浙江大学

# 本科实验报告

| | |
|---|---|
| 课程名称： | 计算机网络基础 |
| 实验名称： | 网络协议分析 |
| 姓　　名： | 姚熙源 |
| 学　　院： | 计算机学院 |
| 系： | 计算机科学与技术 |
| 专　　业： | 软件工程 |
| 学　　号： | 3190300677 |
| 指导教师： | 董玮 |

2022 年　　5 月　　28 日

# 浙江大学实验报告

实验名称：网络协议分析　　实验类型：分析实验

同组学生：　　实验地点：计算机网络实验室

## 一、 实验目的

- 进一步学习使用 Wireshark 抓包工具。
- 观察和理解常见网络协议的交互过程
- 理解数据包分层结构和格式。

## 二、 实验内容

- 熟练掌握网络协议分析软件 Wireshark 的使用
- 观察所在网络出现的各类网络协议，了解其种类和分层结构
- 观察捕获到的数据包格式，理解各字段含义
- 根据要求配置 Wireshark，捕获某一类协议的数据包，并分析解读

## 三、 主要仪器设备

- 联网的 PC 机
- WireShark 协议分析软件

## 四、 操作方法与实验步骤

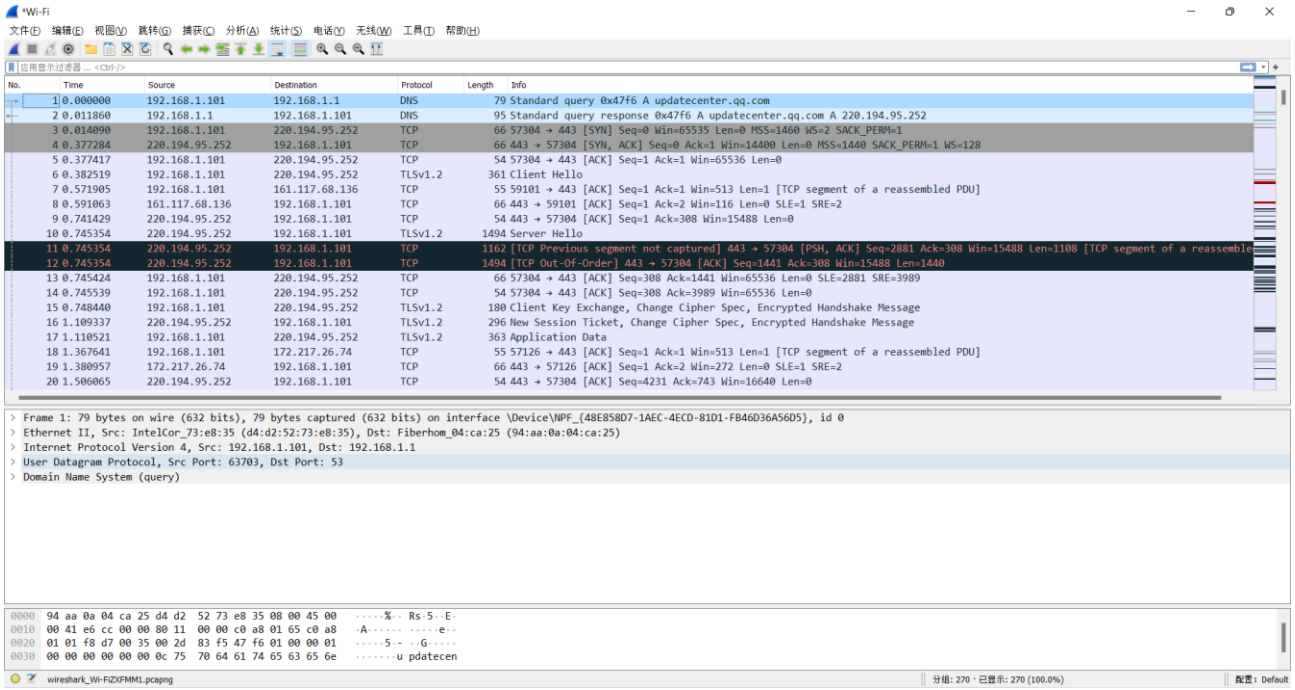
- 配置网络包捕获软件，捕获所有机器的数据包
- 观察捕获到的数据包，并对照解析结果和原始数据包
- 配置网络包捕获软件，只捕获特定 IP 或特定类型的包
- 抓取以下通信协议数据包，观察通信过程和数据包格式
  - ✓ PING：测试一个目标地址是否可达（在实验一基础上）
  - ✓ TRACE ROUTE：跟踪一个目标地址的途经路由（在实验一基础上）
  - ✓ NSLOOKUP：查询一个域名（在实验一基础上）
  - ✓ HTTP：访问一个网页
  - ✓ FTP：上传或下载一个文件
  - ✓ SMTP：发送一封邮件
  - ✓ POP3/IMAP：接收一封邮件
  - ✓ RTP：抓取一段音频流

提醒：为了避免捕获到大量无关数据包，影响实验观察，建议关闭所有无关软件。

## 五、 实验数据记录和处理

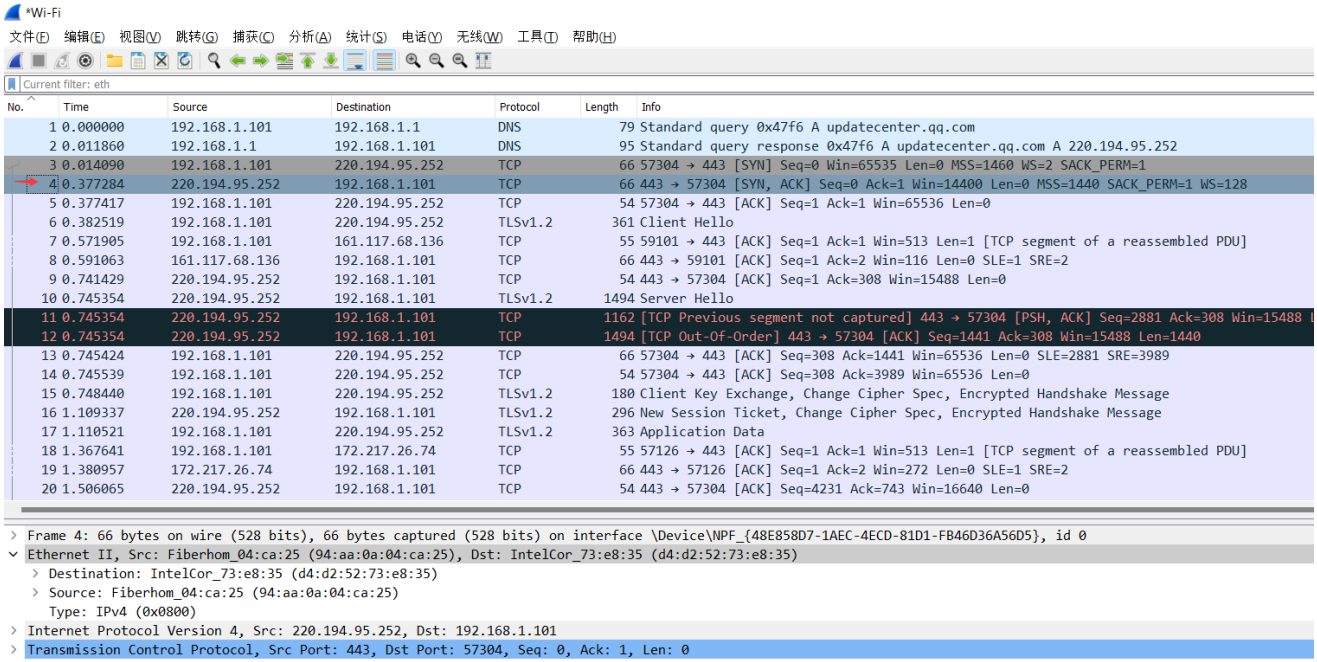
### ✧ Part One

- **打开 WireShark，开始捕获网络数据包后，你看到了什么？有哪些协议？**




看到了很多的数据包不停地在发送和接收，可看到每个数据包的具体内容和解析。其中也包含很多种协议，有 TCP、DNS、TLSv1.2、UDP、SSL、ARP、HTTP 等协议。

- **找一个包含 Ethernet 的数据包，这是什么协议？标出源和目标 MAC 地址。**




这是 TCP 协议。

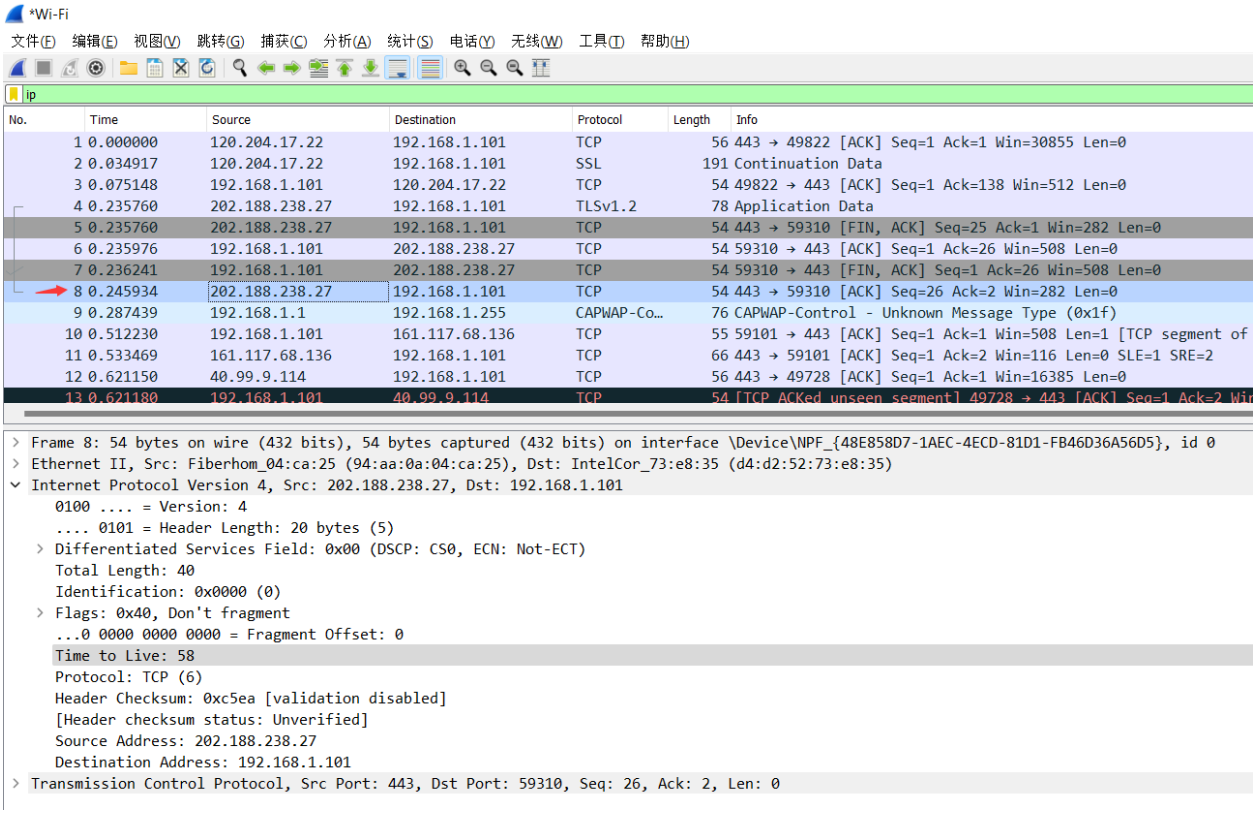
源 MAC 地址：

```
Source: Fiberhom_04:ca:25 (94:aa:0a:04:ca:25)
    Address: Fiberhom_04:ca:25 (94:aa:0a:04:ca:25)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
```

目标 MAC 地址：

```
Ethernet II, Src: Fiberhom_04:ca:25 (94:aa:0a:04:ca:25), Dst: IntelCor_73:e8:35 (d4:d2:52:73:e8:35)
  Destination: IntelCor_73:e8:35 (d4:d2:52:73:e8:35)
    Address: IntelCor_73:e8:35 (d4:d2:52:73:e8:35)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
```

- **找一个包含 IP 的数据包，这是什么协议？标出源 IP 地址、目标 IP 地址。**

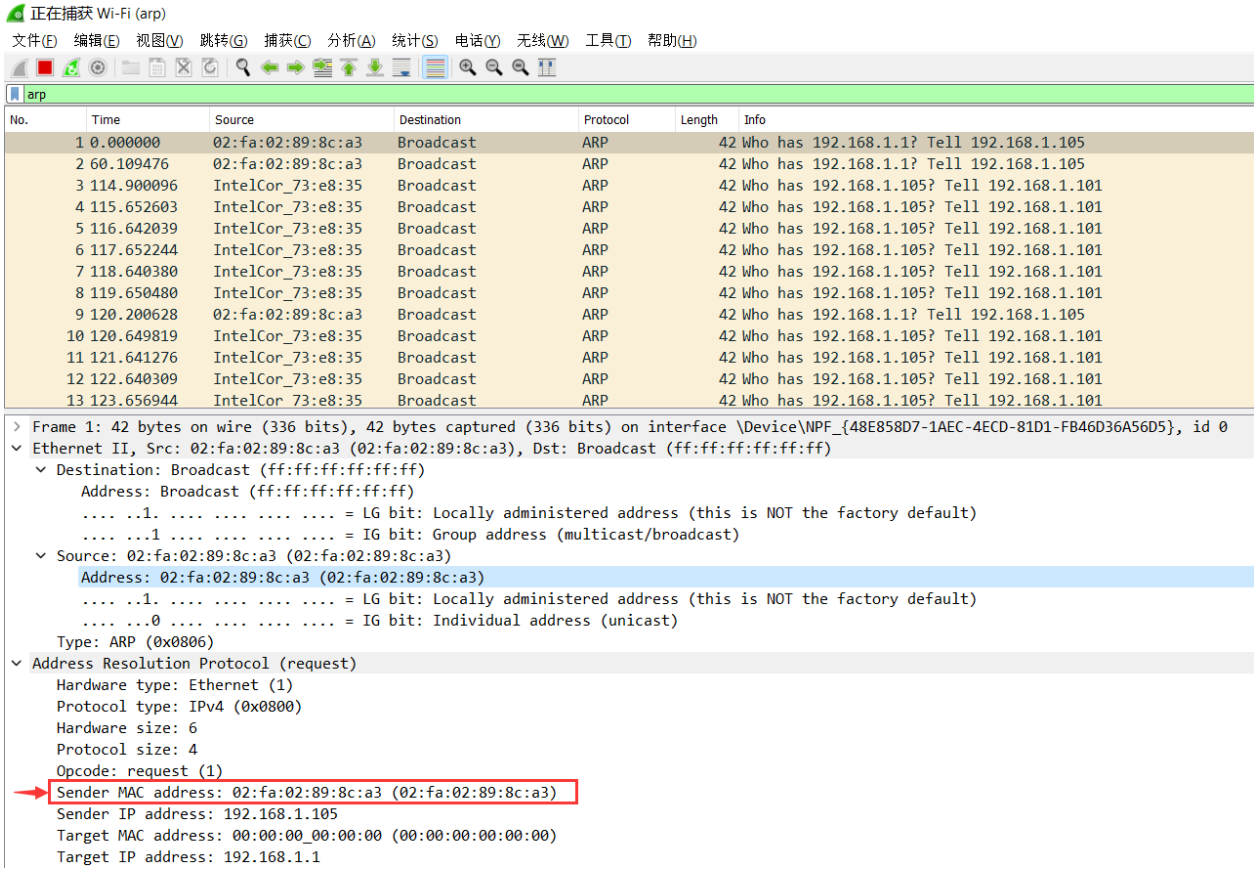



这是 TCP 协议

源 IP 地址：Source Address: 192.168.1.101

目标 IP 地址：Destination Address: 202.188.238.27

- **找一个 ARP 数据包，这是请求还是应答？标注发送者的 MAC 地址。**

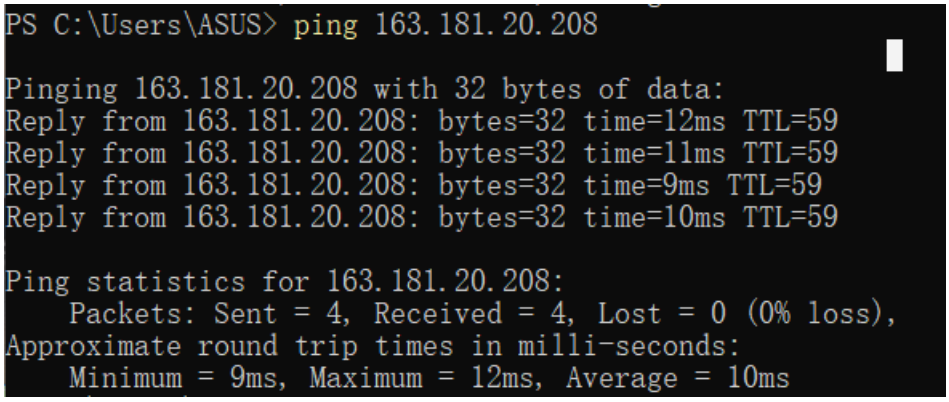


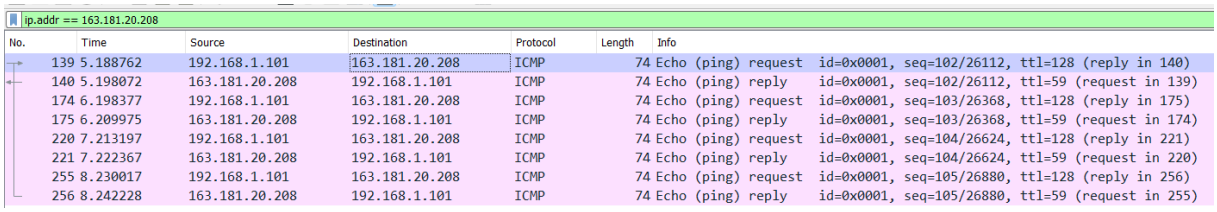

这是请求包，发送者的 MAC 地址已用红色框标注。

请在下面的每次捕获任务完成后，保存 Wireshark 抓包记录（.pcap 格式），随报告一起提交。每一个协议一个单独文件，文件名请取得便于理解。

## ✧ Part Two

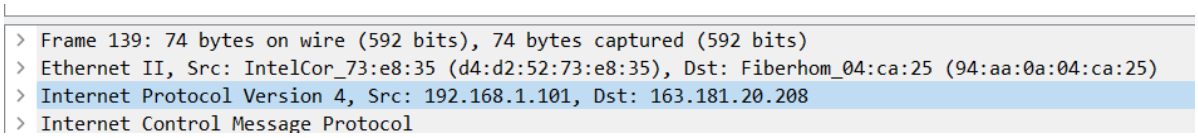
- **使用 Ping 命令，测试某个 IP 地址的连通性，并捕获这次的数据包。数据包由几层协议构成？分别是什么协议？选择一个请求包和一个响应包，展开最高层协议的详细内容，标出请求包和应答包、类型、序号。**





ip.addr == 163.181.20.208

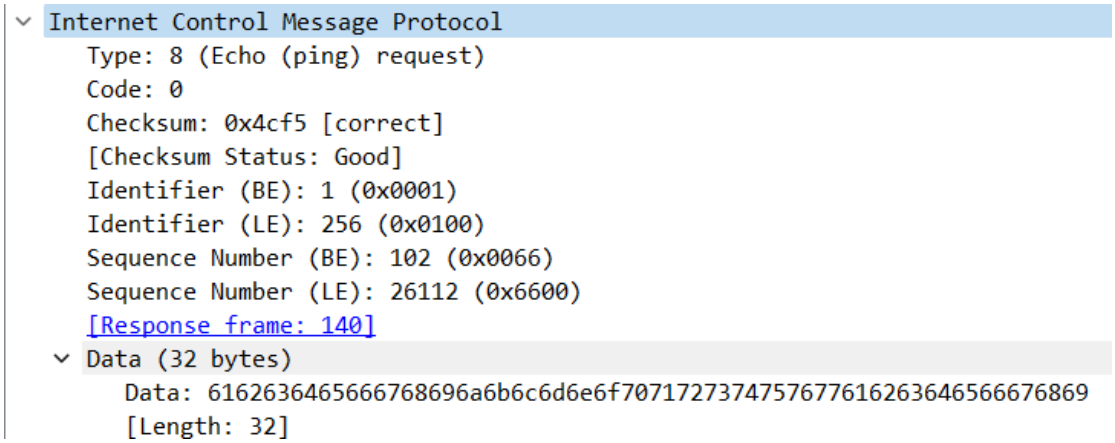
| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 139 | 5.188762 | 192.168.1.101 | 163.181.20.208 | ICMP | 74 | Echo (ping) request id=0x0001, seq=102/26112, ttl=128 (reply in 140) |
| 140 | 5.198072 | 163.181.20.208 | 192.168.1.101 | ICMP | 74 | Echo (ping) reply id=0x0001, seq=102/26112, ttl=59 (request in 139) |
| 174 | 6.198377 | 192.168.1.101 | 163.181.20.208 | ICMP | 74 | Echo (ping) request id=0x0001, seq=103/26368, ttl=128 (reply in 175) |
| 175 | 6.209975 | 163.181.20.208 | 192.168.1.101 | ICMP | 74 | Echo (ping) reply id=0x0001, seq=103/26368, ttl=59 (request in 174) |
| 220 | 7.213197 | 192.168.1.101 | 163.181.20.208 | ICMP | 74 | Echo (ping) request id=0x0001, seq=104/26624, ttl=128 (reply in 221) |
| 221 | 7.222367 | 163.181.20.208 | 192.168.1.101 | ICMP | 74 | Echo (ping) reply id=0x0001, seq=104/26624, ttl=59 (request in 220) |
| 255 | 8.230017 | 192.168.1.101 | 163.181.20.208 | ICMP | 74 | Echo (ping) request id=0x0001, seq=105/26880, ttl=128 (reply in 256) |
| 256 | 8.242228 | 163.181.20.208 | 192.168.1.101 | ICMP | 74 | Echo (ping) reply id=0x0001, seq=105/26880, ttl=59 (request in 255) |



```
> Frame 139: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
> Ethernet II, Src: IntelCor_73:e8:35 (d4:d2:52:73:e8:35), Dst: Fiberhom_04:ca:25 (94:aa:0a:04:ca:25)
> Internet Protocol Version 4, Src: 192.168.1.101, Dst: 163.181.20.208
> Internet Control Message Protocol
```

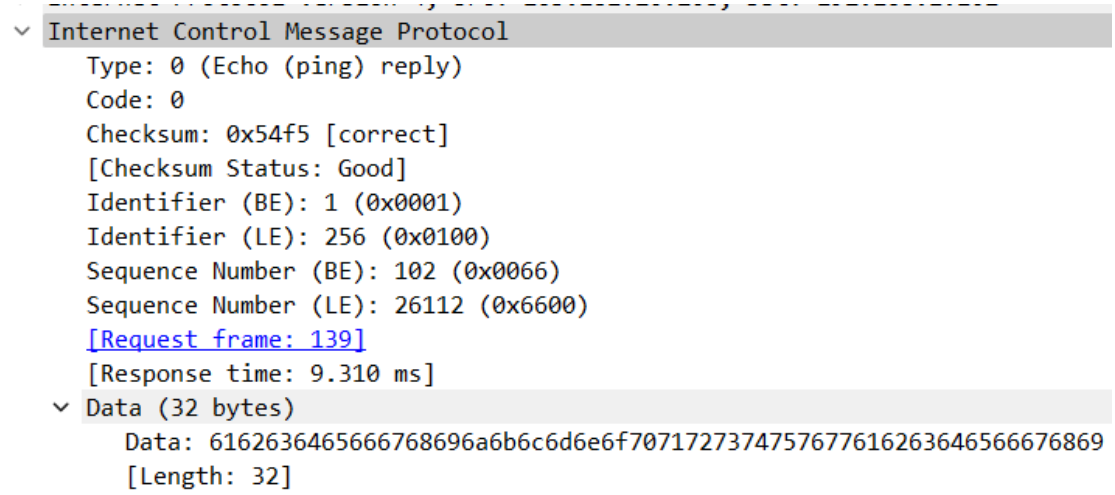
数据包由 3 层协议构成，分别是以太网协议、IPv4 协议、ICMP 协议

请求包，类型 Type: 8(Echo (ping) request)，序号 Sequence Number(BE): 102(0x0066)



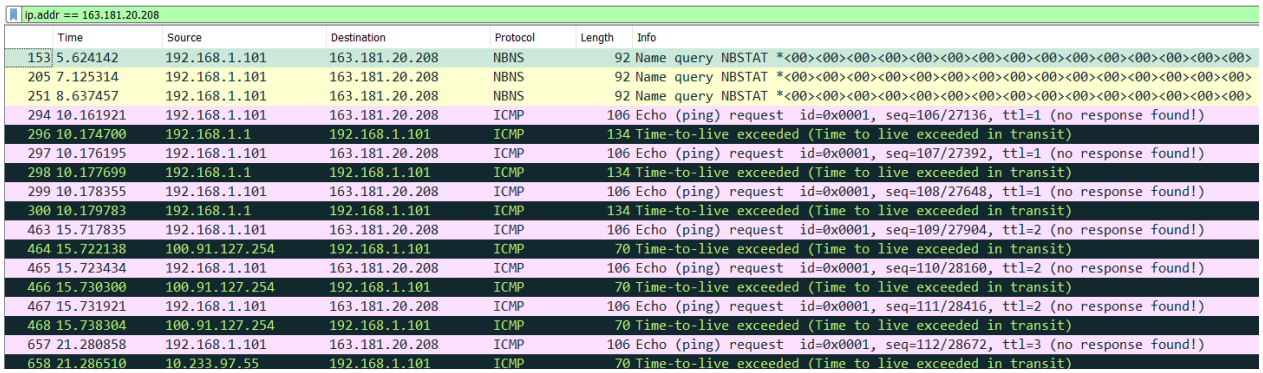
```
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4cf5 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 102 (0x0066)
    Sequence Number (LE): 26112 (0x6600)
    [Response frame: 140]
  Data (32 bytes)
      Data: 6162636465666768696a6b6c6d6e6f707172737475767761626364656667686 9
      [Length: 32]
```

响应包，类型 Type: 0(Echo (ping) reply)，序号 Sequence Number (BE): 102(0x0066)



```
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x54f5 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 102 (0x0066)
    Sequence Number (LE): 26112 (0x6600)
    [Request frame: 139]
    [Response time: 9.310 ms]
  Data (32 bytes)
      Data: 6162636465666768696a6b6c6d6e6f707172737475767761626364656667686 9
      [Length: 32]
```

- **使用 Tracert 命令（Mac 下使用 Traceroute 命令），跟踪某个外部 IP 地址的路由，并捕获这次的数据包。数据包由几层协议构成？分别是什么协议？查看并标记多个请求包的 IP 协议层的 TTL 字段，发现了什么规律？选择一个请求包和一个响应包，展开最高层协议的详细内容，标出类型、序号等关键字段。与 Ping 命令的数据包有什么不同？**



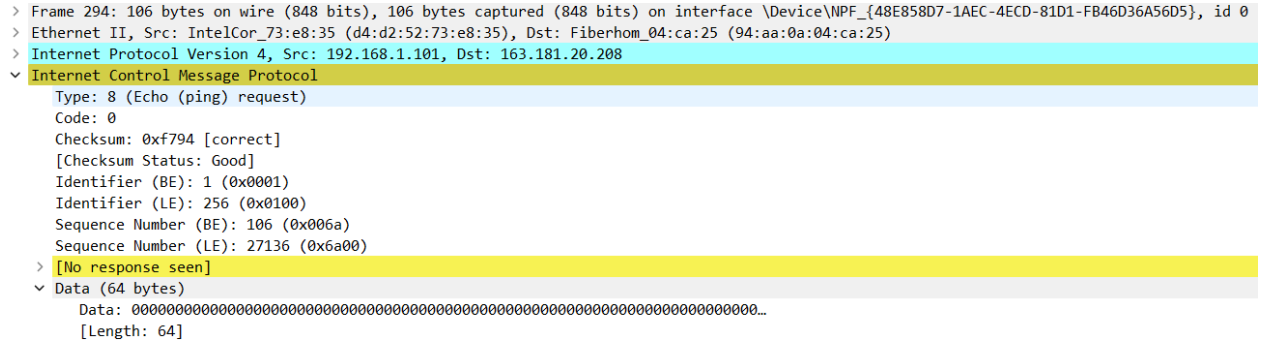
ip.addr == 163.181.20.208

| | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 153 | 5.624142 | 192.168.1.101 | 163.181.20.208 | NBNS | 92 | Name query NBSTAT *<00><00><00><00><00><00><00><00><00><00><00><00><00><00><00><00> |
| 205 | 7.125314 | 192.168.1.101 | 163.181.20.208 | NBNS | 92 | Name query NBSTAT *<00><00><00><00><00><00><00><00><00><00><00><00><00><00><00><00> |
| 251 | 8.637457 | 192.168.1.101 | 163.181.20.208 | NBNS | 92 | Name query NBSTAT *<00><00><00><00><00><00><00><00><00><00><00><00><00><00><00><00> |
| 294 | 10.161921 | 192.168.1.101 | 163.181.20.208 | ICMP | 106 | Echo (ping) request id=0x0001, seq=106/27136, ttl=1 (no response found!) |
| 296 | 10.174700 | 192.168.1.1 | 192.168.1.101 | ICMP | 134 | Time-to-live exceeded (Time to live exceeded in transit) |
| 297 | 10.176195 | 192.168.1.101 | 163.181.20.208 | ICMP | 106 | Echo (ping) request id=0x0001, seq=107/27392, ttl=1 (no response found!) |
| 298 | 10.177699 | 192.168.1.1 | 192.168.1.101 | ICMP | 134 | Time-to-live exceeded (Time to live exceeded in transit) |
| 299 | 10.178355 | 192.168.1.101 | 163.181.20.208 | ICMP | 106 | Echo (ping) request id=0x0001, seq=108/27648, ttl=1 (no response found!) |
| 300 | 10.179783 | 192.168.1.1 | 192.168.1.101 | ICMP | 134 | Time-to-live exceeded (Time to live exceeded in transit) |
| 463 | 15.717835 | 192.168.1.101 | 163.181.20.208 | ICMP | 106 | Echo (ping) request id=0x0001, seq=109/27904, ttl=2 (no response found!) |
| 464 | 15.722138 | 100.91.127.254 | 192.168.1.101 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 465 | 15.723434 | 192.168.1.101 | 163.181.20.208 | ICMP | 106 | Echo (ping) request id=0x0001, seq=110/28160, ttl=2 (no response found!) |
| 466 | 15.730300 | 100.91.127.254 | 192.168.1.101 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 467 | 15.731921 | 192.168.1.101 | 163.181.20.208 | ICMP | 106 | Echo (ping) request id=0x0001, seq=111/28416, ttl=2 (no response found!) |
| 468 | 15.738304 | 100.91.127.254 | 192.168.1.101 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 657 | 21.280858 | 192.168.1.101 | 163.181.20.208 | ICMP | 106 | Echo (ping) request id=0x0001, seq=112/28672, ttl=3 (no response found!) |
| 658 | 21.286510 | 10.233.97.55 | 192.168.1.101 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |

由 3 层协议构成，分别是以太网协议、IPv4 协议和 ICMP 协议

TTL 字段从 1 开始慢慢递增；

请求包，类型 Type: 8 (Echo (ping) request)，序号 Sequence Number (BE): 106(0x006a)



```
> Frame 294: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface \Device\NPF_{48E858D7-1AEC-4ECD-81D1-FB46D36A56D5}, id 0
> Ethernet II, Src: IntelCor_73:e8:35 (d4:d2:52:73:e8:35), Dst: Fiberhom_04:ca:25 (94:aa:0a:04:ca:25)
> Internet Protocol Version 4, Src: 192.168.1.101, Dst: 163.181.20.208
v Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xf794 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 106 (0x006a)
    Sequence Number (LE): 27136 (0x6a00)
  > [No response seen]
  v Data (64 bytes)
      Data: 0000000000000000000000000000000000000000000000000000000000000000…
      [Length: 64]
```

响应包，类型 Type: 11 (Time-to-live exceeded)，序号 Sequence Number(BE): 106(0x006a)

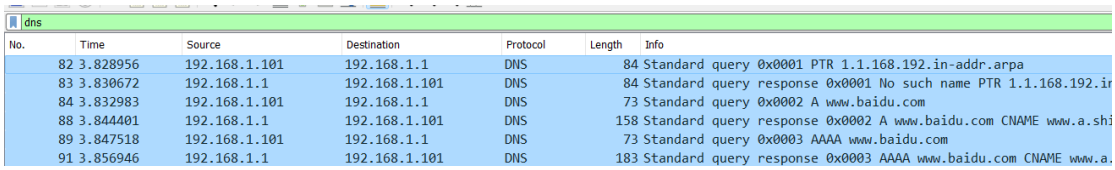
```
Internet Control Message Protocol
   Type: 11 (Time-to-live exceeded)
   Code: 0 (Time to live exceeded in transit)
   Checksum: 0xf4ff [correct]
   [Checksum Status: Good]
   Unused: 00000000
 Internet Protocol Version 4, Src: 192.168.1.101, Dst: 163.181.20.208
      0100 .... = Version: 4
      .... 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: 92
      Identification: 0xd690 (54928)
    Flags: 0x00
         0... .... = Reserved bit: Not set
         .0.. .... = Don't fragment: Not set
         ..0. .... = More fragments: Not set
      ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 1
      Protocol: ICMP (1)
      Header Checksum: 0x687e [validation disabled]
      [Header checksum status: Unverified]
      Source Address: 192.168.1.101
      Destination Address: 163.181.20.208
 Internet Control Message Protocol
      Type: 8 (Echo (ping) request)
      Code: 0
      Checksum: 0xf794 [unverified] [in ICMP error packet]
      [Checksum Status: Unverified]
      Identifier (BE): 1 (0x0001)
      Identifier (LE): 256 (0x0100)
      Sequence Number (BE): 106 (0x006a)
      Sequence Number (LE): 27136 (0x6a00)
    Data (64 bytes)
         Data: 0000000000000000000000000000000000000000000000000000000000000000…
         [Length: 64]
```

和 ping 命令的数据包相比，tracert 命令的 ICMP 数据包 data 字段总是为 0，响应包中包含了类型为 11TTL 降为 0 的超时信息。

- **使用 nslookup 命令，查询某个域名，并捕获这次的数据包。数据包由几层协议构成？分别是什么协议？标记 UDP 协议层的端口字段。选择一个请求包和一个响应包，展开最高层协议的详细内容，标出类型、序号、域名信息。**

```
PS C:\Users\ASUS> nslookup www.baidu.com
Server:  UnKnown
Address:  192.168.1.1

Non-authoritative answer:
Name:    www.wshifen.com
Addresses:  45.113.192.102
          45.113.192.101
Aliases:  www.baidu.com
          www.a.shifen.com
```



dns

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 82 | 3.828956 | 192.168.1.101 | 192.168.1.1 | DNS | 84 | Standard query 0x0001 PTR 1.1.168.192.in-addr.arpa |
| 83 | 3.830672 | 192.168.1.1 | 192.168.1.101 | DNS | 84 | Standard query response 0x0001 No such name PTR 1.1.168.192.ir |
| 84 | 3.832983 | 192.168.1.101 | 192.168.1.1 | DNS | 73 | Standard query 0x0002 A www.baidu.com |
| 88 | 3.844401 | 192.168.1.1 | 192.168.1.101 | DNS | 158 | Standard query response 0x0002 A www.baidu.com CNAME www.a.shi |
| 89 | 3.847518 | 192.168.1.101 | 192.168.1.1 | DNS | 73 | Standard query 0x0003 AAAA www.baidu.com |
| 91 | 3.856946 | 192.168.1.1 | 192.168.1.101 | DNS | 183 | Standard query response 0x0003 AAAA www.baidu.com CNAME www.a. |

由 4 层协议构成，分别是以太网、IPv4、UDP、DN 协议。

UDP 协议层的端口字段

```
> Frame 82: 84 bytes on wire (672 bits), 84 bytes capture
> Ethernet II, Src: IntelCor_73:e8:35 (d4:d2:52:73:e8:35)
> Internet Protocol Version 4, Src: 192.168.1.101, Dst: 1
v User Datagram Protocol, Src Port: 61308, Dst Port: 53
    Source Port: 61308
    Destination Port: 53
    Length: 50
    Checksum: 0x83fa [unverified]
    [Checksum Status: Unverified]
    [Stream index: 2]
  > [Timestamps]
    UDP payload (42 bytes)
```

请求包

```
v Domain Name System (query)
    Transaction ID: 0x0002
  v Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v www.baidu.com: type A, class IN
        Name: www.baidu.com
        [Name Length: 13]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
```

类型：Flags: 0x0100 Standard query

序号：Transaction ID: 0x0002

域名信息：www.baidu.com : type A, class IN

响应包

```
Domain Name System (response)
    Transaction ID: 0x0002
  Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 4
    Authority RRs: 0
    Additional RRs: 0
  > Queries
  > Answers
    > www.baidu.com: type CNAME, class IN, cname www.a.shifen.com
    > www.a.shifen.com: type CNAME, class IN, cname www.wshifen.com
    > www.wshifen.com: type A, class IN, addr 45.113.192.102
    > www.wshifen.com: type A, class IN, addr 45.113.192.101
```

类型：Flags: 0x8180 Standard query response, No error

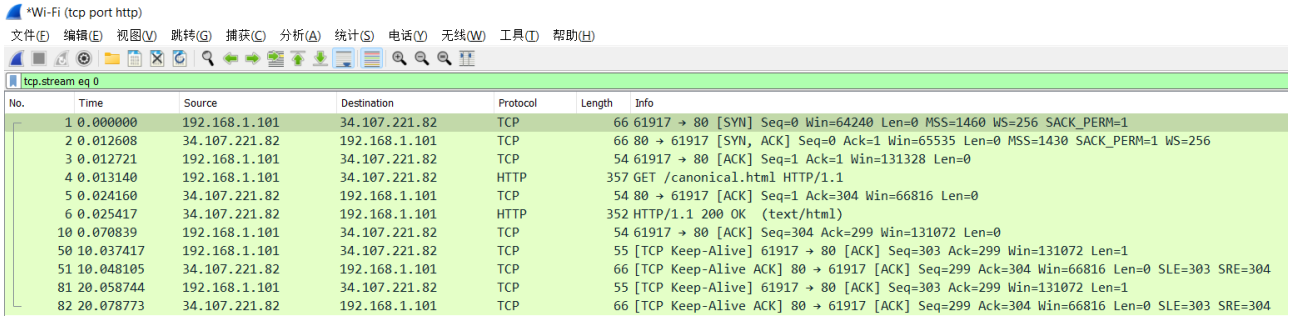
序号：Transaction ID: 0x0002

域名信息：

```
Answers
  www.baidu.com: type CNAME, class IN, cname www.a.shifen.com
      Name: www.baidu.com
      Type: CNAME (Canonical NAME for an alias) (5)
      Class: IN (0x0001)
      Time to live: 808 (13 minutes, 28 seconds)
      Data length: 15
      CNAME: www.a.shifen.com
  > www.a.shifen.com: type CNAME, class IN, cname www.wshifen.com
  > www.wshifen.com: type A, class IN, addr 45.113.192.102
  > www.wshifen.com: type A, class IN, addr 45.113.192.101
[Request In: 84]
```

## ✧ Part Three

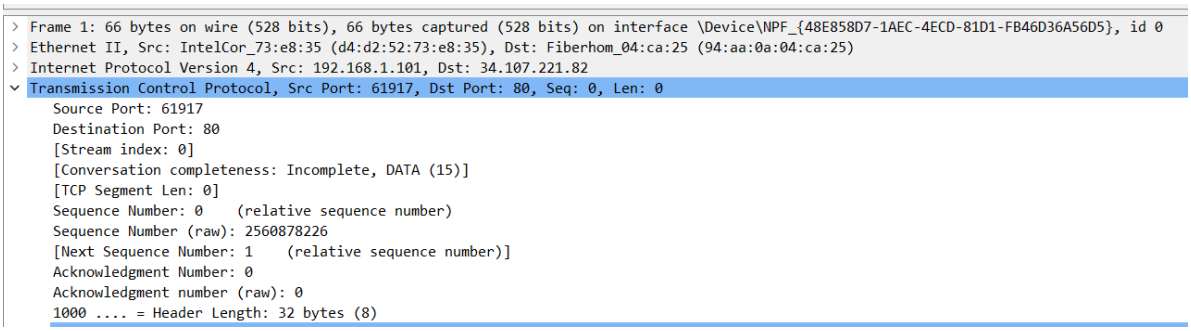
- **运行 ipconfig /flushdns 命令清空 DNS 缓存，然后打开浏览器，访问一个网页，并捕获这次的数据包（网页完全打开后，停止捕获）。数据包由几层协议构成？分别是什么协议？标出数据包的源和目标 IP 地址、源和目标端口。**




由 4 层协议构成，分别是以太网、IPv4、TCP、HTTP

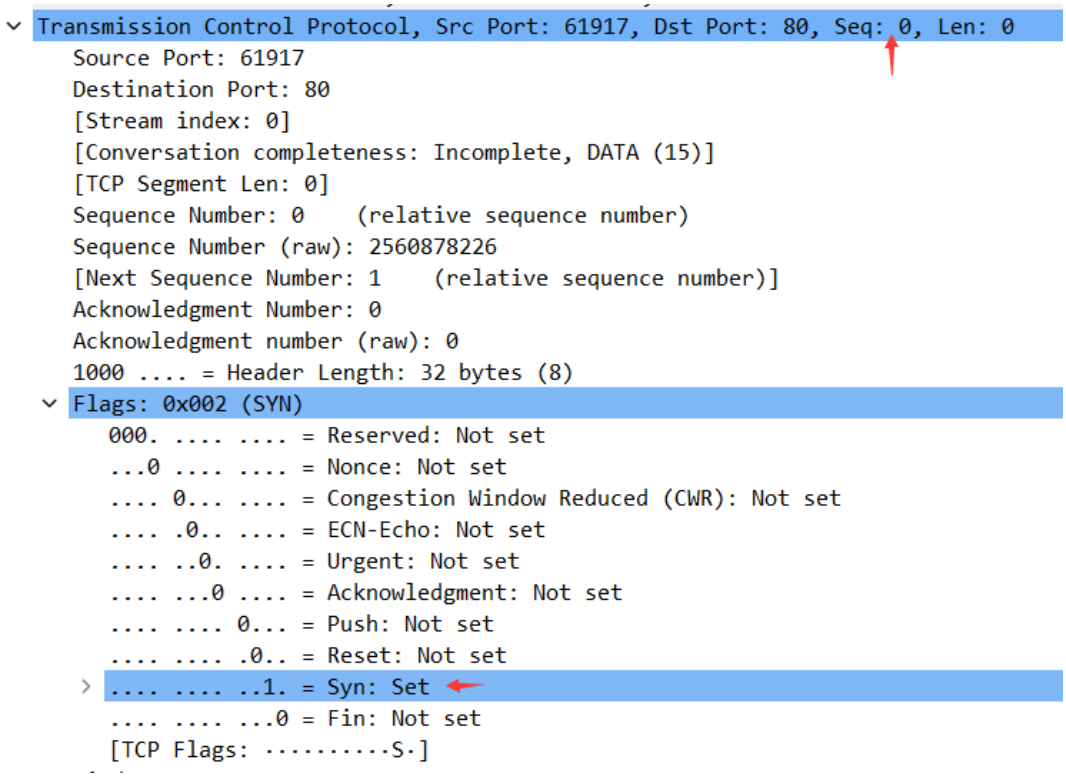
Source : 192.168.1.101 ; Port : 61917

Destination : 34.107.221.82 ; Port : 80

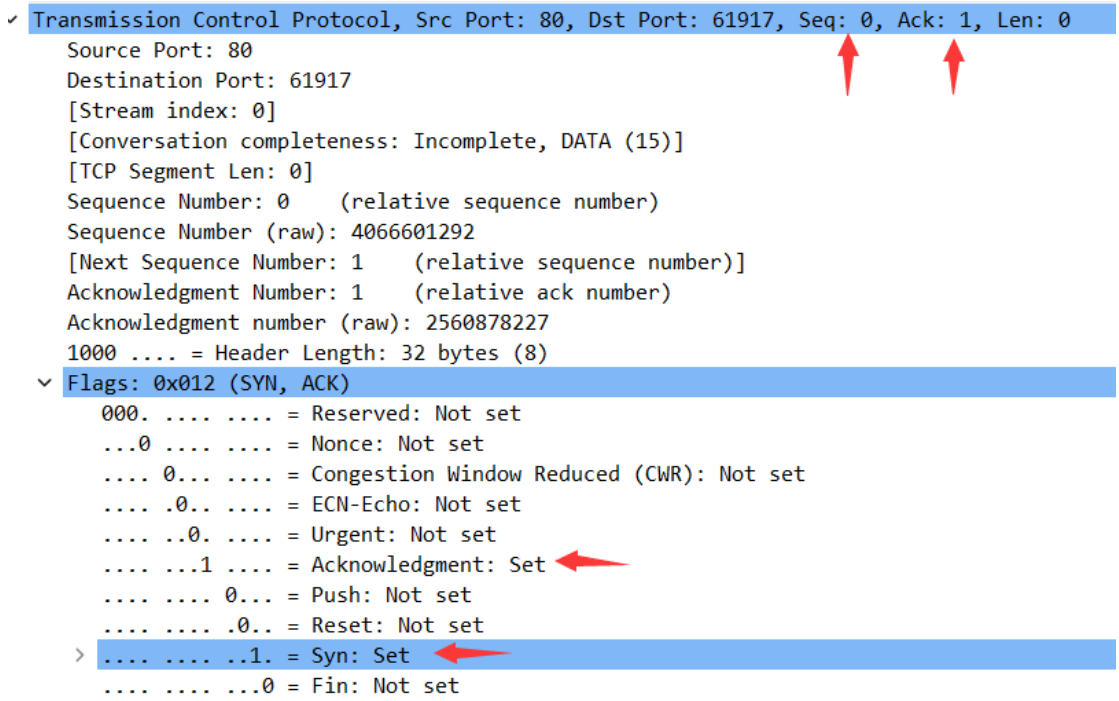



- **找到建立 TCP 连接的三个数据包（称为三次握手），展开 TCP 协议层的 Flags 字段，分别标记三个数据包的 SYN 标志位和 ACK 标志位。**

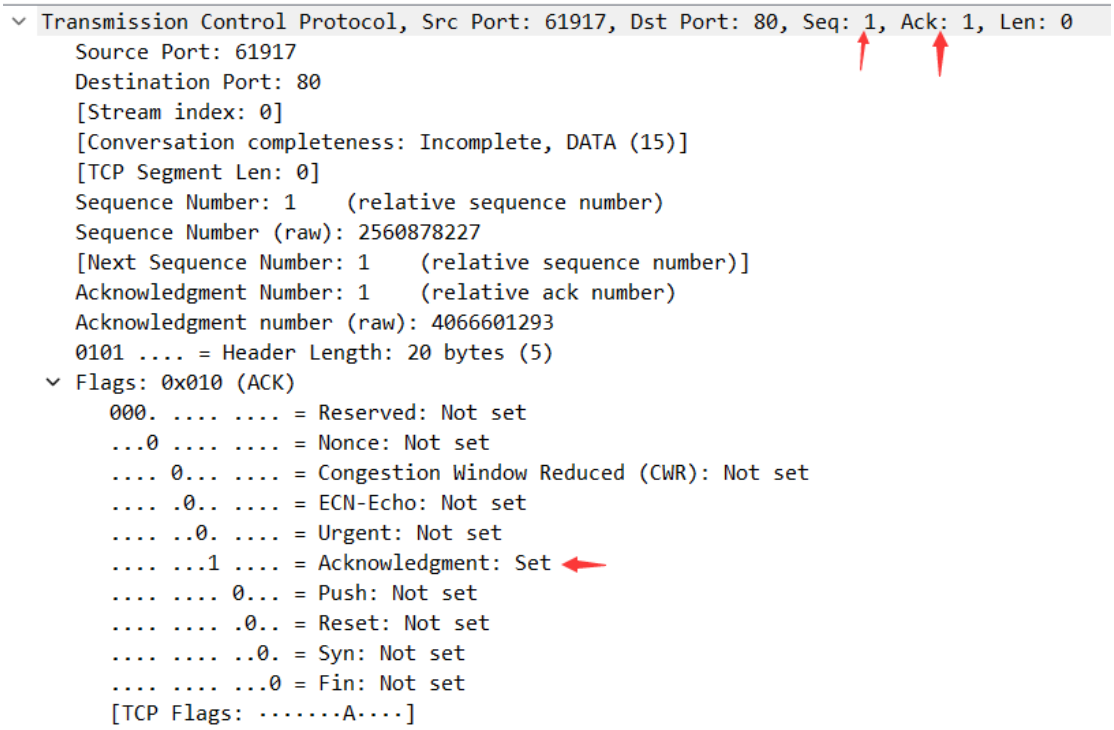
第一次握手，由 port 61917 向 port 80 发送第一个数据包，序号 seq=0 且 syn 标志位是 Set 状态。




第二次握手，由 port 80 向 port 61917 返回数据个数据包给我的 ip，序号 seq=0，ack=1(0+1)且 ack 和 syn 标志位是 Set 状态。



第三次握手，由 port 61917 向 port 80 返回数据包表示连接已建立，序号 seq=1，ack=1(0+1)且 Ack 是 Set 状态表示双方的连接已经建立。




- **选择一个包，点击右键，选择跟踪一个 TCP 流，截取完整的 HTTP 请求消息和部分响应消息，标记 HTTP 请求头部的 Method 字段、URI 字段和 Host 字段，标记 HTTP 响应头部的 Status Code 字段、Content-Type 和 Content-Length 字段，以及区分响应头部和体部的标记（单独的回车换行符）。**

  HTTP 请求头部头部的 Method 字段、URI 字段和 Host 字段：

  GET /canonical.html HTTP/1.1

  Host: detectportal.firefox.com

  HTTP 响应头部的 Status Code 字段、Content-Type 和 Content-Length 字段

  HTTP/1.1 200 OK

  Content-Type: text/html

  Content-Length: 90

```
GET /canonical.html HTTP/1.1
Host: detectportal.firefox.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:101.0) Gecko/20100101 Firefox/101.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cache-Control: no-cache
Pragma: no-cache
Connection: keep-alive

HTTP/1.1 200 OK
Server: nginx
Content-Length: 90
Via: 1.1 google
Date: Wed, 25 May 2022 22:18:18 GMT
Age: 55130
Content-Type: text/html
Cache-Control: public,must-revalidate,max-age=0,s-maxage=3600

<meta http-equiv="refresh" content="0;url=https://support.mozilla.org/kb/captive-portal"/>
```

- **使用过滤器 tcp.stream eq X，让 X 从 0 开始变化，直到没有数据。观察总共捕获到了几个 TCP 连接（一个 TCP 流对应一个 TCP 连接）？存在几个 HTTP 会话（一对 HTTP 请求和响应对应一次 HTTP 会话）？注意：一个 TCP 流上可能存在多个 HTTP 会话。**

  tcp.stream eq 0，9 个 TCP 连接，1 个 HTTP 请求和 1 个 HTTP 响应

  tcp.stream eq 1，9 个 TCP 连接，1 个 HTTP 请求和 1 个 HTTP 响应

  tcp.stream eq 2，9 个 TCP 连接，1 个 OCSP 请求和 1 个 OCSP 响应

  tcp.stream eq 3，9 个 TCP 连接，1 个 OCSP 请求和 1 个 OCSP 响应

  tcp.stream eq 4，9 个 TCP 连接，1 个 OCSP 请求和 1 个 OCSP 响应

  tcp.stream eq 5，9 个 TCP 连接，1 个 OCSP 请求和 1 个 OCSP 响应

  tcp.stream eq 6，7 个 TCP 连接，1 个 OCSP 请求和 1 个 OCSP 响应

  tcp.stream eq 7，9 个 TCP 连接，2 个 OCSP 请求和 2 个 OCSP 响应

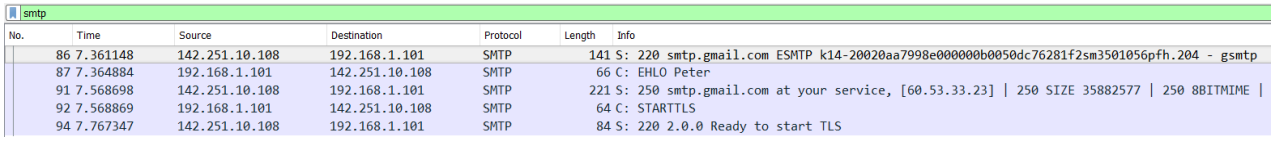
  tcp.stream eq 8，6 个 TCP 连接

  tcp.stream eq 9，6 个 TCP 连接，1 个 HTTP 请求和 1 个 HTTP 响应

  总共捕获了 82 个 TCP 连接。

## ✧ Part Four

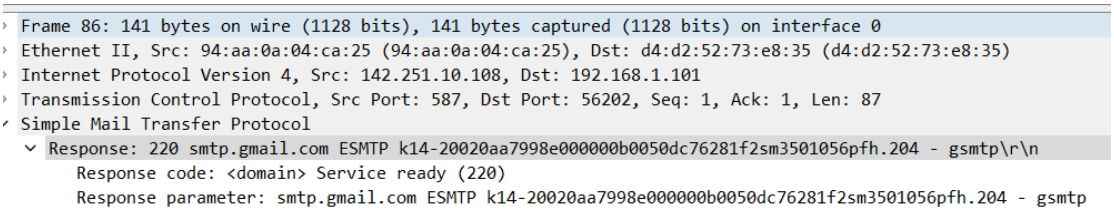
- **打开邮件客户端 Foxmail 或 Outlook，写一封电子邮件（建议采用直接送达方式），并捕获这次的数据包。捕获到的数据包由几层协议构成？分别是什么协议？标出数据包的源和目标 IP 地址、源和目标端口。**




捕获到的数据包由 4 层协议构成，分别是以太网、IPv4、TCP、SMTP

Source : 142.251.10.108 ; Destination : 192.168.1.101

Source Port : 587 ; Destination Port : 56202




- **跟踪 TCP 流，查看 SMTP 握手消息采用的是什么（HELO 还是 EHLO）？标出 SMTP 协议层中的客户端机器名、发件人地址、收件人地址、认证的用户名和密码（如果是 EHLO 握手方式）、邮件正文（内容过长可截取关键部分）。**

由于我的 ISP 阻挡了 port 25，所以捕获到的邮件是不允许不加密的，因此 smtp 服务器只能使用 port587(TLS 加密)，大部分的内容都是截图内容都是加密的。

```
220 smtp.gmail.com ESMTP k14-20020aa7998e000000b0050dc76281f2sm3501056pfh.204 - gsmtp
EHLO Peter
250-smtp.gmail.com at your service, [60.53.33.23]
250-SIZE 35882577
250-8BITMIME
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMTPUTF8
STARTTLS
220 2.0.0 Ready to start TLS
....6...2...=E.Yl.)yT.......\.=..X2u.l..A~....0.,.(.$...
.........k.j.i.h.9.8.7.6..........2...*.&.....=.5.../.+.'.#...      .........g.@.?.>.
%.......<./.A.....................................
...
...
.....U.........
.....................
.....     .
.#...
. .....;...7..b.....=.....Q..J.`......DOWNGRD...+...
0...0..p.................
U."..(.0
.        *.H..
.....0F1.0        ..U....US1"0 ..U.
..Google Trust Services LLC1.0...U...
GTS CA 1C30..
220504171250Z.
```

SMTP 握手消息采用的是 EHLO。由于捕获到的数据包都是通过加密的，所以没办法看到发件人地址、收件人地址、认证的用户名和密码、邮件正文等，只能我手动添加这些内容了。

客户端机器名：Peter

发件人地址：westcircle888@gmail.com、

收件人地址：westcircle@163.com

认证用户名和密码：AUTH LOGIN

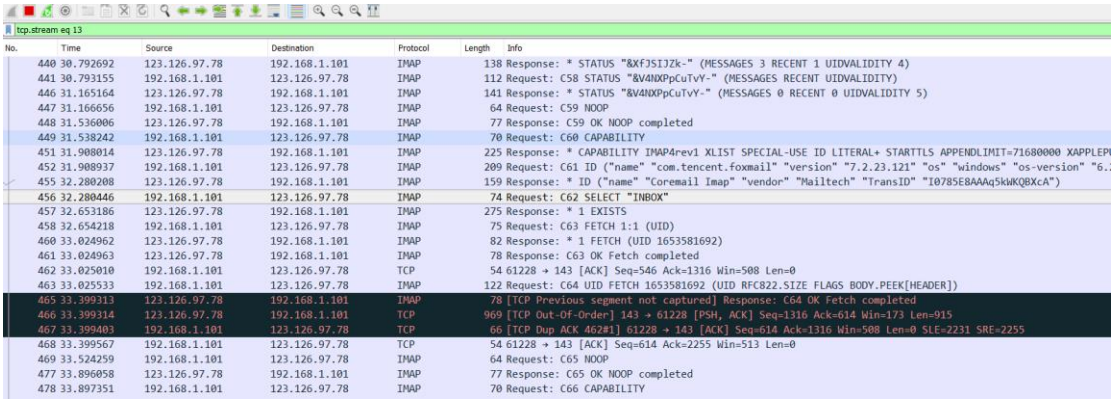
邮件正文：Lab6


- **打开邮件客户端 Foxmail 或 Outlook，收取自己邮箱中的邮件（请在邮件服务器中设置允许 POP3 或者 IMAP），并捕获这次的数据包。捕获到的数据包由几层协议构成？分别是什么协议？标出数据包的源和目标 IP 地址、源和目标端口。**

在邮件服务器中设置了 IMAP，使用其中一个邮箱发送邮件给另一个邮箱，并在另一个邮箱中收取邮件。捕获到的数据包是 IMAP，由 4 层协议构成，分别是以太网、IPv4、TCP、IMAP

Source Port : 61228 ; Destination Port : 143

Source : 192.168.1.101 ; Destination : 123.126.97.78



tcp.stream eq 13

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 440 | 30.792692 | 123.126.97.78 | 192.168.1.101 | IMAP | 138 | Response: * STATUS "&XfJSIJZk-" (MESSAGES 3 RECENT 1 UIDVALIDITY 4) |
| 441 | 30.793155 | 192.168.1.101 | 123.126.97.78 | IMAP | 112 | Request: C58 STATUS "&V4NXPpCuTvY-" (MESSAGES RECENT UIDVALIDITY) |
| 446 | 31.165164 | 123.126.97.78 | 192.168.1.101 | IMAP | 141 | Response: * STATUS "&V4NXPpCuTvY-" (MESSAGES 0 RECENT 0 UIDVALIDITY 5) |
| 447 | 31.166656 | 192.168.1.101 | 123.126.97.78 | IMAP | 64 | Request: C59 NOOP |
| 448 | 31.536006 | 123.126.97.78 | 192.168.1.101 | IMAP | 77 | Response: C59 OK NOOP completed |
| 449 | 31.538242 | 192.168.1.101 | 123.126.97.78 | IMAP | 70 | Request: C60 CAPABILITY |
| 451 | 31.908014 | 123.126.97.78 | 192.168.1.101 | IMAP | 225 | Response: * CAPABILITY IMAP4rev1 XLIST SPECIAL-USE ID LITERAL+ STARTTLS APPENDLIMIT=71680000 XAPPLEP |
| 452 | 31.908937 | 192.168.1.101 | 123.126.97.78 | IMAP | 209 | Request: C61 ID ("name" "com.tencent.foxmail" "version" "7.2.23.121" "os" "windows" "os-version" "6. |
| 455 | 32.280208 | 123.126.97.78 | 192.168.1.101 | IMAP | 159 | Response: * ID ("name" "Coremail Imap" "vendor" "Mailtech" "TransID" "I0785E8AAAq5kWKQBXcA") |
| 456 | 32.280446 | 192.168.1.101 | 123.126.97.78 | IMAP | 74 | Request: C62 SELECT "INBOX" |
| 457 | 32.653186 | 123.126.97.78 | 192.168.1.101 | IMAP | 275 | Response: * 1 EXISTS |
| 458 | 32.654218 | 192.168.1.101 | 123.126.97.78 | IMAP | 75 | Request: C63 FETCH 1:1 (UID) |
| 460 | 33.024962 | 123.126.97.78 | 192.168.1.101 | IMAP | 82 | Response: * 1 FETCH (UID 1653581692) |
| 461 | 33.024963 | 123.126.97.78 | 192.168.1.101 | IMAP | 78 | Response: C63 OK Fetch completed |
| 462 | 33.025010 | 192.168.1.101 | 123.126.97.78 | TCP | 54 | 61228 → 143 [ACK] Seq=546 Ack=1316 Win=508 Len=0 |
| 463 | 33.025533 | 192.168.1.101 | 123.126.97.78 | IMAP | 122 | Request: C64 UID FETCH 1653581692 (UID RFC822.SIZE FLAGS BODY.PEEK[HEADER]) |
| 465 | 33.399313 | 123.126.97.78 | 192.168.1.101 | IMAP | 78 | [TCP Previous segment not captured] Response: C64 OK Fetch completed |
| 466 | 33.399314 | 123.126.97.78 | 192.168.1.101 | TCP | 969 | [TCP Out-Of-Order] 143 → 61228 [PSH, ACK] Seq=1316 Ack=614 Win=173 Len=915 |
| 467 | 33.399403 | 192.168.1.101 | 123.126.97.78 | TCP | 66 | [TCP Dup ACK 462#1] 61228 → 143 [ACK] Seq=614 Ack=1316 Win=508 Len=0 SLE=2231 SRE=2255 |
| 468 | 33.399567 | 192.168.1.101 | 123.126.97.78 | TCP | 54 | 61228 → 143 [ACK] Seq=614 Ack=2255 Win=513 Len=0 |
| 469 | 33.524259 | 192.168.1.101 | 123.126.97.78 | IMAP | 64 | Request: C65 NOOP |
| 477 | 33.896058 | 123.126.97.78 | 192.168.1.101 | IMAP | 77 | Response: C65 OK NOOP completed |
| 478 | 33.897351 | 192.168.1.101 | 123.126.97.78 | IMAP | 70 | Request: C66 CAPABILITY |

```
> Frame 456: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
> Ethernet II, Src: d4:d2:52:73:e8:35 (d4:d2:52:73:e8:35), Dst: 94:aa:0a:04:ca:25 (94:aa:0a:04:ca:25)
> Internet Protocol Version 4, Src: 192.168.1.101, Dst: 123.126.97.78
> Transmission Control Protocol, Src Port: 61228, Dst Port: 143, Seq: 505, Ack: 1043, Len: 20
v Internet Message Access Protocol
  v Line: C62 SELECT "INBOX"\r\n
      Request Tag: C62
      Request Command: SELECT
      Request Folder: "INBOX"
      Request: SELECT "INBOX"
```

- **跟踪 TCP 流，标出 POP3 或 IMAP 协议层中的认证用户名和密码、以及接收的邮件正文（内容过长可截取关键部分）。**

  UID 1653581692，Received from : Peter(我的机器名)、From : westcircle@163.com、To : westcircle8@163.com（收件人邮箱），邮件正文：lab6

```
C60 CAPABILITY
* CAPABILITY IMAP4rev1 XLIST SPECIAL-USE ID LITERAL+ STARTTLS APPENDLIMIT=71680000 XAPPLEPUSHSERVICE UIDPLUS X-CM-EXT-1 SASL-IR
AUTH=XOAUTH2
C60 OK CAPABILITY completed
C61 ID ("name" "com.tencent.foxmail" "version" "7.2.23.121" "os" "windows" "os-version" "6.2" "vendor" "tencent limited" "contact"
"foxmail@foxmail.com")
* ID ("name" "Coremail Imap" "vendor" "Mailtech" "TransID" "I0785E8AAAq5kWKQBXcA")
C61 OK ID completed
C62 SELECT "INBOX"
* 1 EXISTS
* 1 RECENT
* OK [UIDVALIDITY 1] UIDs valid
* FLAGS (\Answered \Seen \Deleted \Draft \Flagged)
* OK [PERMANENTFLAGS (\Answered \Seen \Deleted \Draft \Flagged)] Limited
C62 OK [READ-WRITE] SELECT completed
C63 FETCH 1:1 (UID)
* 1 FETCH (UID 1653581692)
C63 OK Fetch completed
C64 UID FETCH 1653581692 (UID RFC822.SIZE FLAGS BODY.PEEK[HEADER])
* 1 FETCH (UID 1653581692 FLAGS () RFC822.SIZE 1768 BODY[HEADER] {840}
Received: from Peter (unknown [60.53.33.23])
        by smtp5 (Coremail) with SMTP id HdxpCgDHr0hAupFiapG+Eg--.3604S2;
        Sat, 28 May 2022 13:59:31 +0800 (CST)
Date: Sat, 28 May 2022 13:59:30 +0800
From: "westcircle@163.com" <westcircle@163.com>
To: westcircle8 <westcircle8@163.com>
Subject: lab6
X-Priority: 3
X-GUID: D01DC4D2-E5C8-47CE-8F5F-0A298B781D4A
X-Has-Attach: no
X-Mailer: Foxmail 7.2.23.121[cn]
Mime-Version: 1.0
Message-ID: <2022052813592665780 82@163.com>
Content-Type: multipart/alternative;
        boundary="----=_001_NextPart608280284834_=----"
X-CM-TRANSID:HdxpCgDHr0hAupFiapG+Eg--.3604S2
X-Coremail-Antispam: 1Uf129KBjDUn29KB7ZKAUJUUUUU529EdanIXcx71UUUUU7v73
        VFW2AGmfu7bjvjm3AaLaJ3UbIYCTnIWIevJa73UjIFyTuYvjxUINeoUUUUU
X-Originating-IP: [60.53.33.23]
X-CM-SenderInfo: xzhv3uxlufzvi6rwjhhfrp/1tbiNwgPrFWBoJBV4wABsP
```

## ✧ Part Five

本部分需要边操作，边捕获，请在每次操作后暂停捕获，或者使用过滤器。建议通过 FTP 命令行进行实验，也可以使用 FTP 图形客户端。

- **运行 FTP xxx.com 命令，连接并登录服务器，输入用户名和帐号（如果是免费服务器，可以使用匿名帐号 Anonymous，密码是任意的邮箱）。捕获到的数据包由几层协议构成？分别是什么协议？标出数据包的源和目标 IP 地址、源和目标端口。**

  在网上找的公用的 ftp 服务器。

```
PS C:\Users\ASUS> ftp
ftp> open ftp.acc.umu.se
Connected to ftp.acc.umu.se.
220 Please use http://ftp.acc.umu.se/ whenever possible.
200 Always in UTF8 mode.
User (ftp.acc.umu.se:(none)): anonymous
331 Please specify the password.
Password:
230 Login successful.
```

ip.addr == 194.71.11.173

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 320 | 13.153368 | 192.168.1.101 | 194.71.11.173 | TCP | 66 | 52594 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=1 SACK_PERM=1 |
| 324 | 13.466186 | 194.71.11.173 | 192.168.1.101 | TCP | 58 | 21 → 52594 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1460 |
| 325 | 13.466297 | 192.168.1.101 | 194.71.11.173 | TCP | 54 | 52594 → 21 [ACK] Seq=1 Ack=1 Win=8192 Len=0 |
| 334 | 13.733895 | 194.71.11.173 | 192.168.1.101 | FTP | 112 | Response: 220 Please use http://ftp.acc.umu.se/ whenever possible. |
| 335 | 13.739443 | 192.168.1.101 | 194.71.11.173 | FTP | 68 | Request: OPTS UTF8 ON |
| 338 | 13.785990 | 194.71.11.173 | 192.168.1.101 | TCP | 54 | 21 → 52594 [ACK] Seq=59 Ack=15 Win=32106 Len=0 |
| 342 | 14.068829 | 194.71.11.173 | 192.168.1.101 | FTP | 80 | Response: 200 Always in UTF8 mode. |
| 344 | 14.122702 | 192.168.1.101 | 194.71.11.173 | TCP | 54 | 52594 → 21 [ACK] Seq=15 Ack=85 Win=8108 Len=0 |
| 394 | 16.291402 | 192.168.1.101 | 194.71.11.173 | FTP | 70 | Request: USER anonymous |
| 396 | 16.311980 | 194.71.11.173 | 192.168.1.101 | TCP | 54 | 21 → 52594 [ACK] Seq=85 Ack=31 Win=32104 Len=0 |
| 400 | 16.542157 | 194.71.11.173 | 192.168.1.101 | FTP | 88 | Response: 331 Please specify the password. |
| 407 | 16.596022 | 192.168.1.101 | 194.71.11.173 | TCP | 54 | 52594 → 21 [ACK] Seq=31 Ack=119 Win=8074 Len=0 |
| 447 | 18.627690 | 192.168.1.101 | 194.71.11.173 | FTP | 70 | Request: PASS anonymous |
| 448 | 18.636298 | 194.71.11.173 | 192.168.1.101 | TCP | 54 | 21 → 52594 [ACK] Seq=119 Ack=47 Win=32104 Len=0 |
| 456 | 18.868624 | 194.71.11.173 | 192.168.1.101 | FTP | 77 | Response: 230 Login successful. |
| 458 | 18.916607 | 192.168.1.101 | 194.71.11.173 | TCP | 54 | 52594 → 21 [ACK] Seq=47 Ack=142 Win=8051 Len=0 |
| 645 | 26.514650 | 192.168.1.101 | 194.71.11.173 | FTP | 82 | Request: PORT 192,168,1,101,205,140 |
| 649 | 26.568034 | 194.71.11.173 | 192.168.1.101 | TCP | 54 | 21 → 52594 [ACK] Seq=142 Ack=75 Win=32092 Len=0 |
| 656 | 26.798342 | 194.71.11.173 | 192.168.1.101 | FTP | 105 | Response: 200 PORT command successful. Consider using PASV. |
| 657 | 26.803518 | 192.168.1.101 | 194.71.11.173 | FTP | 60 | Request: NLST |
| 658 | 26.812258 | 194.71.11.173 | 192.168.1.101 | TCP | 54 | 21 → 52594 [ACK] Seq=193 Ack=81 Win=32114 Len=0 |
| 662 | 27.046005 | 194.71.11.173 | 192.168.1.101 | TCP | 74 | 36555 → 52620 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval= |
| 664 | 27.046205 | 192.168.1.101 | 194.71.11.173 | TCP | 74 | 52620 → 36555 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 |

```
> Internet Protocol Version 4, Src: 194.71.11.173, Dst: 192.168.1.101
v Transmission Control Protocol, Src Port: 21, Dst Port: 52594, Seq: 1, Ack: 1, Len: 58
    Source Port: 21
    Destination Port: 52594
    [Stream index: 18]
    [TCP Segment Len: 58]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2521361115
    [Next Sequence Number: 59    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1651787642
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    Window: 32120
    [Calculated window size: 32120]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0xadcf [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (58 bytes)
```

TCP3 次握手后，与 ftp 服务器建立连接后，FTP 服务器回应数据包，其中由 4 层协议构成，分别是以太网、IPv4、TCP、FTP。

Source : 194.71.11.173 ; Destination : 192.168.1.101

Source Port : 21 ; Destionation Port : 52594

- **跟踪 TCP 流，标注客户端发出的登录命令、用户名、密码以及服务器的响应。**

```
220 Please use http://ftp.acc.umu.se/ whenever possible.
OPTS UTF8 ON
200 Always in UTF8 mode.
USER anonymous
331 Please specify the password.
PASS anonymous
230 Login successful.
PORT 192,168,1,101,205,140
200 PORT command successful. Consider using PASV.
```

- **执行列目录操作(ls)，在新捕获的数据包中跟踪 TCP 流，标注客户端发出的命令、以及服务器的响应。查看是否建立了一个新的 TCP 连接，跟踪该连接的 TCP 流。建议连接校内服务器，如果服务器在校外，可能需要先执行 passive 命令（下同）。**

```
PORT 192,168,1,101,205,140
200 PORT command successful. Consider using PASV.
NLST
150 Here comes the directory listing.
226 Directory send OK.
```

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
HEADER.html
Public
about
cdimage
conspiracy
debian
debian-cd
favicon.ico
images
mirror
pub
releases
robots.txt
tails
ubuntu
226 Directory send OK.
ftp: 142 bytes received in 0.01Seconds 20.29Kbytes/sec.
```

- 执行更换目录操作（**cd**），在新捕获的数据包中跟踪 **TCP** 流，标注客户端发出的命令、以及服务器的响应。

```
CWD about
250 Directory successfully changed.
PORT 192,168,1,101,205,160
200 PORT command successful. Consider using PASV.
NLST
150 Here comes the directory listing.
226 Directory send OK.
```

```
ftp> cd about
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
ftp-about-SPmkII.html
ftp.sunet.se-history_sv.html
graph.png
index.html
largefile.dot
largefile.png
largefile.svg
smallcachedfile.dot
smallcachedfile.png
smallcachedfile.svg
smalluncachedfile.dot
smalluncachedfile.png
smalluncachedfile.svg
zfs-stats
226 Directory send OK.
ftp: 267 bytes received in 0.01Seconds 26.70Kbytes/sec.
```

- 执行下载文件操作（**get filename**），如果是二进制文件，先执行 **binary** 命令。在新捕获的数据包中跟踪 **TCP** 流，标注客户端发出的命令、以及服务器的响应。查看是否建立了一个新的 **TCP** 连接，跟踪该连接的 **TCP** 流（内容较长时截取部分关键内容）。

```
PORT 192,168,1,101,205,174
200 PORT command successful. Consider using PASV.
RETR graph.png
150 Opening BINARY mode data connection for graph.png (2550 bytes).
226 Transfer complete.
QUIT
221 Goodbye.
```

```
ftp> get graph.png
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for graph.png (2550 bytes).
226 Transfer complete.
ftp: 2550 bytes received in 0.00Seconds 2550.00Kbytes/sec.
ftp> quit
221 Goodbye.
```

## 六、 实验结果与分析

根据你看到的数据包，分别解答以下协议的问题（看完请删除本句）：

- Ping 发送的是什么类型的协议数据包？什么时候会出现 ARP 消息？Ping 一个域名和 Ping 一个 IP 地址出现的数据包有什么不同？

  答：Ping 发送的是 ICMP 协议的数据包。当我们向一个 IP 地址发送数据的时候就需要用到 ARP 缓存表，这个表记录着 IP 地址与 MAC 地址的映射关系可查询 MAC 地址。但是当缓存表中没有这样的记录时，就会发送一个 ARP 请求给该 IP 所对应的 MAC 地址。Ping 一个域名时还需要发送 DNS 协议请求数据来获得对应的 IP 地址。

- Tracert/Traceroute 发送的是什么类型的协议数据包，整个路由跟踪过程是如何进行的？

  答：ICMP 类型的协议数据包。通过 TTL 字段递增来实现的，发送 TTL 字段为 1 的时候到达第一跳的路由器数据包超时时，本机就可以受到 ICMP Error 的信息知道路由器的地址，字段为 2 的第二跳也超时，同样收到错误信息，以此类推。

- 建立 TCP 连接的数据包由几个构成？各自的 SYN 和 ACK 标志字段是什么？

  答：由三个数据包构成。第一次 SYN=1(SET)，第二次 SYN=1(SET)、ACK=1(SET)，第三次 ACK=1(SET)

- 浏览器打开一个网页，可能会看到多个 TCP 连接，多次 HTTP 会话。一个 TCP 连接上是否会存在多个 HTTP 会话？什么情况下会出现 DNS 数据包？

  答：会存在多个 HTTP 会话。当使用域名访问网页的时候，而本地又没有相关信息就会出现 DNS 数据包。

- 邮件客户端发送一封电子邮件，需要几次请求、响应消息的交互？消息的一般格式是什么？邮件正文结束的标记是什么？

  使用 smtp 协议发送邮件给服务器时需要 6 次请求、响应消息的交互。

  1. 使用 EHLO 命令与连接上的 smtp 服务器“打招呼”
  2. 使用“AUTH LOGIN”登录到 smtp 服务器
  3. 指明发件人和收件人 mail from : xxx, rcpt to : xxx
  4. 输入 data 命令，“data”命令表示准备开始输入邮件内容，其邮件格式为，邮件头包含 from、to 和 subject（主题），邮件正文内容在下方，还有其他的内容。
  5. 直遇到“.”告诉邮件服务器邮件内容已经写完
  6. 输入 quit 命令断开与邮件服务器的连接。

- 邮件客户端接收一封电子邮件，需要几次请求、响应消息的交互？消息的一般格式是什么？用户名和密码是否经过了加密处理？

  答：邮件客户端接收一封邮件大致过程：

  1. Login
  2. Select Inbox
  3. Search New
  4. Fetch
  5. Logout

  Received : from 机器名

  by smtp5 (Coremail) with SMTP id HdxpCgDHr0hAupFiapG+Eg--.3604S2;

  Sat, 28 May 2022 13:59:31 +0800 (CST)

  Date: Sat, 28 May 2022 13:59:30 +0800

  From: "xxx@163.com" <xxx@163.com>

  To: xxxx<xxxx@163.com>

  Subject: 正文内容

  X-Priority: 3

  X-GUID: 。。。

  X-Has-Attach: no

  X-Mailer: Foxmail 7.2.23.121[cn]

Mime-Version: 1.0

Message-ID: 。。。

Content-Type: multipart/alternative;

boundary="----=_001_NextPart608280284834_=----"

X-CM-TRANSID:。。。

X-Coremail-Antispam: 。。。

X-Originating-IP: 。。。

X-CM-SenderInfo: 。。。

用户名和密码均经过了加密处理。

- 登录 FTP 服务器时，会产生几个 TCP 连接？列目录和上传或者下载文件时，会产生几个 TCP 连接？

  答：登录时产生一个控制 TCP 连接，之后列目录又产生一次连接，下载文件时也产生一个连接。

## 七、 讨论、心得

这次的实验对我来说算是相当困难因为一直在实验过程中遇到很多的麻烦，花费了 2 天的时间一直在找问题所在，尤其是 Part4 的部分，SMTP 的数据包一直捕获不到，在 Foxmail、Outlook、163 等邮件服务器找了很多相关问题和解决方案都找不了，最后发现是 SMTP 协议是不能加密 SSL 的，如果加密了就不能捕获到了，但是不使用 SSL 的话，就只能使用 port 25 或 587，而 port 25 又被我的 ISP 所阻挡了所以也不能用 port 25 发邮件，587 则是使用 TLS 加密，虽然能被捕获到但是追踪 TCP 流的时候，内容都是加密的，导致我只能在网上找一些案列来看其流程。这次的实验真的耗费了我很多的精力和时间来完成，在这次的实验过程中也学到了很多的知识点，加深了我对计算机网络的认知，虽然过程很艰辛但是收获还是很多的。