实验二：IP 和 TCP 数据分组的捕获和解析

1. 实验内容：

a) 捕获在连接 Internet 过程中产生的网络层分组：DHCP 分组，ARP 分组，IP 数据分组，ICMP 分组。

b) 分析各种分组的格式，说明各种分组在建立网络连接过程中的作用。

c) 分析 IP 数据分组分片的结构。

2. 实验过程及分析：

a) 捕获 DHCP 分组：

DHCP（Dynamic Host Configuration Protocol）称为动态主机分配协议。

为什么要使用 DHCP?

大致可以概括为以下四点：

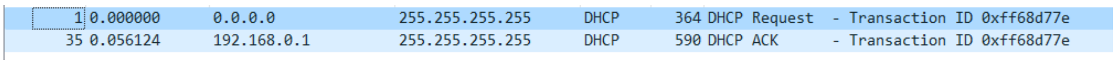
第一、　可以准确地配置 IP 地址

第二、　减少 IP 地址的冲突

第三、　实现 IP 地址管理的自动化

第四、　高效的变更管理

使用 wireshark 捕获到的 DHCP 分组：

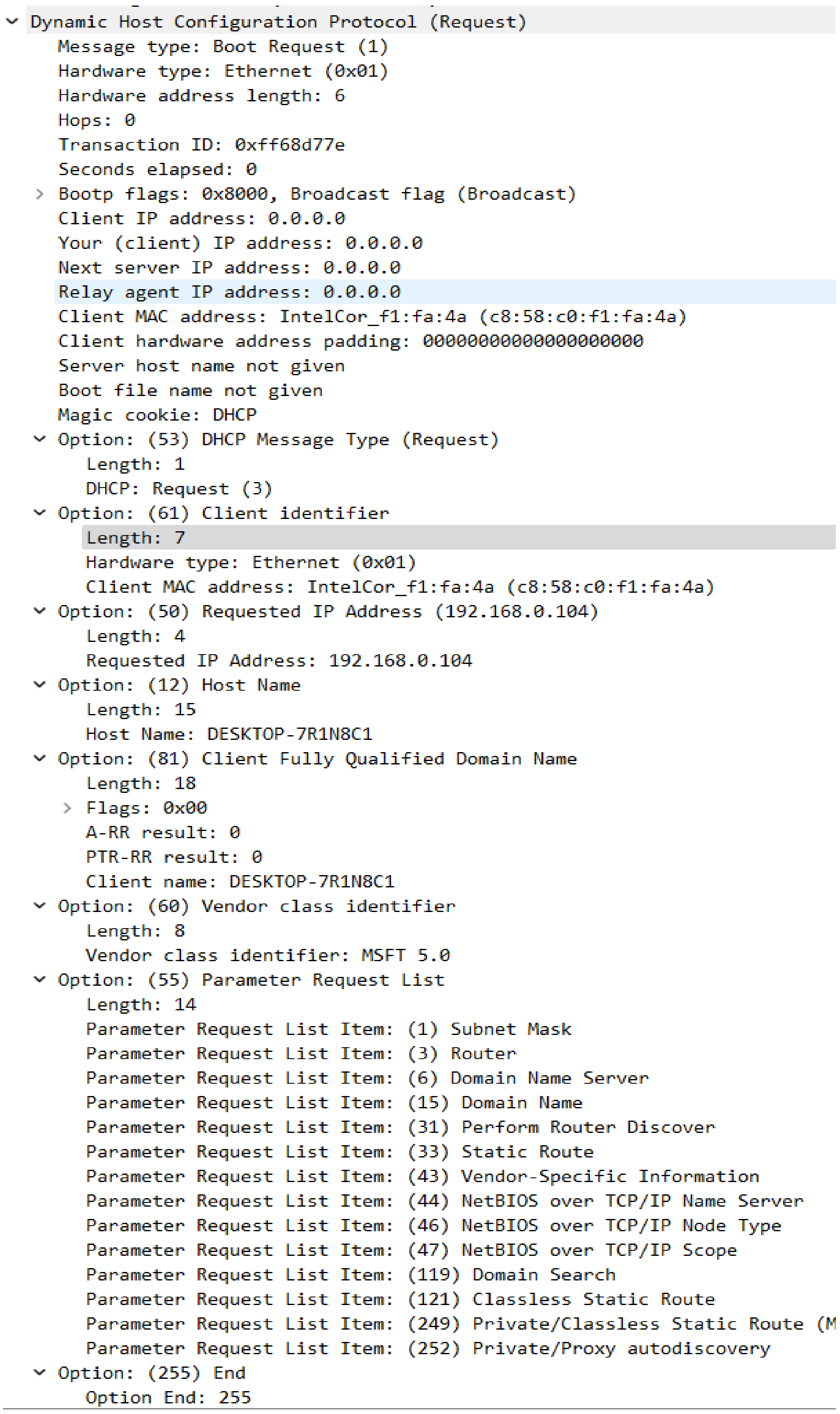


```
 1 0.000000   0.0.0.0       255.255.255.255  DHCP   364 DHCP Request  - Transaction ID 0xff68d77e
35 0.056124   192.168.0.1   255.255.255.255  DHCP   590 DHCP ACK      - Transaction ID 0xff68d77e
```

操作：直接关掉网络然后重新连接，在这个过程中捕获 DHCP

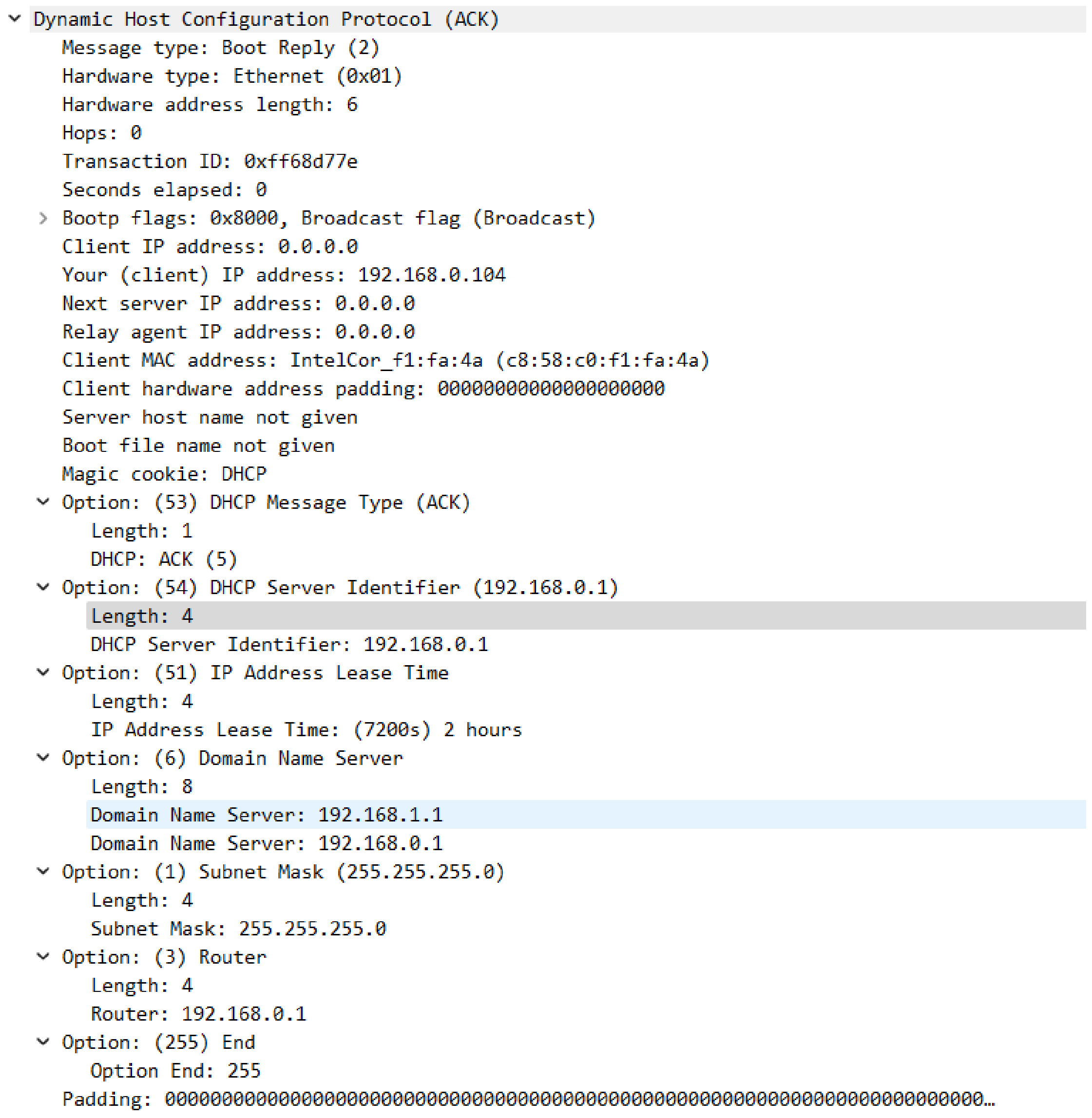
报文。捕获到以上的 request 报文和 ack 报文。

第一条报文（request）如下：



```
⌄ Dynamic Host Configuration Protocol (Request)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0xff68d77e
    Seconds elapsed: 0
  > Bootp flags: 0x8000, Broadcast flag (Broadcast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: IntelCor_f1:fa:4a (c8:58:c0:f1:fa:4a)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ⌄ Option: (53) DHCP Message Type (Request)
      Length: 1
      DHCP: Request (3)
  ⌄ Option: (61) Client identifier
      Length: 7
      Hardware type: Ethernet (0x01)
      Client MAC address: IntelCor_f1:fa:4a (c8:58:c0:f1:fa:4a)
  ⌄ Option: (50) Requested IP Address (192.168.0.104)
      Length: 4
      Requested IP Address: 192.168.0.104
  ⌄ Option: (12) Host Name
      Length: 15
      Host Name: DESKTOP-7R1N8C1
  ⌄ Option: (81) Client Fully Qualified Domain Name
      Length: 18
    > Flags: 0x00
      A-RR result: 0
      PTR-RR result: 0
      Client name: DESKTOP-7R1N8C1
  ⌄ Option: (60) Vendor class identifier
      Length: 8
      Vendor class identifier: MSFT 5.0
  ⌄ Option: (55) Parameter Request List
      Length: 14
      Parameter Request List Item: (1) Subnet Mask
      Parameter Request List Item: (3) Router
      Parameter Request List Item: (6) Domain Name Server
      Parameter Request List Item: (15) Domain Name
      Parameter Request List Item: (31) Perform Router Discover
      Parameter Request List Item: (33) Static Route
      Parameter Request List Item: (43) Vendor-Specific Information
      Parameter Request List Item: (44) NetBIOS over TCP/IP Name Server
      Parameter Request List Item: (46) NetBIOS over TCP/IP Node Type
      Parameter Request List Item: (47) NetBIOS over TCP/IP Scope
      Parameter Request List Item: (119) Domain Search
      Parameter Request List Item: (121) Classless Static Route
      Parameter Request List Item: (249) Private/Classless Static Route (M
      Parameter Request List Item: (252) Private/Proxy autodiscovery
  ⌄ Option: (255) End
      Option End: 255
```

第二条报文（ack）如下：



```
⌄ Dynamic Host Configuration Protocol (ACK)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0xff68d77e
    Seconds elapsed: 0
  › Bootp flags: 0x8000, Broadcast flag (Broadcast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 192.168.0.104
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: IntelCor_f1:fa:4a (c8:58:c0:f1:fa:4a)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ⌄ Option: (53) DHCP Message Type (ACK)
      Length: 1
      DHCP: ACK (5)
  ⌄ Option: (54) DHCP Server Identifier (192.168.0.1)
      Length: 4
      DHCP Server Identifier: 192.168.0.1
  ⌄ Option: (51) IP Address Lease Time
      Length: 4
      IP Address Lease Time: (7200s) 2 hours
  ⌄ Option: (6) Domain Name Server
      Length: 8
      Domain Name Server: 192.168.1.1
      Domain Name Server: 192.168.0.1
  ⌄ Option: (1) Subnet Mask (255.255.255.0)
      Length: 4
      Subnet Mask: 255.255.255.0
  ⌄ Option: (3) Router
      Length: 4
      Router: 192.168.0.1
  ⌄ Option: (255) End
      Option End: 255
    Padding: 000000000000000000000000000000000000000000000000000000000000000000000000…
```

分析如下：

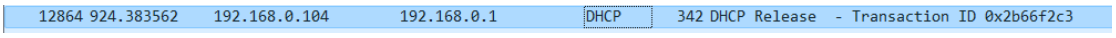
DHCP 在非第一次接入网络的时候只有两个阶段：

一、选择阶段：直接向 DHCP 发送 DHCP REQUEST 而不是 DHCP DISCOVER，发送的 REQUEST 中 Option 50 存储有之前使用的 IP 地址，即向 DHCP 再次请求使用这个 IP 地址。

二、确认阶段：DHCP 服务器获取到这个报文之后，根据 DHCP REQUEST 报文中带有的 MAC 地址来查找是否存在此租约。如

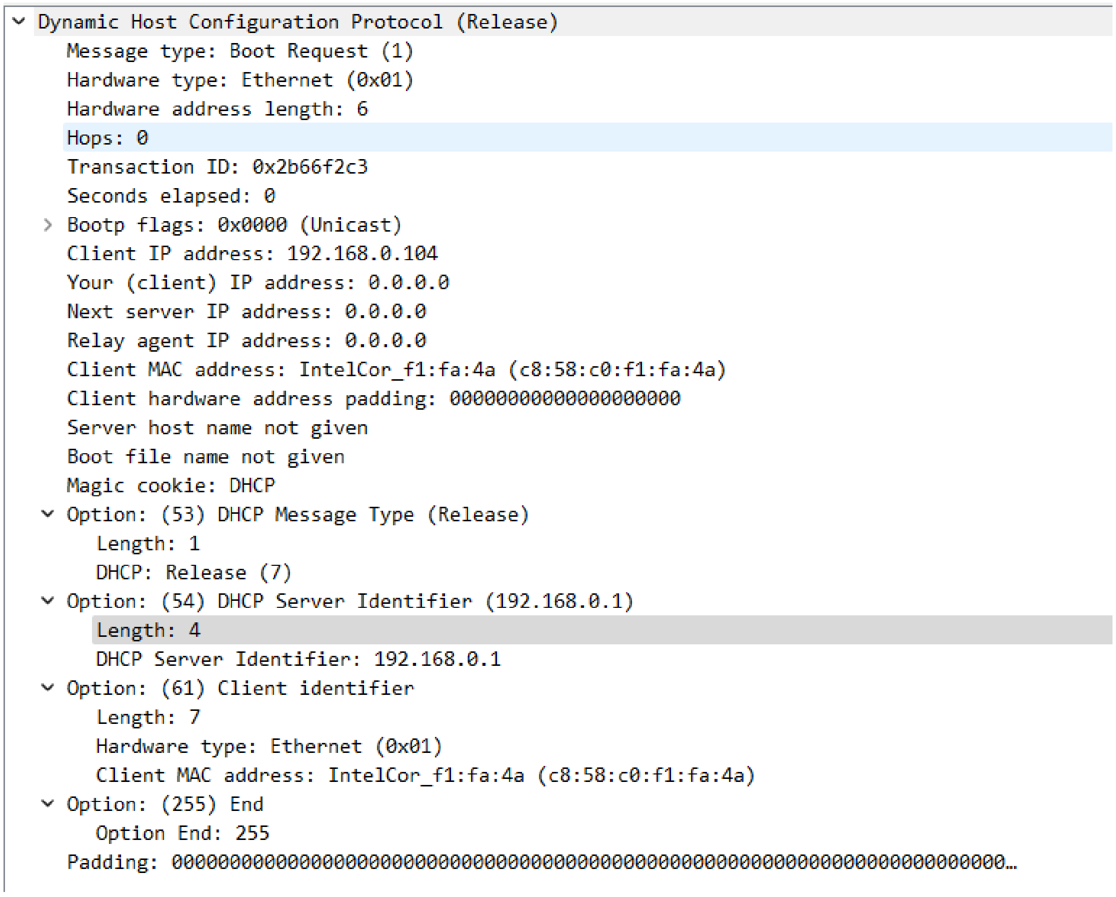
果存在则回复 DHCP ACK，告知请求方可以继续使用这个 IP 地址；如果不存在则保持沉默，一直等到请求方发送 DHCP DISCOVER 请求新的 IP 地址。

在 CMD 界面执行 ipconfig /release，wireshark 捕获到此报文：



```
12864 924.383562   192.168.0.104   192.168.0.1   DHCP   342 DHCP Release  - Transaction ID 0x2b66f2c3
```

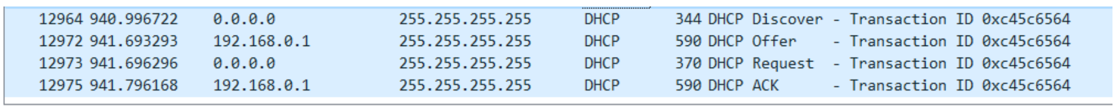
详细信息如下：



```
˅ Dynamic Host Configuration Protocol (Release)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x2b66f2c3
    Seconds elapsed: 0
  > Bootp flags: 0x0000 (Unicast)
    Client IP address: 192.168.0.104
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: IntelCor_f1:fa:4a (c8:58:c0:f1:fa:4a)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ˅ Option: (53) DHCP Message Type (Release)
      Length: 1
      DHCP: Release (7)
  ˅ Option: (54) DHCP Server Identifier (192.168.0.1)
      Length: 4
      DHCP Server Identifier: 192.168.0.1
  ˅ Option: (61) Client identifier
      Length: 7
      Hardware type: Ethernet (0x01)
      Client MAC address: IntelCor_f1:fa:4a (c8:58:c0:f1:fa:4a)
  ˅ Option: (255) End
      Option End: 255
    Padding: 000000000000000000000000000000000000000000000000000000000000000000000000…
```

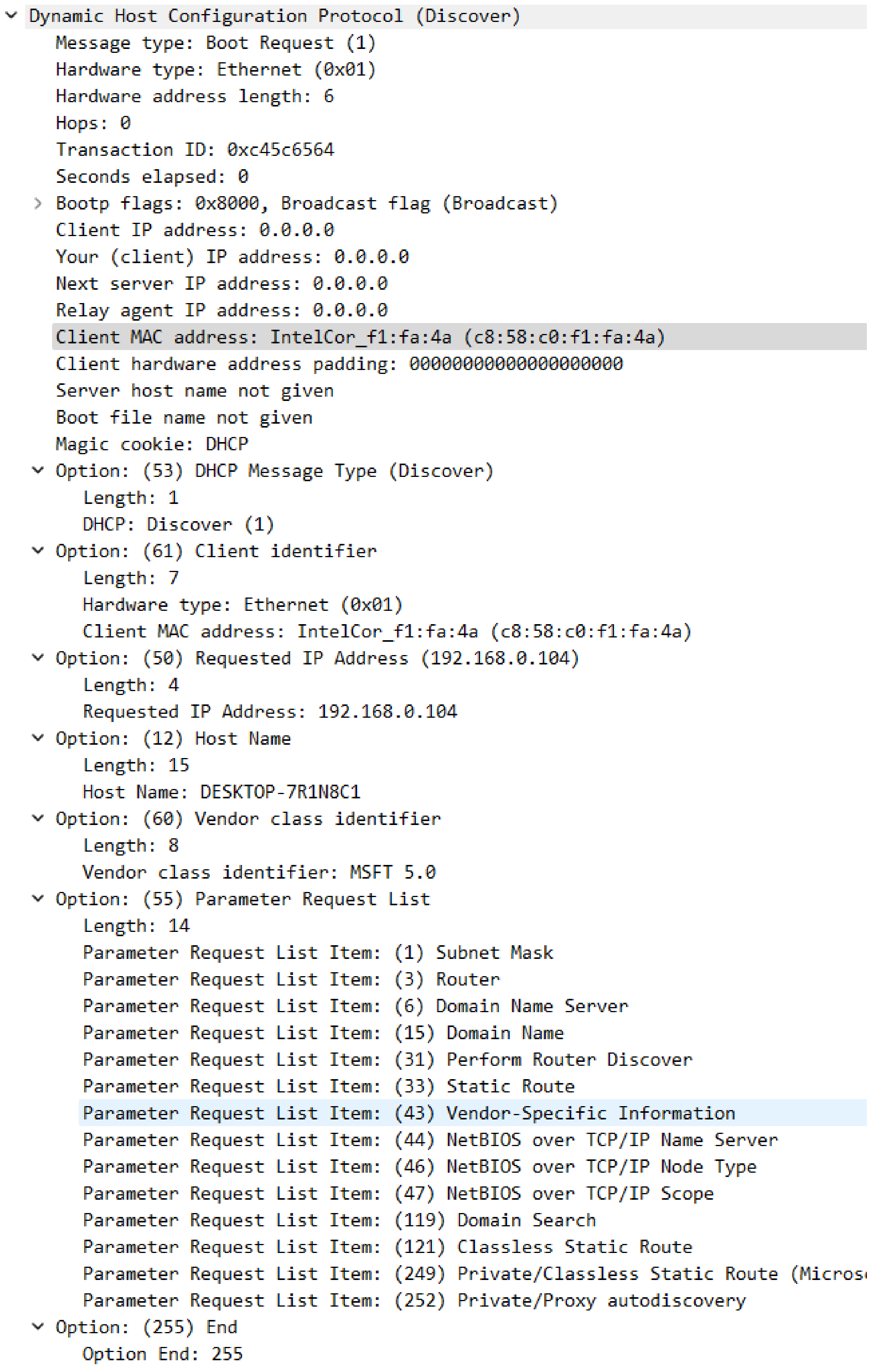
这个报文告诉 DHCP 服务器释放此 MAC 地址正在使用的 IP 地址。

接着执行 ipconfig /renew，捕获到如下报文：



```
12964 940.996722   0.0.0.0       255.255.255.255   DHCP   344 DHCP Discover - Transaction ID 0xc45c6564
12972 941.693293   192.168.0.1   255.255.255.255   DHCP   590 DHCP Offer    - Transaction ID 0xc45c6564
12973 941.696296   0.0.0.0       255.255.255.255   DHCP   370 DHCP Request  - Transaction ID 0xc45c6564
12975 941.796168   192.168.0.1   255.255.255.255   DHCP   590 DHCP ACK      - Transaction ID 0xc45c6564
```

DISCOVER 报文：

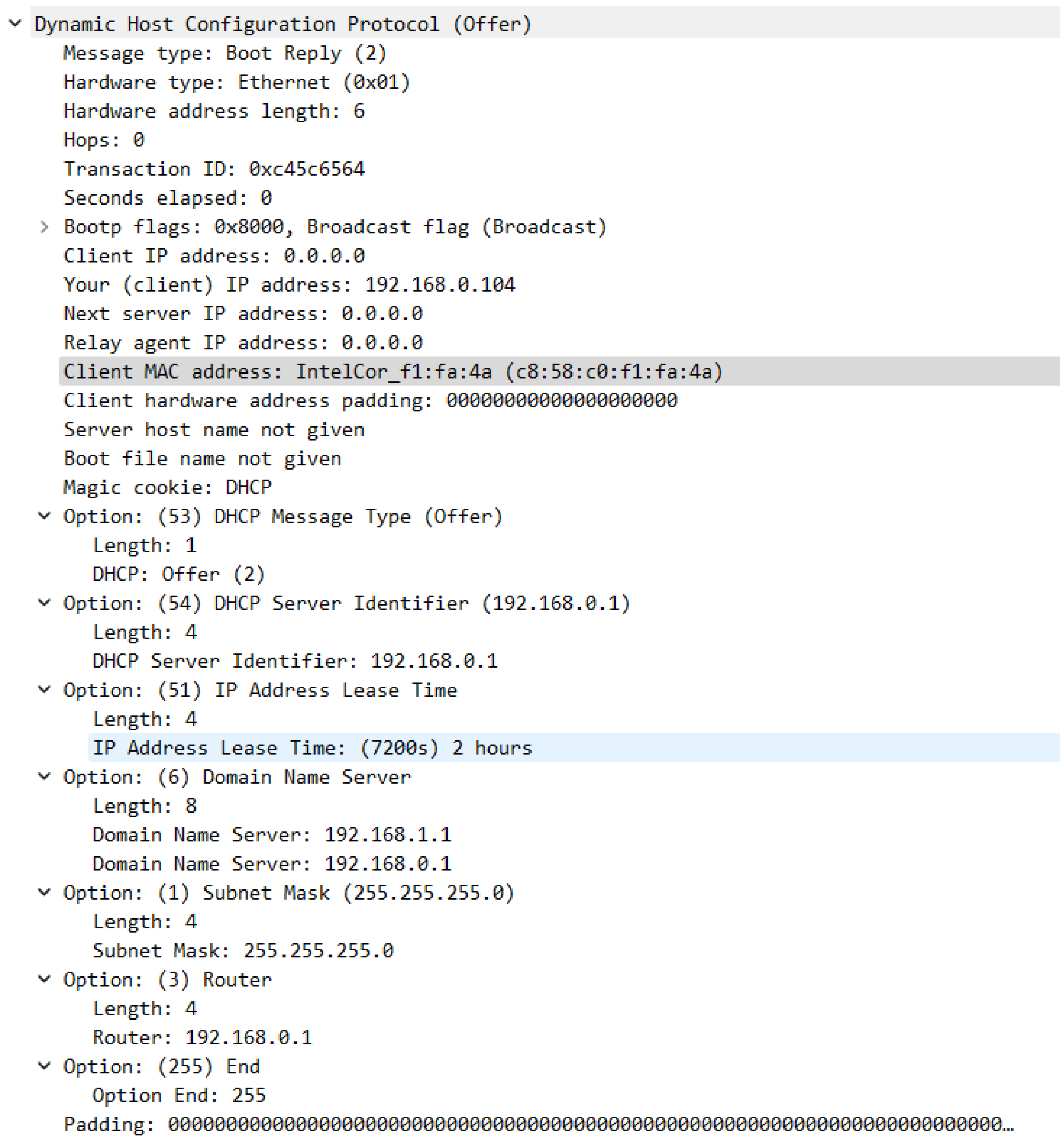


```
⌄ Dynamic Host Configuration Protocol (Discover)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0xc45c6564
    Seconds elapsed: 0
  › Bootp flags: 0x8000, Broadcast flag (Broadcast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: IntelCor_f1:fa:4a (c8:58:c0:f1:fa:4a)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ⌄ Option: (53) DHCP Message Type (Discover)
      Length: 1
      DHCP: Discover (1)
  ⌄ Option: (61) Client identifier
      Length: 7
      Hardware type: Ethernet (0x01)
      Client MAC address: IntelCor_f1:fa:4a (c8:58:c0:f1:fa:4a)
  ⌄ Option: (50) Requested IP Address (192.168.0.104)
      Length: 4
      Requested IP Address: 192.168.0.104
  ⌄ Option: (12) Host Name
      Length: 15
      Host Name: DESKTOP-7R1N8C1
  ⌄ Option: (60) Vendor class identifier
      Length: 8
      Vendor class identifier: MSFT 5.0
  ⌄ Option: (55) Parameter Request List
      Length: 14
      Parameter Request List Item: (1) Subnet Mask
      Parameter Request List Item: (3) Router
      Parameter Request List Item: (6) Domain Name Server
      Parameter Request List Item: (15) Domain Name
      Parameter Request List Item: (31) Perform Router Discover
      Parameter Request List Item: (33) Static Route
      Parameter Request List Item: (43) Vendor-Specific Information
      Parameter Request List Item: (44) NetBIOS over TCP/IP Name Server
      Parameter Request List Item: (46) NetBIOS over TCP/IP Node Type
      Parameter Request List Item: (47) NetBIOS over TCP/IP Scope
      Parameter Request List Item: (119) Domain Search
      Parameter Request List Item: (121) Classless Static Route
      Parameter Request List Item: (249) Private/Classless Static Route (Micros
      Parameter Request List Item: (252) Private/Proxy autodiscovery
  ⌄ Option: (255) End
      Option End: 255
```

分析：

DISCOVER 报文中广播位为 1，表示 DISCOVER 是广播报文 BOARDCAST。报文里还携带了其他信息，比如 Option 53 表示需要请求的参数列表、Option 50 表示要请求的 IP 地址、还自带了请求方的 MAC 地址。

OFFER 报文：



```
⌄ Dynamic Host Configuration Protocol (Offer)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0xc45c6564
    Seconds elapsed: 0
  › Bootp flags: 0x8000, Broadcast flag (Broadcast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 192.168.0.104
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: IntelCor_f1:fa:4a (c8:58:c0:f1:fa:4a)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ⌄ Option: (53) DHCP Message Type (Offer)
      Length: 1
      DHCP: Offer (2)
  ⌄ Option: (54) DHCP Server Identifier (192.168.0.1)
      Length: 4
      DHCP Server Identifier: 192.168.0.1
  ⌄ Option: (51) IP Address Lease Time
      Length: 4
      IP Address Lease Time: (7200s) 2 hours
  ⌄ Option: (6) Domain Name Server
      Length: 8
      Domain Name Server: 192.168.1.1
      Domain Name Server: 192.168.0.1
  ⌄ Option: (1) Subnet Mask (255.255.255.0)
      Length: 4
      Subnet Mask: 255.255.255.0
  ⌄ Option: (3) Router
      Length: 4
      Router: 192.168.0.1
  ⌄ Option: (255) End
      Option End: 255
    Padding: 000000000000000000000000000000000000000000000000000000000000000000000…
```

分析：

与 DHCP 客户端位于同一网段的 DHCP 服务器收到 DISCOVER

报文之后，选择跟接收 DHCP DISCOVER 报文接口的 IP 地址处于同一网段的地址池，并且从中选择一个可用的 IP 地址，然后通过 DHCP OFFER 报文发送给 DHCP 客户端。

DHCP 地址池中会指定 IP 地址的租期，如果 DHCP 客户端发送的 DHCP DISCOVER 中含有期望租期信息，则会将空余的 IP 地址的租期与期望租期比较，最后发送一个租期时间较短的 IP 地址给请求方。OFFER 报文中含有 DISCOVER 中需要请求参数的信息，Option 3 给出路由的 IP 地址，Option 4 给出了子网掩码，Option 6 给出了 DNS 服务器，Option 51 给出了租期时间。

DHCP 服务器在地址池中给 DHCP 客户端配置 IP 地址的优先级如下:

一、 DHCP 服务器中已配置的和客户端 MAC 地址静态绑定的 IP 地址

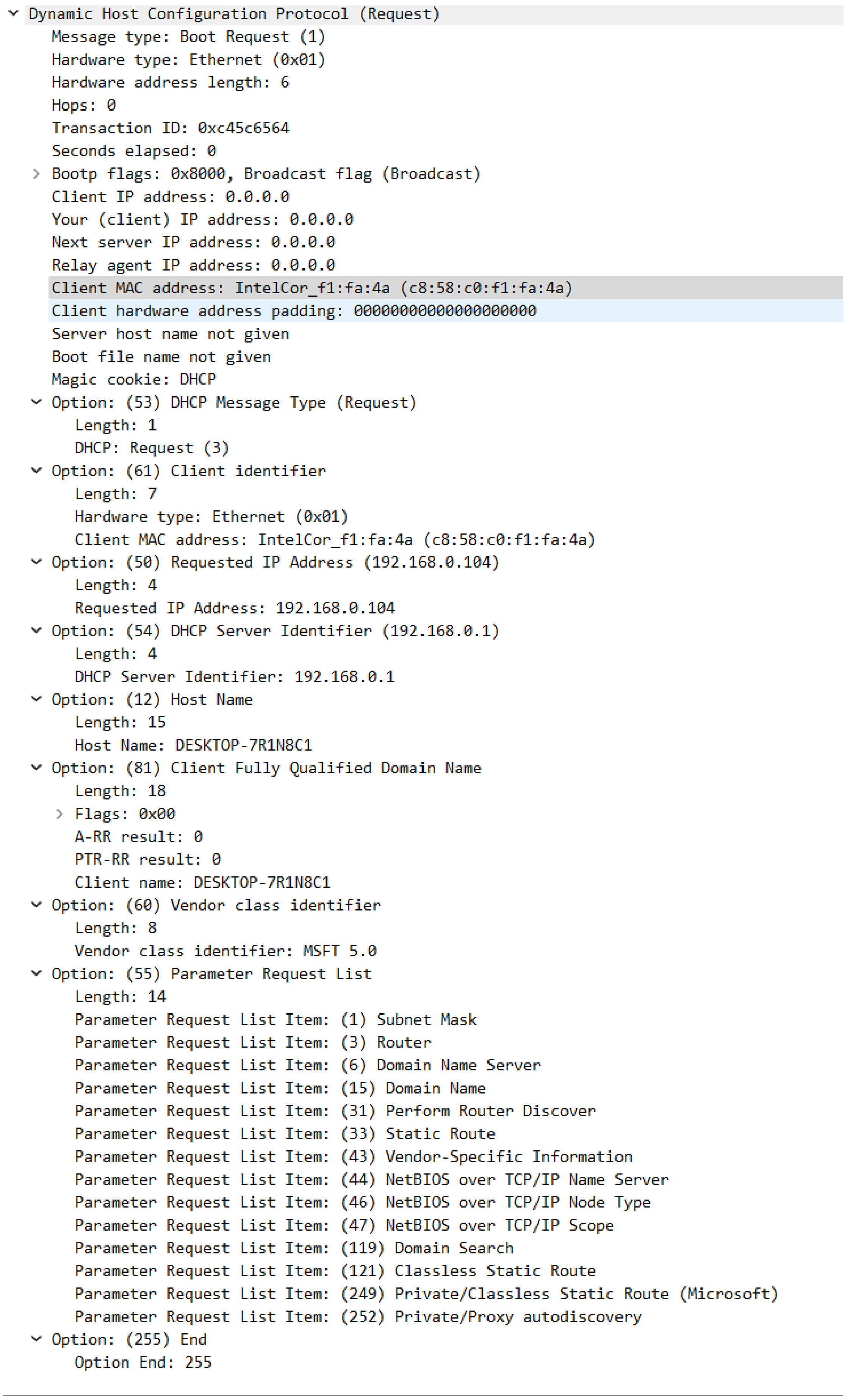
二、 DHCP DISCOVER 中请求的 IP 地址

三、 地址池中处于“expired”状态的 IP 地址，即之前分配给客户端使用但是租约过期的 IP 地址。

四、 地址池中处于“idle”状态的 IP 地址

五、 回收处于 conflict 状态和 expired 状态的 IP 地址。回收之后可以分配则分配，否则等到客户端发送下一个 DHCP DISCOVER 请求 IP 地址。

REQUEST 报文:

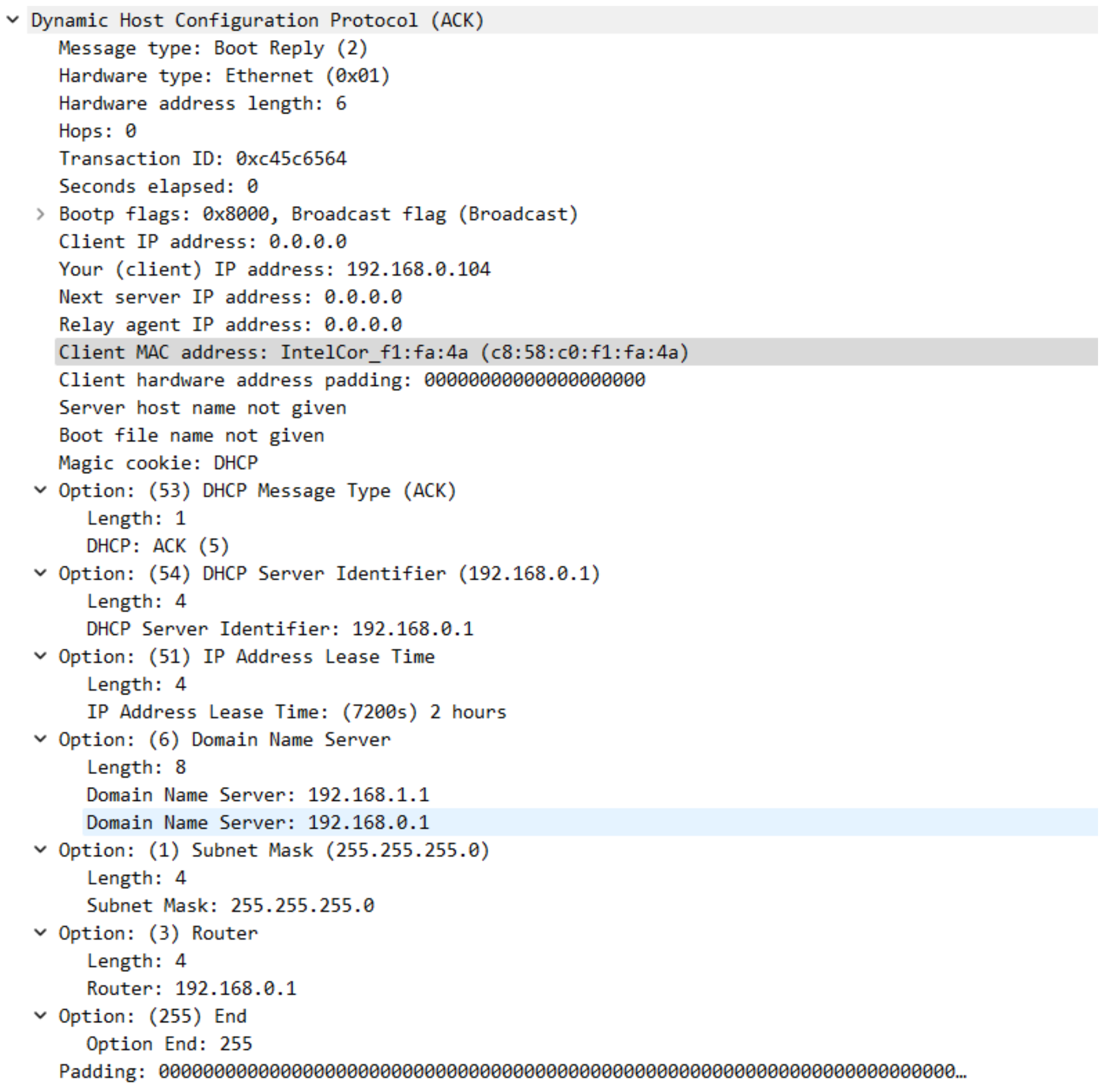


```
v Dynamic Host Configuration Protocol (Request)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0xc45c6564
    Seconds elapsed: 0
  > Bootp flags: 0x8000, Broadcast flag (Broadcast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: IntelCor_f1:fa:4a (c8:58:c0:f1:fa:4a)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  v Option: (53) DHCP Message Type (Request)
      Length: 1
      DHCP: Request (3)
  v Option: (61) Client identifier
      Length: 7
      Hardware type: Ethernet (0x01)
      Client MAC address: IntelCor_f1:fa:4a (c8:58:c0:f1:fa:4a)
  v Option: (50) Requested IP Address (192.168.0.104)
      Length: 4
      Requested IP Address: 192.168.0.104
  v Option: (54) DHCP Server Identifier (192.168.0.1)
      Length: 4
      DHCP Server Identifier: 192.168.0.1
  v Option: (12) Host Name
      Length: 15
      Host Name: DESKTOP-7R1N8C1
  v Option: (81) Client Fully Qualified Domain Name
      Length: 18
    > Flags: 0x00
      A-RR result: 0
      PTR-RR result: 0
      Client name: DESKTOP-7R1N8C1
  v Option: (60) Vendor class identifier
      Length: 8
      Vendor class identifier: MSFT 5.0
  v Option: (55) Parameter Request List
      Length: 14
      Parameter Request List Item: (1) Subnet Mask
      Parameter Request List Item: (3) Router
      Parameter Request List Item: (6) Domain Name Server
      Parameter Request List Item: (15) Domain Name
      Parameter Request List Item: (31) Perform Router Discover
      Parameter Request List Item: (33) Static Route
      Parameter Request List Item: (43) Vendor-Specific Information
      Parameter Request List Item: (44) NetBIOS over TCP/IP Name Server
      Parameter Request List Item: (46) NetBIOS over TCP/IP Node Type
      Parameter Request List Item: (47) NetBIOS over TCP/IP Scope
      Parameter Request List Item: (119) Domain Search
      Parameter Request List Item: (121) Classless Static Route
      Parameter Request List Item: (249) Private/Classless Static Route (Microsoft)
      Parameter Request List Item: (252) Private/Proxy autodiscovery
  v Option: (255) End
      Option End: 255
```

分析：

如果有多个 DHCP 服务器向客户端发送 DHCP OFFER 报文，一般客户端只选择第一个向它发送的服务器配置的 IP 地址。REQUSET 报文为广播形式的，会发送到多个 DHCP 服务器中，REQUSET 报文里面带有选择的 DHCP 服务器 IP 地址（Option 54)，只有对应的 DHCP 服务器会对这个报文做相应。REQUSET 报文中的 Option 50 填充的是 OFFER 报文中的 IP 地址

ACK 报文：



```
v Dynamic Host Configuration Protocol (ACK)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0xc45c6564
    Seconds elapsed: 0
  > Bootp flags: 0x8000, Broadcast flag (Broadcast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 192.168.0.104
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: IntelCor_f1:fa:4a (c8:58:c0:f1:fa:4a)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  v Option: (53) DHCP Message Type (ACK)
      Length: 1
      DHCP: ACK (5)
  v Option: (54) DHCP Server Identifier (192.168.0.1)
      Length: 4
      DHCP Server Identifier: 192.168.0.1
  v Option: (51) IP Address Lease Time
      Length: 4
      IP Address Lease Time: (7200s) 2 hours
  v Option: (6) Domain Name Server
      Length: 8
      Domain Name Server: 192.168.1.1
      Domain Name Server: 192.168.0.1
  v Option: (1) Subnet Mask (255.255.255.0)
      Length: 4
      Subnet Mask: 255.255.255.0
  v Option: (3) Router
      Length: 4
      Router: 192.168.0.1
  v Option: (255) End
      Option End: 255
    Padding: 000000000000000000000000000000000000000000000000000000000000000000000000…
```

分析：

DHCP 服务器在接收到客户端发出的 REQUEST 报文之后向客户端发送 ACK 报文表示客户端可以使用 IP 地址。

客户端在收到了 ACK 之后会广播免费 ARP 报文探测本网段是否有其他终端在使用服务器分配给它的 IP 地址。如果过了一段时间没有人回应则表示 IP 地址没有冲突；如果有回应则客户端会向服务器发送 DHCP DECLINE 报文表示 IP 地址冲突，并重新请求 IP 地址，同时服务器会把这个 IP 地址列为冲突 IP 地址。当服务器没有空闲 IP 地址可以分配的时候，会选择冲突 IP 地址分配，服务器尽量让分配出去的 IP 地址不冲突。

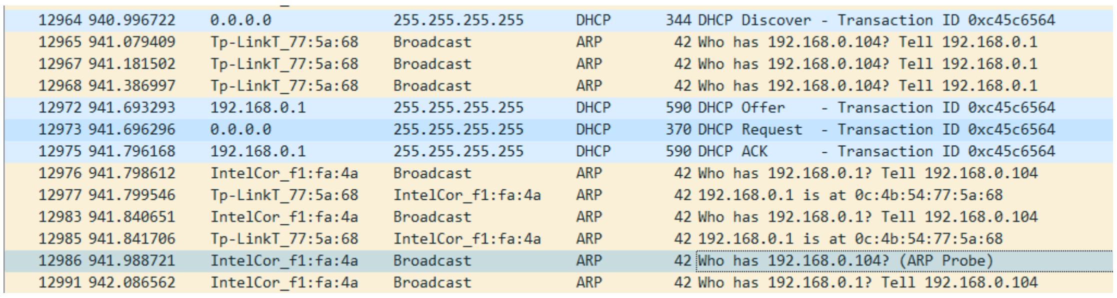
如果服务器收到 DHCP DISCOVER 后无法分配 IP 地址，则会回复 DHCP NAK 给客户端要求客户端重新发送 DHCP DISCOVER 请求 IP 地址。

b) **捕获 ARP 分组**：

ARP（Address Resolution Protocol）称为地址解析协议，用来将 IP 地址解析为 MAC 地址。

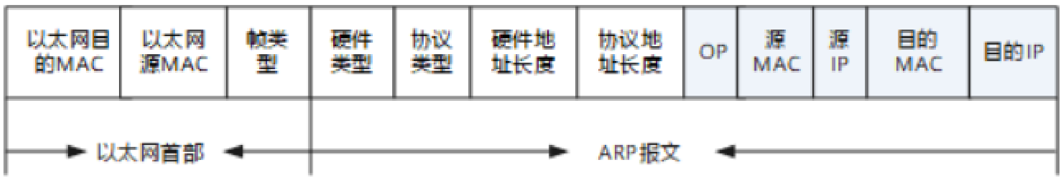
如上述所说客户端在收到 DHCP ACK 之后会广播免费 ARP 检测是否有人使用了分配给自己的 IP 地址。

Wireshark 捕获的 ARP 报文如下：



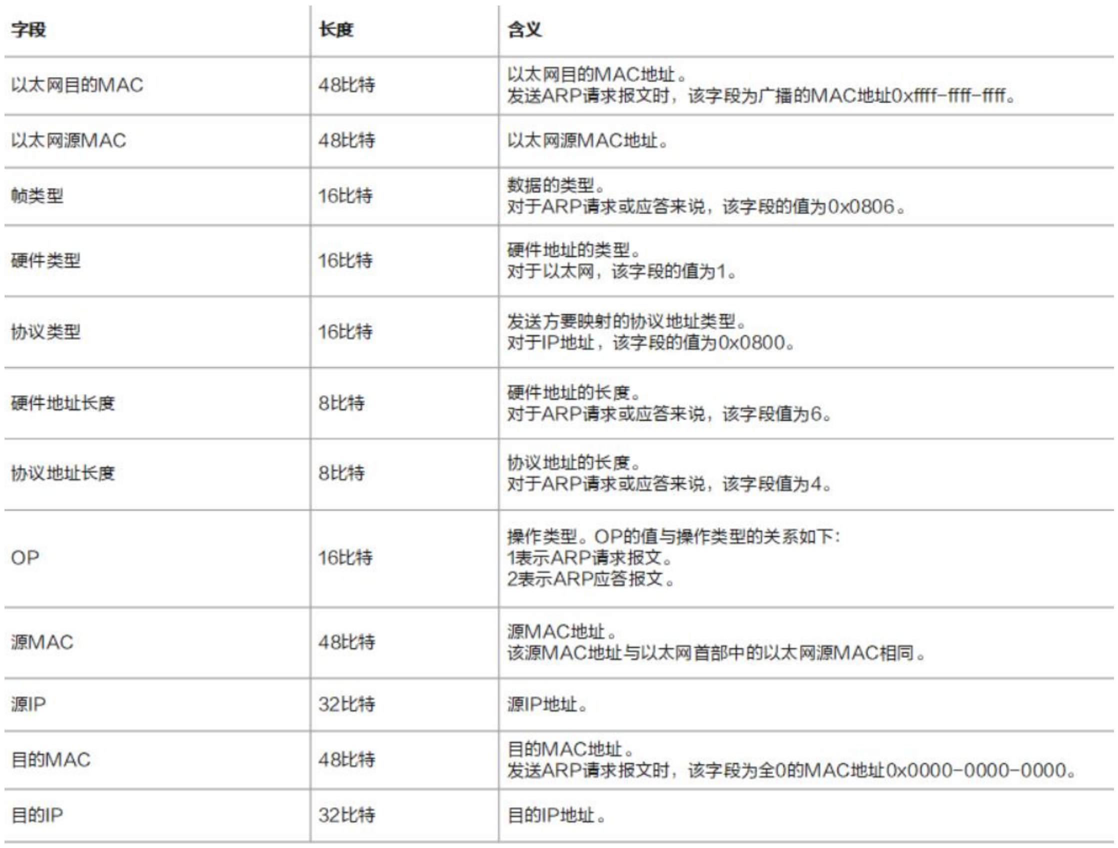
```
12964 940.996722  0.0.0.0            255.255.255.255   DHCP   344 DHCP Discover - Transaction ID 0xc45c6564
12965 941.079409  Tp-LinkT_77:5a:68  Broadcast         ARP     42 Who has 192.168.0.104? Tell 192.168.0.1
12967 941.181502  Tp-LinkT_77:5a:68  Broadcast         ARP     42 Who has 192.168.0.104? Tell 192.168.0.1
12968 941.386997  Tp-LinkT_77:5a:68  Broadcast         ARP     42 Who has 192.168.0.104? Tell 192.168.0.1
12972 941.693293  192.168.0.1        255.255.255.255   DHCP   590 DHCP Offer    - Transaction ID 0xc45c6564
12973 941.696296  0.0.0.0            255.255.255.255   DHCP   370 DHCP Request  - Transaction ID 0xc45c6564
12975 941.796168  192.168.0.1        255.255.255.255   DHCP   590 DHCP ACK      - Transaction ID 0xc45c6564
12976 941.798612  IntelCor_f1:fa:4a  Broadcast         ARP     42 Who has 192.168.0.1? Tell 192.168.0.104
12977 941.799546  Tp-LinkT_77:5a:68  IntelCor_f1:fa:4a ARP     42 192.168.0.1 is at 0c:4b:54:77:5a:68
12983 941.840651  IntelCor_f1:fa:4a  Broadcast         ARP     42 Who has 192.168.0.1? Tell 192.168.0.104
12985 941.841706  Tp-LinkT_77:5a:68  IntelCor_f1:fa:4a ARP     42 192.168.0.1 is at 0c:4b:54:77:5a:68
12986 941.988721  IntelCor_f1:fa:4a  Broadcast         ARP     42 Who has 192.168.0.104? (ARP Probe)
12991 942.086562  IntelCor_f1:fa:4a  Broadcast         ARP     42 Who has 192.168.0.1? Tell 192.168.0.104
```

ARP 报文格式如下：



以太网首部占 14 字节，ARP 报文占 28 个字节，一共占 42 个字节。

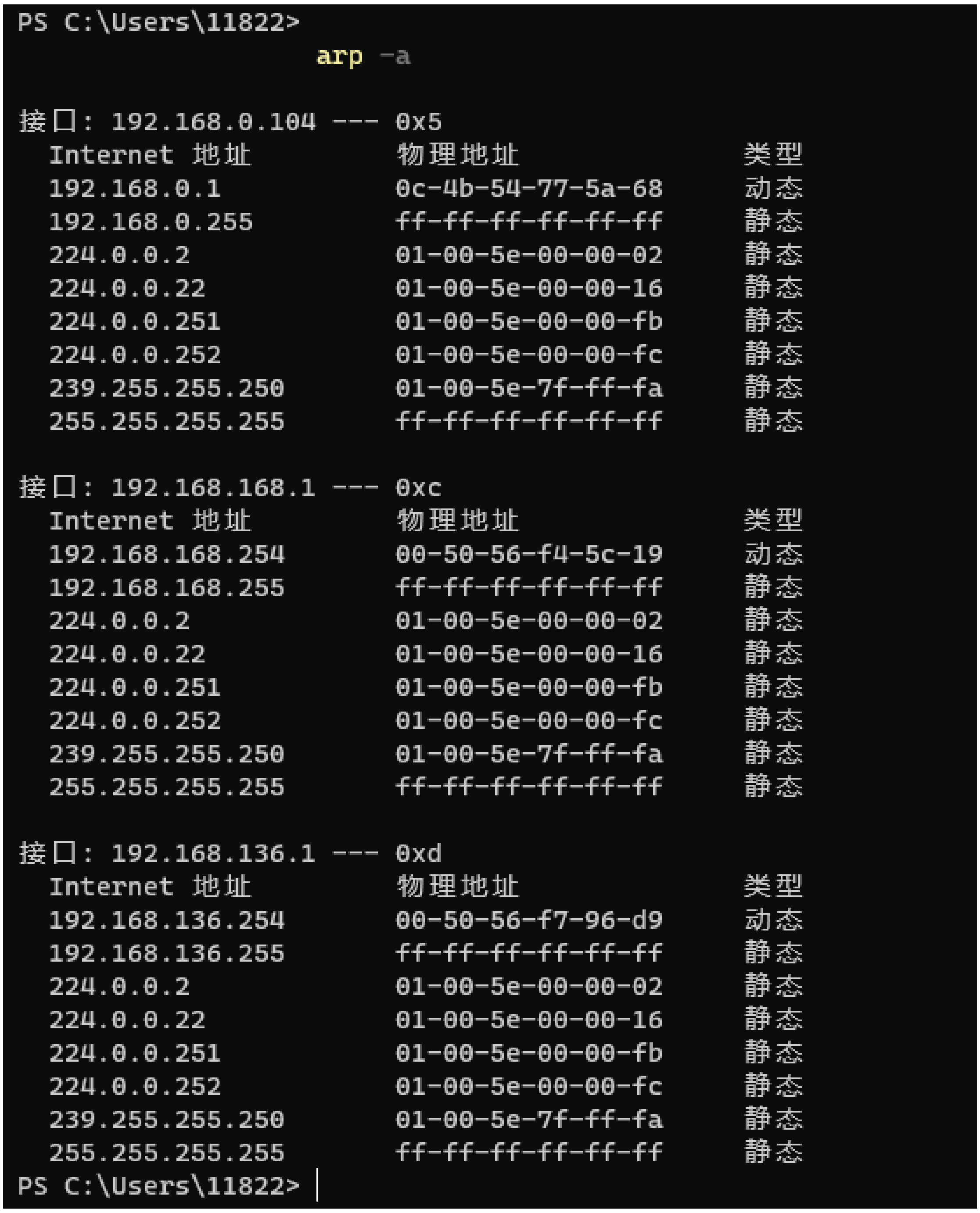
各个字段的含义如下图所示：



字段	长度	含义
以太网目的MAC	48比特	以太网目的MAC地址。 发送ARP请求报文时，该字段为广播的MAC地址0xffff-ffff-ffff。
以太网源MAC	48比特	以太网源MAC地址。
帧类型	16比特	数据的类型。 对于ARP请求或应答来说，该字段的值为0x0806。
硬件类型	16比特	硬件地址的类型。 对于以太网，该字段的值为1。
协议类型	16比特	发送方要映射的协议地址类型。 对于IP地址，该字段的值为0x0800。
硬件地址长度	8比特	硬件地址的长度。 对于ARP请求或应答来说，该字段值为6。
协议地址长度	8比特	协议地址的长度。 对于ARP请求或应答来说，该字段值为4。
OP	16比特	操作类型。OP的值与操作类型的关系如下： 1表示ARP请求报文。 2表示ARP应答报文。
源MAC	48比特	源MAC地址。 该源MAC地址与以太网首部中的以太网源MAC相同。
源IP	32比特	源IP地址。
目的MAC	48比特	目的MAC地址。 发送ARP请求报文时，该字段为全0的MAC地址0x0000-0000-0000。
目的IP	32比特	目的IP地址。

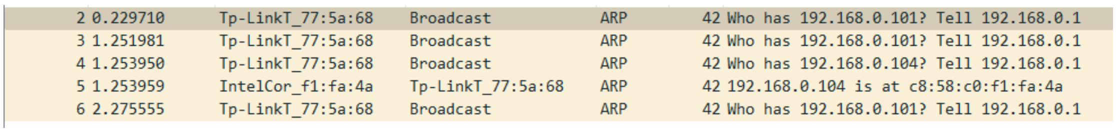
源 IP 和源 MAC 存的是请求方的 IP 地址和 MAC 地址，目的 MAC 由目的 IP 通过 IP-MAC 映射表获取。ARP 请求报文里目的 MAC 地址全 0。免费 ARP 为设备主动使用自己的 IP 地址当作目的 IP 的请求 ARP 报文。

执行 arp -a 之后：



```
PS C:\Users\11822>
                    arp -a

接口: 192.168.0.104 --- 0x5
  Internet 地址         物理地址              类型
  192.168.0.1           0c-4b-54-77-5a-68     动态
  192.168.0.255         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: 192.168.168.1 --- 0xc
  Internet 地址         物理地址              类型
  192.168.168.254       00-50-56-f4-5c-19     动态
  192.168.168.255       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: 192.168.136.1 --- 0xd
  Internet 地址         物理地址              类型
  192.168.136.254       00-50-56-f7-96-d9     动态
  192.168.136.255       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
PS C:\Users\11822>
```

这个过程中 Wireshark 捕获到的 ARP 报文：

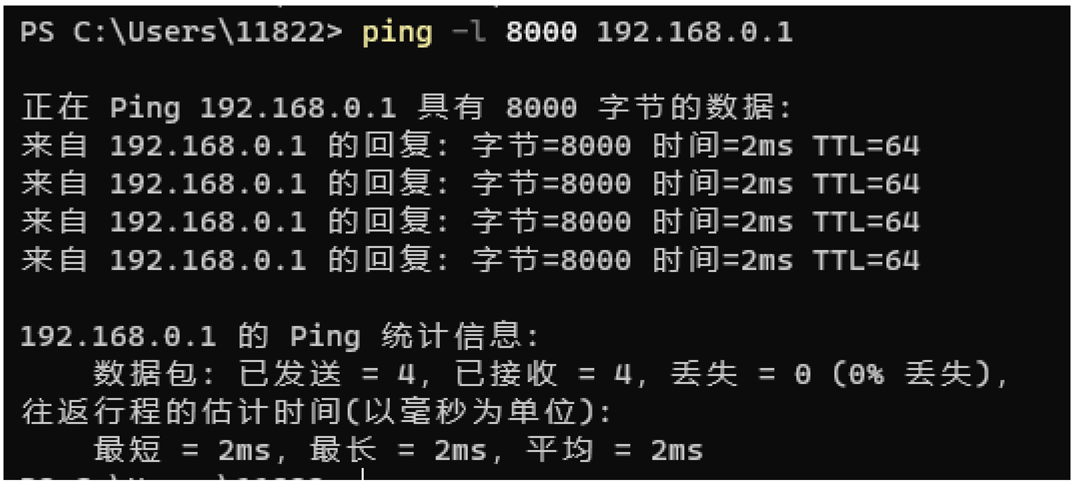


```
2 0.229710    Tp-LinkT_77:5a:68   Broadcast           ARP   42 Who has 192.168.0.101? Tell 192.168.0.1
3 1.251981    Tp-LinkT_77:5a:68   Broadcast           ARP   42 Who has 192.168.0.101? Tell 192.168.0.1
4 1.253950    Tp-LinkT_77:5a:68   Broadcast           ARP   42 Who has 192.168.0.104? Tell 192.168.0.1
5 1.253959    IntelCor_f1:fa:4a   Tp-LinkT_77:5a:68   ARP   42 192.168.0.104 is at c8:58:c0:f1:fa:4a
6 2.275555    Tp-LinkT_77:5a:68   Broadcast           ARP   42 Who has 192.168.0.101? Tell 192.168.0.1
```

输入 arp -a 的时候计算机在网络上广播 ARP 报文来获取 MAC 地址打印在命令行窗口里。

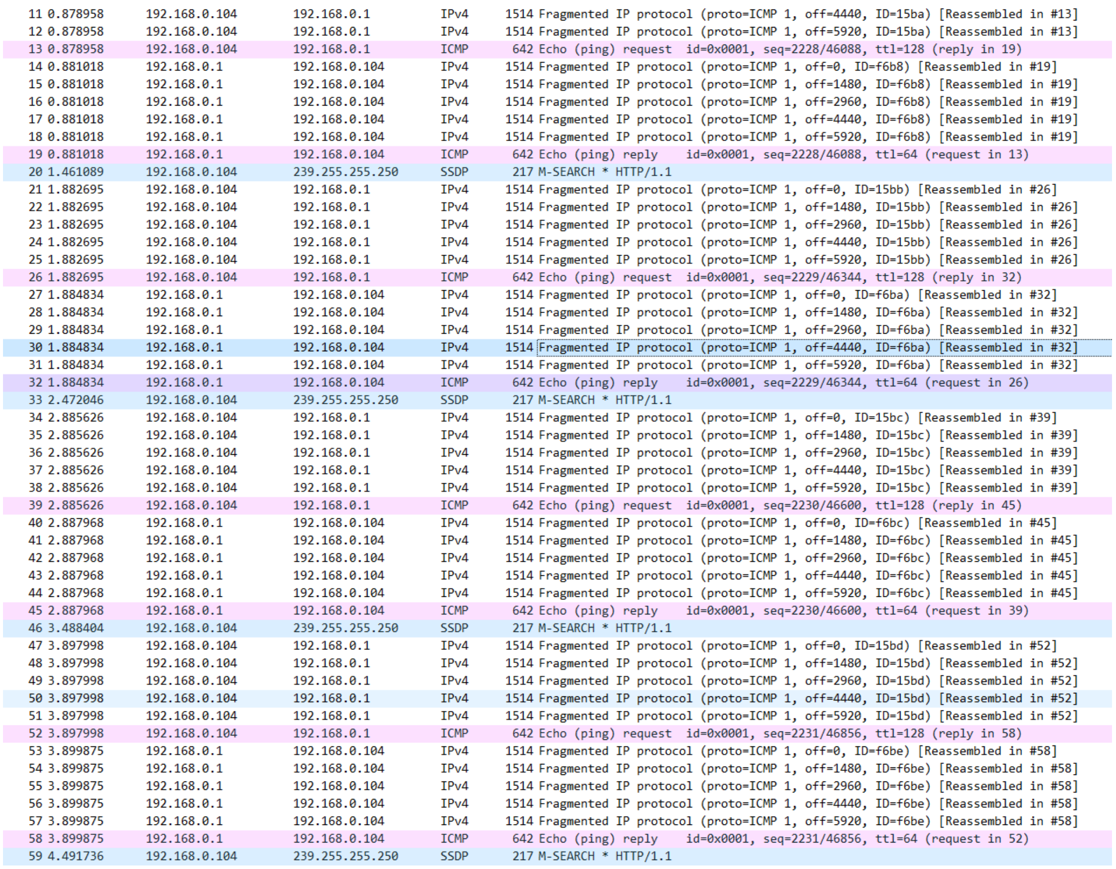
c) IP 数据分组：

在 CMD 窗口输入 ping -l 8000 192.168.0.1 显示如下信息：



```
PS C:\Users\11822> ping -l 8000 192.168.0.1

正在 Ping 192.168.0.1 具有 8000 字节的数据:
来自 192.168.0.1 的回复: 字节=8000 时间=2ms TTL=64
来自 192.168.0.1 的回复: 字节=8000 时间=2ms TTL=64
来自 192.168.0.1 的回复: 字节=8000 时间=2ms TTL=64
来自 192.168.0.1 的回复: 字节=8000 时间=2ms TTL=64

192.168.0.1 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 = 2ms, 最长 = 2ms, 平均 = 2ms
```

捕获的 IPv4 分组如下：



```
11 0.878958   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=15ba) [Reassembled in #13]
12 0.878958   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=15ba) [Reassembled in #13]
13 0.878958   192.168.0.104   192.168.0.1       ICMP    642 Echo (ping) request  id=0x0001, seq=2228/46088, ttl=128 (reply in 19)
14 0.881018   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=f6b8) [Reassembled in #19]
15 0.881018   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=f6b8) [Reassembled in #19]
16 0.881018   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=f6b8) [Reassembled in #19]
17 0.881018   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=f6b8) [Reassembled in #19]
18 0.881018   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=f6b8) [Reassembled in #19]
19 0.881018   192.168.0.1     192.168.0.104     ICMP    642 Echo (ping) reply    id=0x0001, seq=2228/46088, ttl=64 (request in 13)
20 1.461089   192.168.0.104   239.255.255.250   SSDP    217 M-SEARCH * HTTP/1.1
21 1.882695   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=15bb) [Reassembled in #26]
22 1.882695   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=15bb) [Reassembled in #26]
23 1.882695   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=15bb) [Reassembled in #26]
24 1.882695   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=15bb) [Reassembled in #26]
25 1.882695   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=15bb) [Reassembled in #26]
26 1.882695   192.168.0.104   192.168.0.1       ICMP    642 Echo (ping) request  id=0x0001, seq=2229/46344, ttl=128 (reply in 32)
27 1.884834   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=f6ba) [Reassembled in #32]
28 1.884834   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=f6ba) [Reassembled in #32]
29 1.884834   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=f6ba) [Reassembled in #32]
30 1.884834   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=f6ba) [Reassembled in #32]
31 1.884834   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=f6ba) [Reassembled in #32]
32 1.884834   192.168.0.1     192.168.0.104     ICMP    642 Echo (ping) reply    id=0x0001, seq=2229/46344, ttl=64 (request in 26)
33 2.472046   192.168.0.104   239.255.255.250   SSDP    217 M-SEARCH * HTTP/1.1
34 2.885626   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=15bc) [Reassembled in #39]
35 2.885626   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=15bc) [Reassembled in #39]
36 2.885626   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=15bc) [Reassembled in #39]
37 2.885626   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=15bc) [Reassembled in #39]
38 2.885626   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=15bc) [Reassembled in #39]
39 2.885626   192.168.0.104   192.168.0.1       ICMP    642 Echo (ping) request  id=0x0001, seq=2230/46600, ttl=128 (reply in 45)
40 2.887968   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=f6bc) [Reassembled in #45]
41 2.887968   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=f6bc) [Reassembled in #45]
42 2.887968   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=f6bc) [Reassembled in #45]
43 2.887968   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=f6bc) [Reassembled in #45]
44 2.887968   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=f6bc) [Reassembled in #45]
45 2.887968   192.168.0.1     192.168.0.104     ICMP    642 Echo (ping) reply    id=0x0001, seq=2230/46600, ttl=64 (request in 39)
46 3.488404   192.168.0.104   239.255.255.250   SSDP    217 M-SEARCH * HTTP/1.1
47 3.897998   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=15bd) [Reassembled in #52]
48 3.897998   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=15bd) [Reassembled in #52]
49 3.897998   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=15bd) [Reassembled in #52]
50 3.897998   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=15bd) [Reassembled in #52]
51 3.897998   192.168.0.104   192.168.0.1       IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=15bd) [Reassembled in #52]
52 3.897998   192.168.0.104   192.168.0.1       ICMP    642 Echo (ping) request  id=0x0001, seq=2231/46856, ttl=128 (reply in 58)
53 3.899875   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=f6be) [Reassembled in #58]
54 3.899875   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=f6be) [Reassembled in #58]
55 3.899875   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=f6be) [Reassembled in #58]
56 3.899875   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=f6be) [Reassembled in #58]
57 3.899875   192.168.0.1     192.168.0.104     IPv4   1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=f6be) [Reassembled in #58]
58 3.899875   192.168.0.1     192.168.0.104     ICMP    642 Echo (ping) reply    id=0x0001, seq=2231/46856, ttl=64 (request in 52)
59 4.491736   192.168.0.104   239.255.255.250   SSDP    217 M-SEARCH * HTTP/1.1
```

每个 IPv4 报文的数据段都是 1480 字节，对 8000 字节的数据包做了分片，每一包分成五片，每一个分片都是 1480 字节，每次发送整个数据包之后就做一次 ICMP 检验数据的正确性。一共发了四个 8000 字节的数据包，做了四次 ICMP

的校验，每一次都对应 ICMP REQUEST 和 ICMP REPLY

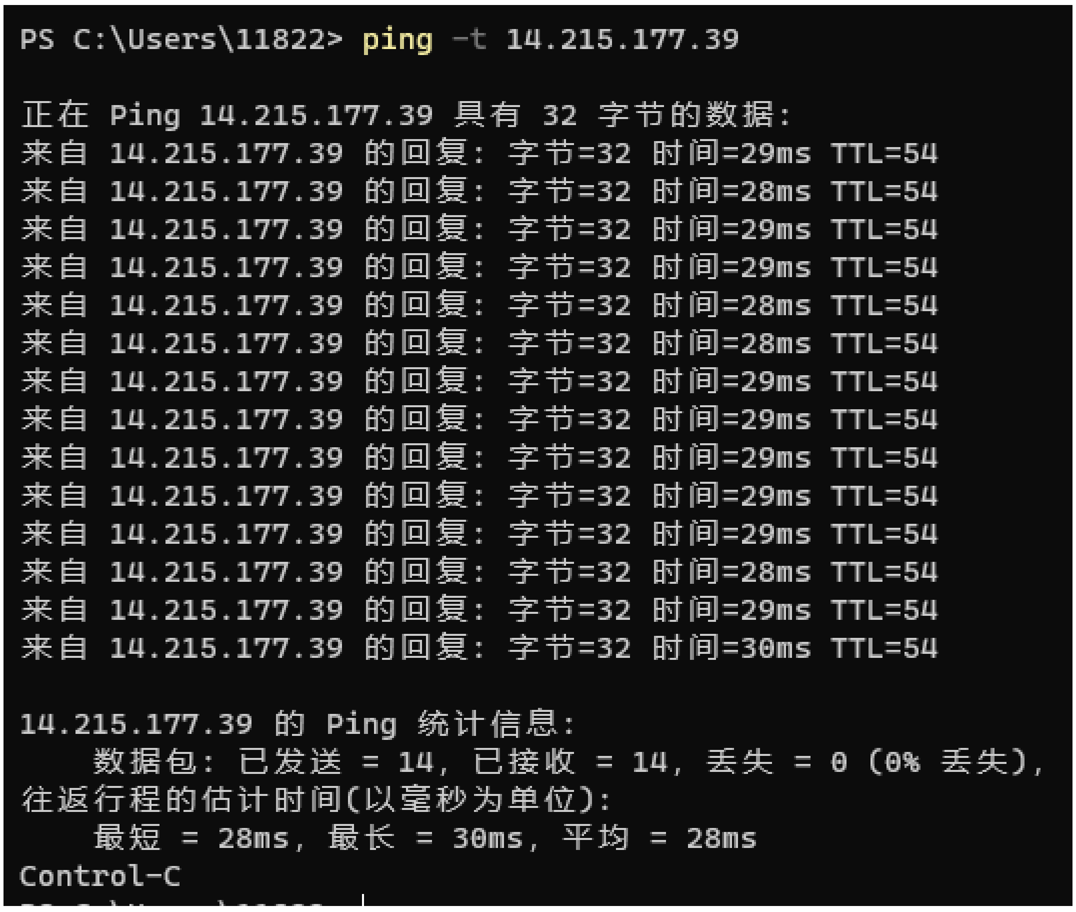
d) ICMP 分组：

ICMP（Internet Control Message Protocol）叫因特网控制报文协议，用来做差错控制的协议。

为什么要有 ICMP?

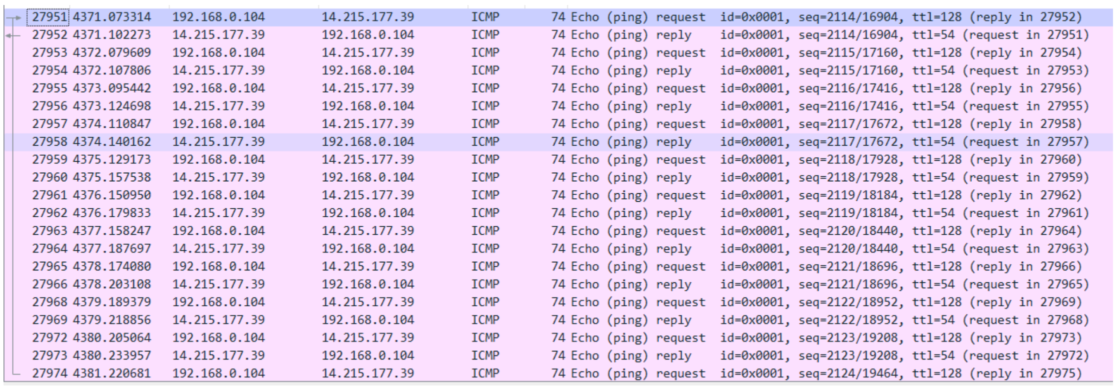
因为 IP 在发送的时候尽力保证的是将数据发送到目的地址，对于发送的数据正确与否并没有验证。ICMP 协议可以检查传输的数据是否出现差错，如果出现错误主机就会向上层协议发出错误信息并提供有关异常信息的报告，让上层可以进行流量控制、差错控制等等，保证服务质量。

执行 ping 命令如下：



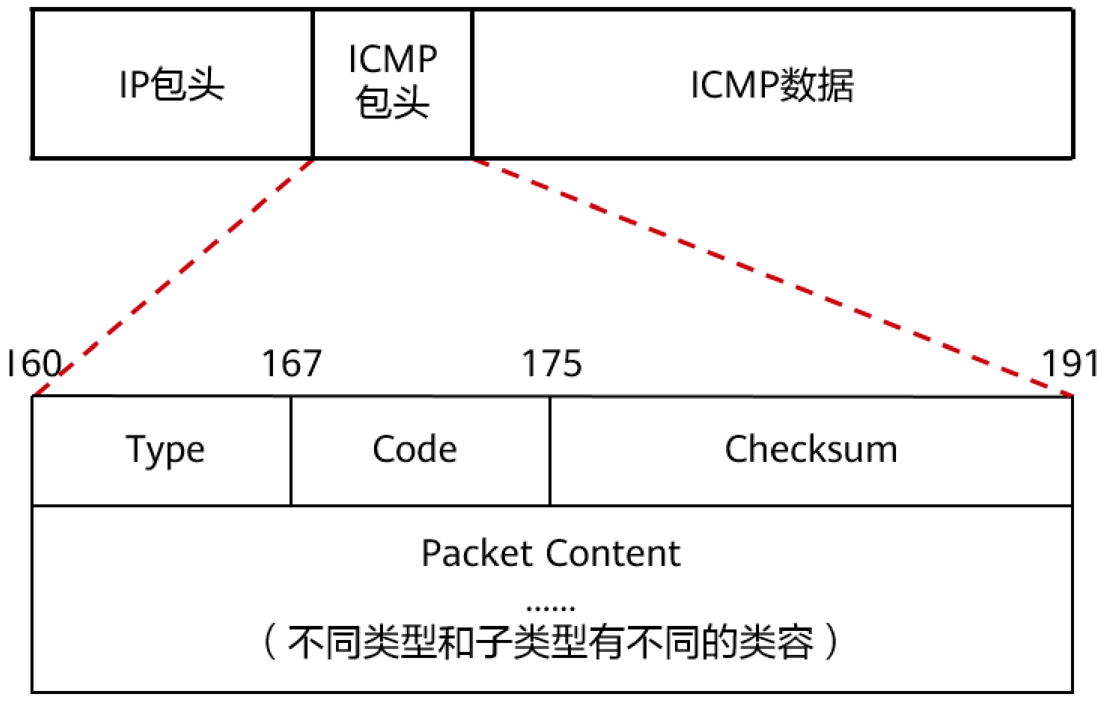
```
PS C:\Users\11822> ping -t 14.215.177.39

正在 Ping 14.215.177.39 具有 32 字节的数据:
来自 14.215.177.39 的回复: 字节=32 时间=29ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=28ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=29ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=29ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=28ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=28ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=29ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=29ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=29ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=29ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=29ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=28ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=29ms TTL=54
来自 14.215.177.39 的回复: 字节=32 时间=30ms TTL=54

14.215.177.39 的 Ping 统计信息:
    数据包: 已发送 = 14，已接收 = 14，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 28ms，最长 = 30ms，平均 = 28ms
Control-C
```

Wireshark 捕获到的 ICMP 分组如下：



```
27951 4371.073314   192.168.0.104     14.215.177.39     ICMP     74 Echo (ping) request  id=0x0001, seq=2114/16904, ttl=128 (reply in 27952)
27952 4371.102273   14.215.177.39     192.168.0.104     ICMP     74 Echo (ping) reply    id=0x0001, seq=2114/16904, ttl=54 (request in 27951)
27953 4372.079609   192.168.0.104     14.215.177.39     ICMP     74 Echo (ping) request  id=0x0001, seq=2115/17160, ttl=128 (reply in 27954)
27954 4372.107806   14.215.177.39     192.168.0.104     ICMP     74 Echo (ping) reply    id=0x0001, seq=2115/17160, ttl=54 (request in 27953)
27955 4373.095442   192.168.0.104     14.215.177.39     ICMP     74 Echo (ping) request  id=0x0001, seq=2116/17416, ttl=128 (reply in 27956)
27956 4373.124698   14.215.177.39     192.168.0.104     ICMP     74 Echo (ping) reply    id=0x0001, seq=2116/17416, ttl=54 (request in 27955)
27957 4374.110847   192.168.0.104     14.215.177.39     ICMP     74 Echo (ping) request  id=0x0001, seq=2117/17672, ttl=128 (reply in 27958)
27958 4374.140162   14.215.177.39     192.168.0.104     ICMP     74 Echo (ping) reply    id=0x0001, seq=2117/17672, ttl=54 (request in 27957)
27959 4375.129173   192.168.0.104     14.215.177.39     ICMP     74 Echo (ping) request  id=0x0001, seq=2118/17928, ttl=128 (reply in 27960)
27960 4375.157538   14.215.177.39     192.168.0.104     ICMP     74 Echo (ping) reply    id=0x0001, seq=2118/17928, ttl=54 (request in 27959)
27961 4376.150950   192.168.0.104     14.215.177.39     ICMP     74 Echo (ping) request  id=0x0001, seq=2119/18184, ttl=128 (reply in 27962)
27962 4376.179833   14.215.177.39     192.168.0.104     ICMP     74 Echo (ping) reply    id=0x0001, seq=2119/18184, ttl=54 (request in 27961)
27963 4377.158247   192.168.0.104     14.215.177.39     ICMP     74 Echo (ping) request  id=0x0001, seq=2120/18440, ttl=128 (reply in 27964)
27964 4377.187697   14.215.177.39     192.168.0.104     ICMP     74 Echo (ping) reply    id=0x0001, seq=2120/18440, ttl=54 (request in 27963)
27965 4378.174080   192.168.0.104     14.215.177.39     ICMP     74 Echo (ping) request  id=0x0001, seq=2121/18696, ttl=128 (reply in 27966)
27966 4378.203108   14.215.177.39     192.168.0.104     ICMP     74 Echo (ping) reply    id=0x0001, seq=2121/18696, ttl=54 (request in 27965)
27968 4379.189379   192.168.0.104     14.215.177.39     ICMP     74 Echo (ping) request  id=0x0001, seq=2122/18952, ttl=128 (reply in 27969)
27969 4379.218856   14.215.177.39     192.168.0.104     ICMP     74 Echo (ping) reply    id=0x0001, seq=2122/18952, ttl=54 (request in 27968)
27972 4380.205064   192.168.0.104     14.215.177.39     ICMP     74 Echo (ping) request  id=0x0001, seq=2123/19208, ttl=128 (reply in 27973)
27973 4380.233957   14.215.177.39     192.168.0.104     ICMP     74 Echo (ping) reply    id=0x0001, seq=2123/19208, ttl=54 (request in 27972)
27974 4381.220681   192.168.0.104     14.215.177.39     ICMP     74 Echo (ping) request  id=0x0001, seq=2124/19464, ttl=128 (reply in 27975)
```

ICMP 报文格式：



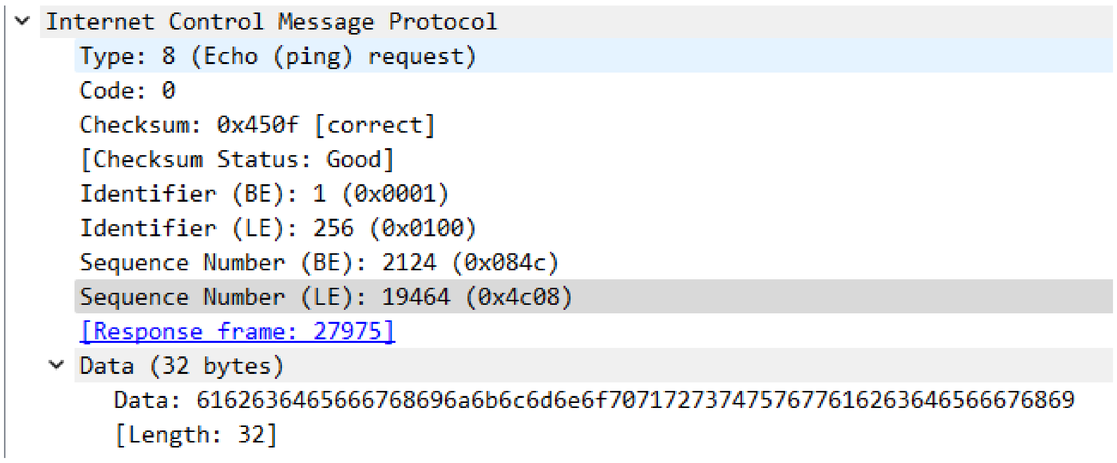
ICMP 报文分类：

表1-1 ICMP报文分类

Type	Code	描述	查询/差错
0-Echo响应	0	Echo响应报文	查询
3-目的不可达	0	目标网络不可达报文	差错
	1	目标主机不可达报文	差错
	2	目标协议不可达报文	差错
	3	目标端口不可达报文	差错
	4	要求分段并设置DF flag标志报文	差错
	5	源路由失败报文	差错
	6	未知的目标网络报文	差错
	7	未知的目标主机报文	差错
	8	源主机隔离报文	差错
	9	禁止访问的网络报文	差错
	10	禁止访问的主机报文	差错
	11	对特定的TOS网络不可达报文	差错
	12	对特定的TOS主机不可达报文	差错
	13	由于过滤 网络流量被禁止报文	差错
	14	主机越权报文	差错
	15	优先权终止生效报文	差错
5-重定向	0	重定向网络报文	差错
	1	重定向主机报文	差错
	2	基于TOS的网络重定向报文	差错
	3	基于TOS的主机重定向报文	差错
8-Echo请求	0	Echo请求报文	查询
9-路由器通告	0	路由通告报文	查询
10-路由器请求	0	路由器的发现/选择/请求报文	查询
11-ICMP超时	0	TTL超时报文	差错
	1	分片重组超时报文	差错
12-参数问题	0	IP报首部参数错误报文	差错
	1	丢失必要选项报文	差错
	2	不支持的长度报文	差错
13-时间戳请求	0	时间戳请求报文	查询
14-时间戳应答	0	时间戳应答报文	查询
15-信息请求	0	信息请求报文	查询
16-信息应答	0	信息应答报文	查询

每个 ICMP 报文中都有一个校验和，用来检测数据是否正确。

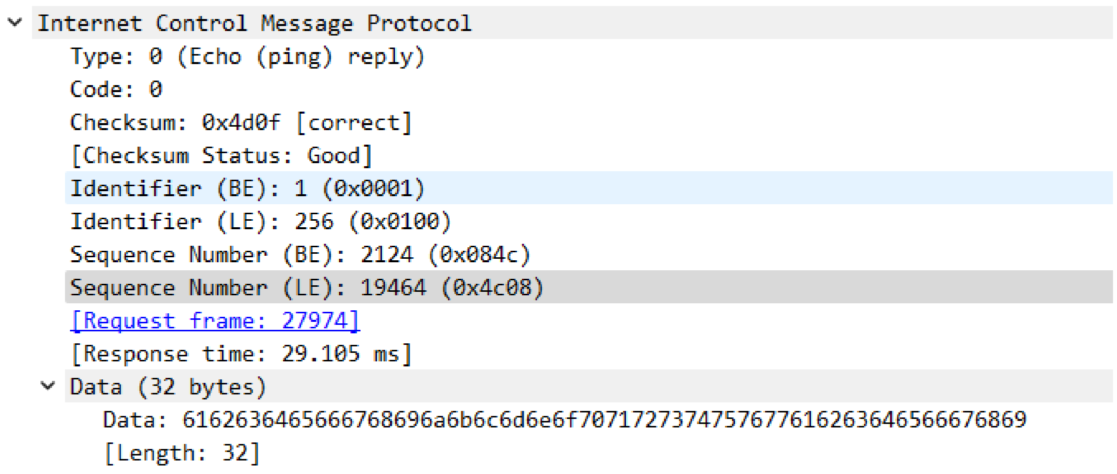
请求 ICMP：



```
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x450f [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 2124 (0x084c)
    Sequence Number (LE): 19464 (0x4c08)
    [Response frame: 27975]
  Data (32 bytes)
        Data: 6162636465666768696a6b6c6d6e6f7071727374757677616263646566676869
        [Length: 32]
```

Type 为 8

回应 ICMP：



```
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x4d0f [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 2124 (0x084c)
    Sequence Number (LE): 19464 (0x4c08)
    [Request frame: 27974]
    [Response time: 29.105 ms]
  Data (32 bytes)
        Data: 6162636465666768696a6b6c6d6e6f7071727374757677616263646566676869
        [Length: 32]
```

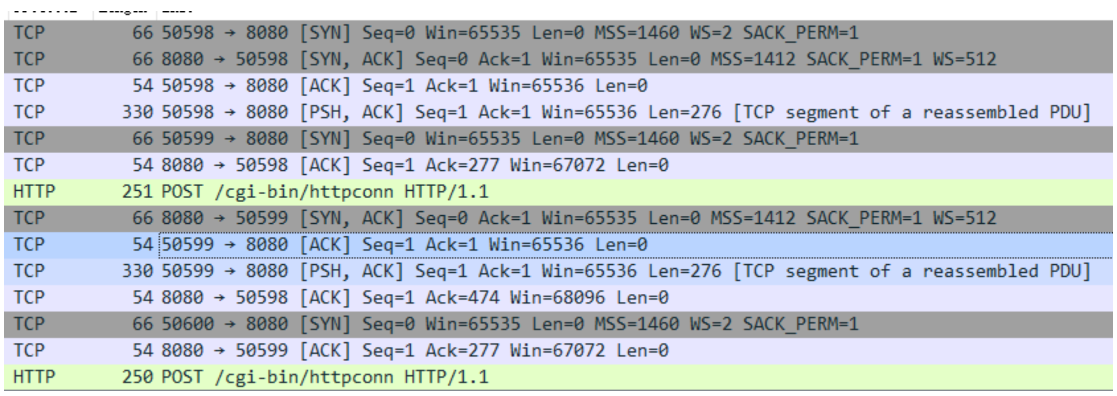
Type 为 0

报文中的 Sequence Number 表示序列号，让请求报文和回应报文可以一一对应。

e) 查看 TCP 三次握手以及其断开：

三次握手：

捕获到以下 TCP 报文：



```
TCP        66 50598 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=2 SACK_PERM=1
TCP        66 8080 → 50598 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM=1 WS=512
TCP        54 50598 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0
TCP       330 50598 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=276 [TCP segment of a reassembled PDU]
TCP        66 50599 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=2 SACK_PERM=1
TCP        54 8080 → 50598 [ACK] Seq=1 Ack=277 Win=67072 Len=0
HTTP      251 POST /cgi-bin/httpconn HTTP/1.1
TCP        66 8080 → 50599 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM=1 WS=512
TCP        54 50599 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0
TCP       330 50599 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=276 [TCP segment of a reassembled PDU]
TCP        54 8080 → 50598 [ACK] Seq=1 Ack=474 Win=68096 Len=0
TCP        66 50600 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=2 SACK_PERM=1
TCP        54 8080 → 50599 [ACK] Seq=1 Ack=277 Win=67072 Len=0
HTTP      250 POST /cgi-bin/httpconn HTTP/1.1
```

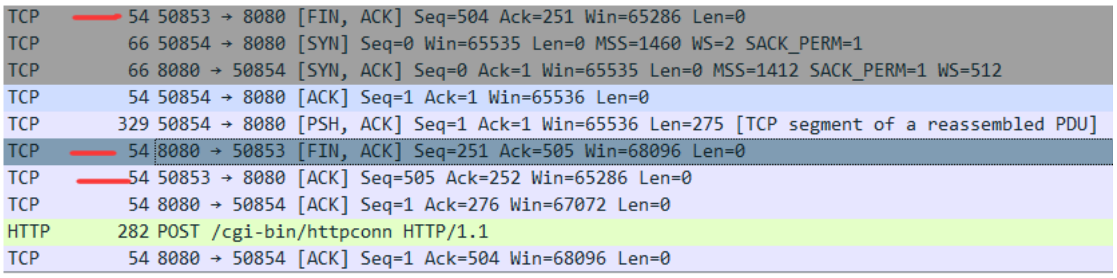
端口 50598 向端口 8080 发送 TCP 报文，其中 SYN=1，ACK=0，表示这个请求连接建立的报文。8080 向 50598 回复带有 SYN 和 ACK 的 TCP 报文表示接受这次连接请求。

TCP 报文里 MSS 表示最大段大小为 1460B，SACK_PERM 表示支持接受 SACK 报文，WS 是窗口缩放的比例因子，可以扩大窗口大小。

之后 50598 又向 8080 发送 ACK 表示收到了回应的 ACK，之后发送的含有 PSH 和 ACK 的报文表示这个 ACK 中家带了数据，数据长度为 276。8080 在收到 50598 发送的 ACK 之后再回复一个 ACK，表示自己收到了夹带数据的 ACK，这时三次握手完毕，两者建立连接成功。

50599 端口和 8080 端口建立连接的方式类似。

断开连接：

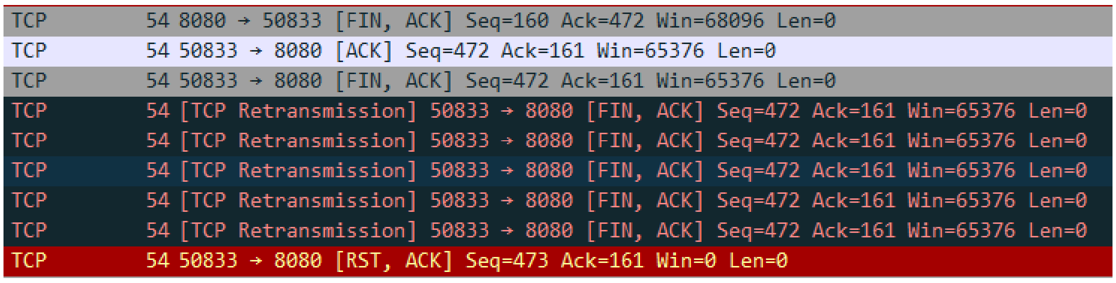


```
TCP        54 50853 → 8080 [FIN, ACK] Seq=504 Ack=251 Win=65286 Len=0
TCP        66 50854 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=2 SACK_PERM=1
TCP        66 8080 → 50854 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM=1 WS=512
TCP        54 50854 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0
TCP       329 50854 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=275 [TCP segment of a reassembled PDU]
TCP        54 8080 → 50853 [FIN, ACK] Seq=251 Ack=505 Win=68096 Len=0
TCP        54 50853 → 8080 [ACK] Seq=505 Ack=252 Win=65286 Len=0
TCP        54 8080 → 50854 [ACK] Seq=1 Ack=276 Win=67072 Len=0
HTTP      282 POST /cgi-bin/httpconn HTTP/1.1
TCP        54 8080 → 50854 [ACK] Seq=1 Ack=504 Win=68096 Len=0
```

50853 端口向 8080 发送含有 FIN 和 ACK 标志的报文表示自己

请求断开连接。8080 在收到请求之后回复含有 FIN 的 ACK 报文，允许断开连接。50853 在收到 8080 的回复之后再发送一次 ACK，表示自己收到了断开连接许可，可以断开连接了。

假如断开连接的时候出现异常，例如下面情况:



```
TCP         54 8080 → 50833 [FIN, ACK] Seq=160 Ack=472 Win=68096 Len=0
TCP         54 50833 → 8080 [ACK] Seq=472 Ack=161 Win=65376 Len=0
TCP         54 50833 → 8080 [FIN, ACK] Seq=472 Ack=161 Win=65376 Len=0
TCP         54 [TCP Retransmission] 50833 → 8080 [FIN, ACK] Seq=472 Ack=161 Win=65376 Len=0
TCP         54 [TCP Retransmission] 50833 → 8080 [FIN, ACK] Seq=472 Ack=161 Win=65376 Len=0
TCP         54 [TCP Retransmission] 50833 → 8080 [FIN, ACK] Seq=472 Ack=161 Win=65376 Len=0
TCP         54 [TCP Retransmission] 50833 → 8080 [FIN, ACK] Seq=472 Ack=161 Win=65376 Len=0
TCP         54 [TCP Retransmission] 50833 → 8080 [FIN, ACK] Seq=472 Ack=161 Win=65376 Len=0
TCP         54 50833 → 8080 [RST, ACK] Seq=473 Ack=161 Win=0 Len=0
```

50833 向 8080 回复允许断开的 ACK 时，一直收不到 8080 的回复 ACK，则一直重传。重传五次之后 50833 停止重传，发送含有 RST 和 ACK 的报文告诉 8080 端口需要重新建立连接，此连接已经被释放。