南開大學
Nankai University

《计算机网络》实验报告

(2023~2024 **学年第一学期**)

实验名称：IP 与 ICMP 分析

学　　院：软件学院

姓　　名：陈高楠

学　　号：2112966

指导老师：张圣林

2023 年 11 月 18 日

目录

实验 2 IP 与 ICMP 分析

1 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

2 实验条件

设备：PC 机一台，连入局域网；软件：Wireshark 软件，建议 3.0 以上版本。

3 实验报告内容及原理

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

执行 ping 命令

启动 Wireshark 监视器，然后在终端输入：ping 10.22.0.1。

```
C:\Users\ASUS>ping 10.22.0.1

正在 Ping 10.22.0.1 具有 32 字节的数据:
来自 10.22.0.1 的回复: 字节=32 时间=15ms TTL=255
来自 10.22.0.1 的回复: 字节=32 时间=14ms TTL=255
来自 10.22.0.1 的回复: 字节=32 时间=17ms TTL=255
来自 10.22.0.1 的回复: 字节=32 时间=15ms TTL=255

10.22.0.1 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 14ms, 最长 = 17ms, 平均 = 15ms
```

图 1: Enter Caption

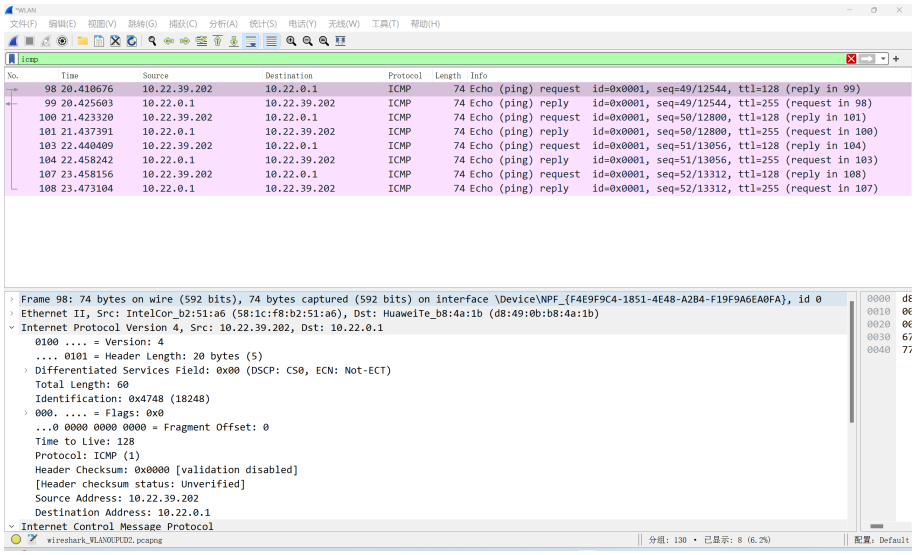

图 2: 筛选结果

接着在 Wireshark 监视器中设置过滤条件。设置过滤条件为 icmp，显示出所捕获的 ICMP 数据包。

观察 ip 数据报

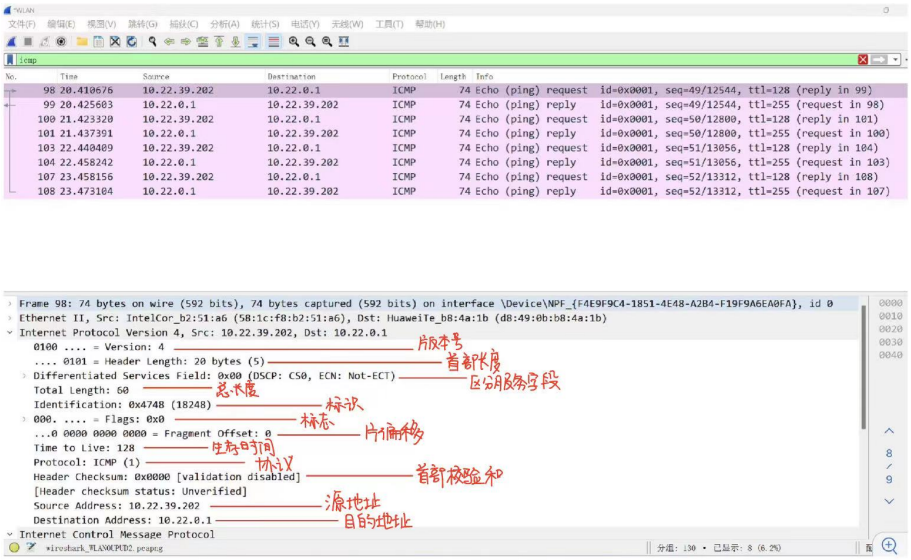

图 3: ip 数据报信息

版本号 (4bit)：表示使用的 IP 协议版本。

首部长度 (4bit)：表示整个 IP 数据包首部的长度。

区分服务字段 (8bit)：该字段一般不使用。

总长度 (16bit)：表示整个 IP 报文的长度，能表示的最大字节是 65535 字节。

标识 (16bit)：IP 软件通过计数器自动产生，每产生 1 个数据包计数器加 1，在 IP 分片后用来标识同一片分片。

标志 (3bit)：目前只有两位有意义，MF，置 1 表示后面还有分片，置 0 表示这是数据包片的最后 1 个；DF，不能分片标志，置 0 时表示允许分片。

片偏移（13bit），表示 IP 分片后，相应的 IP 片在总的 IP 片的相对位置。

生存时间 TTL(8bit)：表示数据包在网络中的生命周期，用通过路由器的数量来计量，即跳数（每经过一个路由器会减 1）。

协议（8bit），标识上层协议（TCP/UDP/ICMP…）。

首部校验和（16bit)：对数据包首部进行校验，不包括数据部分。

源地址（32bit)：标识 IP 片的发送源 IP 地址，

目的地址（32bit)：标识 IP 片的目的地 IP 地址。

观察 ICMP 询问报文的结构

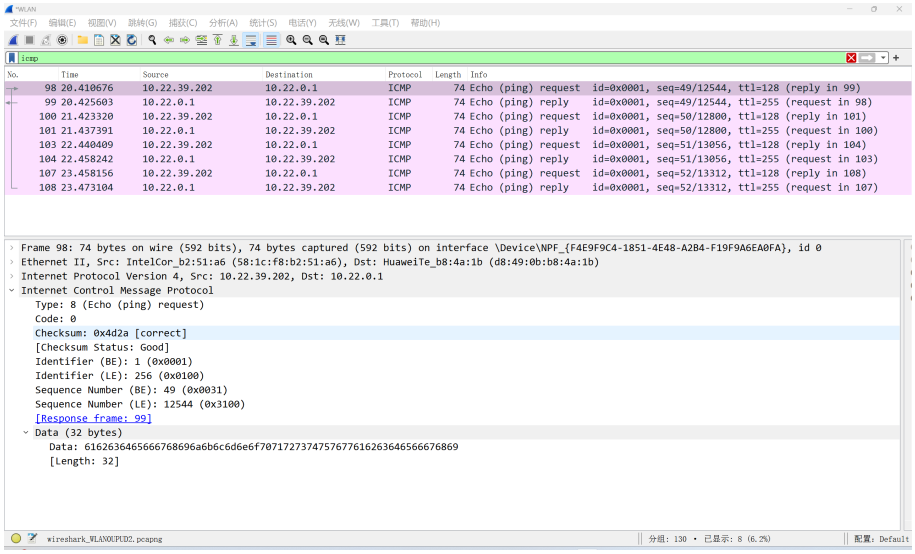

图 4: ICMP 询问报文

Type: 该字段有 1 个字节，表示特定类型的 ICMP 报文。

Code: 该字段有 1 个字节，进一步细分 ICMP 的类型。如上图所示，Type 的值为 8，Code 的值为 0，表示回显请求。

Checksum: 该字段有 2 个字节，表示校验和。

Identifier: 该字段有 2 个字节，用于匹配 Request/Reply 的标识符。

Seq Num: 该字段有 2 个字节，用于匹配 Request/Reply 的序列号。

Data: 数据载荷。

比较 ICMP 请求帧与回应帧

下面是 ICMP 请求帧和回应帧

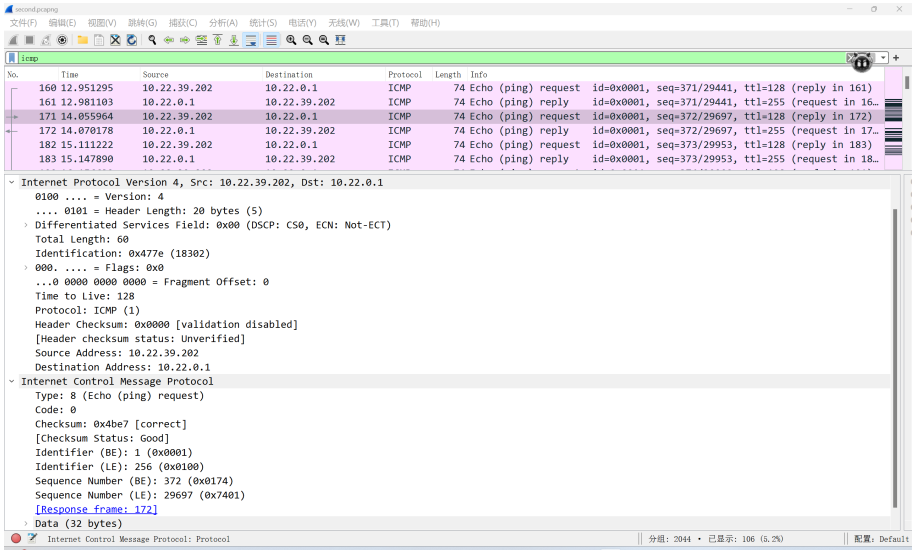

图 5: ICMP 请求帧

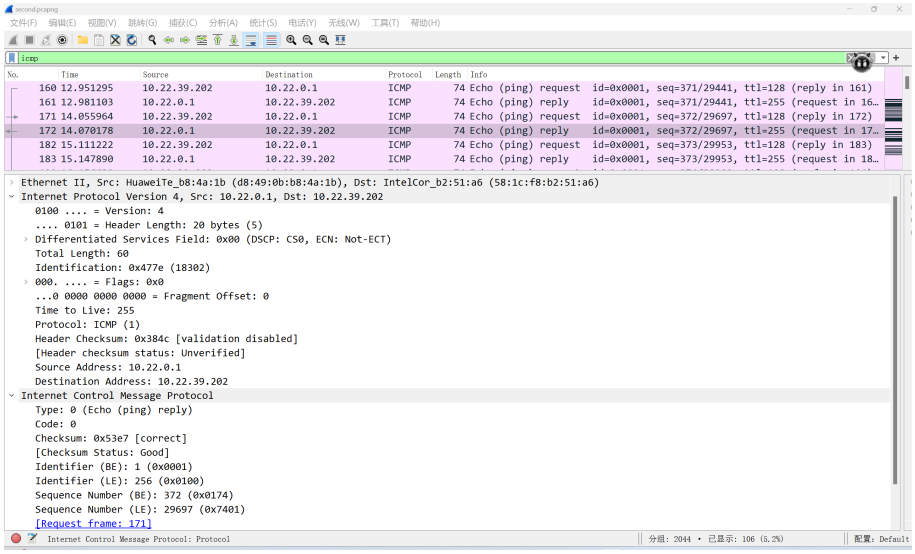

图 6: ICMP 回应帧

比较 IP 数据报：ICMP 请求和回应的版本号，首部长度，总长度，标识，标志，协议都一样。请求的源地址是 10.22.39.202，目标地址是 10.22.0.1，而回应的源地址是 10.22.0.1，目

标地址是 10.22.39.202。他们的生存周期也不一样，请求的生存周期 TTL=128，回应的生存周期 TTL=255。

比较 ICMP 报文：请求 Type 是 8,Code 是 0, 表示请求，回应 Type 是 0,Code 是 0, 表示应答。该请求和回应的 Identifier 一样，说明这个回应是回应这个请求的。

3.2 改变 ping 命令的参数，观察 IP 数据报分片

更改 ping 命令参数，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。更改 ping 的命令参数，分别改为 1000，2000，4000。

```
C:\Users\ASUS>ping 10.22.0.1 -l 1000

正在 Ping 10.22.0.1 具有 1000 字节的数据:
来自 10.22.0.1 的回复: 字节=1000 时间=12ms TTL=255
来自 10.22.0.1 的回复: 字节=1000 时间=16ms TTL=255
来自 10.22.0.1 的回复: 字节=1000 时间=15ms TTL=255
来自 10.22.0.1 的回复: 字节=1000 时间=12ms TTL=255

10.22.0.1 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 12ms, 最长 = 16ms, 平均 = 13ms
```

图 7: 字节长度改为 1000

```
C:\Users\ASUS>ping 10.22.0.1 -l 2000

正在 Ping 10.22.0.1 具有 2000 字节的数据:
来自 10.22.0.1 的回复: 字节=2000 时间=26ms TTL=255
来自 10.22.0.1 的回复: 字节=2000 时间=18ms TTL=255
来自 10.22.0.1 的回复: 字节=2000 时间=20ms TTL=255
来自 10.22.0.1 的回复: 字节=2000 时间=15ms TTL=255

10.22.0.1 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 15ms, 最长 = 26ms, 平均 = 19ms

C:\Users\ASUS>ping 10.22.0.1 -l 4000

正在 Ping 10.22.0.1 具有 4000 字节的数据:
来自 10.22.0.1 的回复: 字节=4000 时间=26ms TTL=255
来自 10.22.0.1 的回复: 字节=4000 时间=41ms TTL=255
来自 10.22.0.1 的回复: 字节=4000 时间=14ms TTL=255
来自 10.22.0.1 的回复: 字节=4000 时间=19ms TTL=255

10.22.0.1 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 14ms, 最长 = 41ms, 平均 = 25ms
```

图 8: 字节长度改为 2000 和 4000

由下图可知，当字节长度为 1000 时不分片

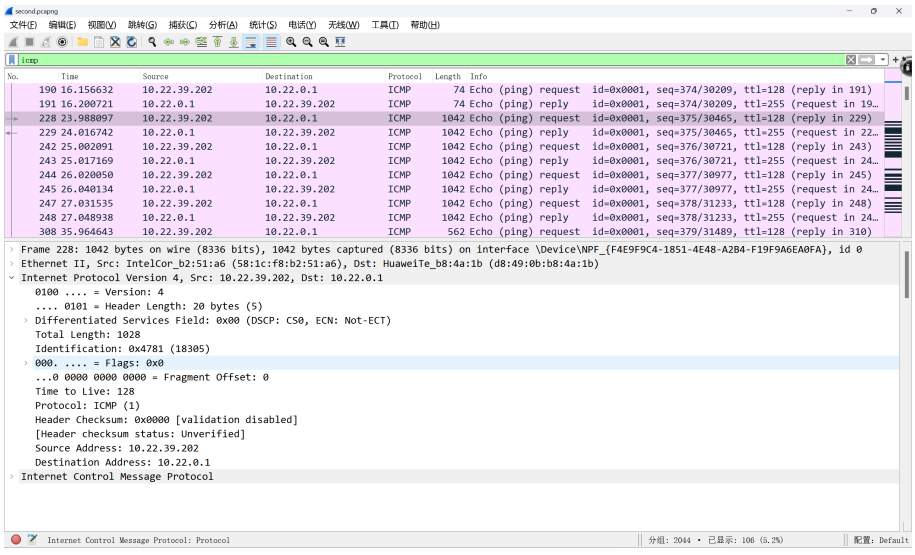

图 9: 字节长度为 1000 时

当字节长度为 2000 时，因为超出了 MTU，所以开始分片，此时分成了两个片。

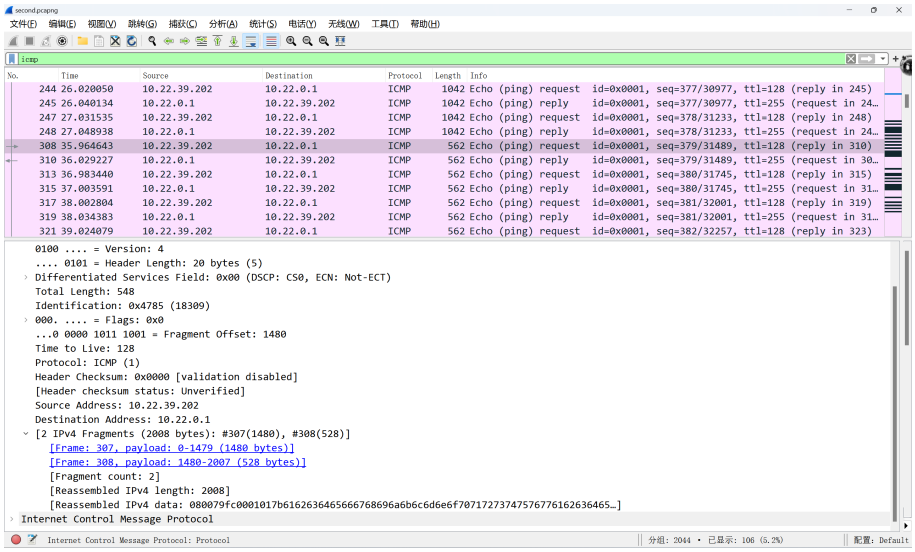

图 10: 分成两片

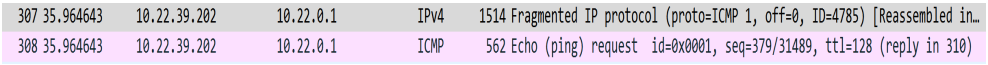

图 11: 分成两片

第一片的 ip 数据报如下图

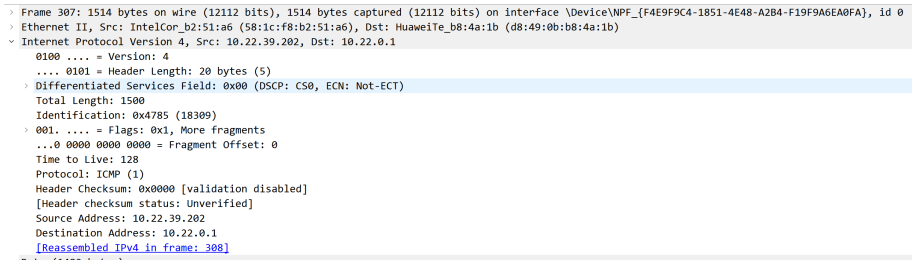

图 12: 分片第一片的 ip 数据报

第二片的 ip 数据报如下图

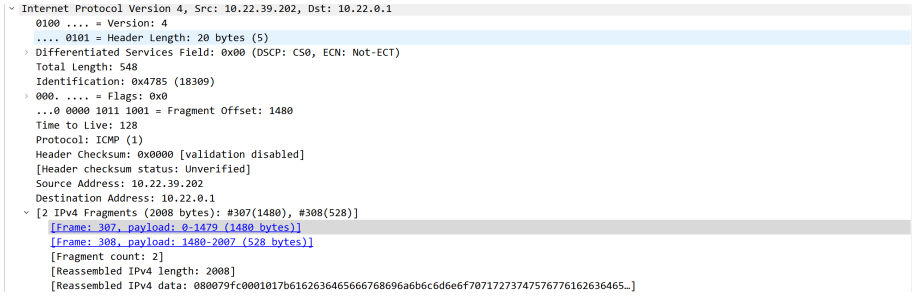

图 13: 分片第二片的 ip 数据报

分析：分片后，第一片的标志的 MF 置 1，表示后面还有分片，下面的片偏移表示分片后相应的 IP 片在哪。第二片的标志的 MF 置 0，表示后面已经没有分片，下面的片偏移表示分片后相应的 IP 片在哪。同时，IP 数据报中还显示了分片的大小，这里第一片分了 1480 字节，第二片分了 528 字节。

分片的原因: 以太网的最大传送单元 MTU 是 1500 字节; 超过则自动分片，减去 IP 首部 20 字节，所以能发 1480 字节。下面为字节长度为 4000 的情况。此时分成了三个片。

336	47.270180	10.22.39.202	10.22.0.1	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=4789) [Reassembled in…
337	47.270180	10.22.39.202	10.22.0.1	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=4789) [Reassembled…
338	47.270180	10.22.39.202	10.22.0.1	ICMP	1082 Echo (ping) request id=0x0001, seq=383/32513, ttl=128 (reply in 341)

图 14: 分成三片

3.3 执行 tracert 命令，观察 ICMP 差错报文的结构，并分析其工作原理

在终端中使用 tracert 命令，目的主机是外网的一台设备 (这里是 39.156.66.10)

```
C:\Users\ASUS>tracert 39.156.66.10

通过最多 30 个跃点跟踪到 39.156.66.10 的路由

  1     9 ms     4 ms     4 ms  10.22.0.1
  2     *        *       16 ms  202.113.18.102
  3    17 ms    15 ms    28 ms  111.33.78.1
  4     *        *        *     请求超时。
  5     *        *        *     请求超时。
  6     *        *       22 ms  221.183.40.13
  7    12 ms    10 ms    11 ms  221.183.49.126
  8     *        *        *     请求超时。
  9    23 ms    28 ms    20 ms  39.156.27.1
 10    40 ms    27 ms    18 ms  39.156.67.1
 11     *        *        *     请求超时。
 12     *        *        *     请求超时。
 13     *        *        *     请求超时。
 14    21 ms    17 ms    19 ms  39.156.66.10

跟踪完成。
```

图 15: tracert 命令

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。捕捉到两种差错报文。

1. 时间超过报文

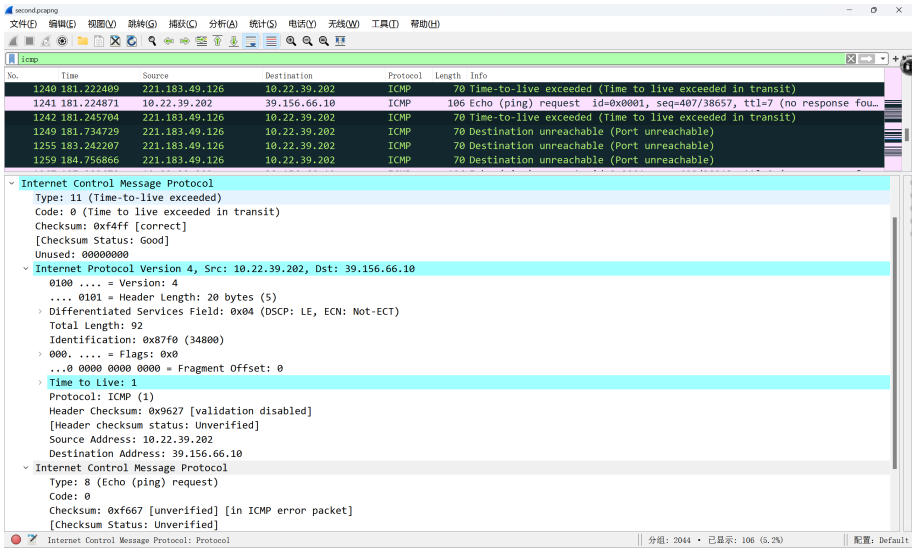

图 16: 时间超过报文

查看 ICMP 差错报文，Type 为 11，表示是超时。Code 为 0，进一步表示是因为数据包在到达目的地之前，它的生存时间值已经减少到 0。。在请求数据包中，生存周期 TTL = 1，再下一次就变为 0 了，就会丢弃该报文，同时向源点发送时间超过报文。这就是为什么会有时间超过报文。

2. 终点不可达报文

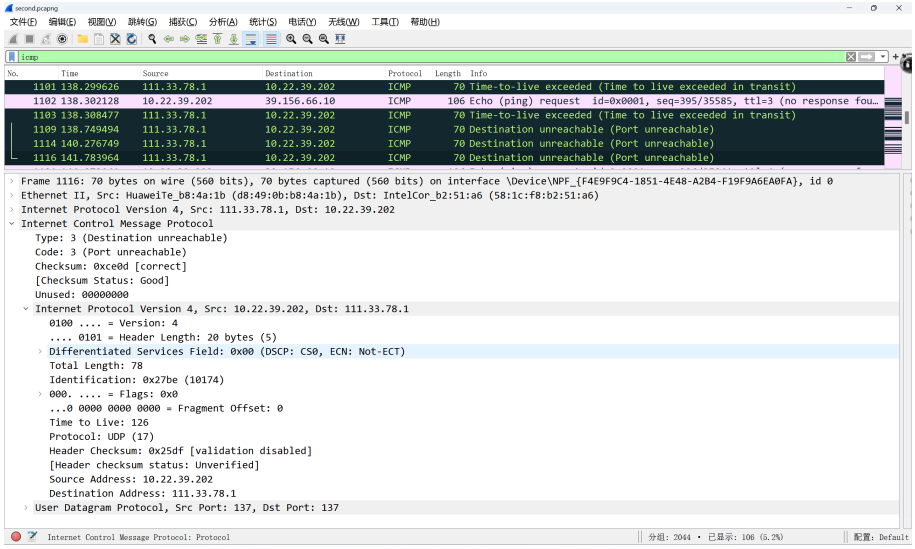

图 17: 终点不可达报文

查看 ICMP 差错报文，Type 为 3，表示目的不可达。Code 为 3，进一步表示是因为端口不可达。

以下为 tracert 工作原理示意图

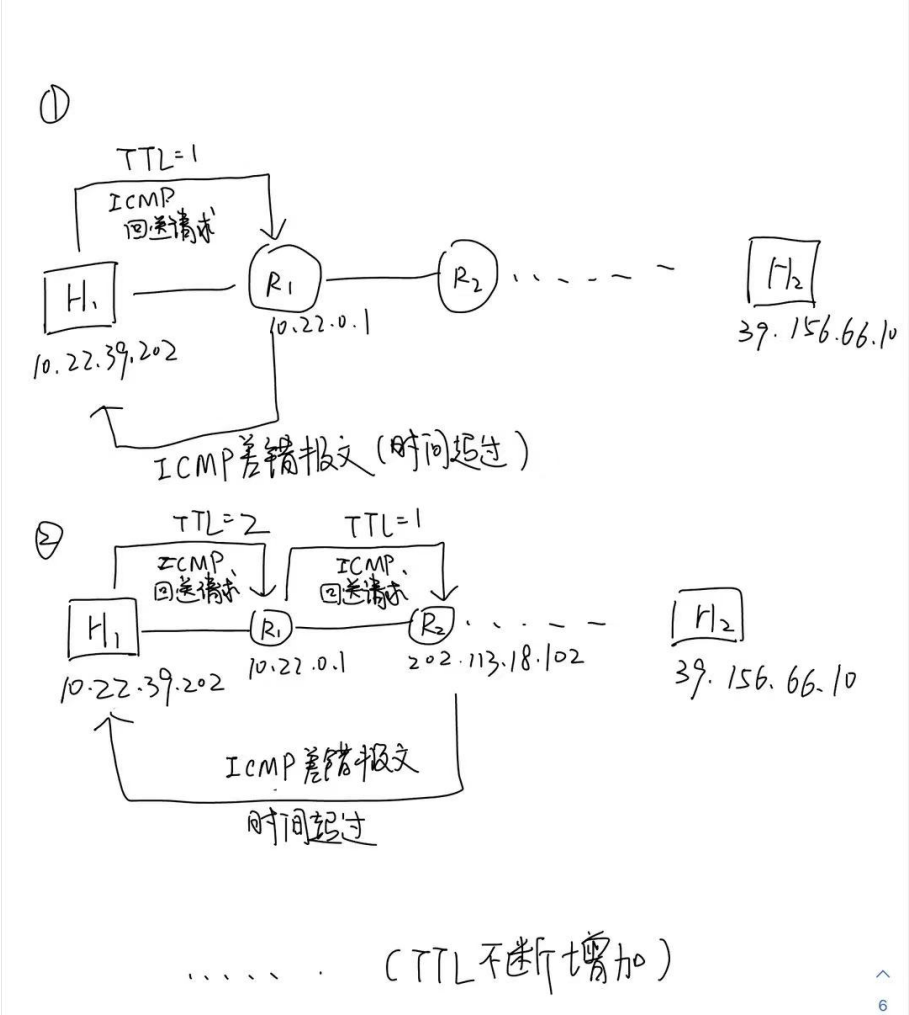

图 18: tracert 示意图 1

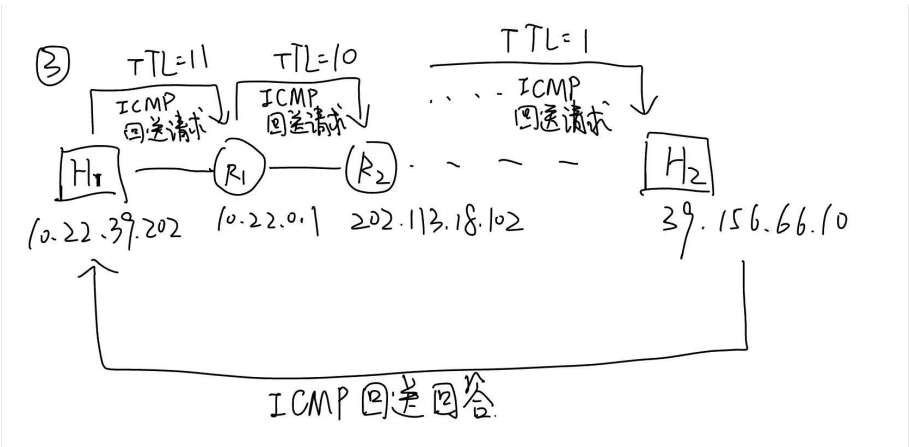

图 19: tracert 示意图 2

tracert 工作原理：tracert 先发送 TTL 为 1 的 ICMP 回应数据包，数据包上的 TTL 在

路由器收到后 TTL 自动减 1 ，某个路由器将 TTL 减 1 后，等于了 0，路由器就将 ICMP 时间差错报文回送给源计算机，源计算机根据收到的信息判断到达的路由器和所用的时间。如图 15 第一步所示。

接着再次发送数据包，将 TTL 递增 1，继续上述测试，直到目标响应或 TTL 达到最大值，从而确定路由。如图 15 第二步所示。通过检查中间路由器发回的 “ICMP 已超时” 的消息确定路由。某些路由器不经询问直接丢弃 TTL 过期的数据包，这在 tracert 实用程序中看不到，因此在终端中会显示超时。

最后当目标主机发送了回答到源主机时，就说明 IP 数据报已经到达，如图三，在 TTL 增加到 11 时可以到达目标主机。

4 实验结论及心得体会

这次实验我直观地观察到 IP 数据报的结构和 ICMP 报文的具体内容。我执行了 ping 命令并通过 Wireshark 观察了 IP 数据报和 ICMP 询问报文的结构，通过比较 ICMP 请求帧与回应帧，对 IP 头部数据字段的异同有了更深入的了解。接着我通过改变 ping 命令的参数来观察 IP 数据报分片，以此来理解分片的原因和原理。最后我执行 tracert 命令并观察 ICMP 差错报文的结构, 充分理解了 tracert 的原理，让我受益匪浅。