北京郵電大學

实验报告

题目：IP 和 TCP 数据分组的捕获和解析

学院：计算机学院（国家示范性软件学院）

专业：计算机科学与技术

班级：2022211305

学号：2022211683

姓名：张晨阳

2024 年 6 月 5 号

目录

1.实验内容和实验环境描述

1.1 实验任务和实验目的

本次实验内容：

1) 捕获在使用网络过程中产生的分组（packet）：IP 数据包、ICMP 报文、DHCP 报文、TCP 报文段。
2) 分析各种分组的格式，说明各种分组在建立网络连接和通信过程中的作用。
3) 分析 IP 数据报分片（片段）的结构：理解长度大于 1500 字节 IP 数据报分片传输的结构
4) 分析 TCP 建立连接、拆除连接和数据通信的过程。

1.2 实验环境

1) 操作系统：Windows 11
2) 协议分析软件：Wireshark 4.2.5
3) 无线网卡：Realtek RTL8852BE WiFi 6 802.11 ax PCIe Adapter
4) 网卡物理地址：9C:2F:9D:91:D5:99

2.实验步骤和协议分析

2.1 实验准备

1) 阅读实验二的指导书，下载 Wireshark 的正确版本，下载 WinPcap 用于抓包；
2) 重启电脑以保证不运行其他网络应用程序，便于查找数据包和分析；
3) 运行 Wireshark。

2.2 捕获和分析 DHCP 报文

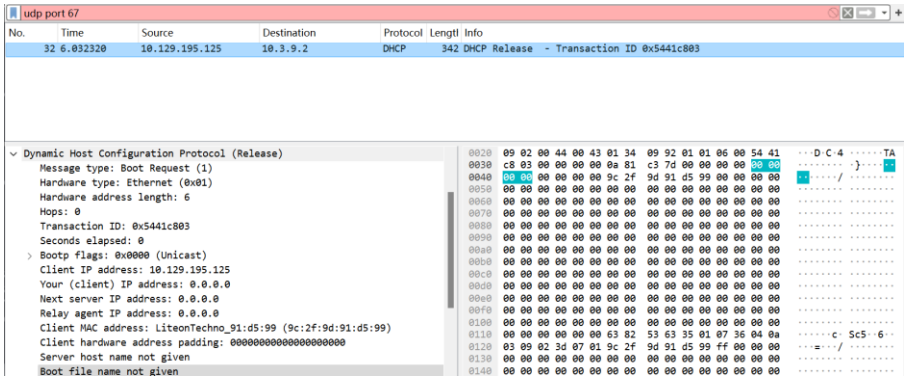
2.2.1 设置捕获过滤器，释放主机 IP 地址

设置捕获过滤器：udp port 67，在终端执行命令 ipconfig /release：

```
C:\Users\SevenGrass>ipconfig /release

Windows IP Configuration

No operation can be performed on 以太网 while it has its media disconnected.
No operation can be performed on 本地连接* 1 while it has its media disconnected.
No operation can be performed on 本地连接* 2 while it has its media disconnected.

Ethernet adapter 以太网:

   Media State . . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Unknown adapter 本地连接:

   Media State . . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

DHCP 1 释放 IP 命令

此时捕获到的 DHCP 如下图：



DHCP 2 报文 Release

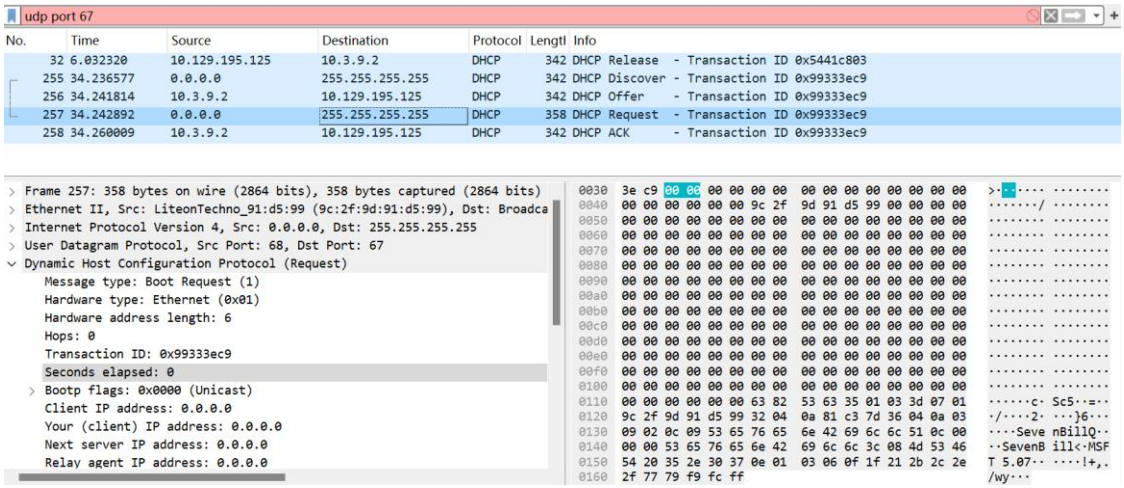
2.2.2 重新分配 IP 地址，捕获 DHCP

在终端执行命令 ipconfig /renew：

```
C:\Users\SevenGrass>ipconfig /renew

Windows IP Configuration

No operation can be performed on 以太网 while it has its media disconnected.
No operation can be performed on 本地连接 while it has its media disconnected.
No operation can be performed on 本地连接* 1 while it has its media disconnected.
No operation can be performed on 本地连接* 2 while it has its media disconnected.

Ethernet adapter 以太网:

   Media State . . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Unknown adapter 本地连接:

   Media State . . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter 本地连接* 1:
```

DHCP 3 重获 IP 命令

此时捕获到的 DHCP 如下：



DHCP 4 报文 Discover、Offer、Request、ACK

2.2.3 DHCP 报文及协议分析

1) DHCP 简介：

DHCP，全称是 Dynamic Host Configuration Protocol，即动态主机配置协议。它是一种网络管理协议，用于自动分配 IP 地址和其他网络配置参数给网络中的设备，如计算机、打印机、路由器等。

2) 实验使用端口分析：

本实验使用的捕获过滤器 **udp port 67** 是指 DHCP 服务器监听的端口。

DHCP 服务器使用这个端口来接收来自客户端的 DHCP 请求消息。当客户端设备需要 IP 地址或其他网络配置信息时，它会通过 UDP 端口 67 向 DHCP 服务器发送请求。

同样的还有 **udp port 68**，这是 DHCP 客户端监听的端口。

客户端设备使用这个端口来接收来自 DHCP 服务器的 DHCP 响应消息。当 DHCP 服务器接收到客户端的请求后，它会通过 UDP 端口 68 发送包含 IP 地址和网络配置信息的响应。

上述两个端口都是 UDP 协议的，因为 DHCP 协议是基于 UDP 的。

3) DHCP 报文格式分析：

经查阅资料，DHCP 报文格式如下表：

0	7	15	23	31
Op	Htype	Hlen	Hops	
Transaction ID				
Secs		Flags		
Client IP addr				
Your IP addr				
Server IP addr				
Gateway IP addr				
Client hardware addr				
Server name				
file				
Options				

DHCP 5 **报文格式**

其中各字段的含义如下：

(1) Op：表示报文的类型。1 表示客户端请求报文；2 表示服务器响应报文；

(2) Htype：表示硬件地址的类型。1 表示以太网 MAC 地址类型；

(3) Hlen：表示客户端的 MAC 地址长度。对于以太网，该值为 6；

(4) Hops：表示跳数。

(5) Transaction ID：由客户端选择的一个随机数，被服务器和客户端用来在它们之间交流请求和响应，客户端用它对请求和应答进行匹配。

(6) Secs：表示从客户端开始获得 IP 地址或 IP 地址续借后所使用了的秒数

(7) Flags：表示标志字段。只有最高位有意义，0 表示采用单播发送方式，1 表示采用广播发送方式其余位置 0.

(8) Cilent IP address：客户端的 IP 地址，仅在服务器发送的 ACK 报文中显示，因为在得到服务器确认前，客户端还没有分配到 IP 地址。

(9) Your IP address：服务器分配给客户端的 IP 地址，仅在服务器发送的 Offer 和 ACK 报文中显示，其他报文中显示为 0。

(10) **Server IP address**：表明 DHCP 协议流程的下一个阶段要使用的服务器的 IP 地址。仅在 Offer 和 ACK 报文中显示，其他报文中显示为 0。

(11) **Gateway IP address**：表示客户端发出请求报文后经过的第一个中继的 IP 地址。如果没有经过中继，则显示为 0。

(12) **Client hardware address**：表示客户端的 MAC 地址，在每个报文中都会显示对应客户端的 MAC 地址

(13) **Server name**：表示客户端获取配置信息的服务器名字。仅在 Offer 和 ACK 报文中显示发送报文的服务器名称，其他报文显示为 0。

(14) **file**：表示客户端的启动配置文件名，仅在 Offer 报文中显示，其他报文中显示为空

(15) **Options**：可选项字段，长度可变，格式为“代码 + 长度 + 数据”

4) **DHCP 常见数据包分类：**

(1) **DISCOVER**：客户端发送的第一个数据包，用于发现网络中的 DHCP 服务器。

(2) **OFFER**：当 DHCP 服务器接收到 DISCOVER 广播后，它会发送一个 OFFER 包作为响应。这个包包含一个 IP 地址和相关的网络配置信息，如子网掩码、默认网关、DNS 服务器等。

(3) **REQUEST**：客户端可以选择一个 OFFER 包，并发送一个 REQUEST 包来请求分配该 IP 地址。这个请求可以发送给最初提供提议的服务器，也可以发送给其他服务器，请求分配一个新的地址。

(4) **DECLINE**：如果客户端发现分配的 IP 地址已经被使用，或者配置信息有问题，它会发送一个 DECLINE 包，拒绝接受该地址。

(5) **ACK**：当 DHCP 服务器接收到 REQUEST 包后，如果同意分配请求的 IP 地址，它会发送一个 ACK 包，确认分配的 IP 地址和网络配置信息。

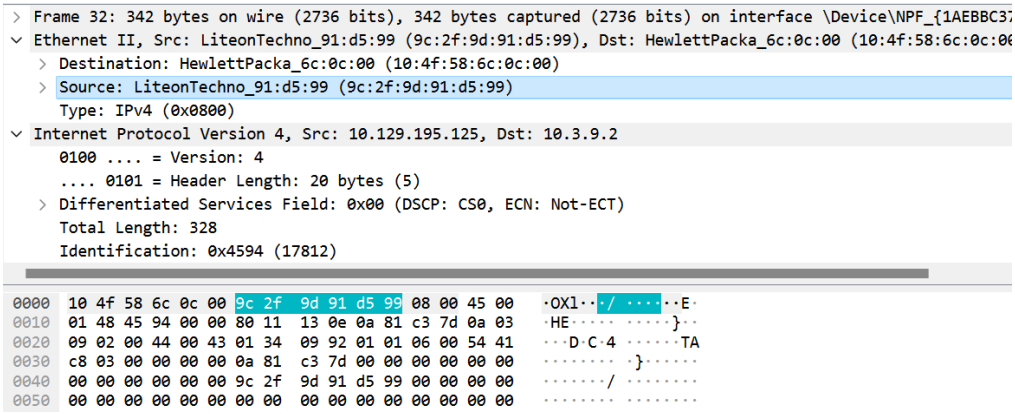
(6) **NAK**：如果 DHCP 服务器无法满足客户端的请求，例如，如果请求的 IP 地址已经被占用，它会发送一个 NAK 包，表示请求失败。

(7) **RELEASE**：当客户端不再需要分配的 IP 地址，或者设备即将关机或重启时，它会发送一个 RELEASE 包，释放当前的 IP 地址。

5) 所捕获的 DHCP 报文分析及 IP 地址分配过程

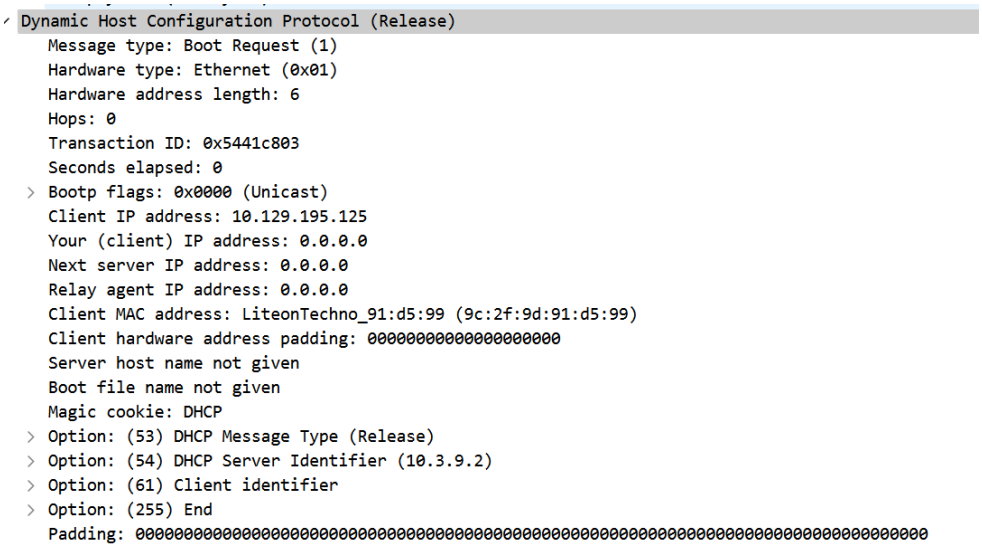
如上图 DHCP 2，当我们释放 IP 地址时，发送了一个长度为 342 字节的 Release 包。

在 Wireshark 中我们可以发现：其源地址为本机无线网卡的物理地址。



DHCP 6 Release **包源地址分析**

按照上面所讲解的 DHCP 的报文格式，我在 Wireshark 里找到了对应的取值如下：



DHCP 7 Release **包字段内容**

各字段的具体取值含义可参考前文的介绍，不多赘述。此时我们释放了当前的 IP 地址。

接下来当我们重新申请分配 IP 地址时，捕获了四个 DHCP 报文，见上图 DHCP 4，可以发现依次是 Discover、Offer、Request、ACK。

这四种包的含义在上述分类中已介绍，故直接在 Wireshark 中查看它们各字段的内容如下图：

```
Dynamic Host Configuration Protocol (Discover)
   Message type: Boot Request (1)
   Hardware type: Ethernet (0x01)
   Hardware address length: 6
   Hops: 0
   Transaction ID: 0x99333ec9
   Seconds elapsed: 0
 > Bootp flags: 0x0000 (Unicast)
   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: LiteonTechno_91:d5:99 (9c:2f:9d:91:d5:99)
   Client hardware address padding: 00000000000000000000
   Server host name not given
   Boot file name not given
   Magic cookie: DHCP
 > Option: (53) DHCP Message Type (Discover)
 > Option: (61) Client identifier
 > Option: (50) Requested IP Address (10.129.195.125)
 > Option: (12) Host Name
 > Option: (60) Vendor class identifier
 > Option: (55) Parameter Request List
 > Option: (255) End
   Padding: 00000000
```

DHCP 8 Discover **字段内容**

```
Dynamic Host Configuration Protocol (Offer)
   Message type: Boot Reply (2)
   Hardware type: Ethernet (0x01)
   Hardware address length: 6
   Hops: 1
   Transaction ID: 0x99333ec9
   Seconds elapsed: 0
 > Bootp flags: 0x0000 (Unicast)
   Client IP address: 0.0.0.0
   Your (client) IP address: 10.129.195.125
   Next server IP address: 0.0.0.0
   Relay agent IP address: 10.129.0.1
   Client MAC address: LiteonTechno_91:d5:99 (9c:2f:9d:91:d5:99)
   Client hardware address padding: 00000000000000000000
   Server host name not given
   Boot file name not given
   Magic cookie: DHCP
 > Option: (53) DHCP Message Type (Offer)
 > Option: (54) DHCP Server Identifier (10.3.9.2)
 > Option: (51) IP Address Lease Time
 > Option: (1) Subnet Mask (255.255.0.0)
 > Option: (3) Router
 > Option: (6) Domain Name Server
 > Option: (255) End
   Padding: 00000000000000000000000000000000000000
```

DHCP 9 Offer **字段内容**

```
Dynamic Host Configuration Protocol (Request)
   Message type: Boot Request (1)
   Hardware type: Ethernet (0x01)
   Hardware address length: 6
   Hops: 0
   Transaction ID: 0x99333ec9
   Seconds elapsed: 0
 > Bootp flags: 0x0000 (Unicast)
   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: LiteonTechno_91:d5:99 (9c:2f:9d:91:d5:99)
   Client hardware address padding: 00000000000000000000
   Server host name not given
   Boot file name not given
   Magic cookie: DHCP
 > Option: (53) DHCP Message Type (Request)
 > Option: (61) Client identifier
 > Option: (50) Requested IP Address (10.129.195.125)
 > Option: (54) DHCP Server Identifier (10.3.9.2)
 > Option: (12) Host Name
 > Option: (81) Client Fully Qualified Domain Name
 > Option: (60) Vendor class identifier
 > Option: (55) Parameter Request List
 > Option: (255) End
```

DHCP 10 Request **字段内容**

```
Dynamic Host Configuration Protocol (ACK)
   Message type: Boot Reply (2)
   Hardware type: Ethernet (0x01)
   Hardware address length: 6
   Hops: 1
   Transaction ID: 0x99333ec9
   Seconds elapsed: 0
 > Bootp flags: 0x0000 (Unicast)
   Client IP address: 0.0.0.0
   Your (client) IP address: 10.129.195.125
   Next server IP address: 0.0.0.0
   Relay agent IP address: 10.129.0.1
   Client MAC address: LiteonTechno_91:d5:99 (9c:2f:9d:91:d5:99)
   Client hardware address padding: 00000000000000000000
   Server host name not given
   Boot file name not given
   Magic cookie: DHCP
 > Option: (53) DHCP Message Type (ACK)
 > Option: (54) DHCP Server Identifier (10.3.9.2)
 > Option: (51) IP Address Lease Time
 > Option: (1) Subnet Mask (255.255.0.0)
 > Option: (3) Router
 > Option: (6) Domain Name Server
 > Option: (255) End
   Padding: 00000000000000000000000000000000000000
```

DHCP 11 ACK **字段内容**

从上图我们可以总结出 IP 地址分配的全过程：

首先，客户端广播一个 DHCP Discover 报文，该报文向服务端请求一个 IP 地址，为了尽可能维持网络的稳定，客户端会尽可能要求获得和之前一样的 IP 地址；

其次，服务器发送一个 DHCP Offer 报文。该报文表示服务器给客户端提供了一个可供使用的 IP 地址，包括该 IP 地址租约的时长等信息，但最终是否使用仍需要客户端自己决定；

接着，客户端发送一个 DHCP Request 报文向服务器请求使用这个 IP 地址，该报文仍然是广播的；

最后，服务器回应 DHCP ACK 报文，表示确认成功使用该 IP 地址。

6) DHCP 协议总结

DHCP 结合 IP 和 UDP 进行工作，主要工作流程由“Discover-Offer-Request-ACK”四个数据包顺序组成，并且合理地使用了 Transaction ID、Secs、Cilent address、Your address、Server address 以及其他字段。最终，客户端在局域网内合理地使用广播方式获取了一条能够使用指定租约时间的 IP 地址。

除此之外，DHCP 还包括 DHCP Relay，即 DHCP 中继，用于实现在不同子网和物理网段之间处理和转发 DHCP 信息的功能，如果果客户端与服务器在同一个物理网段，则需要 DHCP Relay。

在本次实验中，通过 Wireshark 中显示的报文数据，我发现 DHCP Server 的 IP 地址为 10.3.9.2，而客户端的 IP 地址为 10.129.195.125，由已学知识可知：这两个 IP 地址不在同一个子网中。故推断使用了 DHCP Relay 转发 DHCP 请求和响应。在 Wireshark 中也发现使用了路由 10.129.0.1 作为转发的中继站。

2.3 发送 ICMP 报文，捕获并分析格式

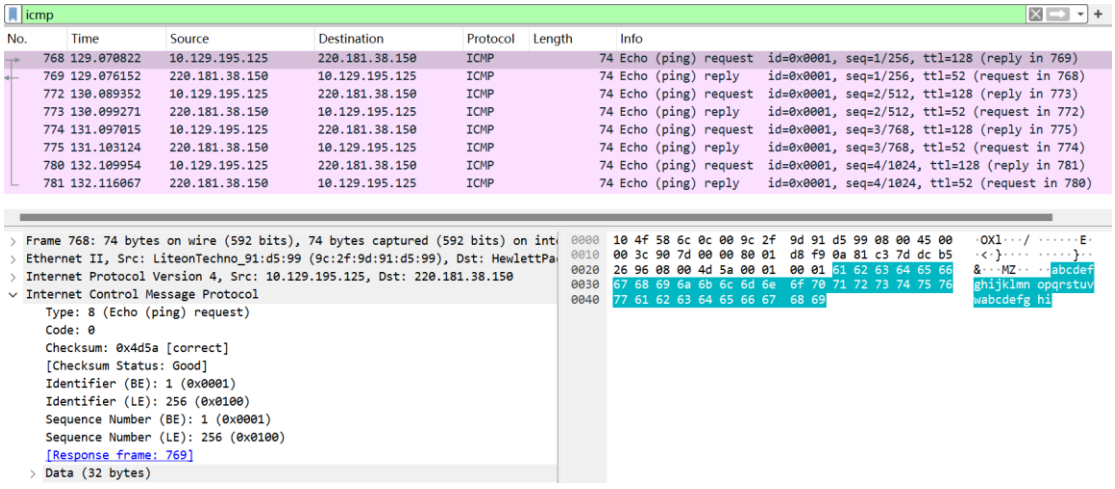
2.3.1 执行 ping 命令，捕获 ICMP 报文

在终端输入 ping www.baidu.com 如下图：

```
C:\Users\SevenGrass>ping www.baidu.com

Pinging www.a.shifen.com [220.181.38.150] with 32 bytes of data:
Reply from 220.181.38.150: bytes=32 time=5ms TTL=52
Reply from 220.181.38.150: bytes=32 time=10ms TTL=52
Reply from 220.181.38.150: bytes=32 time=6ms TTL=52
Reply from 220.181.38.150: bytes=32 time=6ms TTL=52

Ping statistics for 220.181.38.150:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 5ms, Maximum = 10ms, Average = 6ms
```

ICMP 1 执行 ping 命令

设置 Wireshark 显示过滤器为 icmp，过滤其他协议数据，捕获的报文如下：



ICMP 2 捕获 ICMP 报文

2.3.2 ICMP 报文分析

1) ICMP 协议简介：

ICMP，全称 Internet Control Message Protocol，即互联网控制报文协议。是一种用于在 IP 主机、路由器和网络中发送控制消息的网络层协议。它用于报告 IP 数据包传输过程中的错误信息，并执行诊断功能。

2) ICMP 报文格式

Type - 8 bits	Code – 8 bits	Checksum – 16 bits
Identifier		Sequence number
Data section		

ICMP 3 Query messages **格式**

Type - 8 bits	Code – 8 bits	Checksum – 16 bits
Rest of the header		
Data section		

ICMP 4 Error-reporting messages **格式**

对于各种 ICMP 报文，前 32bits 都是三个长度固定的字段，这三个字段的含义作用如下：

(1) Type：表示 ICMP 报文类型，用于标识错误类型的差错报文或者查询类型的报告报文。目前已定义了 14 种，从类型值来看 ICMP 报文可以分为两大类。第一类是取值为 1~127 的差错报文，第 2 类是取值 128 以上的信息报文。

(2) Code：表示 ICMP 差错报文的错误原因，代码值不同对应的错误也不同。如：Type 为 11 且 Code 为 0，表示数据传输过程中超时了，超时的具体原因是 TTL 值为 0，数据报被丢弃。

(3) Checksum：用于检查数据报文是否有错误。

下面是一些常见的 Type 和 Code 的搭配：

类型 Type	**代码 Code**	**描述**
0	0	回显应答（**ping 应答**）
3	0 to 15	目的地不可达
4	0	源端被关闭，即源抑制（用于拥塞控制）
5	0 to 3	重定向
8	0	请求回显（**ping 请求**）
11	0 / 1	超时
12	0 / 1	缺少参数
17 / 18	0	地址掩码请求 / 应答

对于本次实验捕获的 ICMP 报文，都属于 Query messages，我们继续分析该类型报文的剩余字段作用：

(1) Identifier：对于每一个发送的数据报进行标识。

(2) Sequence number：对于发送的每一个数据报文进行编号。

(3) Data section：数据内容。

3) ICMP 捕获内容实例分析：

一共捕获四对 request 包和 reply 包。我们以第一组为例：

No.	Time	Source	Destination	Protocol	Length	Info
768	129.070822	10.129.195.125	220.181.38.150	ICMP	74	Echo (ping) request id=0x0001, seq=1/256, ttl=128 (reply in 769)
769	129.076152	220.181.38.150	10.129.195.125	ICMP	74	Echo (ping) reply id=0x0001, seq=1/256, ttl=52 (request in 768)

ICMP 5 **第一组应答**

首先查看 request 报文的具体内容：

```
Internet Control Message Protocol
   Type: 8 (Echo (ping) request)
   Code: 0
   Checksum: 0x4d5a [correct]
   [Checksum Status: Good]
   Identifier (BE): 1 (0x0001)
   Identifier (LE): 256 (0x0100)
   Sequence Number (BE): 1 (0x0001)
   Sequence Number (LE): 256 (0x0100)
   [Response frame: 769]
 > Data (32 bytes)
```

ICMP 6 request **报文**

如同上述表格中内容，此报文 Type=8，Code=0，表示 ping 请求。

再查看对应的 reply 报文：

```
Internet Control Message Protocol
   Type: 0 (Echo (ping) reply)
   Code: 0
   Checksum: 0x555a [correct]
   [Checksum Status: Good]
   Identifier (BE): 1 (0x0001)
   Identifier (LE): 256 (0x0100)
   Sequence Number (BE): 1 (0x0001)
   Sequence Number (LE): 256 (0x0100)
   [Request frame: 768]
   [Response time: 5.330 ms]
 > Data (32 bytes)
```

ICMP 7 reply **报文**

此时 Type=0，Code=0，表示 ping 应答。

可以发现：第一组的 Sequence number 为 1（BE），点开第二组可以发现序列号为 2，第三组、第四组的序列号依次为 3，4.

2.4 分析 IP 数据报的分片传输过程

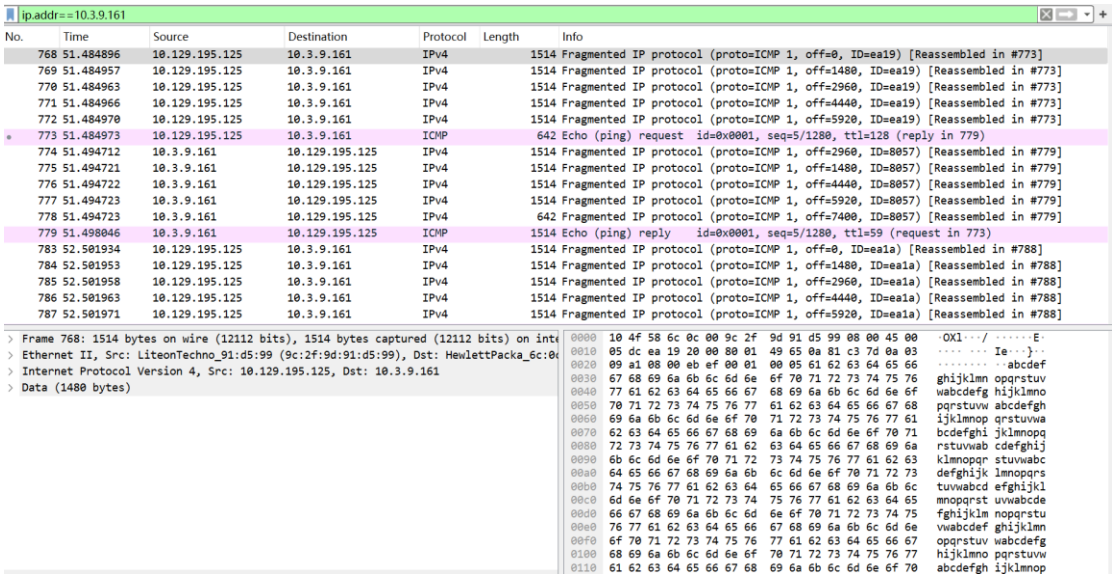
2.4.1 制作 8000 字节的 IP 数据报并发送、捕获

在终端使用命令：ping -l 8000 www.bupt.edu.cn ，向学校官网发送 8000 字节的数据报，结果如下图：

```
C:\Users\SevenGrass>ping -l 8000 www.bupt.edu.cn

Pinging vn46.bupt.edu.cn [10.3.9.161] with 8000 bytes of data:
Reply from 10.3.9.161: bytes=8000 time=13ms TTL=59
Reply from 10.3.9.161: bytes=8000 time=45ms TTL=59
Reply from 10.3.9.161: bytes=8000 time=3ms TTL=59
Reply from 10.3.9.161: bytes=8000 time=12ms TTL=59

Ping statistics for 10.3.9.161:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 45ms, Average = 18ms
```

IP 1 执行发送数据报命令

设置 Wireshark 的显示过滤器为 ip.addr==10.3.9.161，捕获的内容如下：

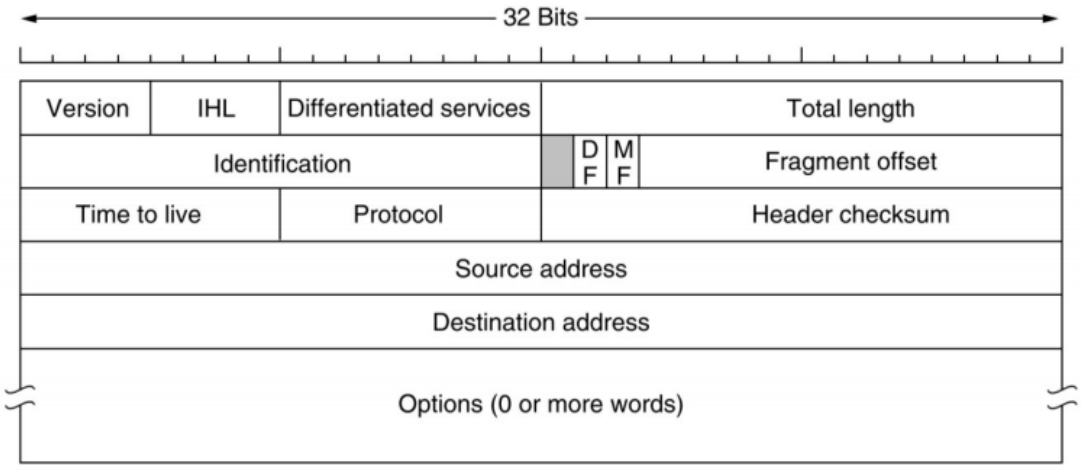


IP 2 捕获的数据报内容

2.4.2 IP 数据报及协议分析

1) IPv4 包头定义：

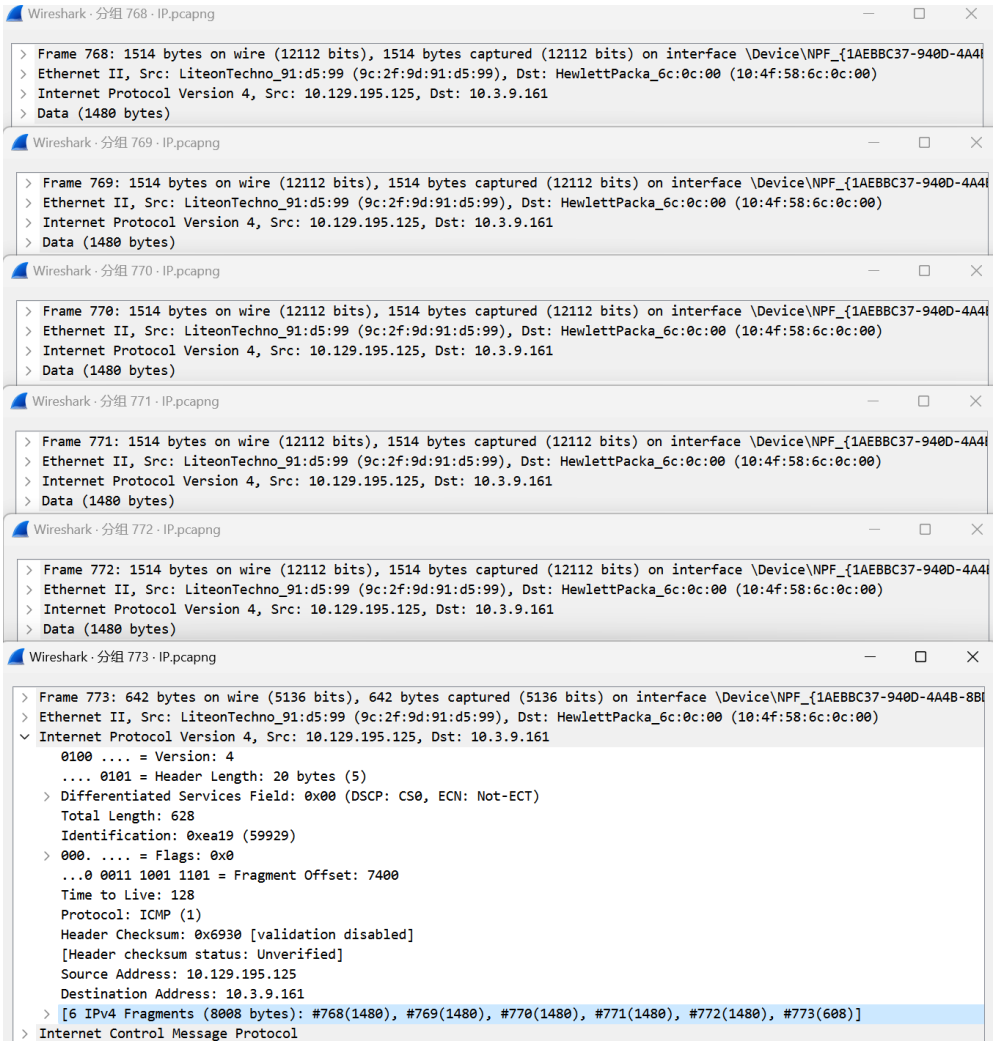
由课内理论知识的学习可知，IPv4 包头格式定义如下：



IP 3 IPv4 **包头定义**

2) 实验 IP 数据报分析：

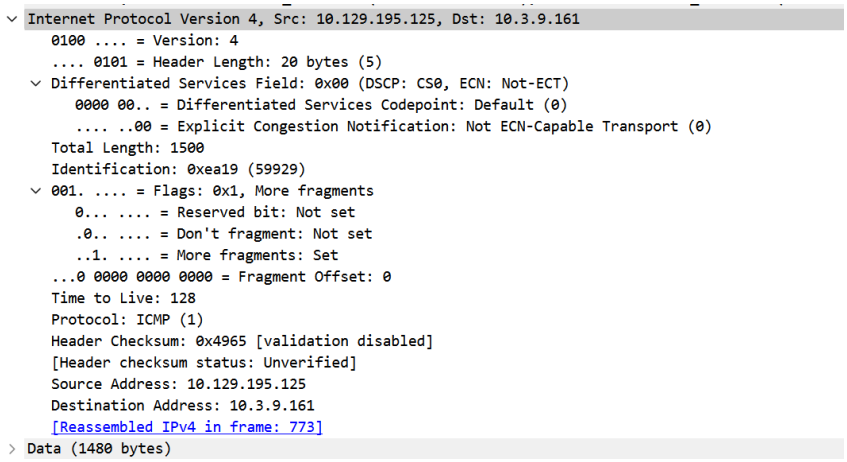
不难发现，第 768～773 号帧为我们发送的数据帧，即制作的 8000 字节数据被分为 6 个分片依次发送，从 Wireshark 中可以看到每个包的大小如下：



IP 4 **各分片大**

由上图可知：去掉 IP 头（20 bytes）后，前五个分片每个包含 1480 字节，最后一片包含 608 字节，一共 8008 字节，即为 8000 字节的数据 +8 字节的 ICMP 头。同时可知链路的 MTU 为 1500.

接下来我们依次分析每片的具体内容。首先分析第 768 号分片：

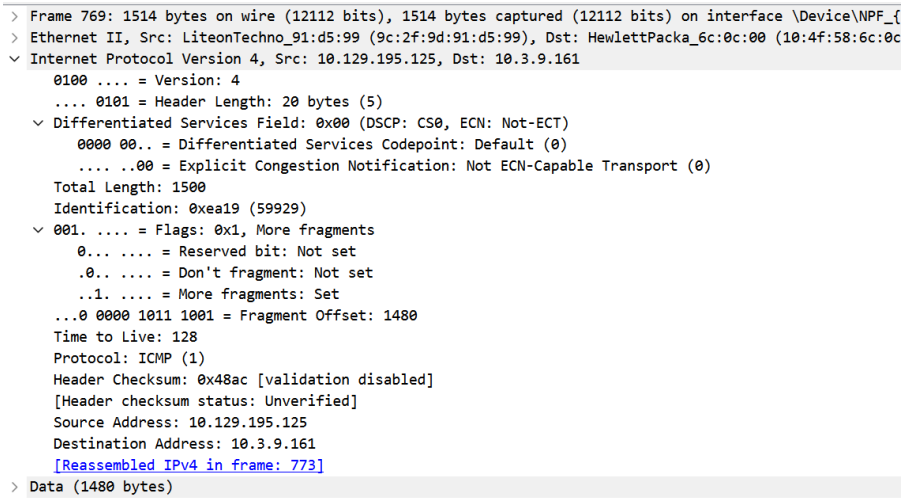


IP 5 **第一个分片**

字段	报文内容 / 取值	描述
包长度	1500	包的总长度为 1500bytes
DF	0	表示允许分片
段标识	0xea19	标识为 0xea19=59929
MF	1	当前分片不是最后一片
偏移量	0	偏移量为 0，即当前片为第一片

IP 6 **第一个分片各字段分析**

接下来是第 769 号分片：



IP 7 **第二个分片**

字段	报文内容 / 取值	描述
包长度	1500	包的总长度为 1500bytes
DF	0	表示允许分片
段标识	0xea19	标识为 0xea19=59929
MF	1	当前分片不是最后一片
偏移量	1480	偏移量为 1480

IP 8 第二个分片各字段分析

接下来是第 770 号分片：

```
Frame 770: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{1AEB
Ethernet II, Src: LiteonTechno_91:d5:99 (9c:2f:9d:91:d5:99), Dst: HewlettPacka_6c:0c:00 (10:4f:58:6c:0c:00)
Internet Protocol Version 4, Src: 10.129.195.125, Dst: 10.3.9.161
   0100 .... = Version: 4
   .... 0101 = Header Length: 20 bytes (5)
v  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: 1500
   Identification: 0xea19 (59929)
v  001. .... = Flags: 0x1, More fragments
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..1. .... = More fragments: Set
   ...0 0001 0111 0010 = Fragment Offset: 2960
   Time to Live: 128
   Protocol: ICMP (1)
   Header Checksum: 0x47f3 [validation disabled]
   [Header checksum status: Unverified]
   Source Address: 10.129.195.125
   Destination Address: 10.3.9.161
   [Reassembled IPv4 in frame: 773]
Data (1480 bytes)
```

IP 9 第三个分片

字段	报文内容 / 取值	描述
包长度	1500	包的总长度为 1500bytes
DF	0	表示允许分片
段标识	0xea19	标识为 0xea19=59929
MF	1	当前分片不是最后一片
偏移量	2960	偏移量为 2960

IP 10 第三个分片各字段分析

接下来是第 771 号分片：

```
Frame 771: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{1AEBBC37-940D
Ethernet II, Src: LiteonTechno_91:d5:99 (9c:2f:9d:91:d5:99), Dst: HewlettPacka_6c:0c:00 (10:4f:58:6c:0c:00)
Internet Protocol Version 4, Src: 10.129.195.125, Dst: 10.3.9.161
    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: 1500
    Identification: 0xea19 (59929)
  ▾ 001. .... = Flags: 0x1, More fragments
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..1. .... = More fragments: Set
    ...0 0010 0010 1011 = Fragment Offset: 4440
    Time to Live: 128
    Protocol: ICMP (1)
    Header Checksum: 0x473a [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.129.195.125
    Destination Address: 10.3.9.161
    [Reassembled IPv4 in frame: 773]
Data (1480 bytes)
```

IP 11 第四个分片

字段	报文内容 / 取值	描述
包长度	1500	包的总长度为 1500bytes
DF	0	表示允许分片
段标识	0xea19	标识为 0xea19=59929
MF	1	当前分片不是最后一片
偏移量	4440	偏移量为 4440

IP 12 第四个分片各字段分析

接下来是第 772 号分片：

```
› Frame 772: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface \Device\NPF_{1AEBBC37-940D-4A
› Ethernet II, Src: LiteonTechno_91:d5:99 (9c:2f:9d:91:d5:99), Dst: HewlettPacka_6c:0c:00 (10:4f:58:6c:0c:00)
▾ Internet Protocol Version 4, Src: 10.129.195.125, Dst: 10.3.9.161
    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: 1500
    Identification: 0xea19 (59929)
  ▾ 001. .... = Flags: 0x1, More fragments
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..1. .... = More fragments: Set
    ...0 0010 1110 0100 = Fragment Offset: 5920
    Time to Live: 128
    Protocol: ICMP (1)
    Header Checksum: 0x4681 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.129.195.125
    Destination Address: 10.3.9.161
    [Reassembled IPv4 in frame: 773]
› Data (1480 bytes)
```

IP 13 第五个分片

字段	报文内容 / 取值	描述
包长度	1500	包的总长度为 1500bytes
DF	0	表示允许分片
段标识	0xea19	标识为 0xea19=59929
MF	1	当前分片不是最后一片
偏移量	5920	偏移量为 5920

IP 14 第五个分片各字段分析

接下来是第 773 号分片：

```
> Frame 773: 642 bytes on wire (5136 bits), 642 bytes captured (5136 bits) on interface \Device\NPF_{1AEBBC37-940D-4A4B-
> Ethernet II, Src: LiteonTechno_91:d5:99 (9c:2f:9d:91:d5:99), Dst: HewlettPacka_6c:0c:00 (10:4f:58:6c:0c:00)
v Internet Protocol Version 4, Src: 10.129.195.125, Dst: 10.3.9.161
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  v 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: 628
    Identification: 0xea19 (59929)
  v 000. .... = Flags: 0x0
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
    ...0 0011 1001 1101 = Fragment Offset: 7400
    Time to Live: 128
    Protocol: ICMP (1)
    Header Checksum: 0x6930 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.129.195.125
    Destination Address: 10.3.9.161
  > [6 IPv4 Fragments (8008 bytes): #768(1480), #769(1480), #770(1480), #771(1480), #772(1480), #773(608)]
> Internet Control Message Protocol
```

IP 15 第六个分片

字段	报文内容 / 取值	描述
包长度	628	包的总长度为 628 bytes
DF	0	表示允许分片
段标识	0xea19	标识为 0xea19=59929
MF	0	当前分片为最后一片
偏移量	7400	偏移量为 7400

IP 16 第六个分片各字段分析

同一个包的所有分片都有相同的段标识(Identification) 字段，从而可以确认哪些分片属于同一个包。

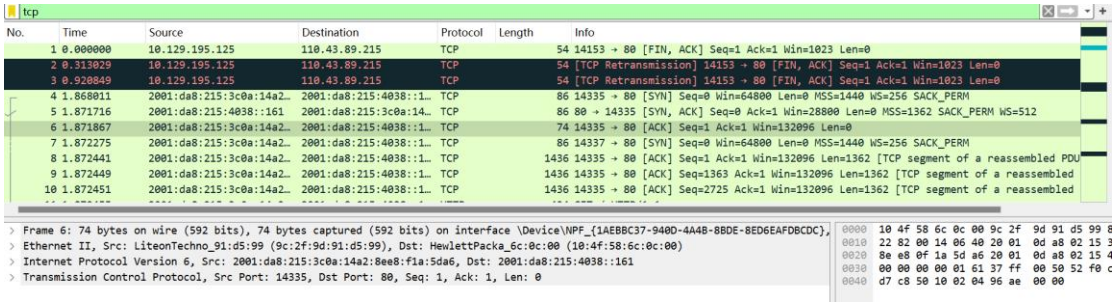
2.5 捕获建立连接和释放连接过程的 TCP 报文段并分析

2.5.1 打开网页，全部显示后关闭，捕获 TCP 报文段

打开 Wireshark 监控，设置捕获过滤器为 tcp port 80；

使用 edge 浏览器打开 http://ucloud.bupt.edu.cn，待完全显示后直接关闭 edge 浏览器。

捕获的 TCP 报文段如下：

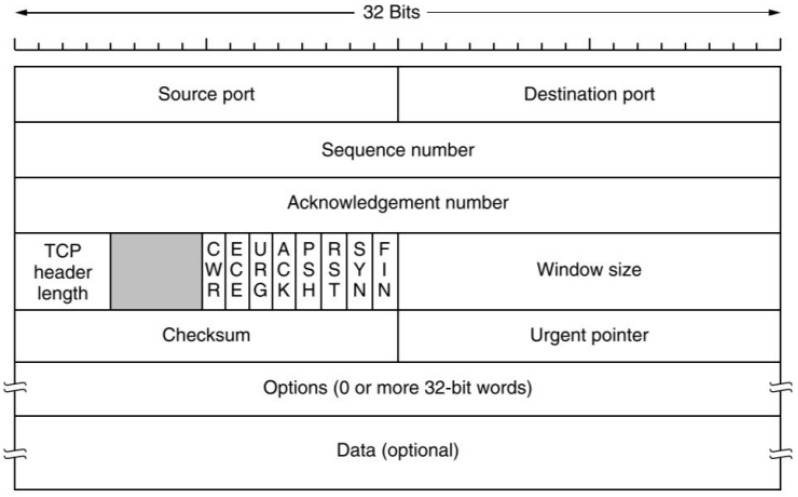


TCP 1 捕获的 TCP 报文段

2.5.2 TCP 协议分析

TCP，全称 Transmission Control Protocol，即传输控制协议，是一种面向连接的、可靠的、基于字节流的传输层通信协议。TCP 在传输数据之前，需要在发送端和接收端之间建立一个连接。这通过“三次握手”完成。

先分析一下 TCP 头各字段作用：

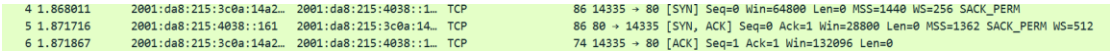


TCP 2 TCP 头格式

1) **Source / Destination port**：表示连接的两个端点；
2) **Sequence number**：表示当前报文的标志；
3) **Acknowledgement number**：表示期望收到的报文的序号；
4) **TCP header length**：表示数据字段在报文中的起始位置；
5) **CWR、ECE**：拥塞控制的信号；
6) **URG**：表示是否使用了紧急指针字段；
7) **ACK**：表示确认字段是否有效；
8) **PSH**：表示当前数据是否需要立即“PUSH”；
9) **RST**：用于处理连接混乱、主机崩溃等问题；
10) **SYN**：用于建立连接过程；
11) **FIN**：用于释放连接；
12) **Window size**：表示从被确认的字节开始，发送端可以发送多少个字节；
13) **Checksum**：校验字段；
14) **Urgent pointer**：指向从当前序号开始找到紧急数据的字节偏移量；
15) **Options**：用于扩展。

2.5.3 TCP 建立连接分析

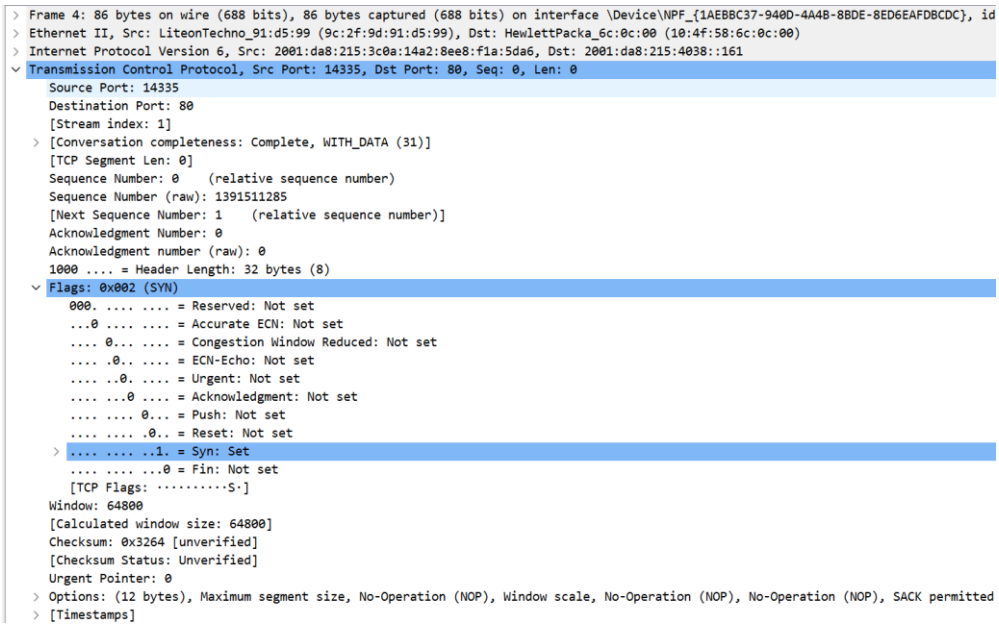
序号 4-6：客户端向服务器发送 SYN 请求建立连接，服务器返回 SYN+ACK 应答，客户端返回 ACK 完成连接建立。即“三次握手”



TCP 3 **建立连接的消息序列**

各消息详细内容如下。

首先分析 TCP 连接建立阶段的第一个数据报内容：

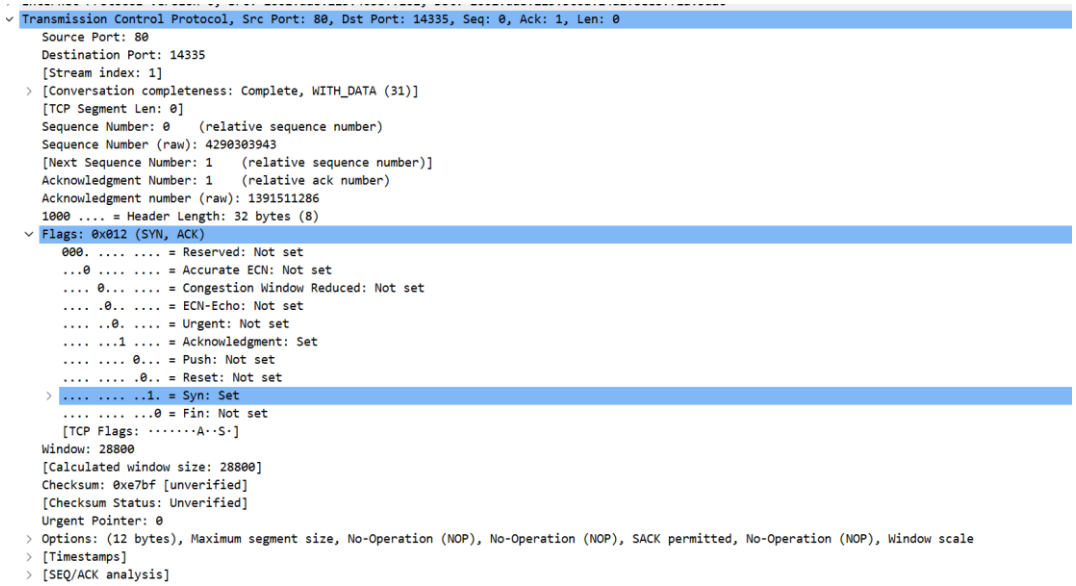


TCP 4 **建立连接阶段** SYN

字段	**值**	**描述**
SYN	1	用于建立连接
ACK	0	ACK 字段无效
FIN	0	FIN 字段无效
Sequence number	0	该报文的起始字节序号为 0
Acknowledgement number	0	接下来期望收到序号为 0 的字节

TCP 5 SYN **字段解释**

接着分析 TCP 连接建立阶段第二个数据报内容：



TCP 6 **建立连接时** SYN+ACK

字段	值	描述
SYN	1	用于建立连接
ACK	1	ACK 字段有效
FIN	0	FIN 字段无效
Sequence number	0	该报文的起始字节序号为 0
Acknowledgement number	1	接下来期望收到序号为 1 的字节

TCP 7SYN+ACK **内容分析**

再分析 TCP 连接建立阶段的第三个数据报内容：

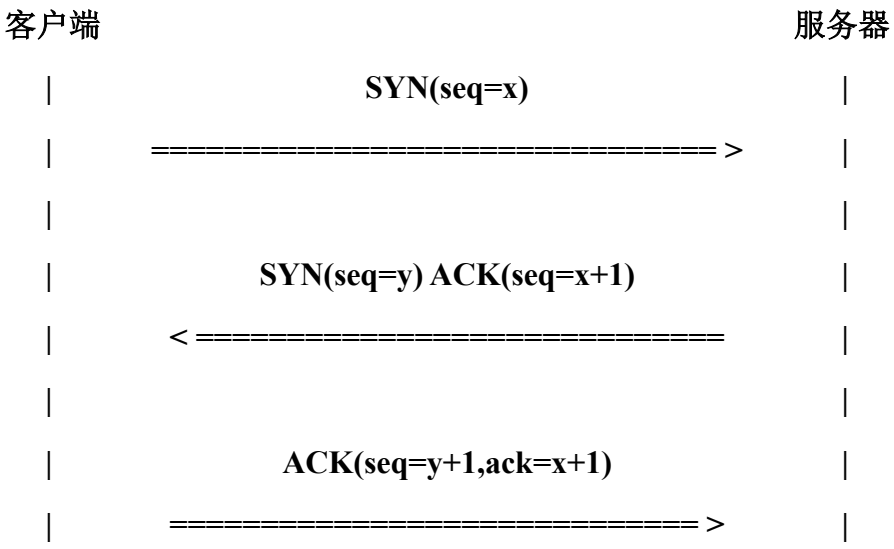
```
Transmission Control Protocol, Src Port: 14335, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
    Source Port: 14335
    Destination Port: 80
    [Stream index: 1]
  > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 1391511286
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 4290303944
    0101 .... = Header Length: 20 bytes (5)
  v Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Accurate ECN: Not set
      .... 0... .... = Congestion Window Reduced: Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A····]
    Window: 516
    [Calculated window size: 132096]
    [Window size scaling factor: 256]
    Checksum: 0x96ae [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
```

TCP 8 TCP **建立连接时** ACK

字段	值	描述
SYN	0	SYN 字段无效
ACK	1	ACK 字段有效
FIN	0	FIN 字段无效
Sequence number	1	该报文的起始字节序号为 1
Acknowledgement number	1	接下来期望收到序号为 1 的字节

TCP 9 ACK **字段解释**

上述 TCP 连接的建立阶段可以总结为“本地发送 SYN、服务端返回 SYN+ACK、本地发送 ACK”，如下图所示：



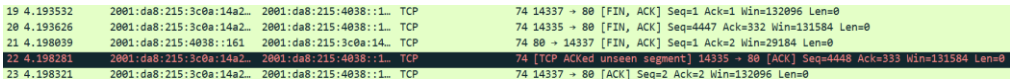
TCP 10 I=连接建立消息序列图

2.5.4 TCP 释放连接分析

序号 19-23 的数据报展示了客户端和服务器分别发送 FIN 请求，对方返回应答，最终返回 ACK 完成连接释放的过程。

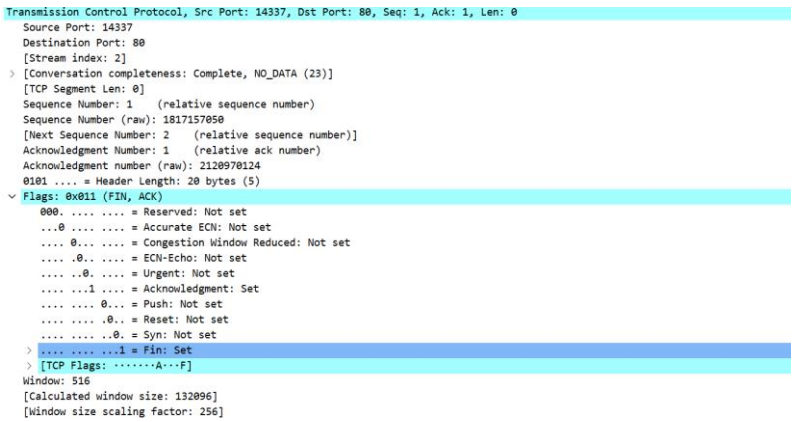
特别需要注意的是：在本次实验中，在释放连接的过程中，只抓到了对应的 3 个包，但对于“四次挥手”，其实是将第二次和第三次合并在了一起，具体分析见下文。

释放连接的过程其实是“四次挥手”：



TCP 11 释放连接报文序列

首先分析“第一次挥手”：

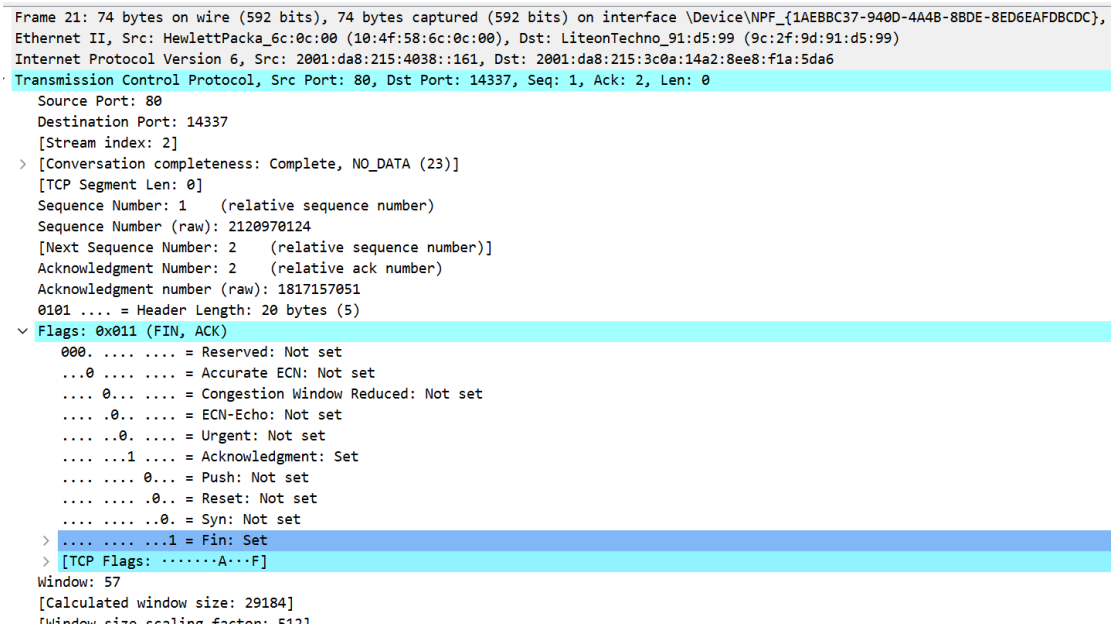


TCP 12 释放连接 FIN

字段	值	描述
SYN	0	SYN 字段无效
ACK	1	ACK 字段有效
FIN	1	FIN 字段有效
Sequence number	1	该报文的起始字节序号为 1
Acknowledgement number	1	接下来期望收到序号为 1 的字节

TCP 13 **释放连接** FIN **字段分析**

接下来其实是“第二次挥手”+“第三次挥手”：



TCP 14 **释放连接** FIN+ACK

字段	值	描述
SYN	0	SYN 字段无效
ACK	1	ACK 字段有效
FIN	1	FIN 字段有效
Sequence number	1	该报文的起始字节序号为 1
Acknowledgement number	2	接下来期望收到序号为 2 的字节

TCP 15FIN+ACK **字段分析**

可以看到，此时的包 seq=1，ack=2.而理论上，如果分为两次挥手，第二次挥手(ACK)时，seq=1,ack=2，之后第三次挥手(FIN)时，seq=x，ack=2；

经查阅资料发现：关闭连接有两种方式，当一方关闭连接，另外一方没有数

据发送时，马上关闭连接，也就将第二步的 ACK 与第三步的 FIN 合并为一步了。

详细解释如下：因为开启了延时 ACK 机制 ，导致收到第一个 FIN 之后，发送 ACK 的条件不能满足立即发送 ACK 的条件，导致 ACK 的发送被延时了，在延时的过程中，应用如果确认没数据要发，并且也要关闭此连接的情况下，会触发发送 FIN，这个 FIN 就会和之前的 ACK 合并被发出。

接下来时最后一次挥手：

```
Frame 23: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{1AEBBC37-940D-4A4B-8BDE-8ED6EAFDBCDC},
Ethernet II, Src: LiteonTechno_91:d5:99 (9c:2f:9d:91:d5:99), Dst: HewlettPacka_6c:0c:00 (10:4f:58:6c:0c:00)
Internet Protocol Version 6, Src: 2001:da8:215:3c0a:14a2:8ee8:f1a:5da6, Dst: 2001:da8:215:4038::161
Transmission Control Protocol, Src Port: 14337, Dst Port: 80, Seq: 2, Ack: 2, Len: 0
   Source Port: 14337
   Destination Port: 80
   [Stream index: 2]
 > [Conversation completeness: Complete, NO_DATA (23)]
   [TCP Segment Len: 0]
   Sequence Number: 2    (relative sequence number)
   Sequence Number (raw): 1817157051
   [Next Sequence Number: 2    (relative sequence number)]
   Acknowledgment Number: 2    (relative ack number)
   Acknowledgment number (raw): 2120970125
   0101 .... = Header Length: 20 bytes (5)
 v Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Accurate ECN: Not set
      .... 0... .... = Congestion Window Reduced: Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A····]
   Window: 516
   [Calculated window size: 132096]
   [Window size scaling factor: 256]
```

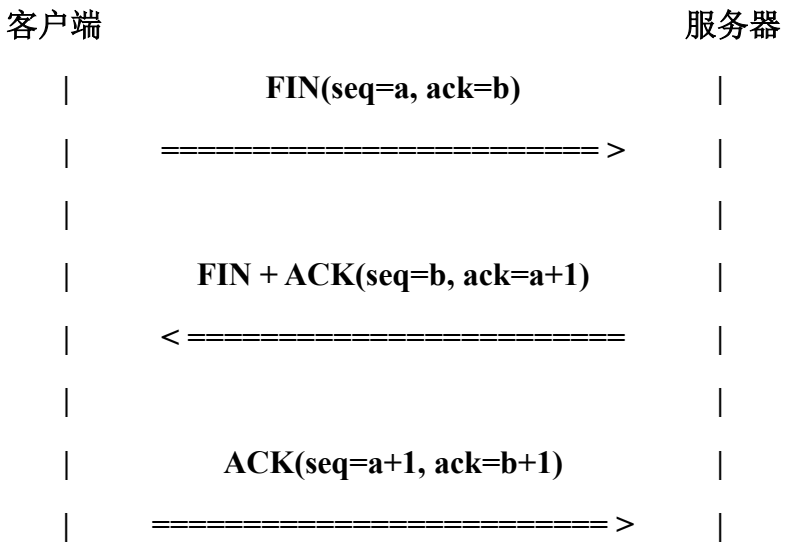
TCP 16 **释放连接** ACK

字段	**值**	**描述**
SYN	0	SYN 字段无效
ACK	1	ACK 字段有效
FIN	0	FIN 字段无效
Sequence number	2	该报文的起始字节序号为 2
Acknowledgement number	2	接下来期望收到序号为 2 的字节

TCP 17 ACK **字段分析**

上述 TCP 连接的断开阶段可以总结为“一端发送 FIN、另一端返回 ACK、另一端发送 FIN、一端返回 ACK”，即“四次挥手”。

由于本次实验实际测得第二次和第三次挥手合并，故画出“三次挥手”序列图，如下所示：



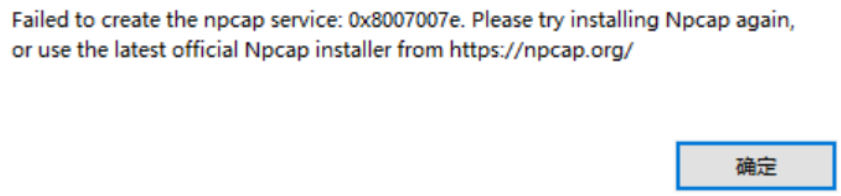
TCP 18 释放连接消息序列图

3.实验结论和实验心得

3.1 所遇问题及解决方案

1) 安装 Wireshark 时 Npcap 组件安装失败

安装 Npcap 组件时报错：



尝试了去 Npcap 的官网下载，依然失败；关闭了杀毒软件等，依然失败。

解决方案：安装 WinPcap 代替 Npcap，因为Wireshark 在找不到 Npcap 的情况下，会使用 WinPcap 作为替代。

2) 捕获 IP 数据报时，显示过滤器中地址如何确定

在执行 ping -l 8000 www.bupt.edu.cn 前，我在思考如何在 Wireshark 中确定显示的是发送给学校官网的报文，而不是其他网址的。考虑设置过滤器为 ip.addr==ip 地址，但又不知道官网的 ip 地址。

解决方案：使用命令 nslookup www.bupt.edu.cn 查看学校官网的 IP 地址如下图：

```
C:\Users\SevenGrass>nslookup www.bupt.edu.cn
Server:  b.resolvers.level3.net
Address:  4.2.2.2

Non-authoritative answer:
Name:    vn46.bupt.edu.cn
Addresses:  2001:da8:215:4038::161
          10.3.9.161
Aliases:  www.bupt.edu.cn
```

从图中可以知道学校官网的 IP 地址为 10.3.9.161 ，但是后来才发现执行完 ping -l 8000 www.bupt.edu.cn 也会自动显示该网站的 IP 地址。

3) 分析 TCP 连接释放过程时，只有“三次挥手”

多次抓包皆为“三次挥手”，浪费较多时间，后来查阅相关资料《Linux 的 TCP 实现之：四次挥手》得到解决。

3.2 实验结论及心得

本次抓包实验完成抓包内容仅花费 1 小时（除去最后 TCP“四次挥手”的多次重复尝试），而报告却花费超过 5 小时，主要原因是分析的内容比较多、比较细，以及一些细枝末节的问题也去查阅了资料，虽然耗时很久，但是觉得很值，获得了较多之前没注意到的内容。

本次实验大大加深了我对 IP、TCP、UDP 等等相关知识的理解和掌握，特别是在我查阅资料的过程中，对于计算机网络的知识面拓展了很多，也大大的开阔了我的眼界。

在理论知识方面，对于各协议的各字段内容都有了不同程度的掌握和理解加深，收获很多，对协议的工作原理也有了更加直观的认识。

除此之外，还掌握了 Wireshark 的使用方法，同时在选择过滤条件时，也复习了一遍各端口的含义，对计网的学习非常有帮助。