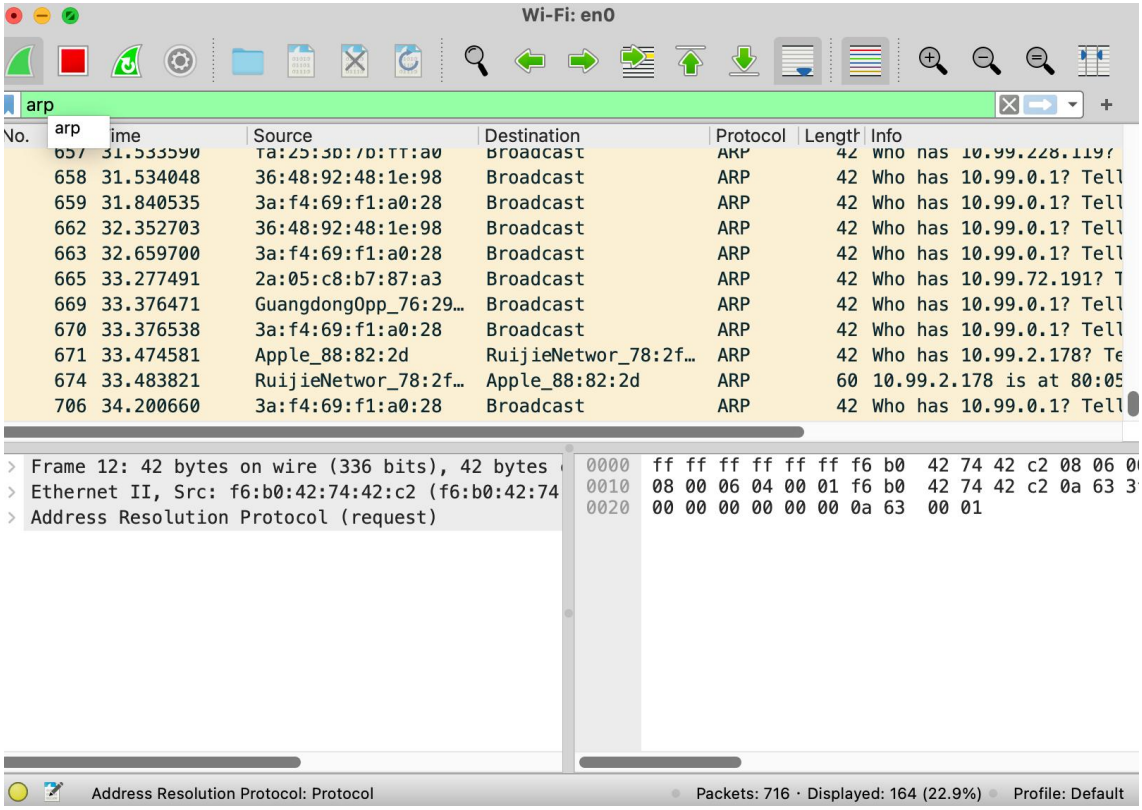


3. 点击工具栏上的红色方块按钮停止捕捉。

4. 菜单、工具栏、状态栏和主窗口如图所示。



5. 使用“显示过滤器”可以方便地从捕获的数据包中筛选出要观察的数据包。显示过滤器支持若干的过滤选项：源 MAC、目的 MAC、源 IP、目的 IP、TCP/UDP 传输协议、应用层协议（HTTP, DHCP）、源端口 Port、目的端口 Port 等。



6. 通过主菜单“文件”/“导出特定分组”，可以保存捕获的网络数据

7. 如果只想捕捉特定的数据包，可以使用菜单“捕获”/“捕获过滤器”选定想要的类型。例如，选择“IPv4 only”，Wireshark 只抓取 ipv4 类型的数据包。

5.2 观察 MAC 地址

启动 Wireshark 捕捉数据包，在命令行窗口分别 ping 网关和 ping 同网段的一台主机，分析本机发出的数据包。重点观察以太网帧的 Destination 和 Source 的 MAC 地址， 辨识 MAC 地址类型，解读 OUI 信息、I/G 和 G/L 位。

ping 网关

Destination：RuijieNetwor_78:2f:cc（80:05:88:78:2f:cc）

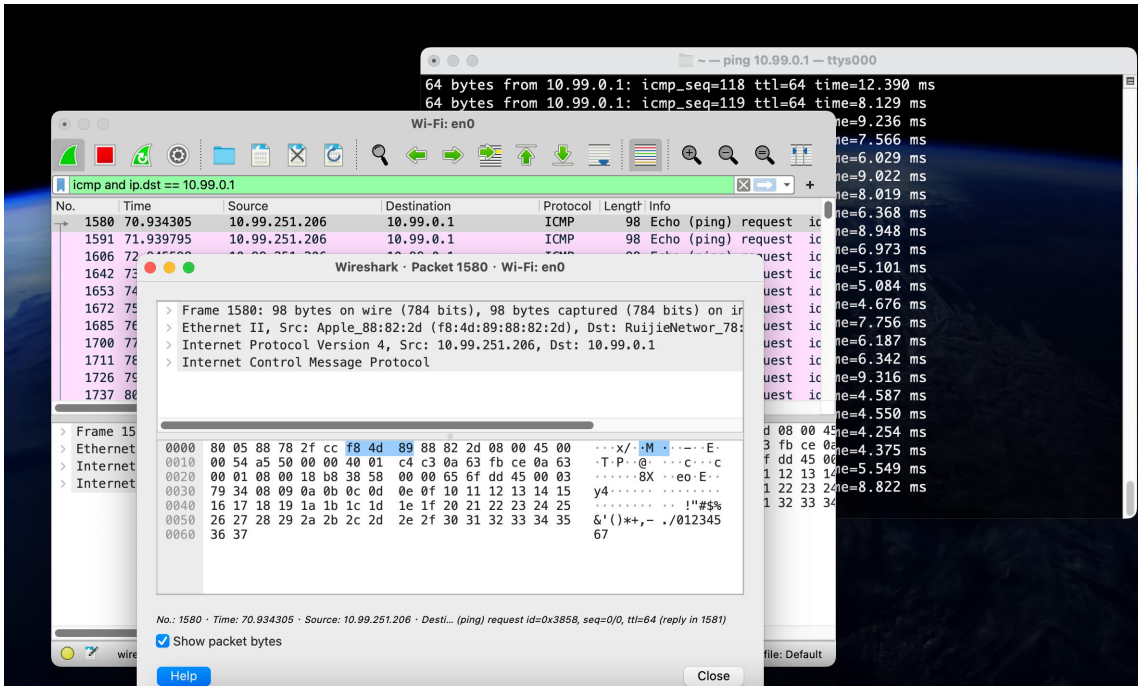
.... ..0. .... = LG bit：Globally unique address (factory default)

....0 .... = IG bit：Individual address (unicast)

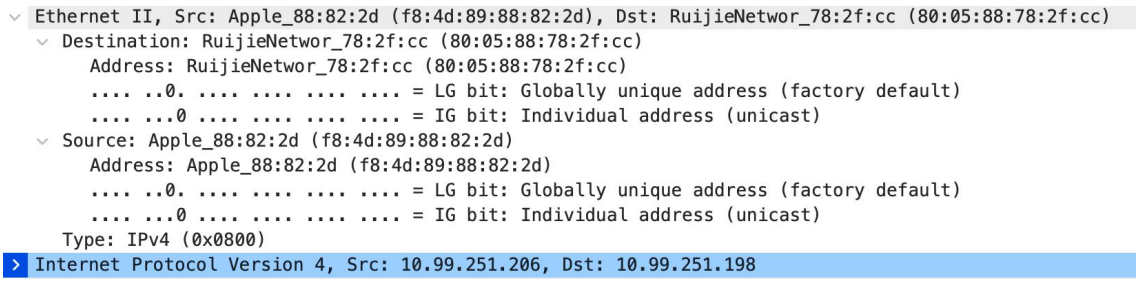
Source：Apple_88:82:2d（f8:4d:89:88:82:2d）

.... ..0. .... = LG bit：Globally unique address (factory default)

....0 .... = IG bit：Individual address (unicast)

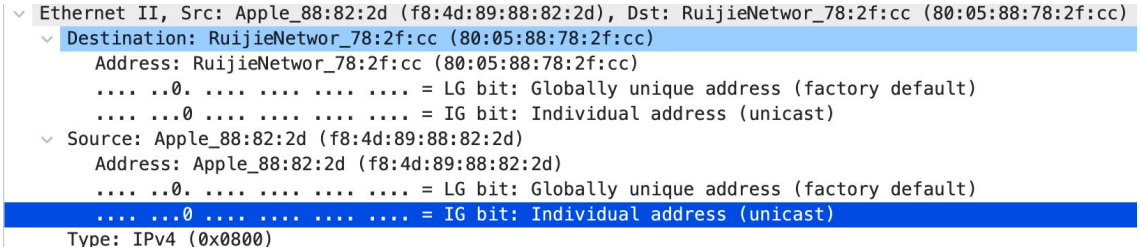


ping 同网段的一台主机



5.3 分析以太网的帧结构

选择其中一个数据包，点击 Ethernet II 展开，查看 MAC 帧的各个字段。



Frame 1580: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface en0, id 0
Ethernet II, Src: Apple_88:82:2d (f8:4d:89:88:82:2d), Dst: RuijieNetwor_78:2f:cc (80:05:88:78:2f:cc)
Destination: RuijieNetwor_78:2f:cc (80:05:88:78:2f:cc)
Address: RuijieNetwor_78:2f:cc (80:05:88:78:2f:cc)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Source: Apple_88:82:2d (f8:4d:89:88:82:2d)
Address: Apple_88:82:2d (f8:4d:89:88:82:2d)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 10.99.251.206, Dst: 10.99.0.1
Internet Control Message Protocol

5.4 ARP 协议分析

1. 使用 arp -F 命令，清空本机的 ARP 缓存，开启 Wireshark， ping 本机的同网段地址，在显示过滤器条框中输入“arp ”，观察捕获的 ARP 报文的各个字段，分析请求/响应的过程。

ARP 请求：本机尝试联系同一局域网内的另一台设备时，会发送一个 ARP 请求。这个请求包含发送者的 MAC 地址和 IP 地址，以及目标设备的 IP 地址。

ARP 响应：收到 ARP 请求的设备会检查请求中的目标 IP 地址是否与自己的 IP 地址匹配。如果匹配，该设备会发送一个 ARP 响应。响应中包含了目标设备的 MAC 地址和 IP 地址，以及发送者的 MAC 地址和 IP 地址。

```
Apple_88:82:2d      Broadcast           ARP      42 Who has 10.99.251.195? Tell 10.99.251.206
RuijieNetwor_78:2f… Apple_88:82:2d      ARP      60 10.99.251.195 is at 80:05:88:78:2f:cc
```

```
> Frame 378: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface en0, id 0
v Ethernet II, Src: Apple_88:82:2d (f8:4d:89:88:82:2d), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  v Destination: Broadcast (ff:ff:ff:ff:ff:ff)
      Address: Broadcast (ff:ff:ff:ff:ff:ff)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory defau
      .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
  v Source: Apple_88:82:2d (f8:4d:89:88:82:2d)
      Address: Apple_88:82:2d (f8:4d:89:88:82:2d)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: ARP (0x0806)
v Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Apple_88:82:2d (f8:4d:89:88:82:2d)
    Sender IP address: 10.99.251.206
    Target MAC address: Xerox_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.99.121.1
```

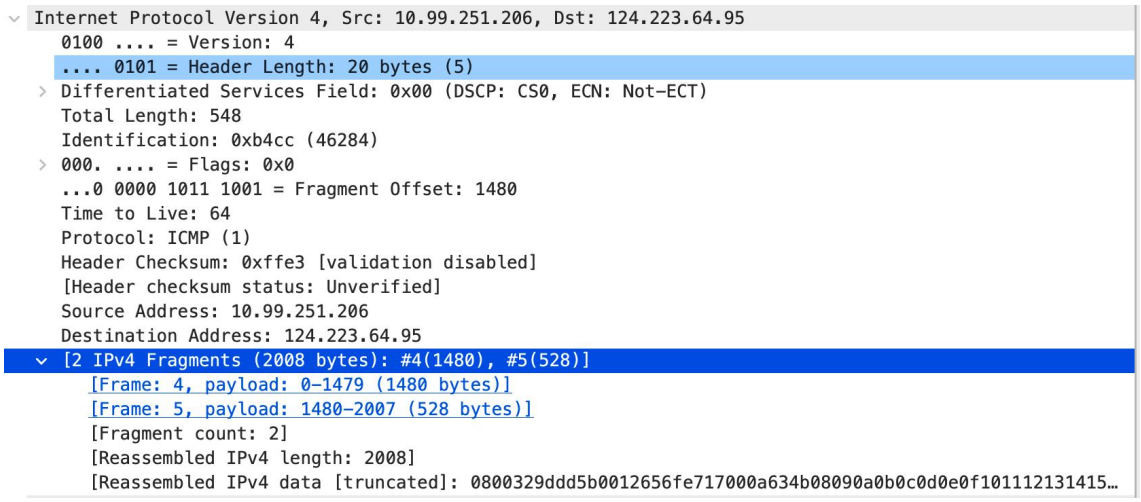
2. 使用 arp －F 命令，清空本机的 ARP 缓存。开启 Wireshark，ping 与本机网段不同的 IP 地址或域名，观察捕获的 ARP 报文的各个字段，分析请求/响应的过程。

数据包需要通过网关转发到其他网络或互联网上。ARP 请求会寻找本地网络上的网关的 MAC 地址，而不是目标 IP 地址的 MAC 地址。

```
Apple_88:82:2d      Broadcast           ARP      42 Who has 10.99.0.1? Tell 10.99.251.206
RuijieNetwor_78:2f… Apple_88:82:2d      ARP      60 10.99.0.1 is at 80:05:88:78:2f:cc
```

1. 使用了显示过滤器后，Wireshark 的抓包工作量会减少吗?
不会减少。显示过滤器只是在 Wireshark 的用户界面上过滤显示已经捕获的数据包，以便用户可以专注于分析特定类型的数据包。
2. MAC 帧的长度和 IP 数据报的长度有怎样的关系？请用你的数据记录进行验证。
一个 IP 数据报可以被封装在一个或多个 MAC 帧中进行传输，取决于网络的 MTU 大小。
使用 ping 发出一个大于 MTU 的 IP 数据包可以观察到分片的现象，如图所示

```
(base) nanmener@Haotians-MacBook-Pro ~ % ping -s 2000 chenhaotian.top
PING chenhaotian.top (124.223.64.95): 2000 data bytes
2008 bytes from 124.223.64.95: icmp_seq=0 ttl=49 time=18.367 ms
2008 bytes from 124.223.64.95: icmp_seq=1 ttl=49 time=18.514 ms
2008 bytes from 124.223.64.95: icmp_seq=2 ttl=49 time=26.646 ms
2008 bytes from 124.223.64.95: icmp_seq=3 ttl=49 time=70.703 ms
```



3. ping 同一局域网内的主机和局域网外的主机，都会产生 ARP 报文么？所产生的 ARP 报文有何不同，为什么？

都有可能产生 ARP 报文。不同在于，局域网内通信的 ARP 是直接查询目标主机的 MAC 地址，而局域网外通信的 ARP 是查询网关的 MAC 地址。

六． 实验心得与体会

通过参与“Wireshark 软件使用与协议分析”的实验，我对网络数据包的捕获和分析有了更深入的了解。在实验中，我不仅学会了如何使用 Wireshark 这一强大的网络分析工具，而且对 ARP 协议、MAC 地址和以太网帧结构有了更加清晰的认识。

Wireshark 的界面友好、功能强大，让我能够轻松地捕获和分析网络数据。通过捕获过滤器和显示过滤器的使用，我学会了如何高效地筛选出我所关注的数据包，这对于网络问题的诊断和性能分析非常有帮助。

我对 MAC 地址的了解更加深入了。在实验中，我观察了 MAC 地址的结构，了解到它包括由 IEEE 分配的组织唯一标志符和制造商指定的产品系列号。通过分析 ping 命令产生的数据包，我进一步理解了 MAC 地址的全球唯一性和单播地址的概念。

在分析 ARP 协议的过程中，我对地址解析的工作原理有了更加透彻的理解。ARP 协议通过请求和应答的方式，将 IP 地址解析为对应的 MAC 地址，确保了数据能够在局域网内正确地传输到目的地。实验中的 ARP 缓存清空和报文捕获步骤，让我体验了 ARP 查询的全过程。

我还学习了以太网帧的结构，包括帧首部和尾部，以及如何将数据封装成帧进行传输。通过对帧结构的分析，我对以太网的工作方式有了更深的理解。

这次实验不仅提高了我的实操能力，也加深了我对网络协议和数据传输原理的理解。我意识到，网络分析是一个复杂但有趣的领域，Wireshark 则是网络工程师不可或缺的工具。通过这次实验，我更加坚定了继续深入学习网络技术的决心，并相信这些知识将在我的未来学习和工作中发挥重要作用。

3.2-----IP 与 ICMP 分析

一． 实验目的

IP 和 ICMP 协议是 TCP/IP 协议簇中的网络层协议，在网络寻址定位、数据分组转发和路由选择等任务中发挥了重要作用。本实验要求熟练使用 Wireshark 软件，观察 IP 数据报的基本结构，分析数据报的分片；掌握基于 ICMP 协议的 ping 和 traceroute 命令及其工作原理。

二． 实验内容

启动 Wireshark，捕捉网络命令执行过程中本机接受和发送的数据报。

1. 执行 **ping** 命令，观察 **IP** 数据报和 **ICMP** 询问报文的结构：通过 Wireshark 监视器观察捕获流量中的 ICMP 询问报文和 IP 数据报的结构。注意比较 ICMP 请求帧与回应帧，及其 IP 头部数据字段的异同。

2. 改变 **ping** 命令的参数，观察 **IP** 数据报分片：更改 ping 命令参数 MTU，使其发出长报文以触发 IP 数据报分片，再观察 IP 数据报的结构变化。

3. 执行 **Traceroute** 命令，观察 **ICMP** 差错报文的结构，并分析其工作原理：使用 Linux 操作系统提供的 traceroute 命令（或者 Windows 系统提供的 tracert 命令）， 捕获和分析该命令所产生的 IP 数据报，特别注意相关的 ICMP 差错报文。结合捕获的具体数据，画出命令执行过程中数据交互的示意图，掌握 traceroute 的工作原理。

三． 实验原理

3.1 IP 协议及数据报格式

网际互连协议（Internet Protocol，IP），是 TCP/IP 体系中的网络层协议，可实现大规模的异构网络互联互通，为主机提供无连接的、尽力而为的数据包传输

服务。在网际协议第 4 版（IPv4）中，IP 数据报是一个可变长分组，包括首部和数据两部分。

3.2 ICMP 协议及报文格式

因特网控制报文协议（Internet Control Message Protocol，ICMP），用于 IP 主机、路由器之间传递控制消息。控制消息是指网络是否连通、主机是否可达、路由是否可用等网络本身的控制管理消息，对网络正常运行起着重要的作用。

ICMP 报文的类型可以分为 ICMP 差错报文和 ICMP 询问报文两种。ICMP 差错报告报文主要有终点不可达、源站抑制、超时、参数问题和路由重定向 5 种。ICMP 询问报文有回送请求和应答、时间戳请求和应答、地址掩码请求和应答以及路由器询问和通告 4 种。

1. ping 命令，是测试网络最有效的工具之一。它是由主机或路由器执行 ping 命令向一个特定的目的主机发送一份 ICMP 回显请求（Echo request）报文，并等待其返回 ICMP 回显应答（Echo Reply）。ping 命令可以检测网络的连通性，简单估测数据报的往返时间（Round Trip Time），确定是否有数据包丢失或损坏，从而帮助分析网络故障。

2. traceroute/tracert 命令，利用 TTL 字段和 ICMP 差错类型报文，查找 IP 数据报的路由转发路径（含路由器信息）。源主机执行该命令向目的主机发送生存时间（TTL）不同的 ICMP 回送请求报文，直至收到目的主机应答，并通过分析应答报文获得转发路径和时延信息。

首先源主机发起一个 TTL=1 的 ICMP 报文。第一个路由器收到该报文后，TTL 减 1 变为 0 并丢弃此报文，返回一个 [ICMP time exceeded] 的消息。源主机通过这个消息获知 IP 数据报转发路径上的第一个路由器信息。然后，依次增加发

送 ICMP 报文的 TTL 值，可以获取路径上的后续路由器的信息。当到达目的地时，目标主机返回一个 [ICMP port unreachable] 的消息，使发起者确认 IP 数据报已经正常到达。至此，traceroute 命令发起者已经获得了通向目标主机路径上的所有路由信息。

四． 实验条件

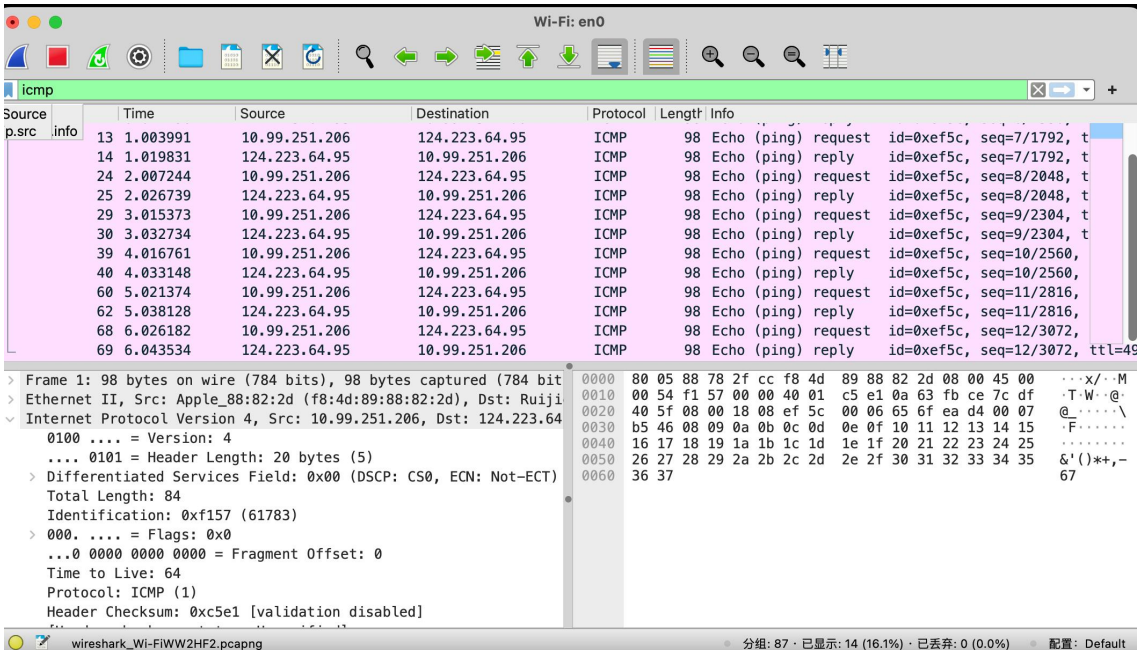
装有 Wireshark 软件的 PC 机一台，处于局域网环境。

五． 实验步骤

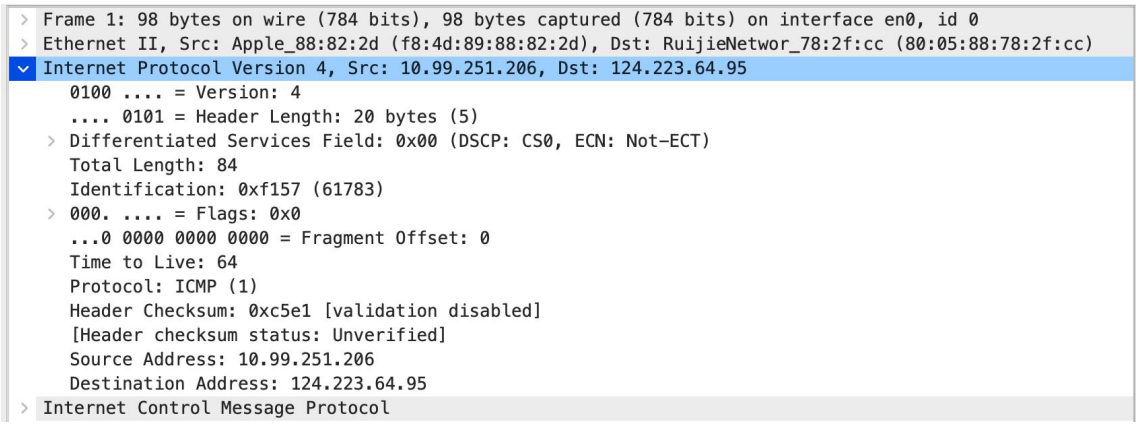
5.1 ping 命令

本机启动 Wireshark 软件，选择要监听的网络接口（如 eth0、wlan0）；然后在终端发起网络命令：ping IP 地址/域名。

1. 在 Wireshark 监视器中设置过滤条件。例如设置过滤条件为 icmp，则显示出所捕获的 ICMP 数据包。



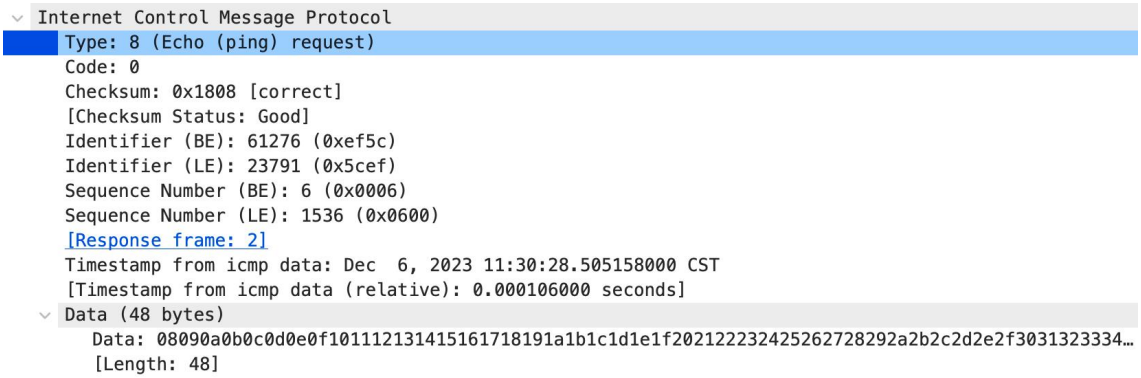
2. 点击 Internet Protocol Version 4 展开，查看 IP 数据报，特别观察 IP 数据报的首部字段内容。

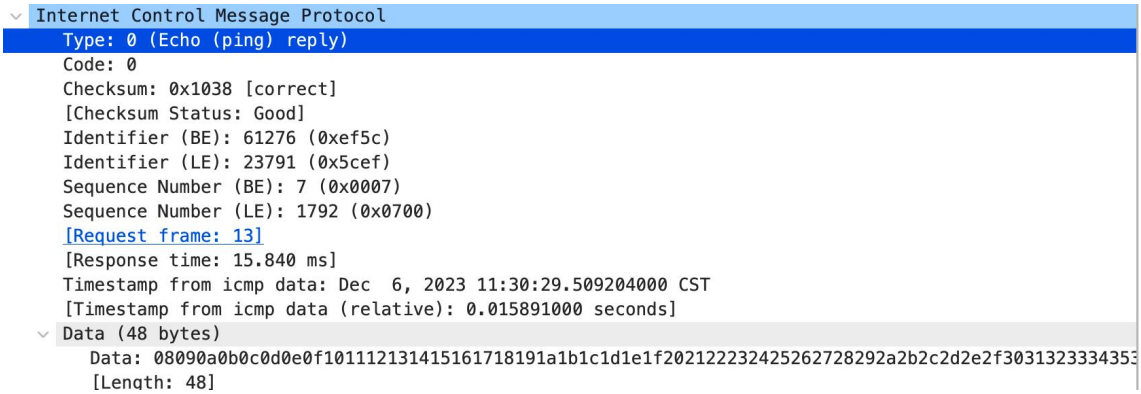


3. 点击 Internet Control Message Protocol 展开，查看 ICMP 报文，并解释回显（Echo Request 和 Echo Reply）报文的首部字段。

Type：8 表示这是 Echo Request

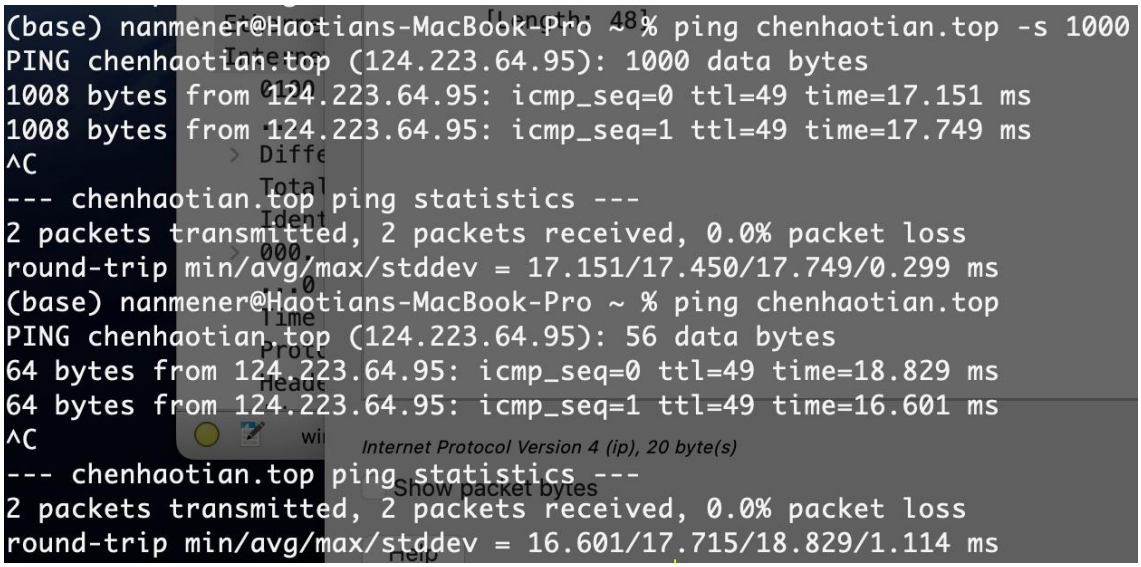
Type：0 表示这是 Echo Reply





4. 清空 Wireshark 监控器，重新发起网络命令：ping IP 地址/域名 –l#length，并解释对比前后两次执行 ping 命令的结果。其中，–l #length 确定 echo 数据报的长度为 #length，其默认值为 32 字节，且小于 65,527 字节。

第一次 ICMP 数据负载为 1008 字节，第二次为 64 字节。



10.99.251.206	124.223.64.95	ICMP	1042	Echo (ping) request
124.223.64.95	10.99.251.206	ICMP	1042	Echo (ping) reply
10.99.251.206	124.223.64.95	ICMP	98	Echo (ping) request
124.223.64.95	10.99.251.206	ICMP	98	Echo (ping) reply

5. 可以多次改变 #length 的大小(例如 1000 字节、2000 字节和 4000 字节),观察 IP 数据报何时会分片?请解释 IP 数据报分片的原因和具体情况。提示:请先确认该网络的 MTU,可在 Wireshark 记录中查找“IPv4 fragments”项目。

MTU 值是 1500 字节。包括 IP 头(20 字节)和 ICMP 头(8 字节)在内,数据部分最大不能超过 1472 字节,否则就会发生分片。

在 ICMP 数据负载为 1008 时,不分片;数据负载为 2008 和 4008 时,发生分片,如图所示。

2008 分片情况:

[2 IPv4 Fragments (2008 bytes): #92(1480), #93(528)]

[Frame: 92, payload: 0–1479 (1480 bytes)]

[Frame: 93, payload: 1480–2007 (528 bytes)]

4008 分片情况:

[3 IPv4 Fragments (4008 bytes): #121(1480), #122(1480), #123(1048)]

[Frame: 121, payload: 0–1479 (1480 bytes)]

[Frame: 122, payload: 1480–2959 (1480 bytes)]

[Frame: 123, payload: 2960–4007 (1048 bytes)]

```
(base) nanmener@Haotians-MacBook-Pro ~ % ping chenhaotian.top -s 1000
PING chenhaotian.top (124.223.64.95): 1000 data bytes
1008 bytes from 124.223.64.95: icmp_seq=0 ttl=49 time=20.526 ms
1008 bytes from 124.223.64.95: icmp_seq=1 ttl=49 time=57.200 ms
^C
--- chenhaotian.top ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 20.526/38.863/57.200/18.337 ms
(base) nanmener@Haotians-MacBook-Pro ~ % ping chenhaotian.top -s 2000
PING chenhaotian.top (124.223.64.95): 2000 data bytes
2008 bytes from 124.223.64.95: icmp_seq=0 ttl=49 time=17.873 ms
2008 bytes from 124.223.64.95: icmp_seq=1 ttl=49 time=20.079 ms
^C
--- chenhaotian.top ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 17.873/18.976/20.079/1.103 ms
(base) nanmener@Haotians-MacBook-Pro ~ % ping chenhaotian.top -s 4000
PING chenhaotian.top (124.223.64.95): 4000 data bytes
4008 bytes from 124.223.64.95: icmp_seq=0 ttl=49 time=17.888 ms
4008 bytes from 124.223.64.95: icmp_seq=1 ttl=49 time=17.982 ms
^C
--- chenhaotian.top ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 17.888/17.935/17.982/0.047 ms
(base) nanmener@Haotians-MacBook-Pro ~ %
```

	Time	Source	Destination	Protocol	Length	Info
31	2.044260	10.99.251.206	124.223.64.95	ICMP	1042	Echo (ping) request id=0x705e, seq=0/0, ttl=64 (re
34	2.064460	124.223.64.95	10.99.251.206	ICMP	1042	Echo (ping) reply id=0x705e, seq=0/0, ttl=49 (re
47	3.049253	10.99.251.206	124.223.64.95	ICMP	1042	Echo (ping) request id=0x705e, seq=1/256, ttl=64 (
48	3.106126	124.223.64.95	10.99.251.206	ICMP	1042	Echo (ping) reply id=0x705e, seq=1/256, ttl=49 (
93	9.461044	10.99.251.206	124.223.64.95	ICMP	562	Echo (ping) request id=0x725e, seq=0/0, ttl=64 (re
95	9.478765	124.223.64.95	10.99.251.206	ICMP	626	Echo (ping) reply id=0x725e, seq=0/0, ttl=49 (re
100	10.466649	10.99.251.206	124.223.64.95	ICMP	562	Echo (ping) request id=0x725e, seq=1/256, ttl=64 (
102	10.485858	124.223.64.95	10.99.251.206	ICMP	626	Echo (ping) reply id=0x725e, seq=1/256, ttl=49 (
123	14.987220	10.99.251.206	124.223.64.95	ICMP	1082	Echo (ping) request id=0x765e, seq=0/0, ttl=64 (re
126	15.004912	124.223.64.95	10.99.251.206	ICMP	1210	Echo (ping) reply id=0x765e, seq=0/0, ttl=49 (re
133	16.000482	10.99.251.206	124.223.64.95	ICMP	1082	Echo (ping) request id=0x765e, seq=1/256, ttl=64 (
136	16.018132	124.223.64.95	10.99.251.206	ICMP	1210	Echo (ping) reply id=0x765e, seq=1/256, ttl=49 (

```
Internet Protocol Version 4, Src: 10.99.251.206, Dst: 124.223.64.95
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 548
    Identification: 0xae23 (44579)
  > 000. .... = Flags: 0x0
    ...0 0000 1011 1001 = Fragment Offset: 1480
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x068d [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.99.251.206
    Destination Address: 124.223.64.95
  v [2 IPv4 Fragments (2008 bytes): #92(1480), #93(528)]
      [Frame: 92, payload: 0-1479 (1480 bytes)]
      [Frame: 93, payload: 1480-2007 (528 bytes)]
      [Fragment count: 2]
      [Reassembled IPv4 length: 2008]
      [Reassembled IPv4 data [truncated]: 08004da4725e0000656fef1a000eab4c08090a0b0c0d0e0f10111213141516
```

```
Internet Protocol Version 4, Src: 10.99.251.206, Dst: 124.223.64.95
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1068
    Identification: 0x2df5 (11765)
  > 000. .... = Flags: 0x0
    ...0 0001 0111 0010 = Fragment Offset: 2960
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x83fa [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.99.251.206
    Destination Address: 124.223.64.95
  v [3 IPv4 Fragments (4008 bytes): #121(1480), #122(1480), #123(1048)]
      [Frame: 121, payload: 0-1479 (1480 bytes)]
      [Frame: 122, payload: 1480-2959 (1480 bytes)]
      [Frame: 123, payload: 2960-4007 (1048 bytes)]
      [Fragment count: 3]
      [Reassembled IPv4 length: 4008]
      [Reassembled IPv4 data [truncated]: 0800bbf6765e0000656fef200007704e08090a0b0c0d0e0f101112131415161
```

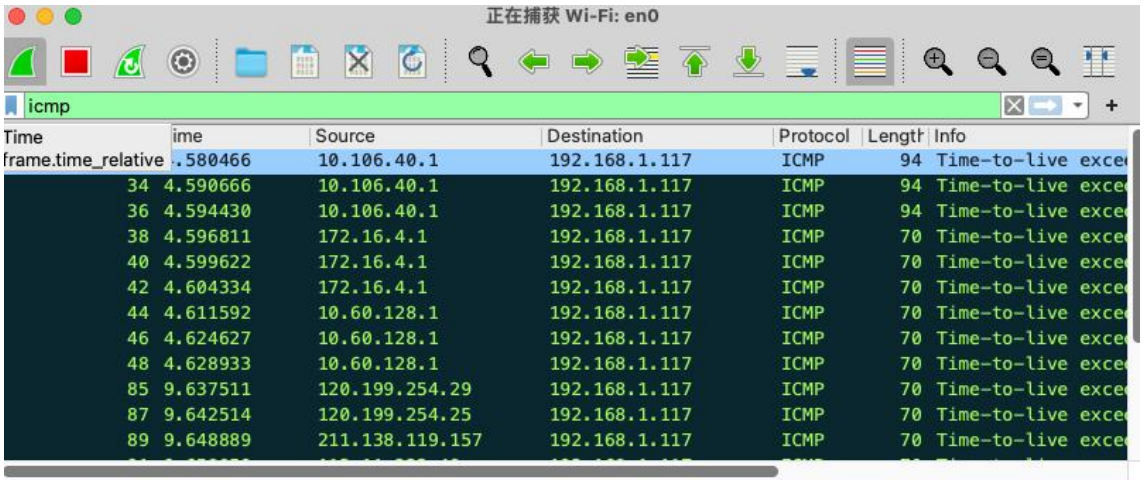
5.2 traceroute 命令

本机启动 Wireshark 软件，选择要监听的网络接口（如 eth0、wlan0）；然后在终端发起网络命令：traceroute IP 地址/域名。

1. 启动 Wireshark 软件，选择要监听的网络接口，设置过滤条件 icmp。

2. 在终端中使用 traceroute 命令，目的主机是外网的一台设备。

```
(base) nanmener@Haotians-MacBook-Pro ~ % traceroute 221.183.45.101
traceroute to 221.183.45.101 (221.183.45.101), 64 hops max, 52 byte packets
 1  10.106.40.1 (10.106.40.1)  2.163 ms  3.909 ms  3.737 ms
 2  172.16.4.1 (172.16.4.1)  2.365 ms  1.857 ms  5.015 ms
 3  10.60.128.1 (10.60.128.1)  6.926 ms  12.114 ms  4.325 ms
 4  * 120.199.254.29 (120.199.254.29)  5.757 ms
    120.199.254.25 (120.199.254.25)  3.719 ms
 5  211.138.119.157 (211.138.119.157)  5.085 ms
    112.11.232.49 (112.11.232.49)  8.068 ms
    117.148.181.25 (117.148.181.25)  4.316 ms
 6  221.183.47.165 (221.183.47.165)  6.139 ms
    221.183.47.169 (221.183.47.169)  4.650 ms
    221.183.47.165 (221.183.47.165)  5.418 ms
 7  221.183.40.149 (221.183.40.149)  9.927 ms *  8.047 ms
 8  * * *
 9  * * *
10  * * *
```



3. 点击 Internet Control Message Protocol 展开，查看 ICMP 差错报文，观察并解释 ICMP 报文结构和字段内容。

Type：11（Time-to-live exceeded）
这个字段表示 ICMP 报文的类型。类型 11 表示时间超出。

Code：0（Time to live exceeded in transit）
代码 0 与类型 11 一起表示数据包在传输过程中 TTL 值达到 0。

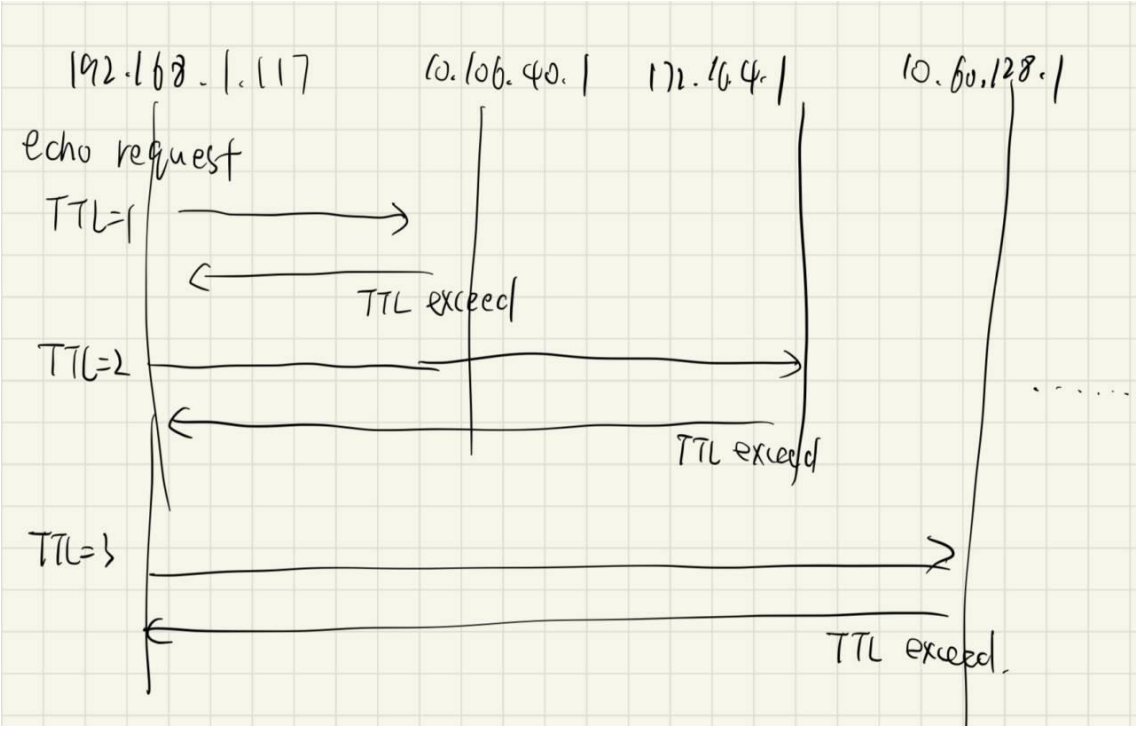
Checksum：0xc26b [correct]
校验和用于检测报文在传输过程中是否出现错误。

Unused：00000000
这个字段在类型 11 的 ICMP 报文中未使用。

```
Internet Control Message Protocol
    Type: 11 (Time-to-live exceeded)
    Code: 0 (Time to live exceeded in transit)
    Checksum: 0xc26b [correct]
    [Checksum Status: Good]
    Unused: 00000000
  Internet Protocol Version 4, Src: 192.168.1.117, Dst: 221.183.45.101
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      Total Length: 52
      Identification: 0x4ffc (20476)
    > 010. .... = Flags: 0x2, Don't fragment
      ...0 0000 0000 0000 = Fragment Offset: 0
    > Time to Live: 1
      Protocol: UDP (17)
      Header Checksum: 0x5c83 [validation disabled]
      [Header checksum status: Unverified]
      Source Address: 192.168.1.117
      Destination Address: 221.183.45.101
  > User Datagram Protocol, Src Port: 58286, Dst Port: 33435
Data (24 bytes)
    Data: 000000000000000000000000000000000000000000000000
    [Length: 24]
```

4．结合 ICMP 报文记录画出数据交互示意图，并描述 traceroute 工作原理。

traceroute 发送 TTL 值设置得很低（=1）的数据包。每当数据包通过一个路由器，它的 TTL 值就会减少 1。当数据包的 TTL 值降至 0 时，路由器会丢弃该数据包，并发送一个 ICMP “时间超出”响应给发送者。traceroute 接收到 ICMP 响应后，会递增 TTL 值，并发送另一个数据包。每次递增 TTL 值，直到达到目的主机或达到预设的最大跳数限制。当数据包到达目的主机时，目标会回复一个不同类型的 ICMP 错误。当收到这种类型的 ICMP 错误时，traceroute 知道它已经到达目的地，并结束路径追踪。



六． 实验心得与体会

通过这次实验，我对网络层协议 IP 和 ICMP 有了更深入的理解。通过操作 Wireshark 软件捕获和分析数据包，我对 IP 数据报的基本结构和 ICMP 报文的工作原理有了更直观的认识。

在执行 ping 命令的过程中，我观察到了 ICMP 询问报文的结构，并通过比较请求帧与回应帧，理解了它们在 IP 头部数据字段上的异同。这个过程让我意识到了 ping 工具在网络诊断中的重要作用，它可以帮助我们快速检测网络连通性和估计数据报的往返时间。

通过改变 ping 命令的参数来触发 IP 数据报的分片，我了解到当数据报的大小超过网络的 MTU 限制时，数据报就会发生分片。这个过程中，我认识到了 MTU 在网络传输中的重要性，并且体会到了分片对网络性能可能产生的影响。

执行 traceroute 命令时，我通过捕获和分析 ICMP 差错报文，理解了 traceroute 命令的工作原理和它如何追踪数据包的路由路径。通过实际操作，我能够将理论知识与实践相结合，这使我对数据包在网络中传输的过程有了更深刻的理解。

这次实验不仅提高了我的实际操作能力，也加深了我对网络层协议的认识。通过动手实践，我更加明白了理论知识的应用价值，并激发了我对网络技术更深入研究的兴趣。在未来的学习和工作中，我将继续探索和实践，以不断提高自己在网络技术领域的专业水平。