实验 1.2

秦华谦 21312683

一． 实验目的

1. 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
2. 掌握基于 ICMP 协议的 ping 和 traceroute 命令及其工作原理

二． 实验内容

1. 执行 ping 命令，观察 IP 数据报和 ICMP 询问报文的结构：

我首先开启个人热点，接入个人电脑和 iPad:

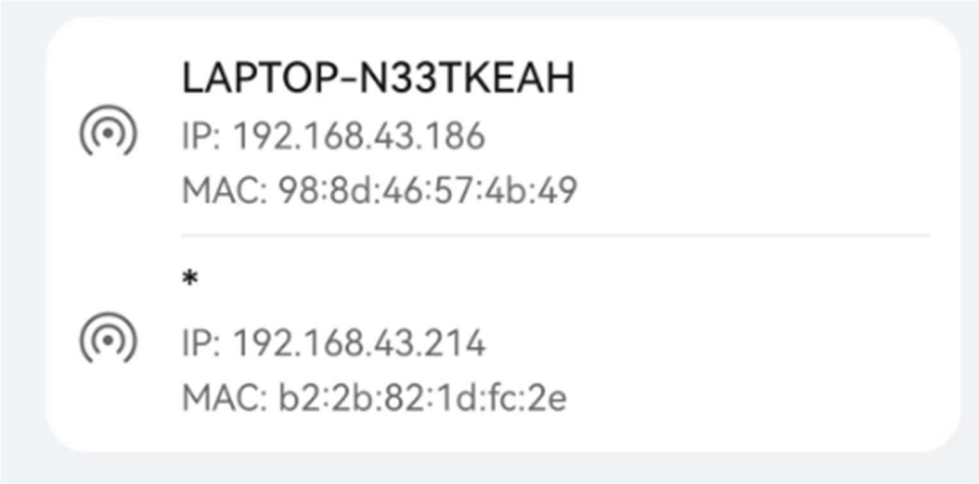


图 2.1.1 电脑及 iPad 的 IP 地址

本次实验，我选择通过电脑（ip：192.168.43.186）ping iPad（ip：192：168：43：214），设置过滤条件为 icmp，则可显示如下所捕获的 ICMP 数据包:

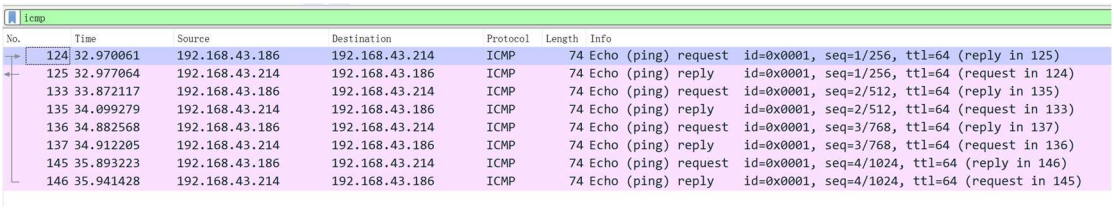


图 2.1.2 Wireshark 监视器界面

选择打开第一个数据包进行分析:

① **IP 数据报分析**

图 2.1.3 IP 数据报结构示意图

实验项	IP包字段名称		含义
1	Version		版本为4，IPV4
2	Differentiated Services		区分服务领域
3	Header Length		头部长度，指向数据开始的位置，这个域的最小合法值是5
4	Total Length		总长
5	Identification		标识符
6	Flags		标记字段，第1位不能使用，第2位是DF位，当DF为1 时，表示路由器不允许分段处理，为0 时，表示允许分段处理。 第3 位是MF位，当MF为1时表示不是最后一个分段，为0时表示是最后一个分段。
7	Fragment offset		分段偏移，表示是首段的偏移。以8个字节为偏移单位。
8	Time to live		生存期
9	Protocol		协议，指定了数据包中的数据类型
10	Header checksum		头部校验和，确保数据的正确性
11	Destination		目的地址
12	Source		源地址

表 2.1.4 IP 数据报结构解释表

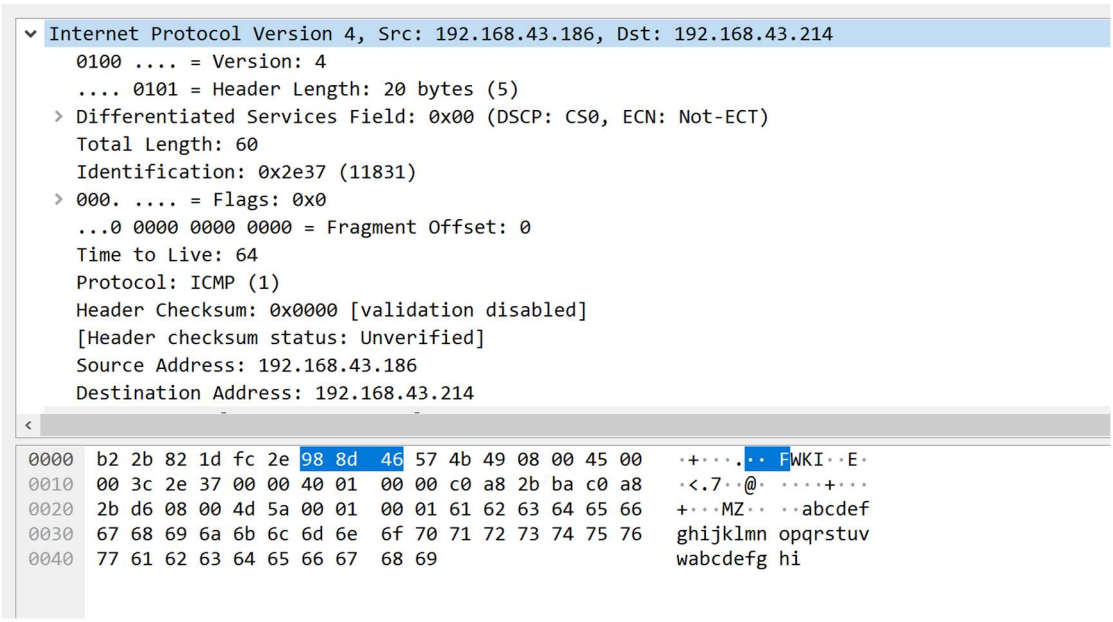


图 2.1.5 查看 IP 数据报

基于此，我们便可以对本 IP 数据包进行分析:

<Ⅰ> 版本（version）的序号为：08:00，由文字部分也可得：版本为 4，即 IPv4;

<Ⅱ> 首部长度（header length）为 20 字节，即 5 个 32 字节;

<Ⅲ>服务类型（differentiated services）为：0x00，全为 0，即没有特殊要求的一般服务;

<Ⅳ>总长度（total length）为 60 个字节;

<Ⅴ>标识（identification）为 0x2e37（11831），即为标识符;

<Ⅵ>标记（flags）为 0x00，第 1 位不使用，第 2 位 DF 位为 0 表示允许分段处理，第 3 位 MF 位为 0 表示这是最后一个分段;

<Ⅶ>片偏移（fragment offset）为 0，表示首段偏移为 0;

<Ⅸ>生存时间（time to live）为 64，表示 IP 数据包还能生存多久，根据操作系统不同而变化;

<Ⅹ>协议（protocol）为 ICMP;

<Ⅺ>头部校验和（header checksum）为 0x0000，确保数据的准确性;

<Ⅻ>源地址（source）为：192.168.43.186，为我的电脑的 IP 地址;

<ⅩⅢ>目的地址（destination）为：192.168.43.214，为我的 iPad 的 IP 地址。

② ICMP 报文分析

图 2.1.6 ICMP 数据报结构示意图

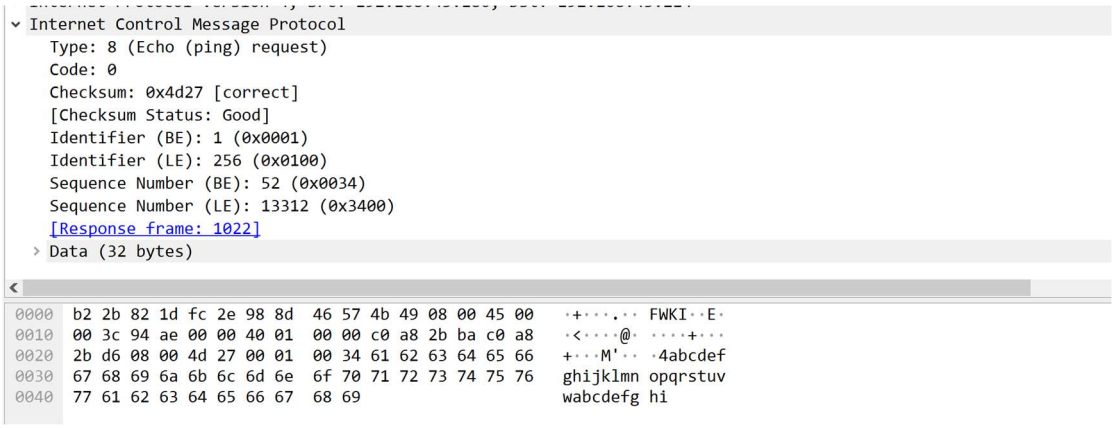


图 2.1.7 查看 ICMP 报文（request）

基于此，我们便可以对本 ICMP 数据包（request）进行分析：

<Ⅰ> 类型（type）为：8（回显请求）；

<Ⅱ> 代码（code）为 0；

<Ⅲ> 校验和（checksum）为 0x4d27，确保数据的准确性，由显示可得本次正确；

<Ⅳ> 标识符，大端顺序（identifier（BE））为：1（0x0001）；

<Ⅴ> 标识符，小端顺序（identifier（LE））为：256（0x0100）；

<Ⅵ> 序列号，大端顺序（sequence number （BE））为：52（0x0034）；

<Ⅶ> 序列号，小端顺序（sequence number （LE））为：13312（0x3400）。

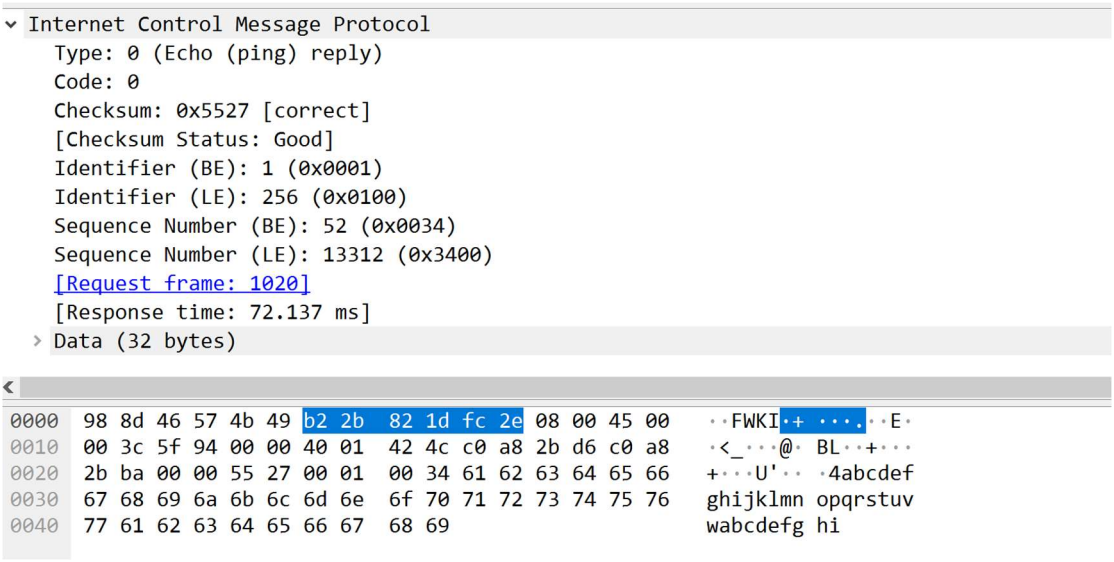


图 2.1.8 查看 ICMP 报文（reply）

基于此，我们便可以对本 ICMP 数据包（reply）进行分析：

<Ⅰ> 类型（type）为：0（回显应答）

<Ⅱ> 代码（code）为 0；

<Ⅲ> 校验和（checksum）为 0x5527，确保数据的准确性，由显示可得本次正确；

<Ⅳ> 标识符，大端顺序（identifier（BE））为：1（0x0001）；

<Ⅴ> 标识符，小端顺序（identifier（LE））为：256（0x0100）；

<Ⅵ> 序列号，大端顺序（sequence number （BE））为：52（0x0034）；

<Ⅶ> 序列号，小端顺序（sequence number （LE））为：13312（0x3400）；

<Ⅸ> 回应时间（response time）为：72.127ms。

对比可得，ICMP Echo Request 和 Echo Reply 的主要区别在于类型（type）和校验和（checksum）。类型不同的原因在于两个报文一个为回显请求，一个为回显应答；校验和不同的原因在于两个报文的内容不同，则校验和不同。同时，Echo Reply 还多了一项回应时间（response time），单位为毫秒。

2. 改变 ping 命令的参数，观察 IP 数据报分片：

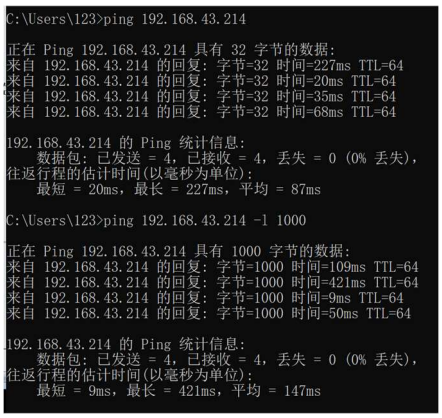


图 2.2.1 ping 命令执行

如图可得，ping 命令默认为 32 字节的数据，由于数据较少，不会出现分片情况。

① 改变 length 长度为 1000

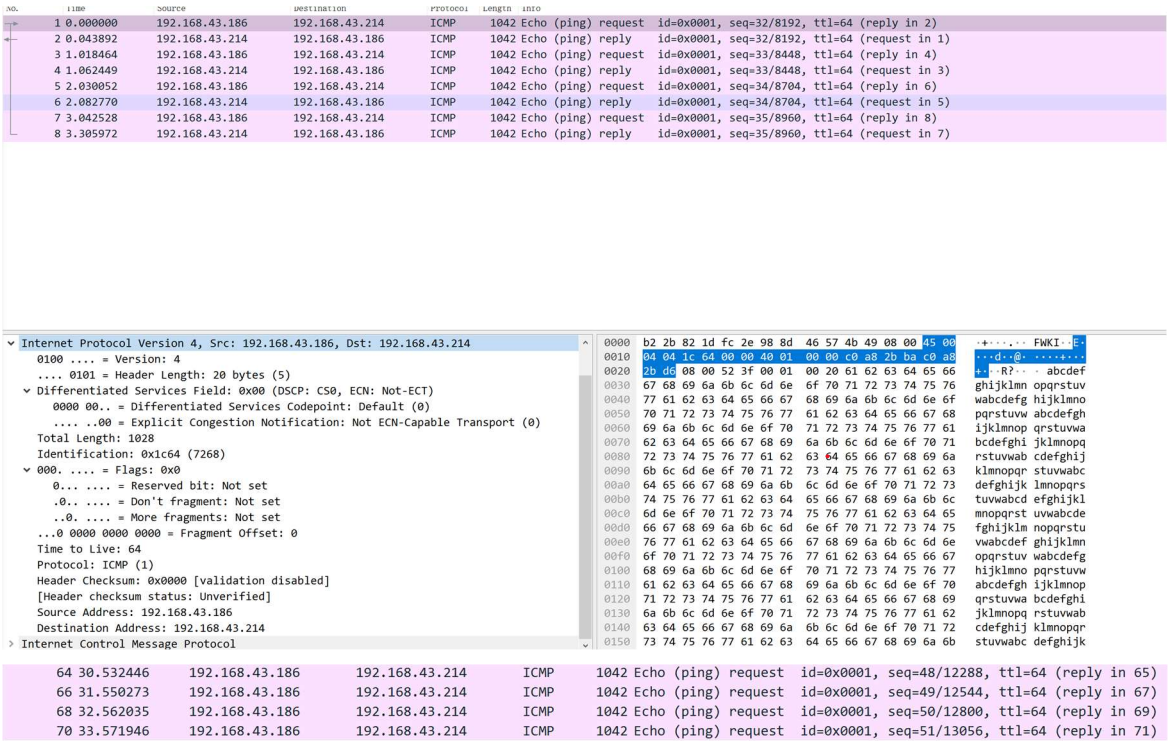
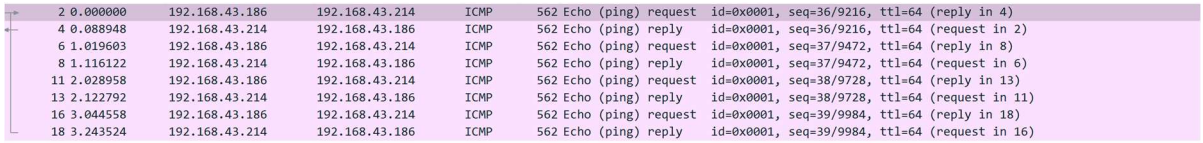


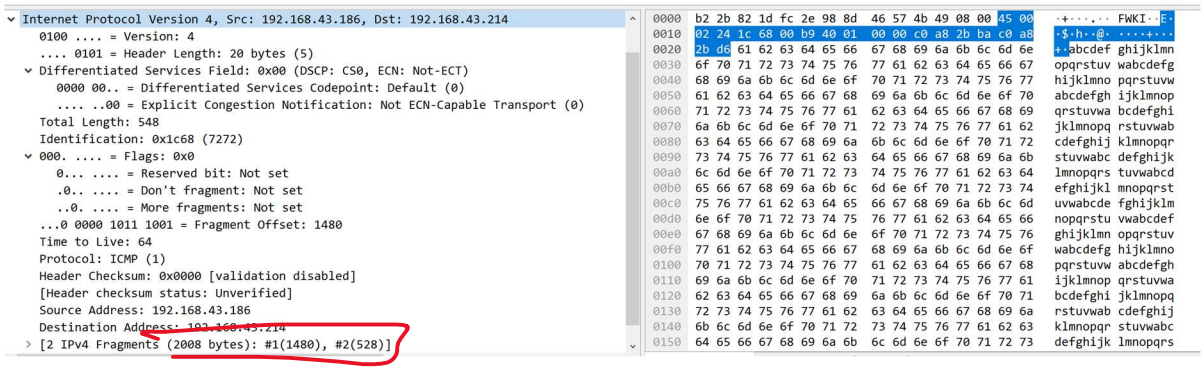
图 2.2.2 改变 length 大小为 1000 结果

由图可见，当字节长度为 1000 时，没有分片。

② 改变 length 长度为 2000



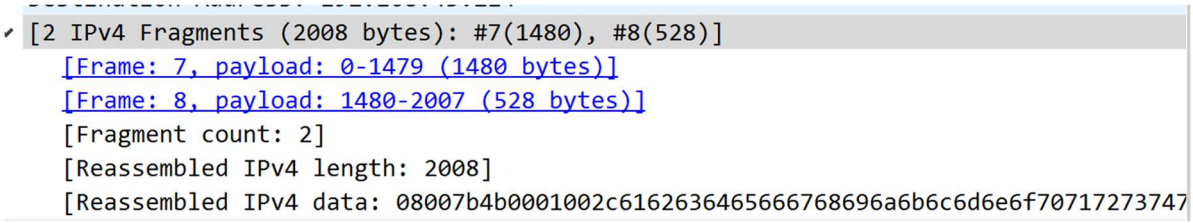
2	0.000000	192.168.43.186	192.168.43.214	ICMP	562 Echo (ping) request id=0x0001, seq=36/9216, ttl=64 (reply in 4)
4	0.088948	192.168.43.214	192.168.43.186	ICMP	562 Echo (ping) reply id=0x0001, seq=36/9216, ttl=64 (request in 2)
6	1.019603	192.168.43.186	192.168.43.214	ICMP	562 Echo (ping) request id=0x0001, seq=37/9472, ttl=64 (reply in 8)
8	1.116122	192.168.43.214	192.168.43.186	ICMP	562 Echo (ping) reply id=0x0001, seq=37/9472, ttl=64 (request in 6)
11	2.028958	192.168.43.186	192.168.43.214	ICMP	562 Echo (ping) request id=0x0001, seq=38/9728, ttl=64 (reply in 13)
13	2.122792	192.168.43.214	192.168.43.186	ICMP	562 Echo (ping) reply id=0x0001, seq=38/9728, ttl=64 (request in 11)
16	3.044558	192.168.43.186	192.168.43.214	ICMP	562 Echo (ping) request id=0x0001, seq=39/9984, ttl=64 (reply in 18)
18	3.243524	192.168.43.214	192.168.43.186	ICMP	562 Echo (ping) reply id=0x0001, seq=39/9984, ttl=64 (request in 16)



```
Internet Protocol Version 4, Src: 192.168.43.186, Dst: 192.168.43.214
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 548
    Identification: 0x1c68 (7272)
  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
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.43.186
    Destination Address: 192.168.43.214
  [2 IPv4 Fragments (2008 bytes): #1(1480), #2(528)]
```

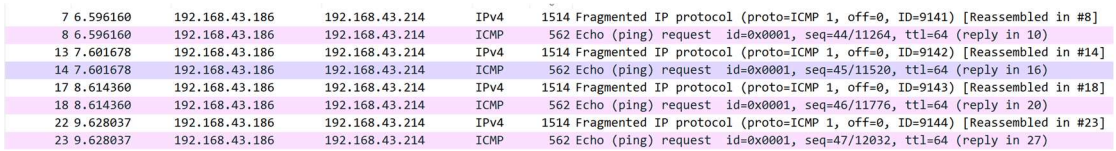
```
0000  b2 2b 82 1d fc 2e 98 8d  46 57 4b 49 08 00 45 00   ·+······ FWKI··E·
0010  02 24 1c 68 00 b9 40 01  00 00 c0 a8 2b ba c0 a8   ·$·h··@· ····+···
0020  2b d6 61 62 63 64 65 66  67 68 69 6a 6b 6c 6d 6e   +·abcdef ghijklmn
0030  6f 70 71 72 73 74 75 76  77 61 62 63 64 65 66 67   opqrstuv wabcdefg
0040  68 69 6a 6b 6c 6d 6e 6f  70 71 72 73 74 75 76 77   hijklmno pqrstuvw
0050  61 62 63 64 65 66 67 68  69 6a 6b 6c 6d 6e 6f 70   abcdefgh ijklmnop
0060  71 72 73 74 75 76 77 61  62 63 64 65 66 67 68 69   qrstuvwa bcdefghi
0070  6a 6b 6c 6d 6e 6f 70 71  72 73 74 75 76 77 61 62   jklmnopq rstuvwab
0080  63 64 65 66 67 68 69 6a  6b 6c 6d 6e 6f 70 71 72   cdefghij klmnopqr
0090  73 74 75 76 77 61 62 63  64 65 66 67 68 69 6a 6b   stuvwabc defghijk
00a0  6c 6d 6e 6f 70 71 72 73  74 75 76 77 61 62 63 64   lmnopqrs tuvwabcd
00b0  65 66 67 68 69 6a 6b 6c  6d 6e 6f 70 71 72 73 74   efghijkl mnopqrst
00c0  75 76 77 61 62 63 64 65  66 67 68 69 6a 6b 6c 6d   uvwabcde fghijklm
00d0  6e 6f 70 71 72 73 74 75  76 77 61 62 63 64 65 66   nopqrstu vwabcdef
00e0  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
00f0  77 61 62 63 64 65 66 67  68 69 6a 6b 6c 6d 6e 6f   wabcdefg hijklmno
0100  70 71 72 73 74 75 76 77  61 62 63 64 65 66 67 68   pqrstuvw abcdefgh
0110  69 6a 6b 6c 6d 6e 6f 70  71 72 73 74 75 76 77 61   ijklmnop qrstuvwa
0120  62 63 64 65 66 67 68 69  6a 6b 6c 6d 6e 6f 70 71   bcdefghi jklmnopq
0130  72 73 74 75 76 77 61 62  63 64 65 66 67 68 69 6a   rstuvwab cdefghij
0140  6b 6c 6d 6e 6f 70 71 72  73 74 75 76 77 61 62 63   klmnopqr stuvwabc
0150  64 65 66 67 68 69 6a 6b  6c 6d 6e 6f 70 71 72 73   defghijk lmnopqrs
```

图 2.2.3 改变 length 大小为 2000 结果



```
[2 IPv4 Fragments (2008 bytes): #7(1480), #8(528)]
    [Frame: 7, payload: 0-1479 (1480 bytes)]
    [Frame: 8, payload: 1480-2007 (528 bytes)]
    [Fragment count: 2]
    [Reassembled IPv4 length: 2008]
    [Reassembled IPv4 data: 08007b4b0001002c6162636465666768696a6b6c6d6e6f70717273747
```

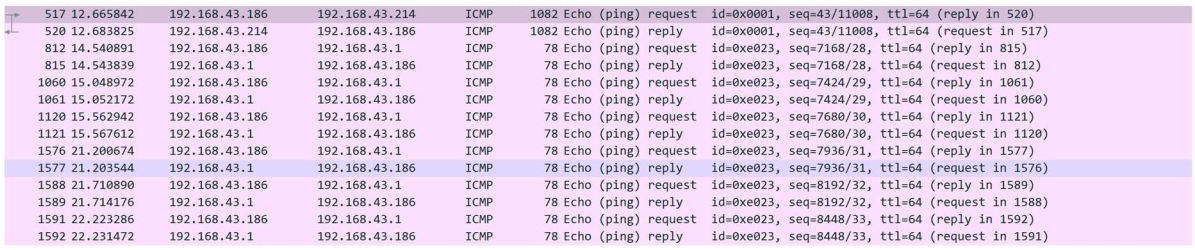
图 2.2.4 改变 length 大小为 2000 结果



7	6.596160	192.168.43.186	192.168.43.214	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=9141) [Reassembled in #8]
8	6.596160	192.168.43.186	192.168.43.214	ICMP	562 Echo (ping) request id=0x0001, seq=44/11264, ttl=64 (reply in 10)
13	7.601678	192.168.43.186	192.168.43.214	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=9142) [Reassembled in #14]
14	7.601678	192.168.43.186	192.168.43.214	ICMP	562 Echo (ping) request id=0x0001, seq=45/11520, ttl=64 (reply in 16)
17	8.614360	192.168.43.186	192.168.43.214	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=9143) [Reassembled in #18]
18	8.614360	192.168.43.186	192.168.43.214	ICMP	562 Echo (ping) request id=0x0001, seq=46/11776, ttl=64 (reply in 20)
22	9.628037	192.168.43.186	192.168.43.214	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=9144) [Reassembled in #23]
23	9.628037	192.168.43.186	192.168.43.214	ICMP	562 Echo (ping) request id=0x0001, seq=47/12032, ttl=64 (reply in 27)

图 2.2.5 改变 length 大小为 2000 分片情况

由图可见，当字节长度为 2000 时，有分片，多了一个数据包。先捕捉到的 IPv4 数据包长度为 1514，而接下来的 ICMP 包长度为 562。



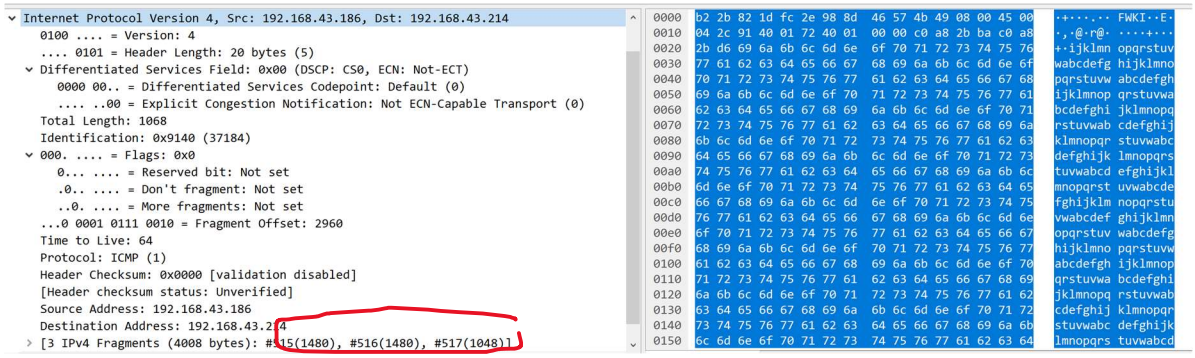


图 2.2.6 改变 length 大小为 4000 结果

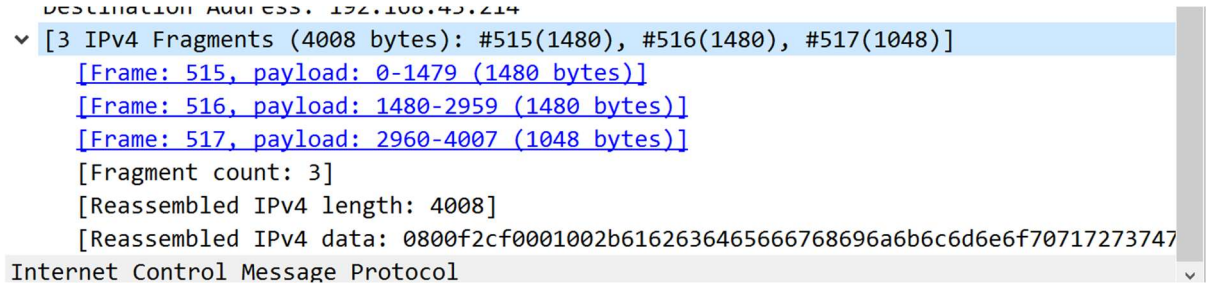


图 2.2.7 改变 length 大小为 4000 结果

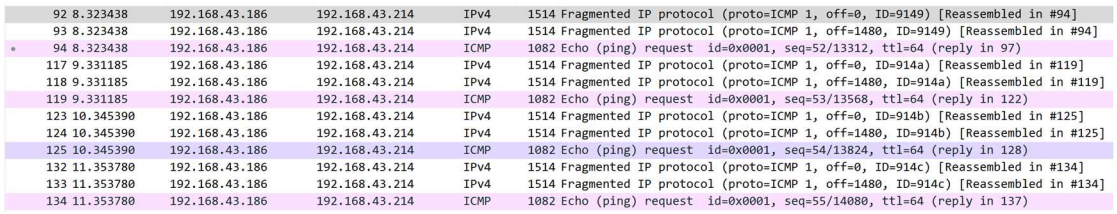


图 2.2.8 改变 length 大小为 4000 分片情况

由图可见，当字节长度为 4000 时，有分片，多了两个数据包。先捕捉到的两个 IPv4 数据包长度为 1514，而接下来的 ICMP 包长度为 1082。

首先确认局域网下的 MTU 为 1500 字节（其限制了数据帧的最大长度，传输的 IP 报文的大小超过最大传输单位 MTU 时就会产生 IP 分片情况）。
经过如上数据包分析，可初步得出结果：length 参数大于 1480 会出现分片的情况。
以 length 参数为 2000 的结果为例，其首先捕捉到的 IPv4 数据包长度为 1514 字节（14 字节以太网帧头+20 字节 IP 包头+1480 字节数据），fragment offset 为 0，代

表手段偏移为 0；而 ICMP 数据包长度为 528 字节（14 字节以太网帧头+20 字节 IP 包头+8 字节 ICMP 包头+剩余 520 字节数据），fragment offset 为 1480，代表前一段已包含 1480 字节的数据，与前一个 IPv4 数据包数据长度吻合，猜想成立。

综上，本局域网网关的最大传送单元 MTU 是 1500 字节，超过则自动分片，减去 IP 首部 20 字节，所以 IP 数据报能发 1480 字节。对于一个 IPv4 数据包，其还需包括 20 字节的 IP 首部和 14 字节的以太网帧头，所以其最大大小应为 1514.

同时我在实验中也发现了一个特殊的情况，就是在只发送一个数据包的情况下，想要保证不出现“fragmented IP protocol”，最大的允许数据字节长度应该为 1472 字节而非 1480 字节：

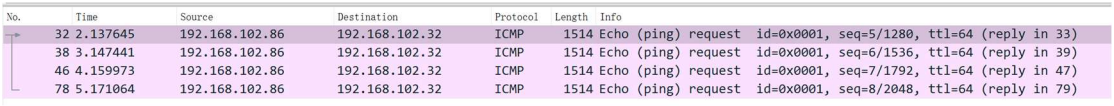


图 2.2.9 改变 length 大小为 1472 分片情况

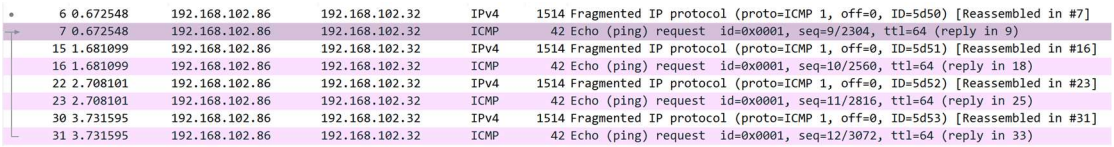


图 2.2.10 改变 length 大小为 1480 分片情况

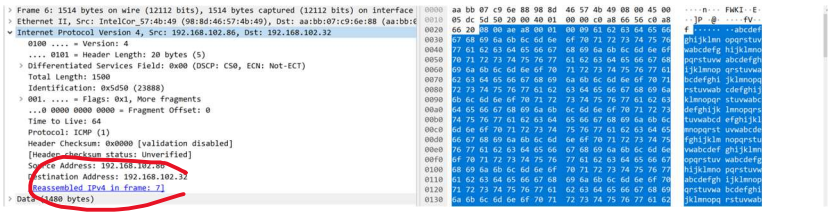


图 2.2.11 改变 length 大小为 1480 的 IPv4 包

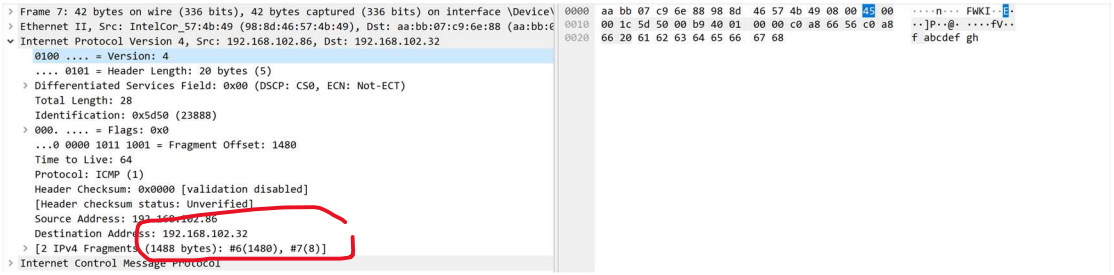


图 2.2.12 改变 length 大小为 1480 的 ICMP 包

由图 2.2.11，图 2.2.12 可见，当字节数为 1472 时不会发生分片，而字节数为 1480 时会发生分片。两个 IP 数据包数据的字节数分别为 1480 字节和 8 字节。但根据已学知识和 length 大小为 1480 的 IPv4 包中所显示的 data 为 1480 字节可猜想得：之所以在只发送一个数据包时，length 大于 1472 就会分片，是因为其只会发送一个 ICMP 报文，而 ICMP 报文比起 IPv4 报文会增加一个 8 字节 ICMP 包头，使得 data 的字节长度从 1480 字节被压缩至 1472 字节。

3. **执行 Traceroute 命令，观察 ICMP 差错报文的结构，并分析其工作原理**:

ping 工具只能测试目的设备的连通性，但是看不到数据包的传输路径。所以在网络不通的情况下，无法知道网络问题发生在哪个位置。tracert 工具可以查看数据包的整条传输路径，包括途中经过的中间设备。

```
C:\Users\123>tracert 210.34.0.1

通过最多 30 个跃点跟踪
到 router.xmu.edu.cn [210.34.0.1] 的路由:

  1     6 ms     4 ms     3 ms  192.168.102.32
  2     *        *        *     请求超时。
  3    35 ms    33 ms    31 ms  172.21.1.1
  4    30 ms    23 ms    18 ms  172.21.6.49
  5     *        *       39 ms  172.21.6.9
  6    26 ms    20 ms    17 ms  221.179.59.241
  7     *       47 ms     *     211.136.248.89
  8     *        *        *     请求超时。
  9     *       28 ms     *     221.183.90.218
 10    36 ms    30 ms    33 ms  221.183.95.218
 11    35 ms    26 ms    26 ms  101.4.118.229
 12    58 ms    39 ms    39 ms  101.4.112.46
 13    52 ms    44 ms    40 ms  101.4.115.90
 14    59 ms    39 ms    38 ms  101.4.118.86
 15     *        *        *     请求超时。
 16     *        *        *     请求超时。
 17     *        *        *     请求超时。
 18     *        *        *     请求超时。
 19     *        *        *     请求超时。
 20     *        *        *     请求超时。
 21     *        *        *     请求超时。
 22     *        *        *     请求超时。
 23     *        *        *     请求超时。
 24     *        *        *     请求超时。
 25     *        *        *     请求超时。
 26     *        *        *     请求超时。
 27     *        *        *     请求超时。
 28     *        *        *     请求超时。
 29     *        *        *     请求超时。
 30     *        *        *     请求超时。

跟踪完成。
```

图 2.3.1 tracert 结果

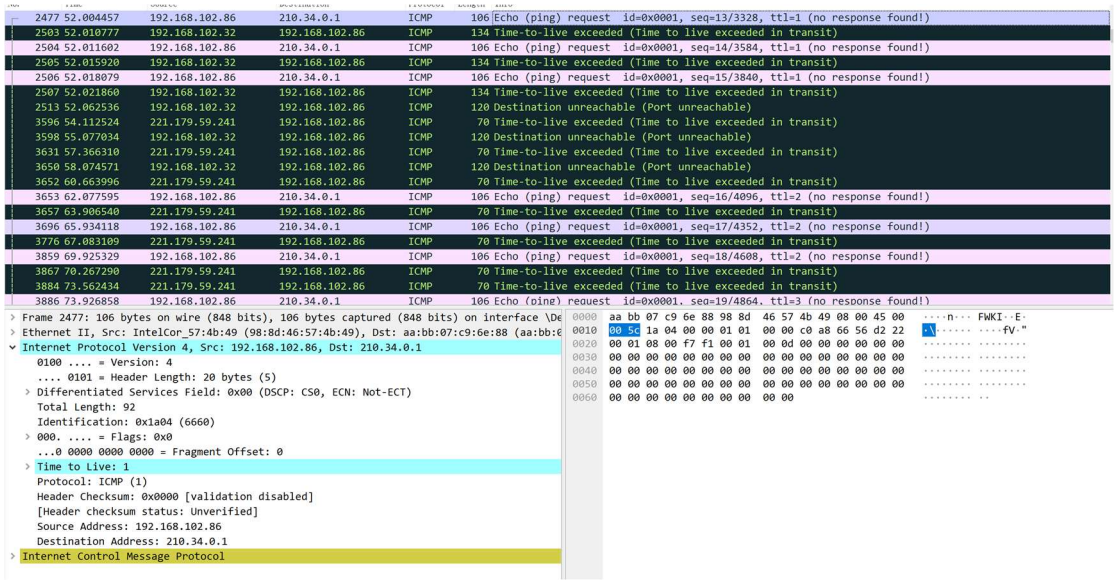


图 2.3.2 差错报文

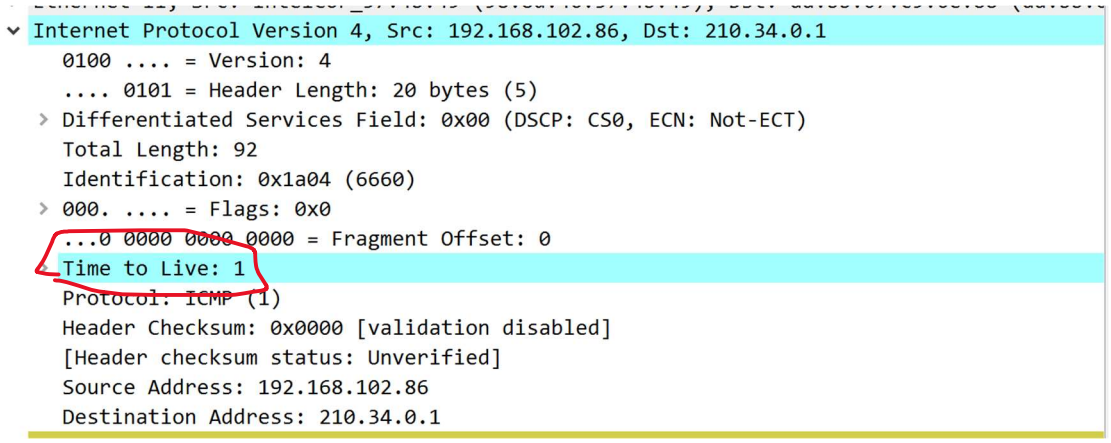


图 2.3.3 发送的第一个 ICMP 报文

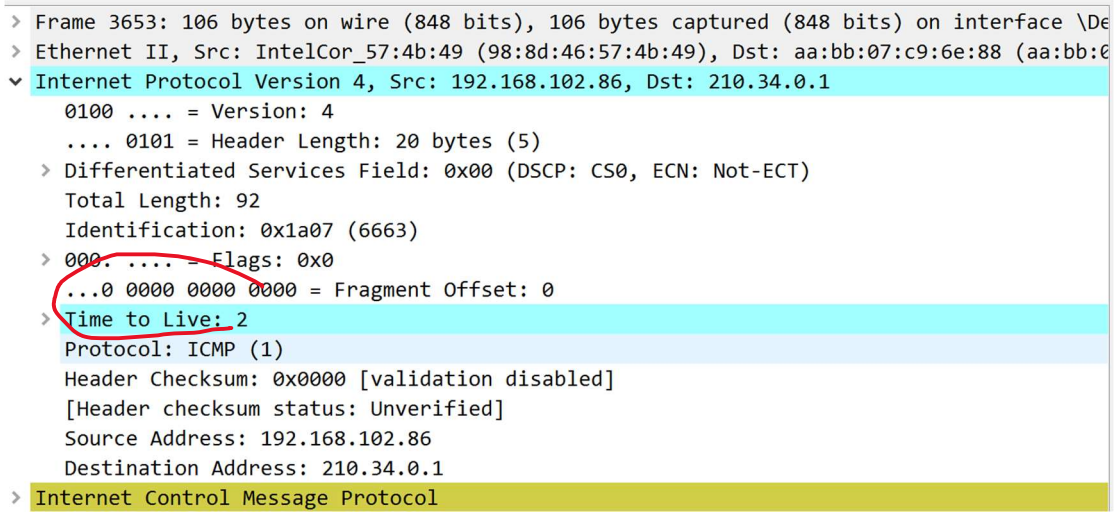


图 2.3.4 发送的第四个 ICMP 报文

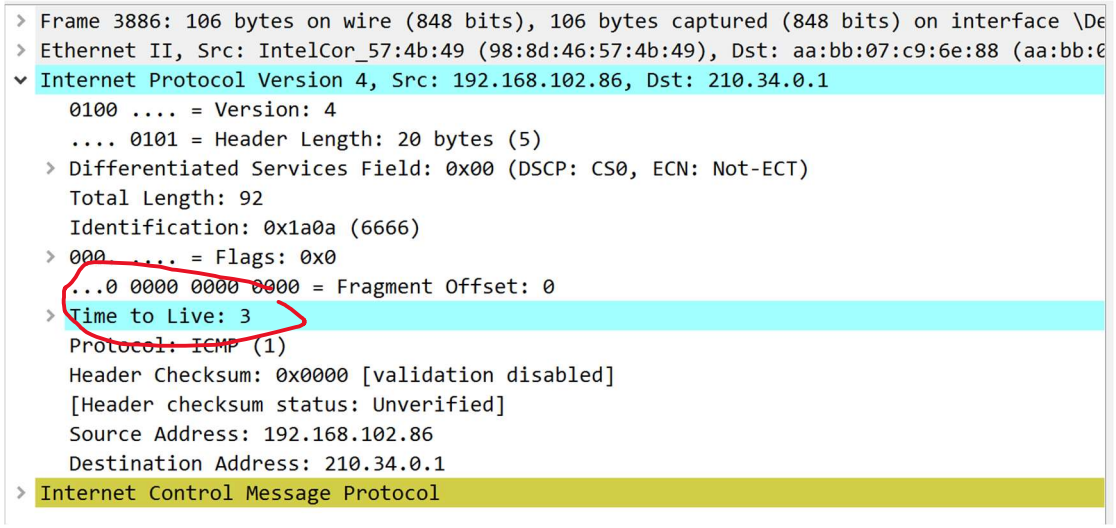


图 2.3.5 发送的第七个 ICMP 报文

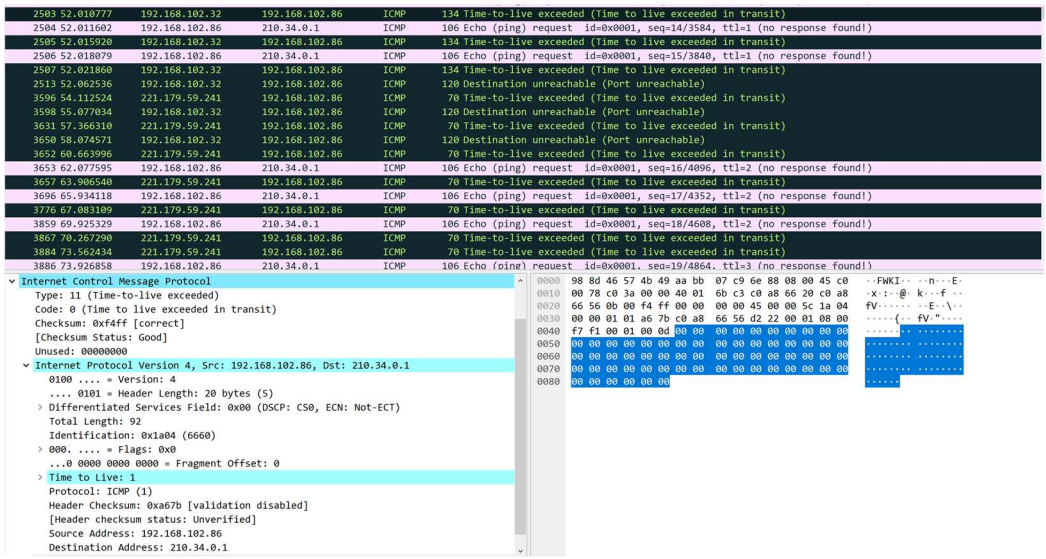


图 2.3.6 Time-to-live exceed 差错报文

由以上分析可得，在使用 tracert 时，原设备会逐跳发送数据包，并等待响应报文。由命令界面可得，原设备会向每一个路由器发送 3 个数据包。给第一个路由器发送数据包时，TTL 值设为 1，第二个设为 2，以此类推。且此数据报文是经过每一个路由器逐级送出的，每一个路由器收到数据包后会将 TTL 减 1，随即将其发向下一个路由器；若 TTL 已减少至 0，路由器则会将数据包丢弃，并返回一个 time exceeded 差错报文，原设备的 tracert 收到响应报文后，取出源 IP 地址。

由此可以去表述一个较为完整的 tracert 过程：

①第一个路由器收到数据包后 TTL 值减 1，随即丢弃数据包，并返回一个 time exceeded 消息。

②源设备的 tracert 收到响应报文后，取出源 IP 地址，即路径上的第一个路由器地址。

③然后 tracert 发送一个 TTL 值为 2 的数据包。第一个路由器将 TTL 值减 1，并转发数据包。

④第二个路由器再将 TTL 值减 1，丢弃数据包并返回一个 time exceeded 消息。

⑤tracert 收到响应报文后，取出源 IP 地址，即路径上的第二个路由器地址。

⑥类似步骤，tracert 逐跳获得每一个路由器的地址，并探测到目的设备的可达性。

⑦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。

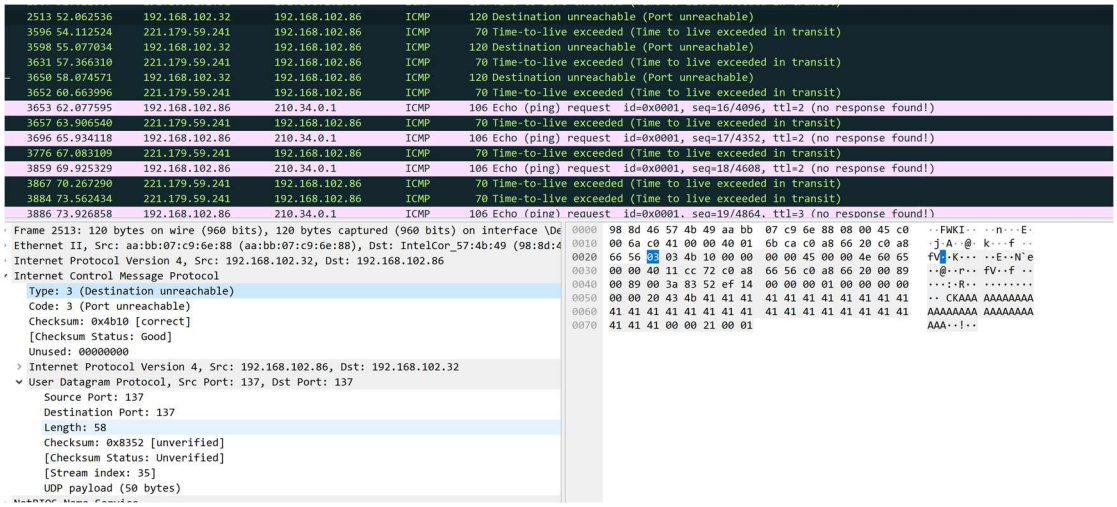


图 2.3.7 destination unreachable 差错报文

ICMP 五种差错报告报文：

① **终点不可达报文**：路由器 / 主机 不能交付数据报时，就会向源点 发送 终点不可达报文；

② **源点抑制报文**：路由器 / 主机 拥塞，丢弃 IP 数据报，向源点发送源点抑制报文，让源点降低发送速率；

③ **时间超过报文**：

- **生存周期为 0**：路由器 生存周期 TTL = 0 时，丢弃该报文，同时向源点发送 时间超过报文；
- **分组丢失**：终点 在预定时间内 没有收到 数据报的全部数据分组时，就会将已收到的数据分组全部丢弃，向源点发送时间超过报文；

④ **参数问题报文**：路由器 / 主机 收到的 数据报 首部 字段由错误值，丢弃该数据报，向源点发送 参数问题报文；

⑤ **改变路由报文**：路由器 将 改变路由报文 发送给主机，让主机下次将数据报发送给另外的路由器；又称为“重定向报文”；

图 2.3.8 ICMP 五种差错报文

由图 2.3.8 可知，差错报文主要包括重点不可达、源点抑制、时间超过、参数问题和改变路由报文等等。

由图 2.3.6 和图 2.3.7 可知，tracert 的差错报文主要包括时间超过报文（time exceeded）和终点不可达报文（destination unreachable）两种：time exceeded 报文主要通过设置 TTL 实现，以用于确认每一阶段所经过的路由器的 IP 地址和传输时间；destination unreachable 报文主要通过填入一个不可能的端口号值作为目标端口号实现。当目的主机，收到数据包后，会返回 ICMP 差错报文消息，且这个差错报文消息的类型是“destination unreachable”，以用于确认发送方发出的数据包到达了目的主机。

由图 2.3.6 和图 2.3.7 进行 ICMP 差错报文分析：

① Type 和 code 部分不同：time exceeded 报文的 type 为 11，code 为 0，代表其问传输期间生存时间为 0 的差错；destination unreachable 报文的 type 为 3，code 为 3，代表其为端口不可达的差错；

② tracert 的 data 不像 ping 命令中的随机字符，内容全为 0，由于其是错误分组，因此有一个 unused 字段，其值为 0；

③ IPv4 中所包含的信息与 ping 报文中无过大差别；

④ 路由跟踪的 ICMP 响应数据包（非 time exceeded）的 ICMP 的 type 为 0，代表 ICMP 响应，每次的序列号都不同。

经过如上分析，我所绘制出的 tracert 交互过程如图 2.3.9：

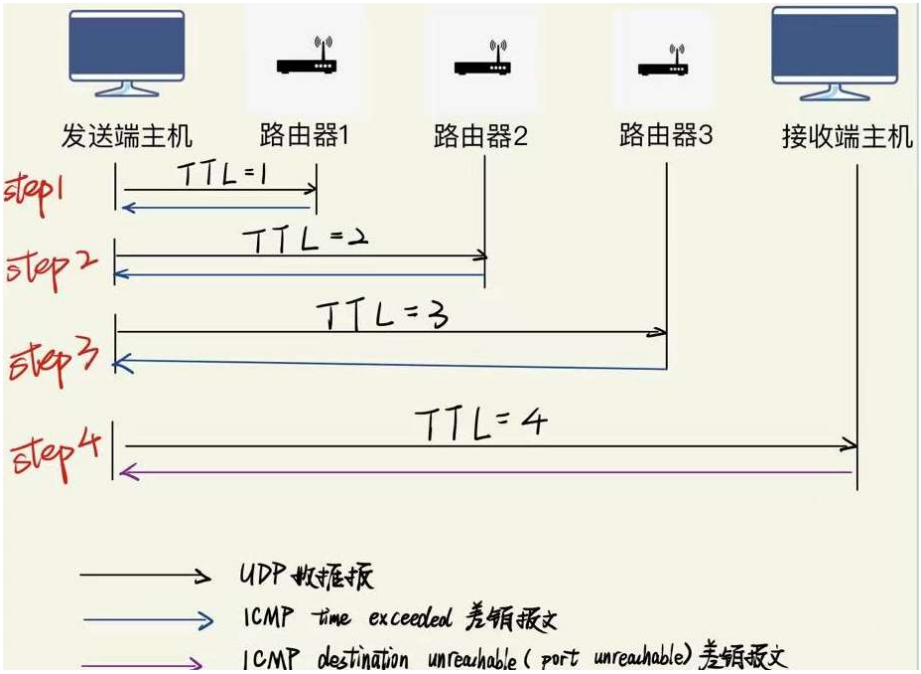


图 2.3.9 tracert 数据交互过程

三． 实验所遇问题

问题：在分析切片问题时，我发现我无法通过在 wireshark 中查找“IPv4 fragments”来确定 MTU

解决：通过在网上查找到的方式解决：

在本机打开dos窗口，执行：ping -f -l 1472 192.168.0.1 其中192.168.0.1是网关IP地址，1472是数据包的长度。请注意，上面的参数是“-l”（小写的L），而不是“-1”。 如果能ping通，表示数据包不需要拆包，可以通过网关发送出去。 如果出现： Packet needs to be fragmented but DF set. 表示数据包需要拆开来发送。此时，减少数据包长度，再执行上面的ping命令。从1400到1472之间多试几次，就能找到合适的数据包长度了。把数据包长度加上数据包头28字节，就得到MTU的值。 如果检测到网关的MTU值是1500，不需要修改。 如果网关有防火墙ping不通，可以试试直接把MTU设为1400。

图 3.1 查询 MTU 方法

四． 思考题

1. 问：在实际操作中，Traceroute 命令返回的某些条目以“*”号表示。请思考有哪些原因可能导致这样的情况。

答：

① 可能是防火墙封掉了 ICMP 的返回信息，使我们得不到什么相关的数据包返回数据。

② 可能使中间任何一个 router 上封了 ICMP Echo Request，traceroute 就不能工

作；中间的 router 看不到，但能看到 packet 到达了最后的 destination；如果封了 ICMP Echo Reply，中间的全能看到，最后的 destination 看不到。

2. 问：发送方要怎样决定 IP 数据报分组大小，才能避免因为不同网络 MTU 不一致而引起分片呢？

   答：

   可以探测路径最小 MTU：发送探测 IP 包，在包头中设置不可分片，然后通过是否受到 ICMP 消息即可确定路径 MTU。

   具体操作：首先在发送端主机发送 IP 数据报时将 IP 包首部的分片禁止标志位设置为 1。根据这个标志位，途中的路由器不会对大数据包进行分片，而是将包丢弃。随后，通过一个 ICMP 的不可达消息将数据链路上 MTU 的值一起给发送主机，不可达消息的类型为“需要进行分片但设置了不分片位”。发送主机端每次收到 ICMP 差错报文时就减少包的大小，以此来定位一个合适的分组大小和路径 MTU 值，就避免因为不同网络 MTU 不一致而引起分片。

五． 实验感悟

本次实验，我借助 wireshark 工具，深入了解了 IP 协议，ICMP 协议，分片操作和 tracert 操作流程。

通过本次实验，也使得抽象的理论知识形象化，使我更加可视化地了解了 IP 报文和 ICMP 报文，让我对本身较为枯燥的理论知识印象更为深刻，同时也了解了课内所不了解的分片和 tracert 知识，让我对本门课程产生了更为深刻的理解。