



**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](#_Toc168518714)

[1.1 实验任务和实验目的 1](#_Toc168518715)

[1.2 实验环境 1](#_Toc168518716)

[2. 实验步骤和协议分析 1](#_Toc168518717)

[2.1 实验准备 1](#_Toc168518718)

[2.2 捕获和分析DHCP报文 2](#_Toc168518719)

[2.2.1 设置捕获过滤器，释放主机IP地址 2](#_Toc168518720)

[2.2.2 重新分配IP地址，捕获DHCP 2](#_Toc168518721)

[2.2.3 DHCP报文及协议分析 3](#_Toc168518722)

[2.3 发送 ICMP 报文，捕获并分析格式 9](#_Toc168518723)

[2.3.1 执行ping命令，捕获ICMP报文 9](#_Toc168518724)

[2.3.2 ICMP报文分析 9](#_Toc168518725)

[2.4 分析 IP 数据报的分片传输过程 12](#_Toc168518726)

[2.4.1 制作8000字节的IP数据报并发送、捕获 12](#_Toc168518727)

[2.4.2 IP数据报及协议分析 12](#_Toc168518728)

[2.5 捕获建立连接和释放连接过程的 TCP 报文段并分析 18](#_Toc168518729)

[2.5.1 打开网页，全部显示后关闭，捕获TCP报文段 18](#_Toc168518730)

[2.5.2 TCP协议分析 18](#_Toc168518731)

[2.5.3 TCP建立连接分析 19](#_Toc168518732)

[2.5.4 TCP释放连接分析 22](#_Toc168518733)

[3. 实验结论和实验心得 25](#_Toc168518734)

[3.1 所遇问题及解决方案 25](#_Toc168518735)

[3.2 实验结论及心得 26](#_Toc168518736)

# 实验内容和实验环境描述

## 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。

## 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 PCle Adapter
4. 网卡物理地址：9C:2F:9D:91:D5:99

# 实验步骤和协议分析

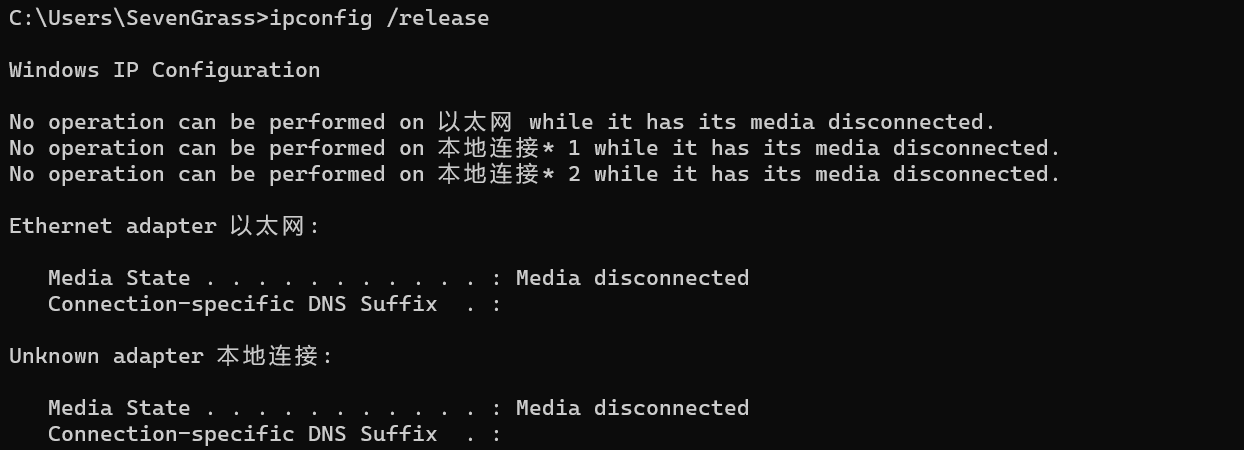
## 实验准备

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。

## 捕获和分析DHCP报文

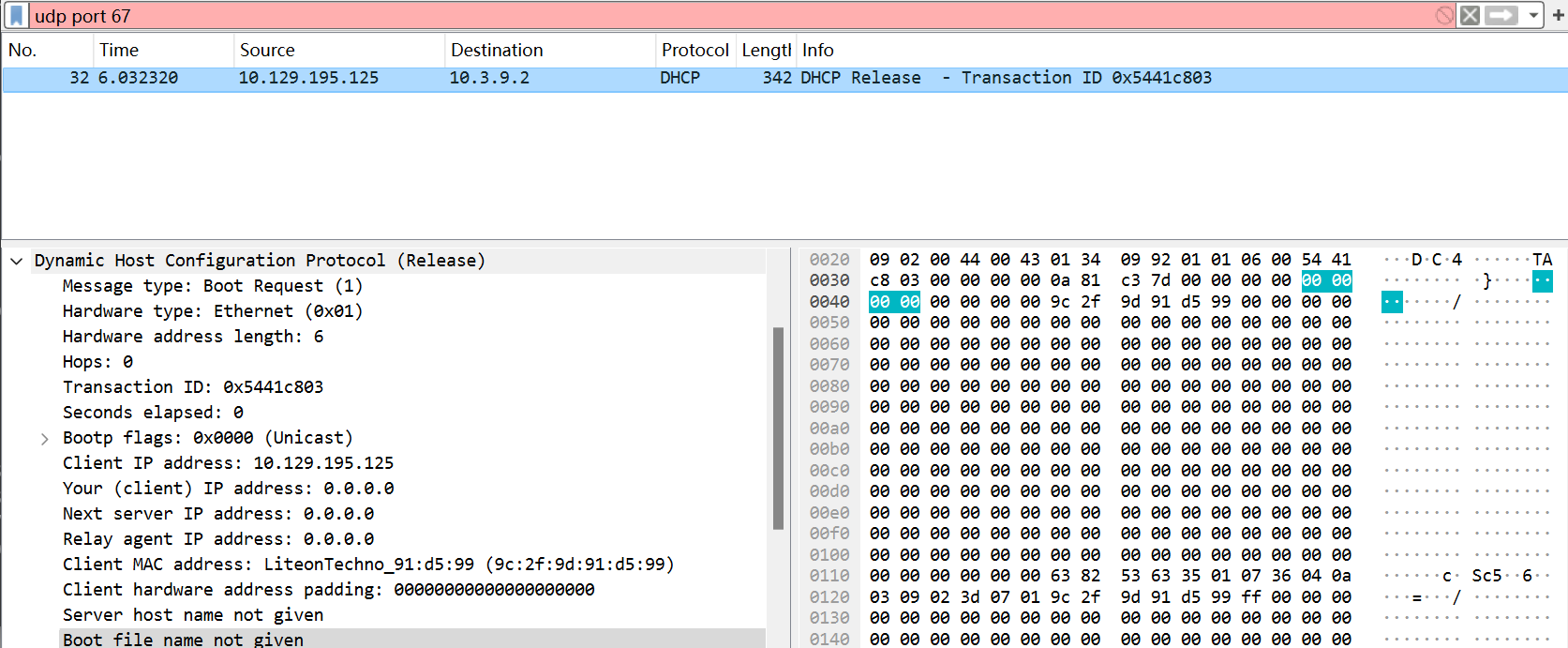
### 设置捕获过滤器，释放主机IP地址

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



DHCP 1释放IP命令

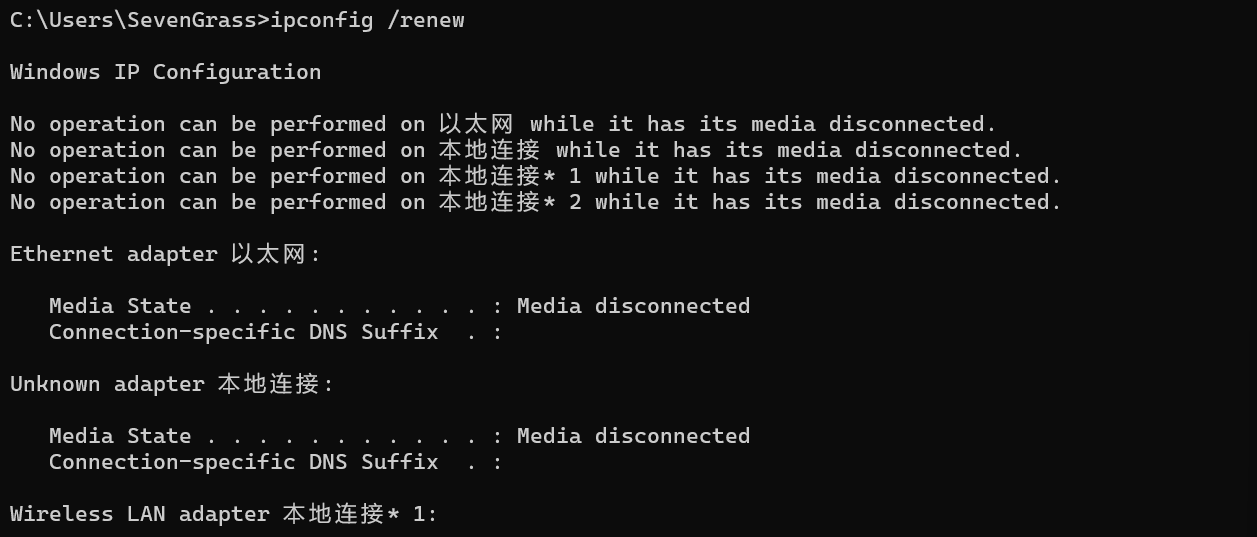
此时捕获到的DHCP如下图：



DHCP 2 报文Release

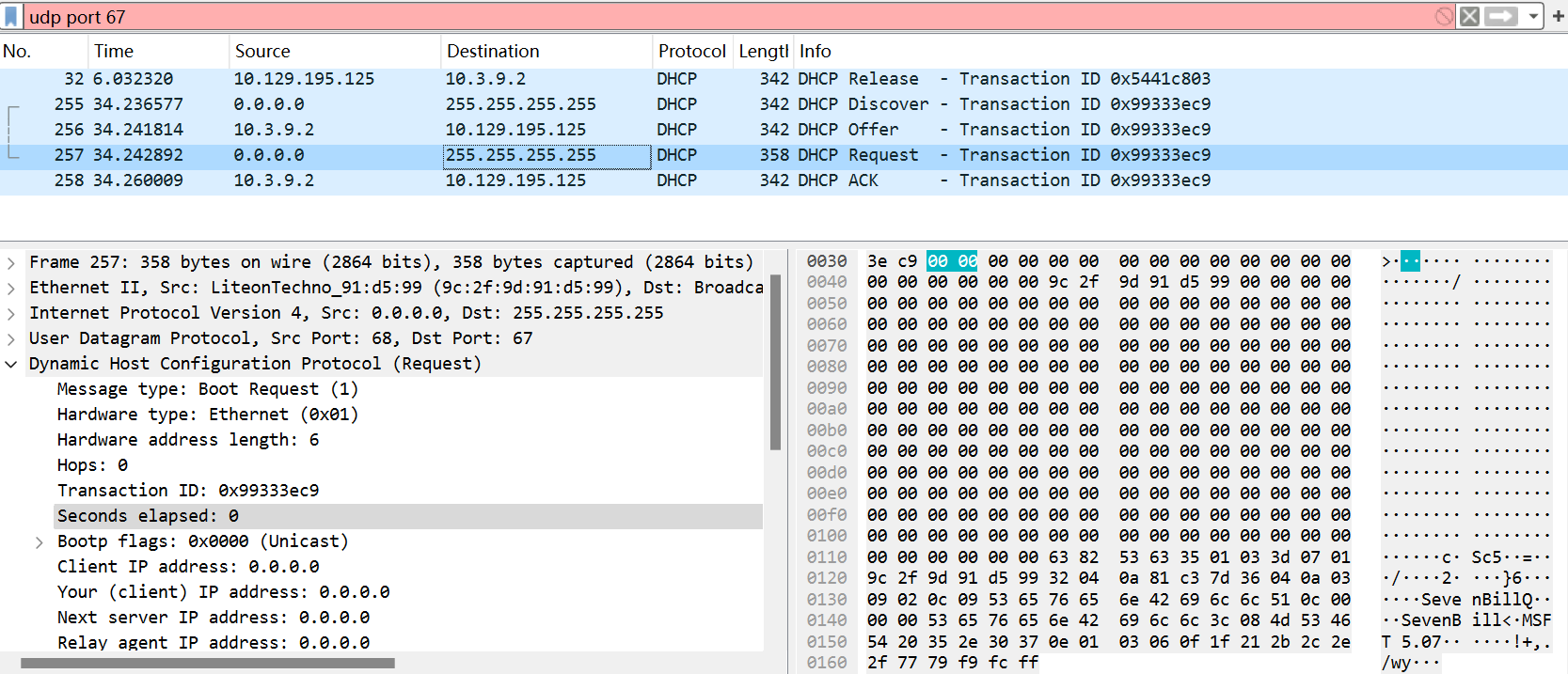
### 重新分配IP地址，捕获DHCP

在终端执行命令ipconfig /renew：



DHCP 3重获IP命令

此时捕获到的DHCP如下：



DHCP 4报文Discover、Offer、Request、ACK

### 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。

1. **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。

1. **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 | | | |

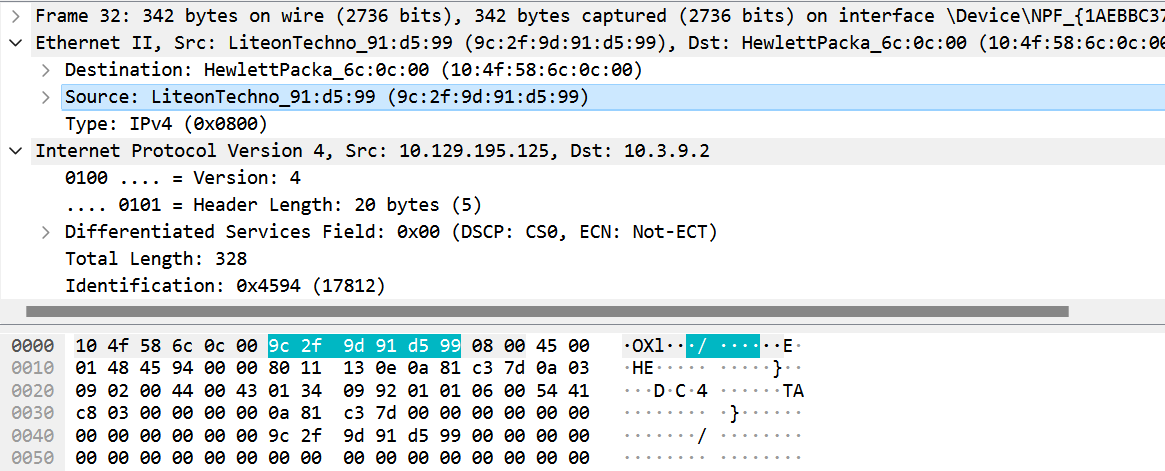
DHCP 5报文格式

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”
16. **DHCP常见数据包分类：**
17. **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。
18. **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。
19. **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。
20. **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。
21. **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。
22. **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。
23. **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。
24. **所捕获的DHCP报文分析及IP地址分配过程**

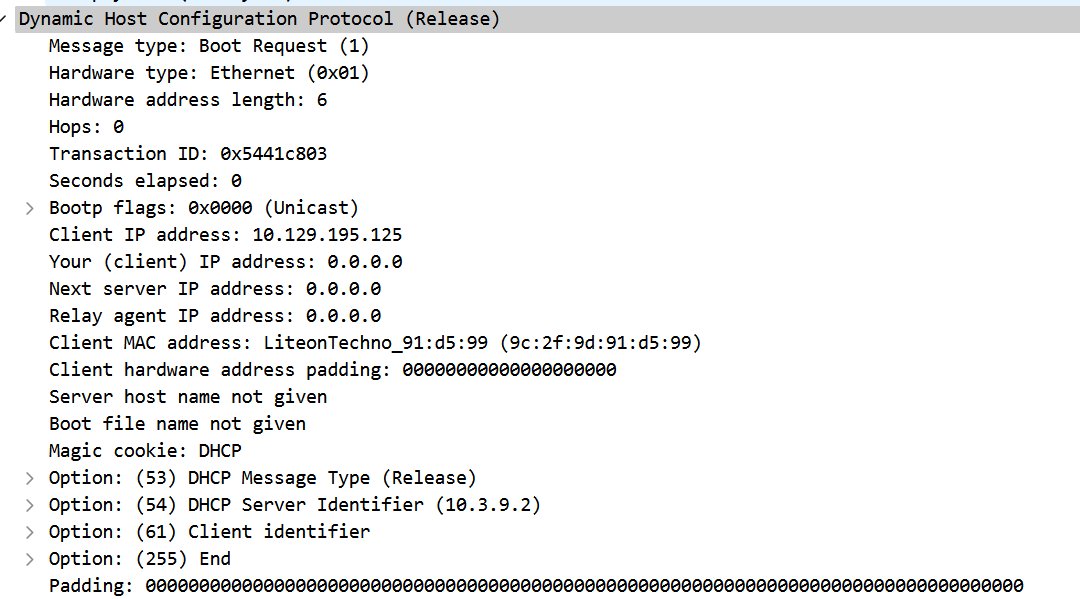
如上图DHCP 2，当我们释放IP地址时，发送了一个长度为342字节的Release包。

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：

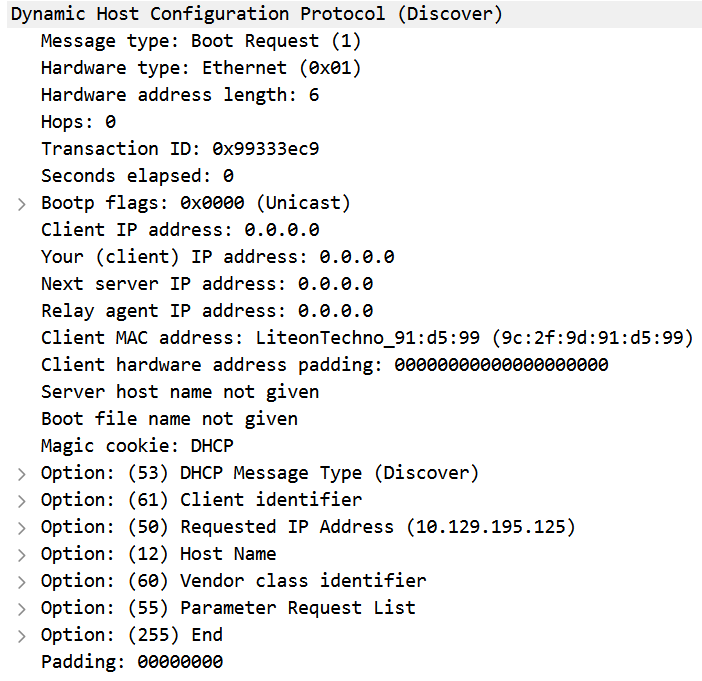
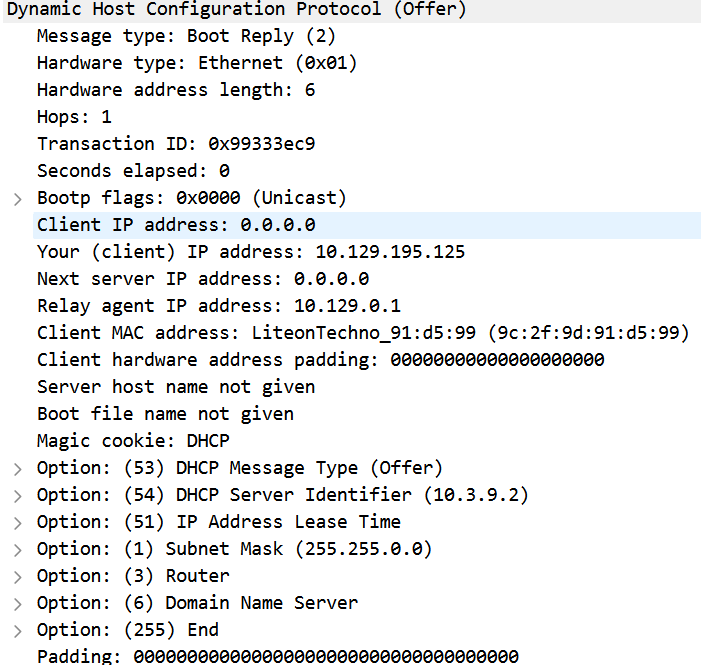


DHCP 7 Release包字段内容

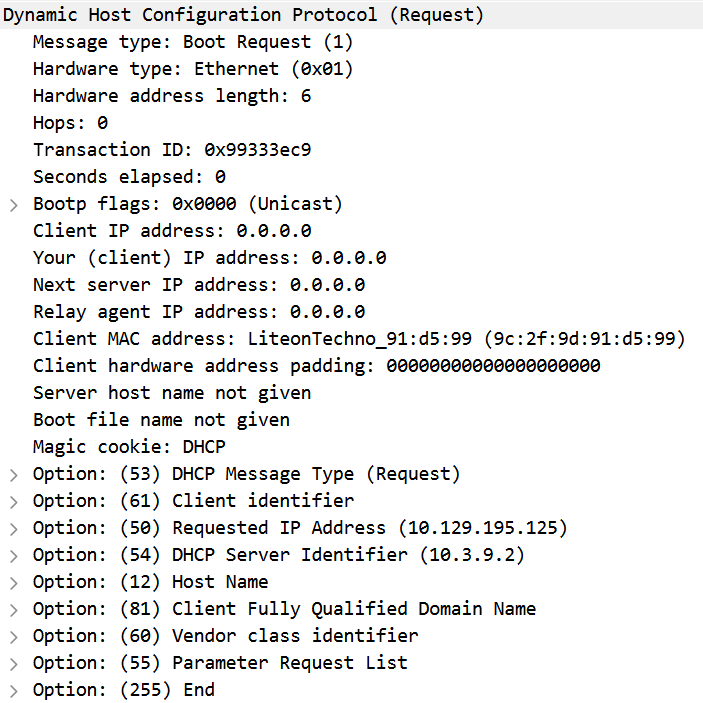
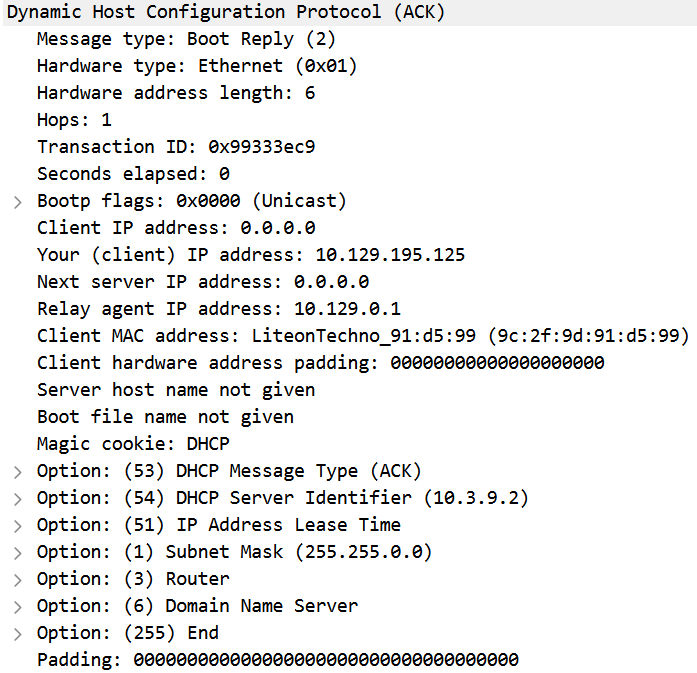
各字段的具体取值含义可参考前文的介绍，不多赘述。此时我们释放了当前的IP地址。

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：

DHCP 8 Discover字段内容 DHCP 9 Offer字段内容

DHCP 10 Request字段内容 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。

1. **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。

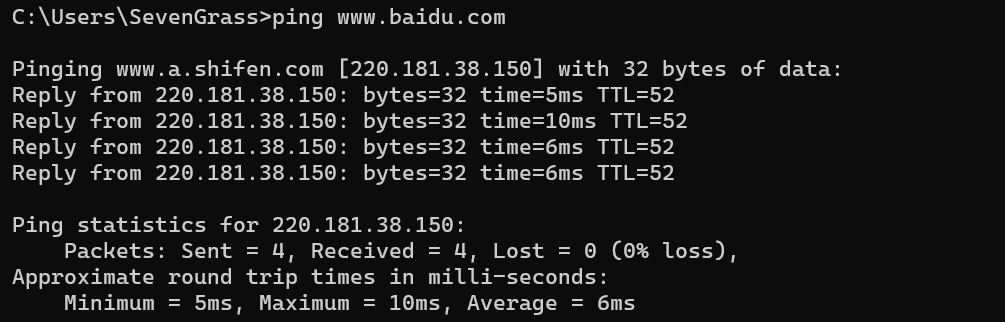
除此之外，DHCP还包括DHCP Relay，即DHCP 中继，用于实现在不同子网和物理网段之间处理和转发 DHCP 信息的功能，如果果客户端与服务器在同一个物理网段，则需要 DHCP Relay。

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。

## 发送 ICMP 报文，捕获并分析格式

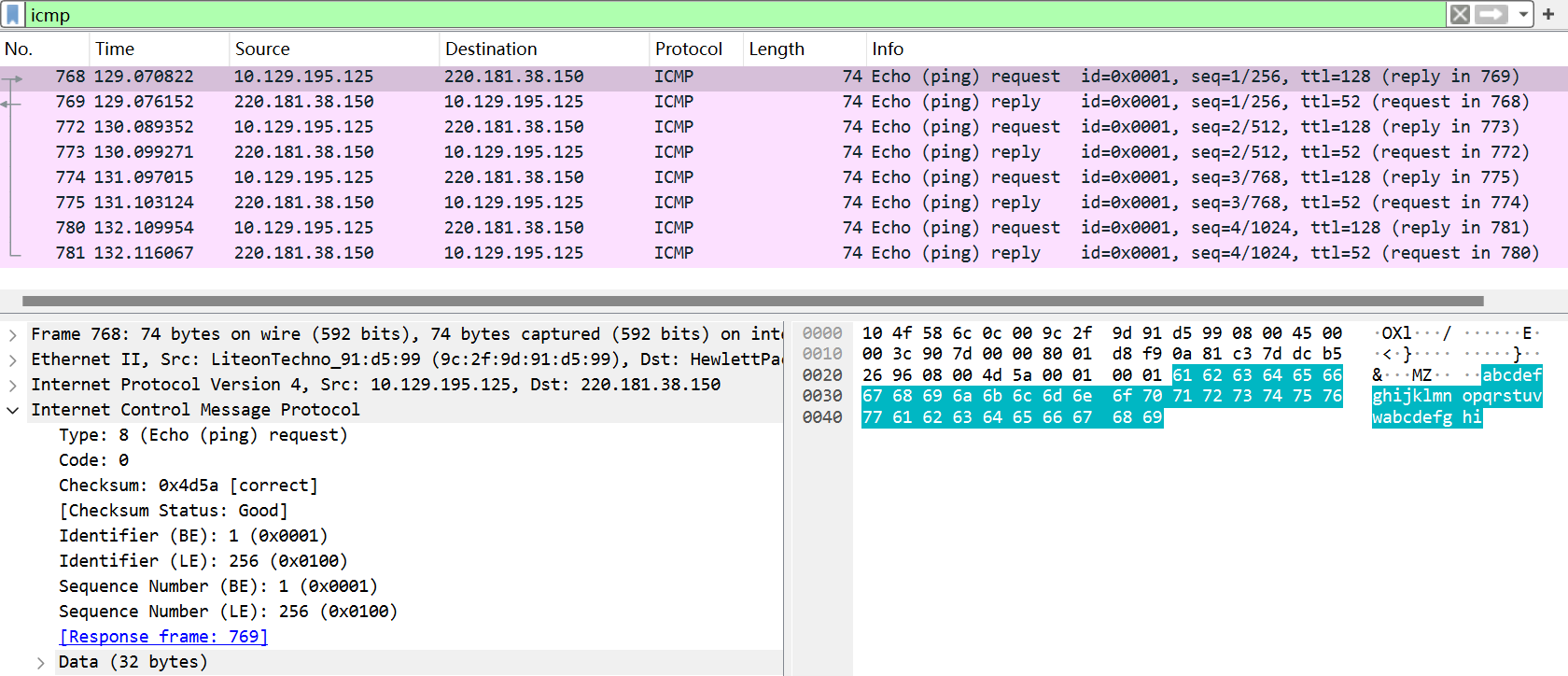
### 执行ping命令，捕获ICMP报文

在终端输入ping [www.baidu.com](http://www.baidu.com)如下图：



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

### 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。

1. **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。

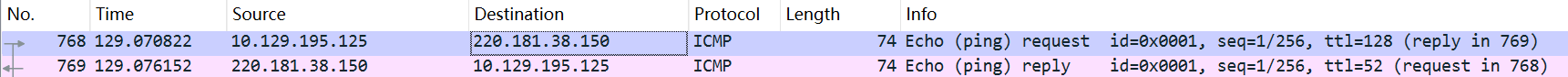
下面是一些常见的Type和Code的搭配：

|  |  |  |
| --- | --- | --- |
| **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：

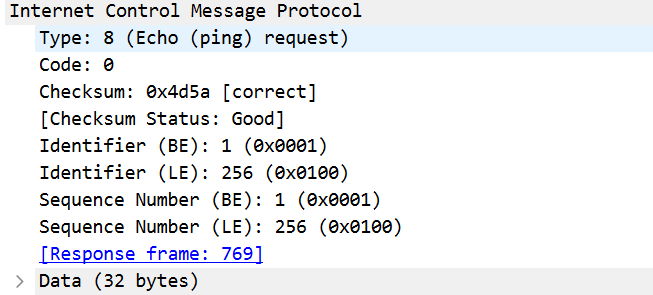
1. **Identifier：**对于每一个发送的数据报进行标识。
2. **Sequence number：**对于发送的每一个数据报文进行编号。
3. **Data section：**数据内容。
4. **ICMP捕获内容实例分析：**

一共捕获四对request包和reply包。我们以第一组为例：



ICMP 5 第一组应答

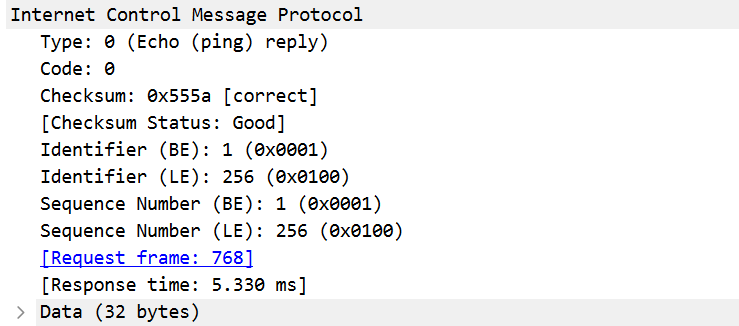
首先查看request报文的具体内容：



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：



ICMP 7 reply报文

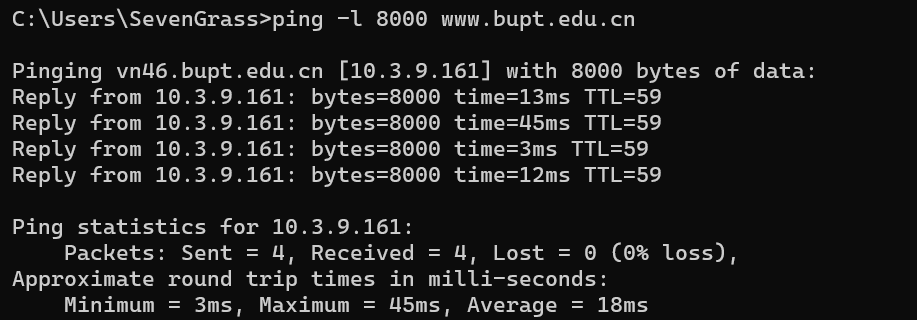
此时Type=0，Code=0，表示ping应答。

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.

## 分析 IP 数据报的分片传输过程

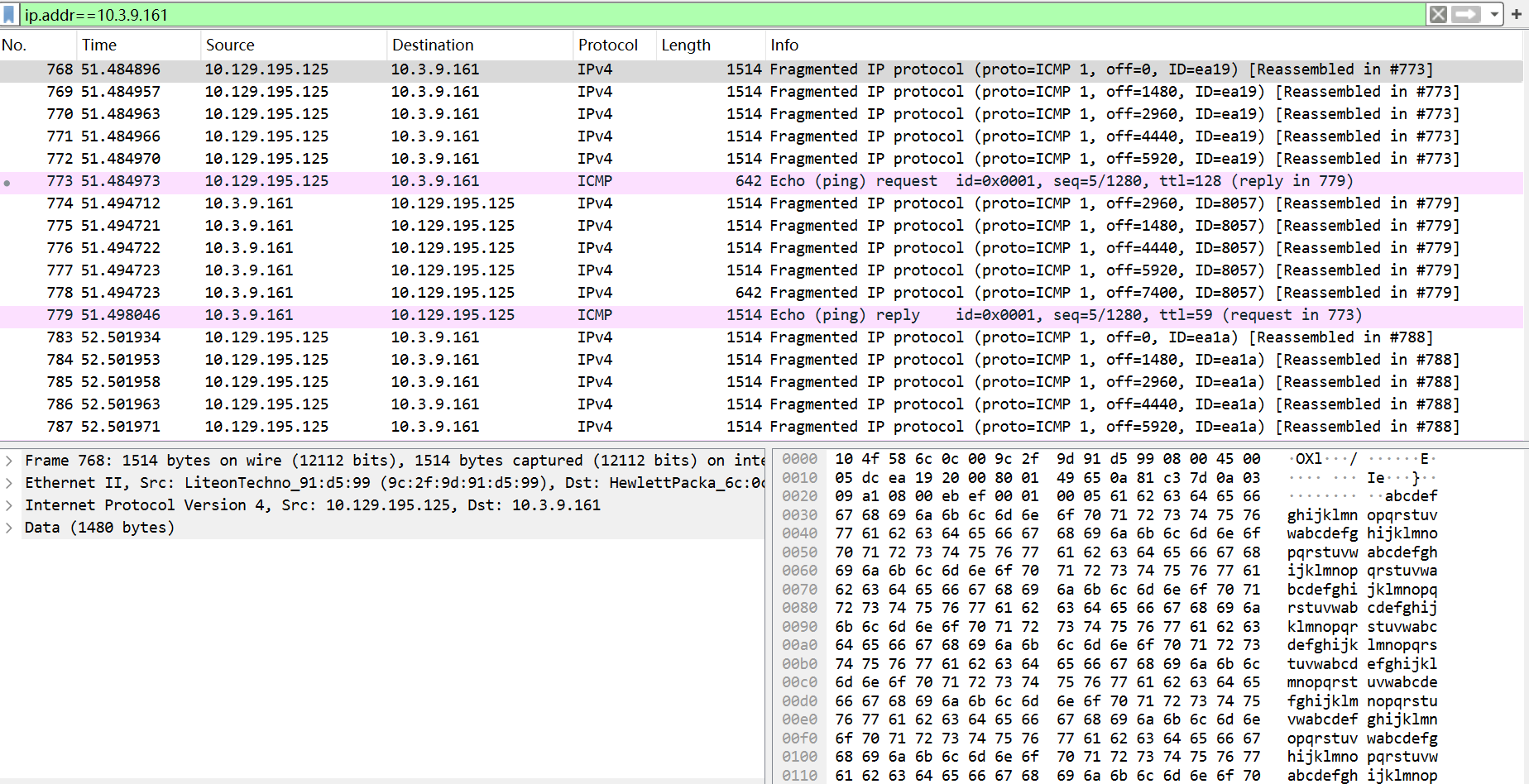
### 制作8000字节的IP数据报并发送、捕获

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



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：

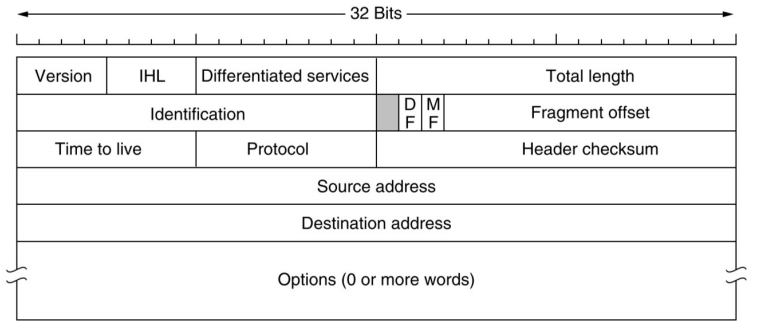


IP 2 捕获的数据报内容

### IP数据报及协议分析

1. **IPv4 包头定义：**

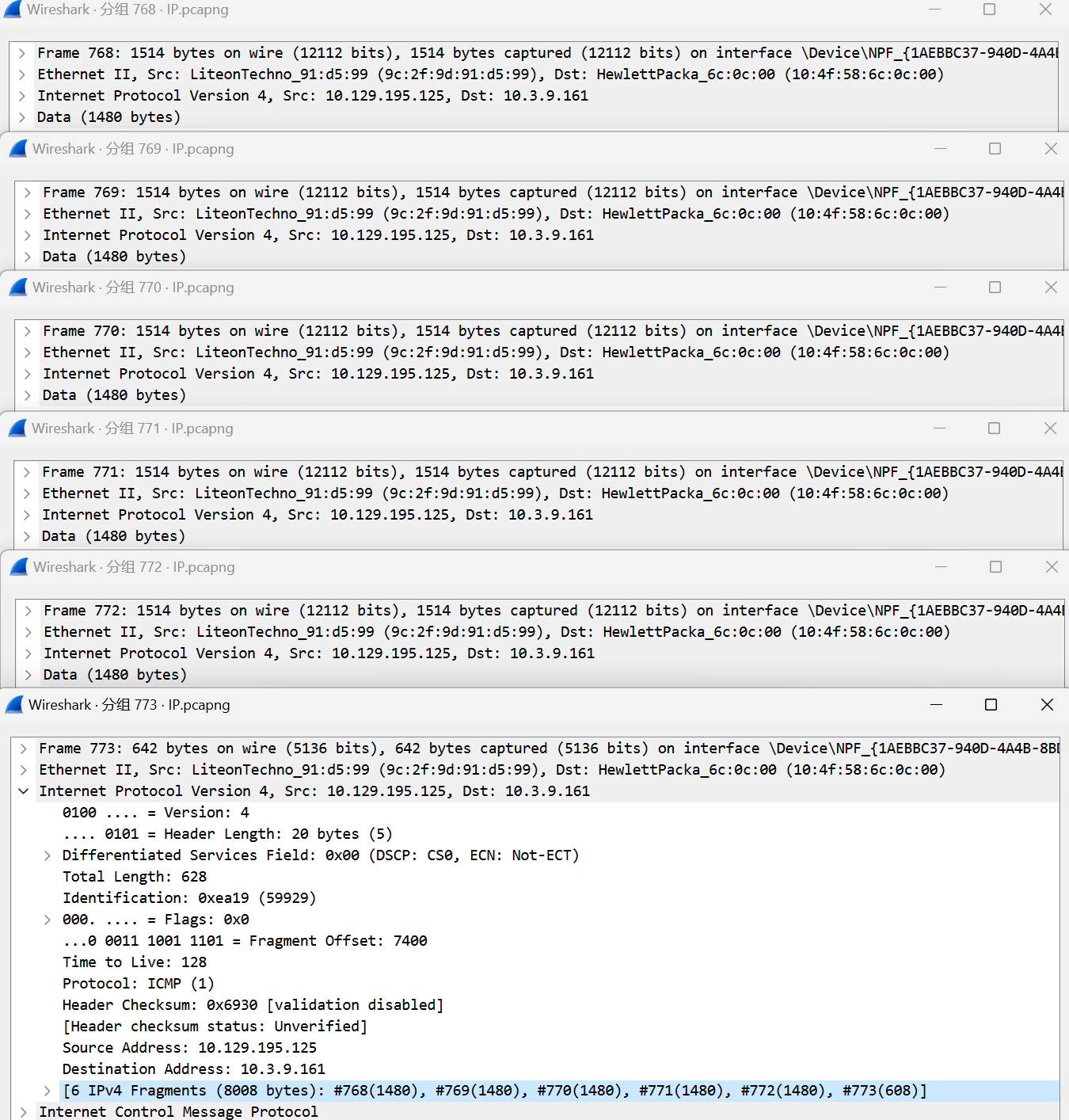
由课内理论知识的学习可知，IPv4包头格式定义如下：



IP 3 IPv4包头定义

1. **实验IP数据报分析：**

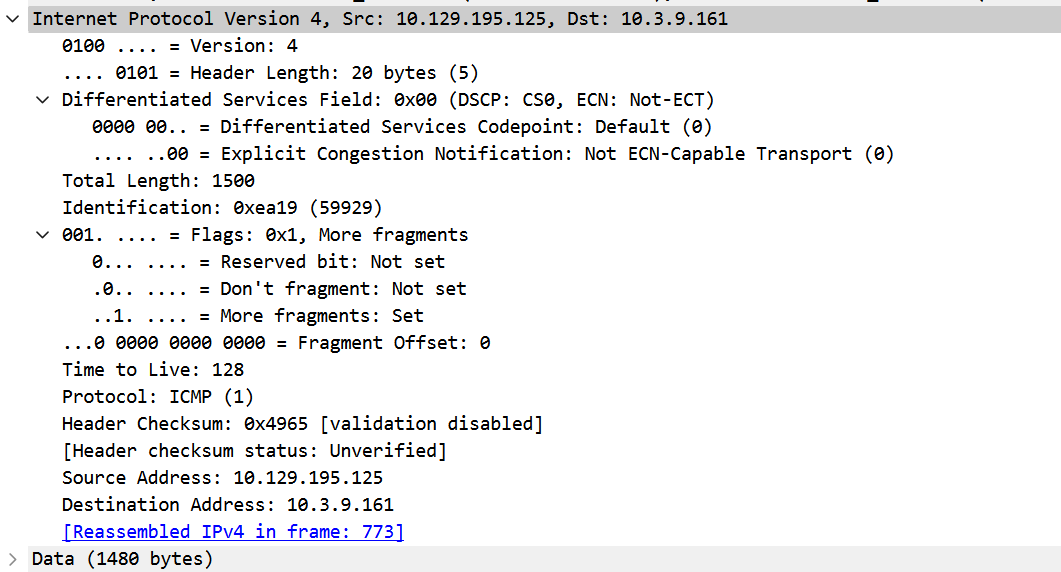
不难发现，第768 ~ 773号帧为我们发送的数据帧，即制作的8000字节数据被分为6个分片依次发送，从Wireshark中可以看到每个包的大小如下：



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：

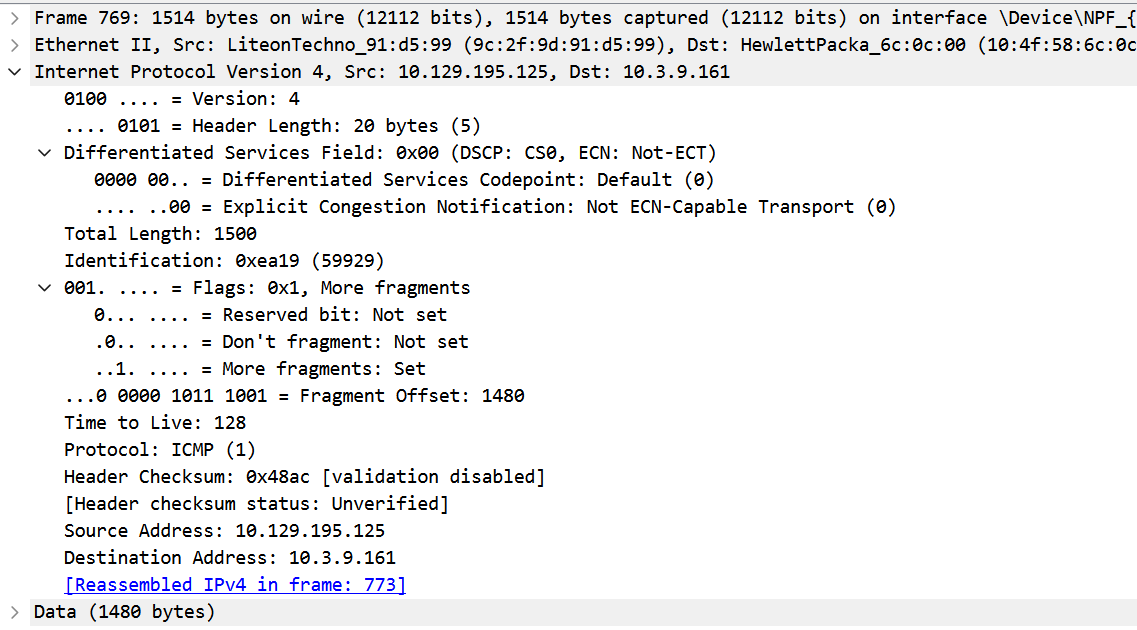


IP 5 第一个分片

|  |  |  |
| --- | --- | --- |
| **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：

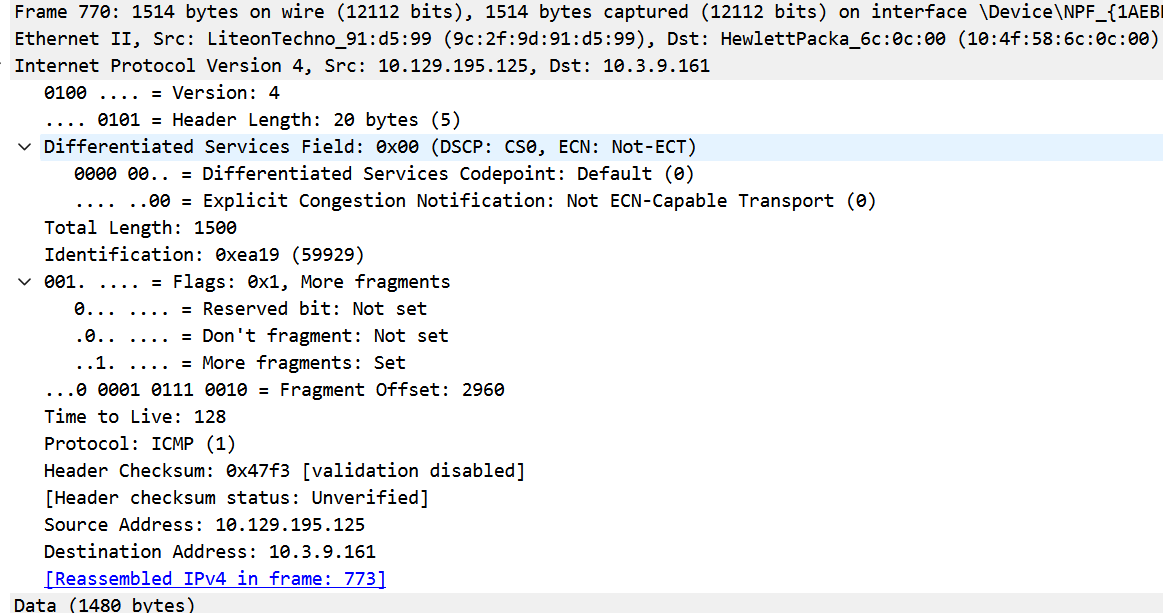


IP 7 第二个分片

|  |  |  |
| --- | --- | --- |
| **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：

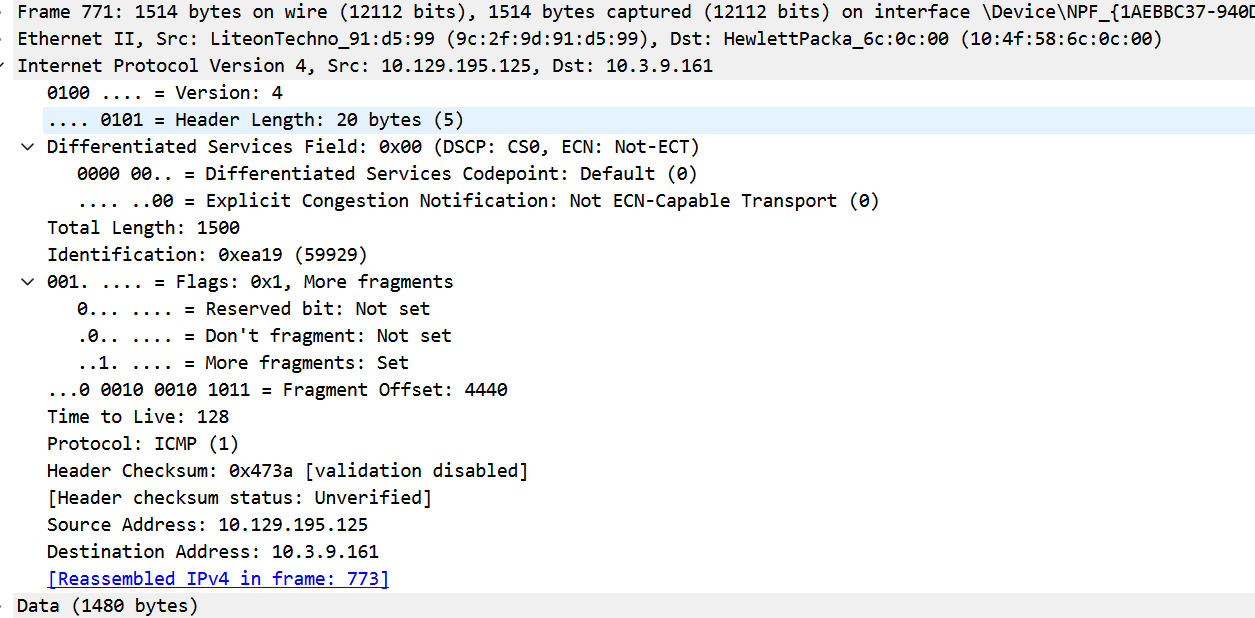


IP 9 第三个分片

|  |  |  |
| --- | --- | --- |
| **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：

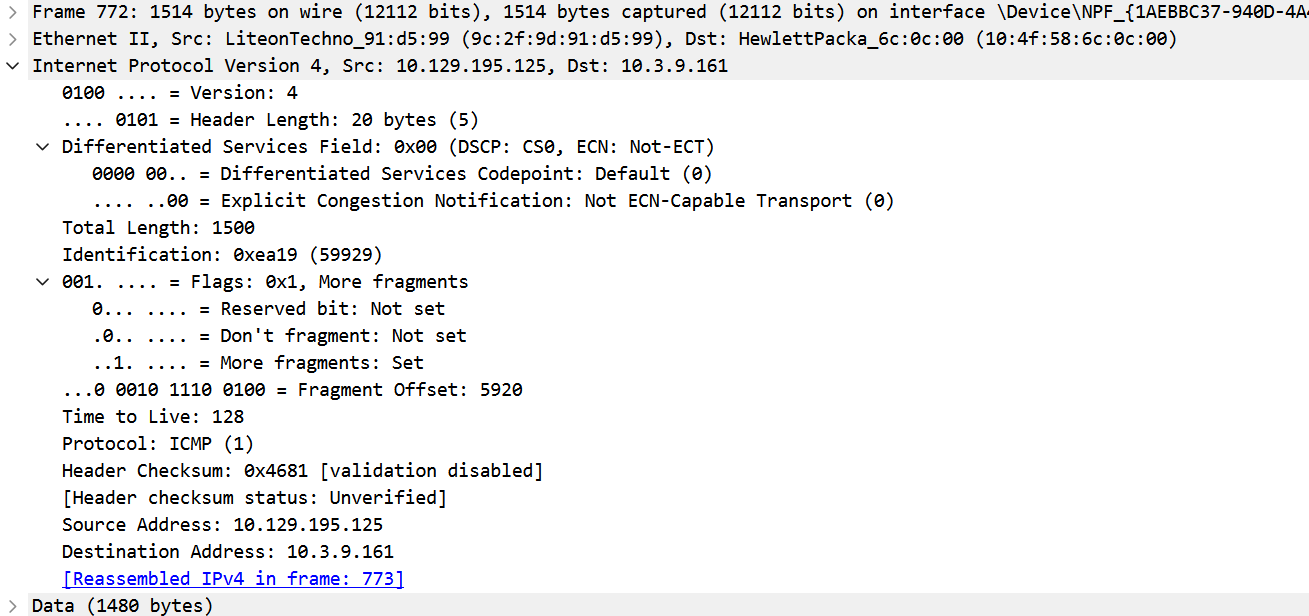


IP 11 第四个分片

|  |  |  |
| --- | --- | --- |
| **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：

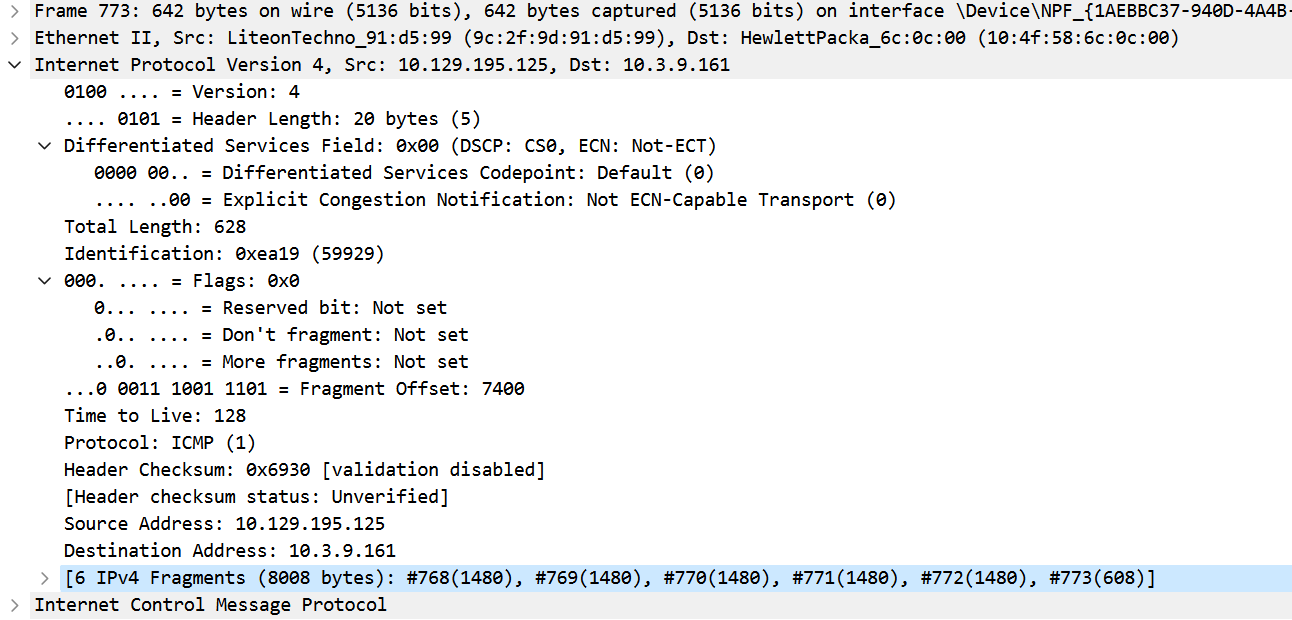


IP 13 第五个分片

|  |  |  |
| --- | --- | --- |
| **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：



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。

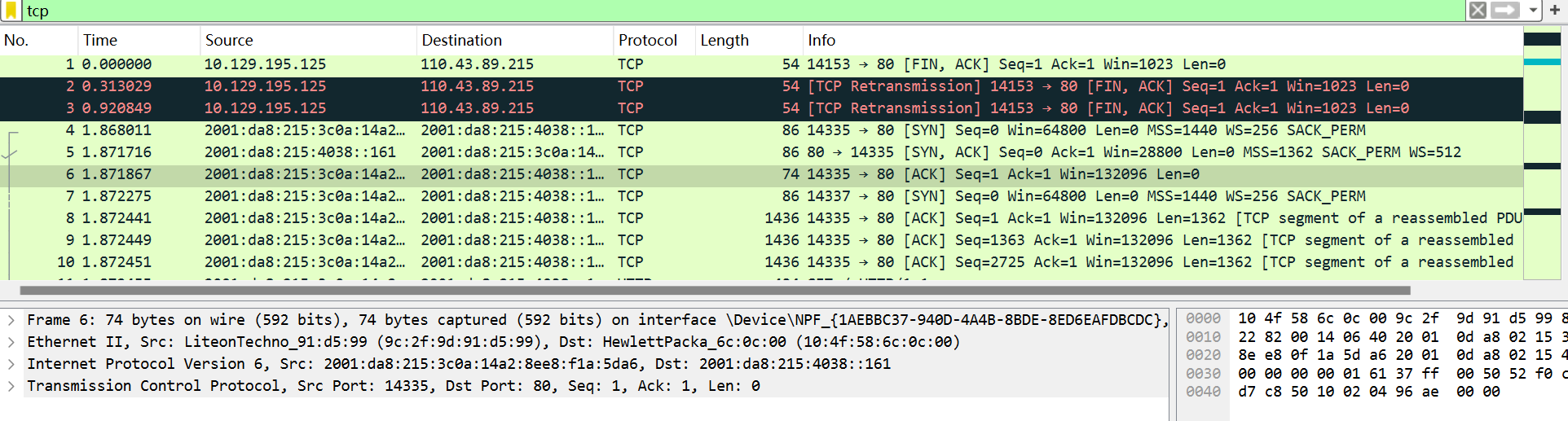
## 捕获建立连接和释放连接过程的 TCP 报文段并分析

### 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：

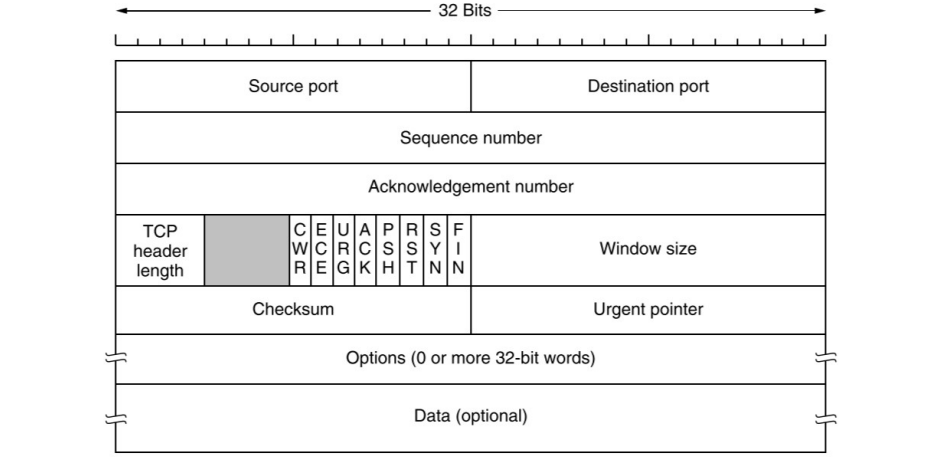


TCP 1 捕获的TCP报文段

### 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：

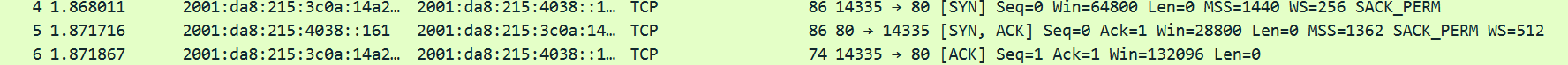


TCP 2 TCP头格式

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。

### TCP建立连接分析

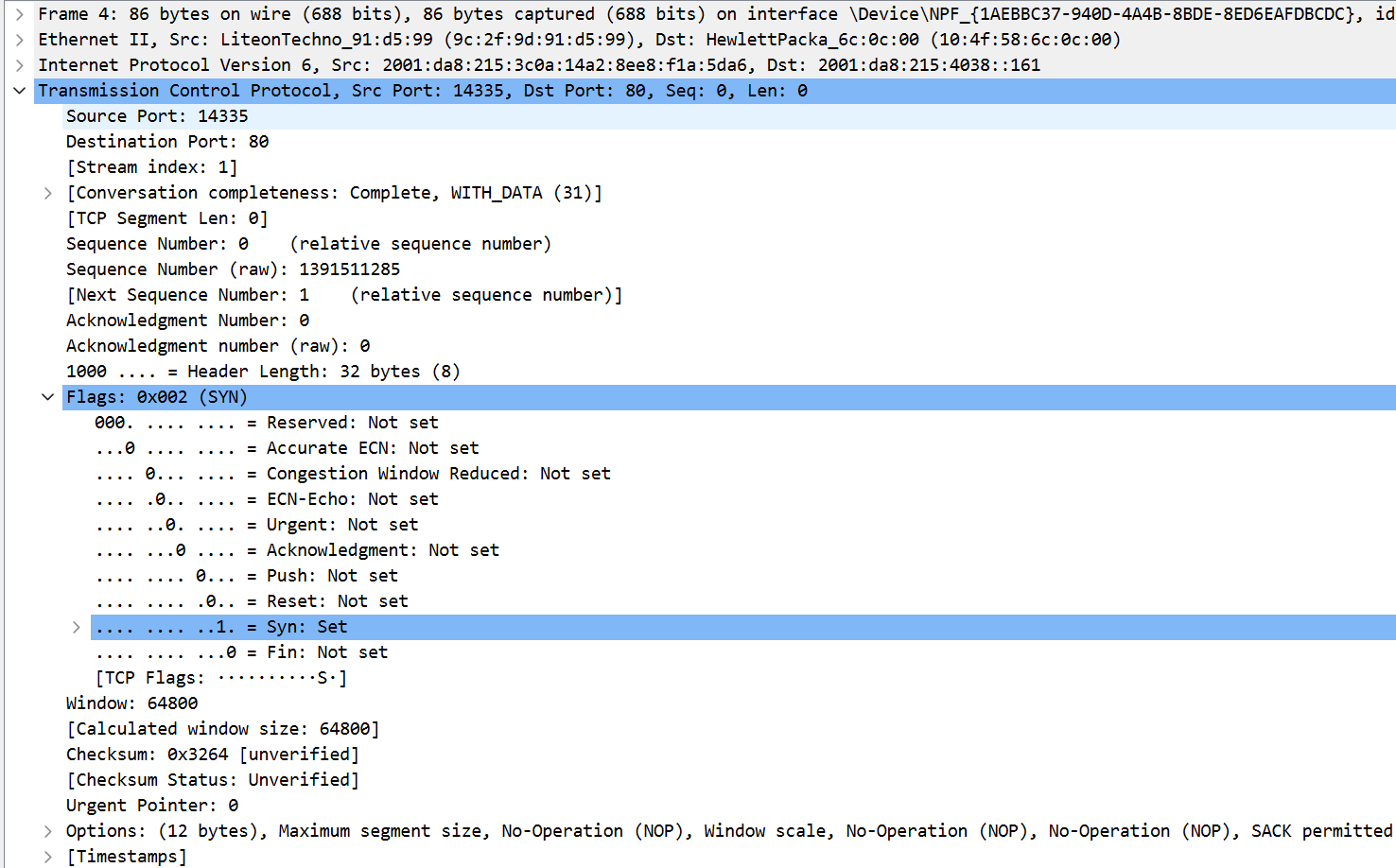
序号4-6：客户端向服务器发送 SYN 请求建立连接，服务器返回 SYN+ACK 应答，客户端返回 ACK 完成连接建立。即“三次握手”



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：

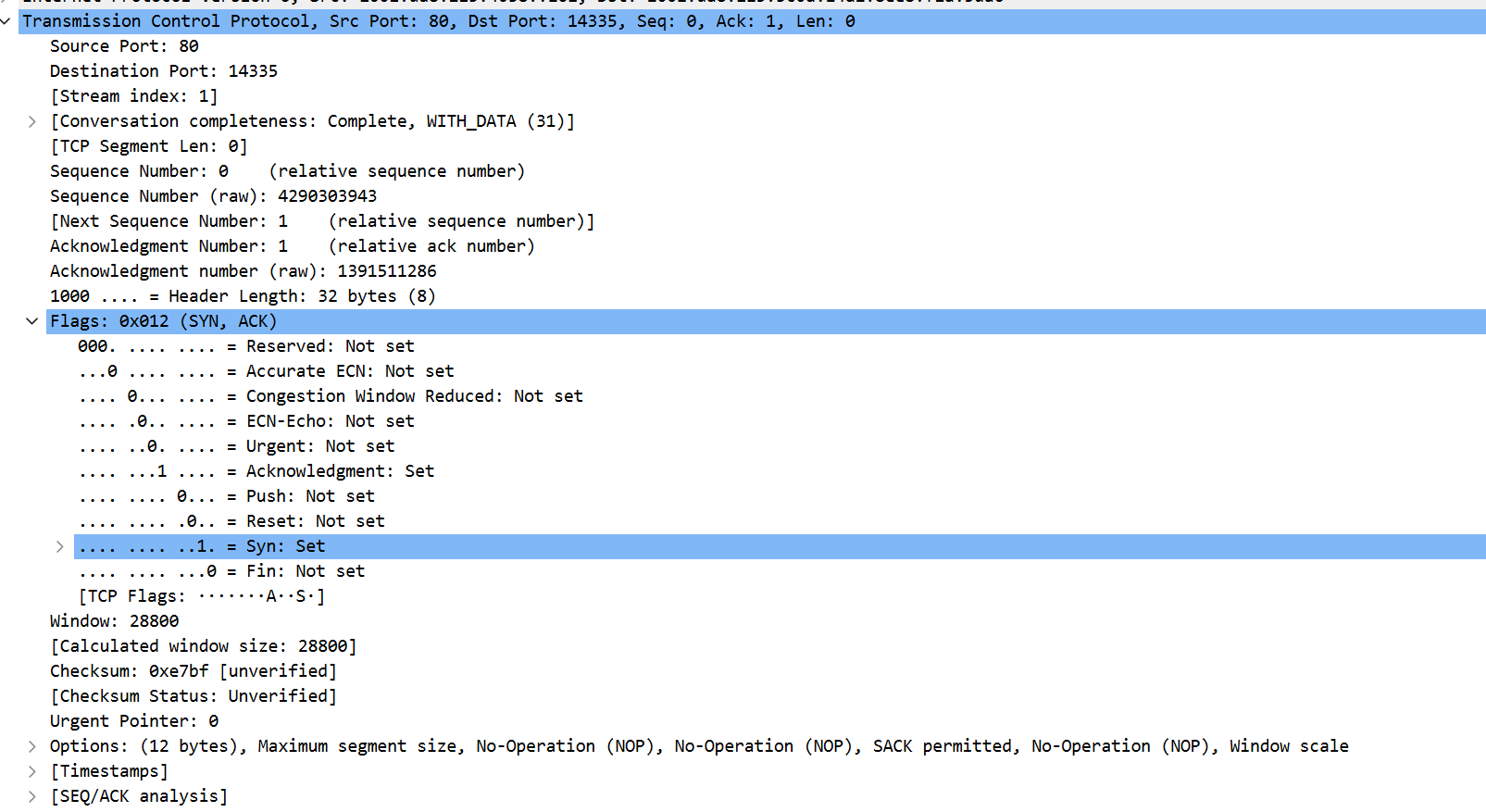


TCP 4 建立连接阶段SYN

|  |  |  |
| --- | --- | --- |
| **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：

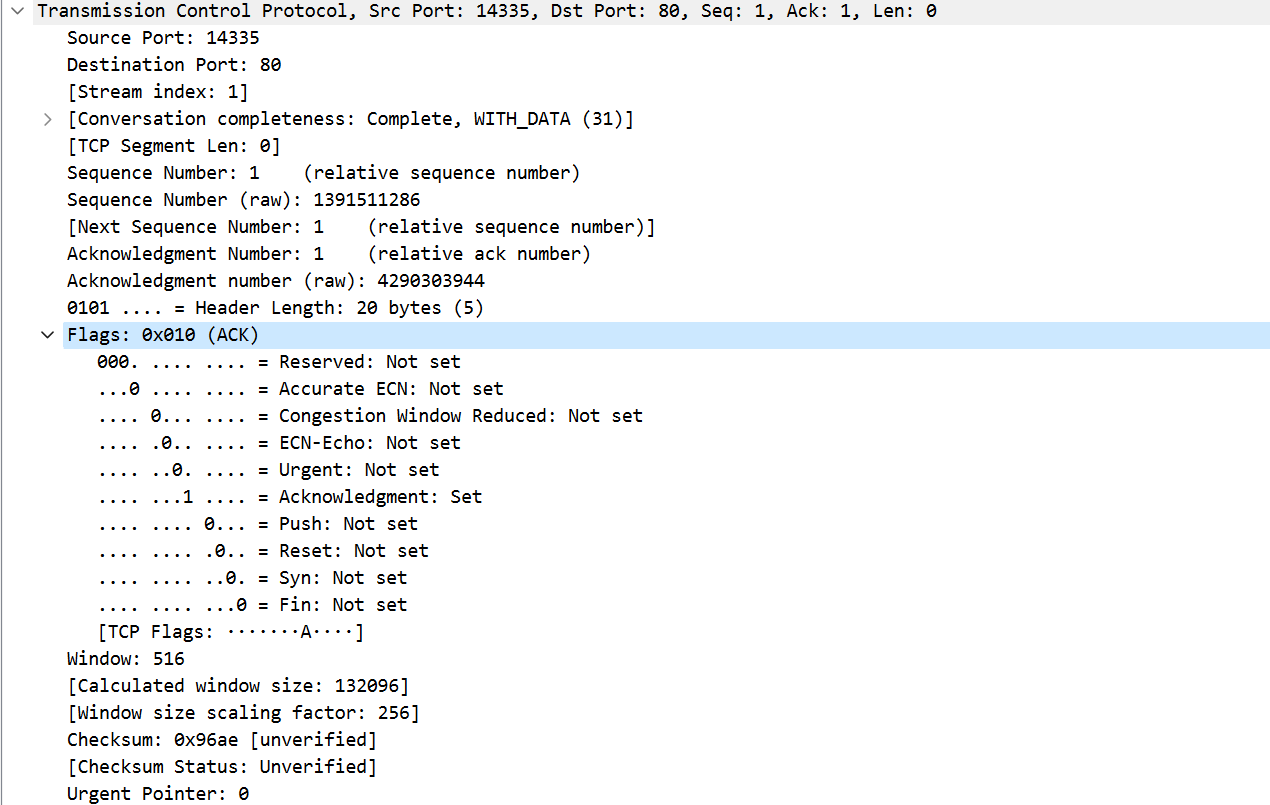


TCP 6 建立连接时SYN+ACK

|  |  |  |
| --- | --- | --- |
| **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：



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：

|  |  |  |
| --- | --- | --- |
| **客户端** |  | **服务器** |
| **|** | **SYN(seq=x)** | **|** |
| **|** | **=============================== >** | **|** |
| **|** |  | **|** |
| **|** | **SYN(seq=y) ACK(seq=x+1)** | **|** |
| **|** | **< =============================** | **|** |
| **|** |  | **|** |
| **|** | **ACK(seq=y+1,ack=x+1)** | **|** |
| **|** | **============================= >** | **|** |

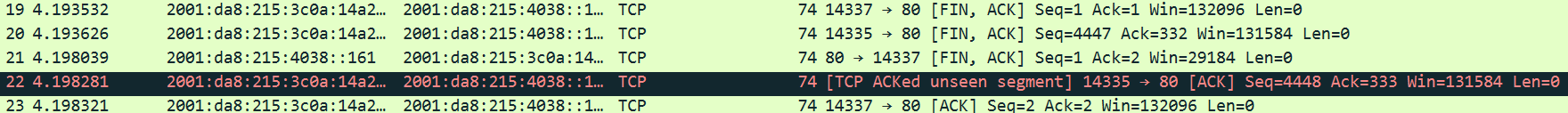
TCP 10 l=连接建立消息序列图

### 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。

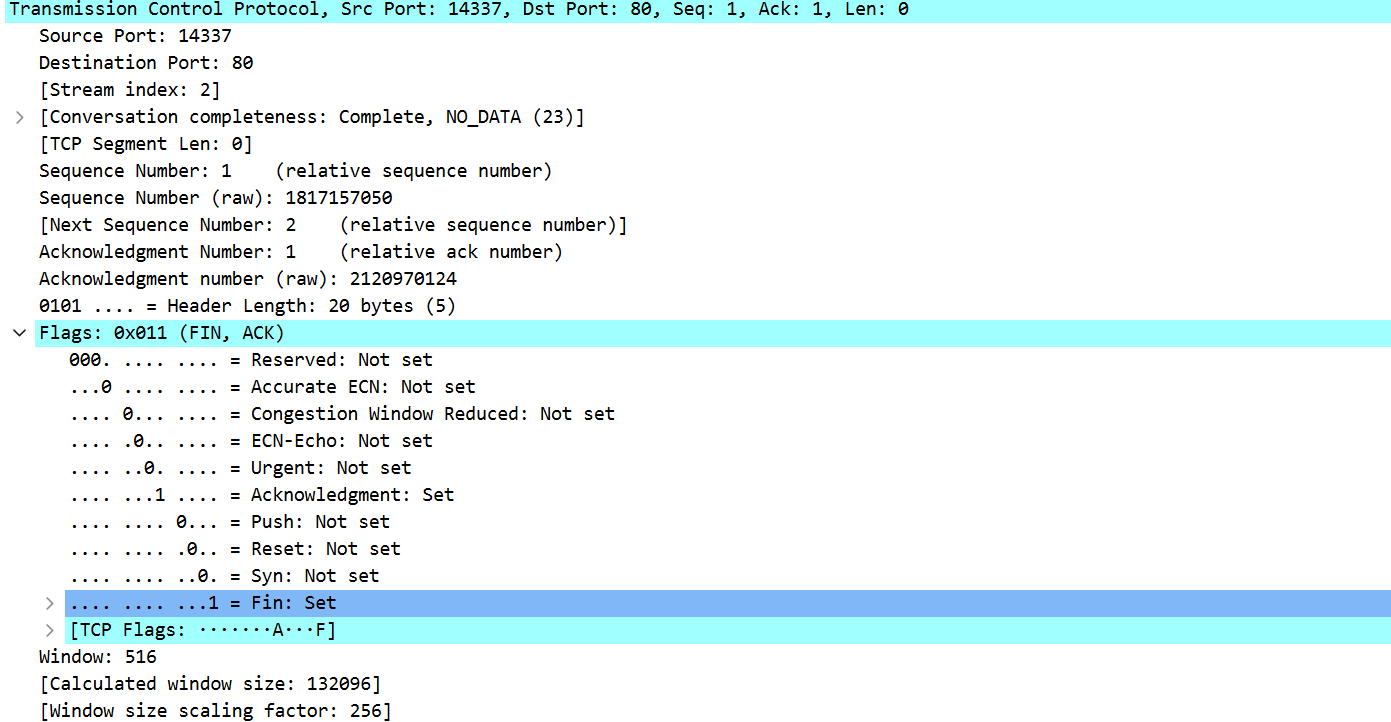
特别需要注意的是：在本次实验中，在释放连接的过程中，只抓到了对应的3个包，但对于“四次挥手”，其实是将第二次和第三次合并在了一起，具体分析见下文。

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”：

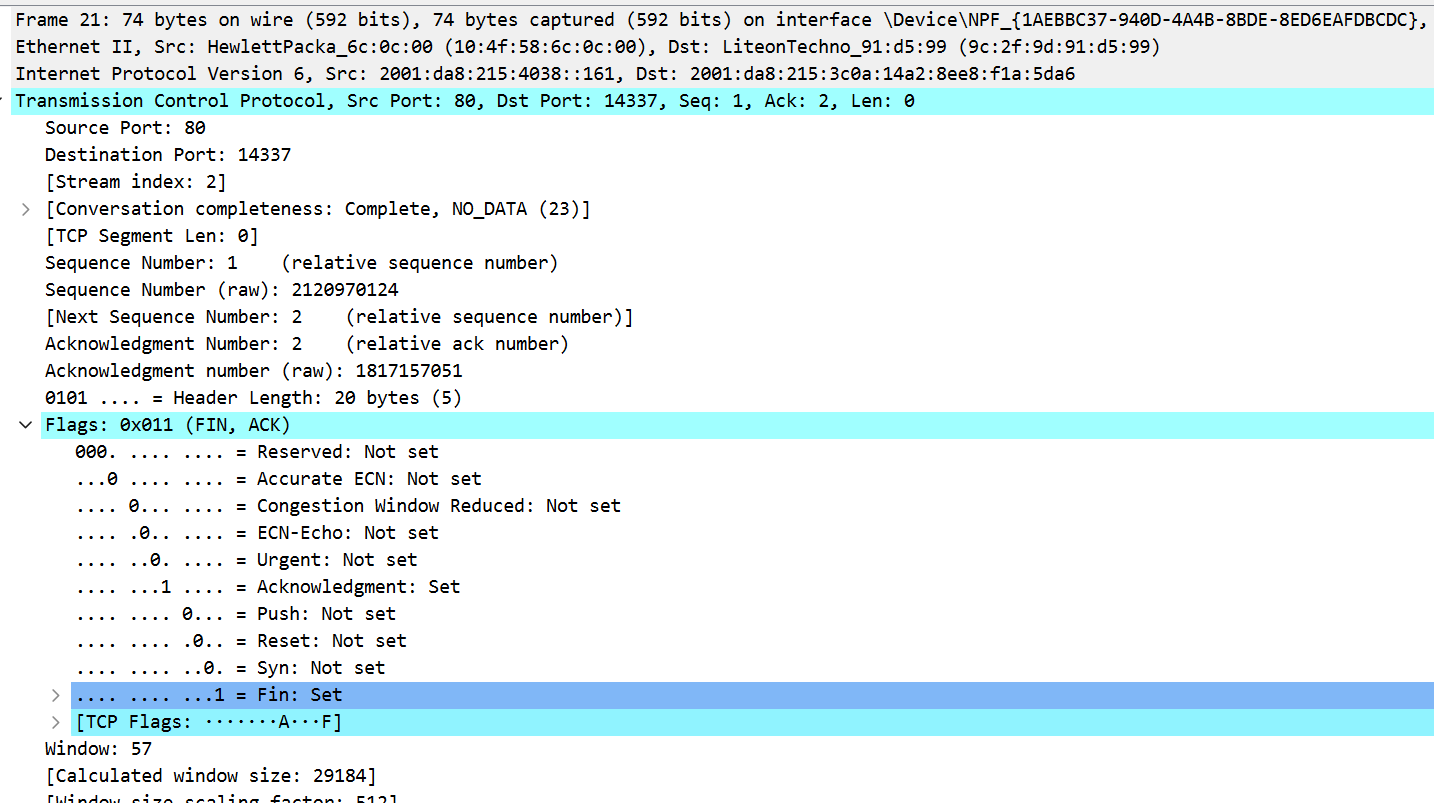


TCP 12 释放连接FIN

|  |  |  |
| --- | --- | --- |
| **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 |

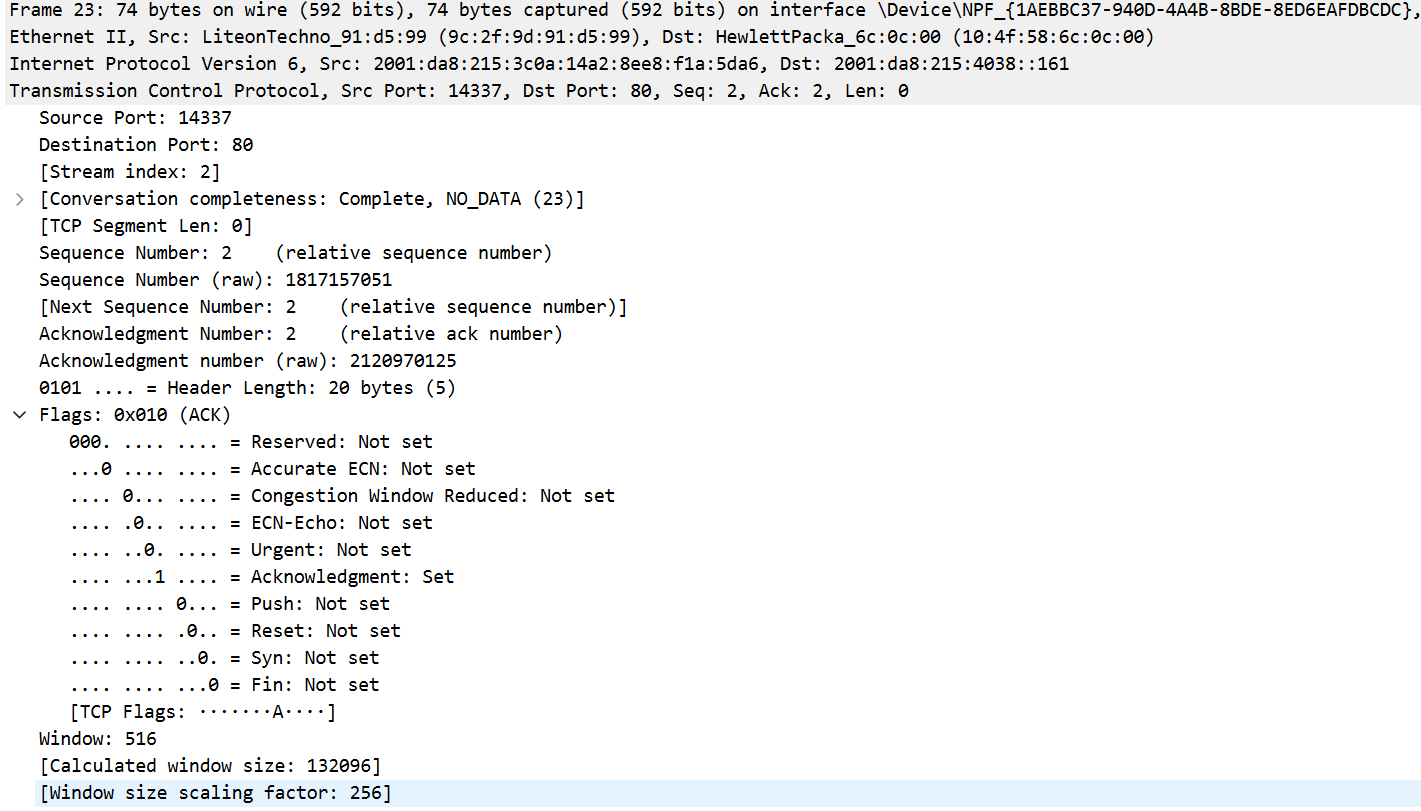
TCP 15FIN+ACK字段分析

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：



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：

|  |  |  |
| --- | --- | --- |
| **客户端** |  | **服务器** |
| **|** | **FIN(seq=a, ack=b)** | **|** |
| **|** | **======================== >** | **|** |
| **|** |  | **|** |
| **|** | **FIN + ACK(seq=b, ack=a+1)** | **|** |
| **|** | **< ========================** | **|** |
| **|** |  | **|** |
| **|** | **ACK(seq=a+1, ack=b+1)** | **|** |
| **|** | **========================= >** | **|** |

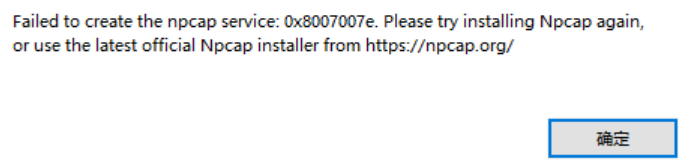
TCP 18 释放连接消息序列图

# 实验结论和实验心得

## 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：



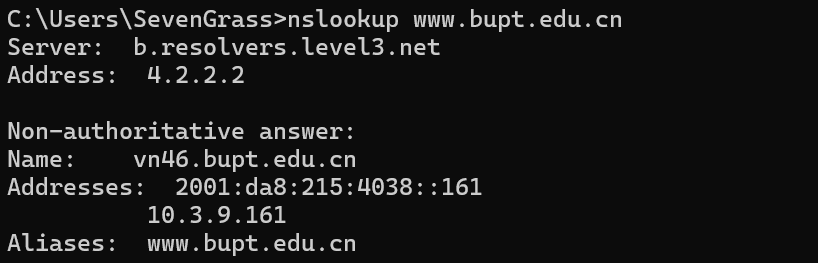
尝试了去Npcap的官网下载，依然失败；关闭了杀毒软件等，依然失败。

**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。

1. **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](http://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](http://www.bupt.edu.cn) 查看学校官网的IP地址如下图：



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](http://www.bupt.edu.cn) 也会自动显示该网站的IP地址。

1. **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》](https://zorrozou.github.io/docs/tcp/wavehand/TCP_Wavehand.html)得到解决。

## 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。