

网络安全实验报告

题 目 捕包软件的使用与实现

专 业 信息安全

学 号 120L020121

学 生 刘旭

指 导 教 师 王彦

**一、实验目的**

理解捕包程序捕包过程，可以自己编程捕包并从数据包中解析出需要的信息。

# 二、实验内容

1. 熟练使用 sniffer 或 wireshark 软件，对协议进行还原（能够找访问网页的四元组）；只需要写报告，不需要在实验课检查。
2. 利用 libpcap 或 winpcap 进行编程，能够对本机的数据包进行捕获分析（比如将本机所有数据包的四元组写到指定文件），按照自己的设想撰写需求分析和详细设计。（实验课检查程序）

# 三、实验过程

**（一）使用 wireshark 软件对协议进行还原**

## 实验基本信息：

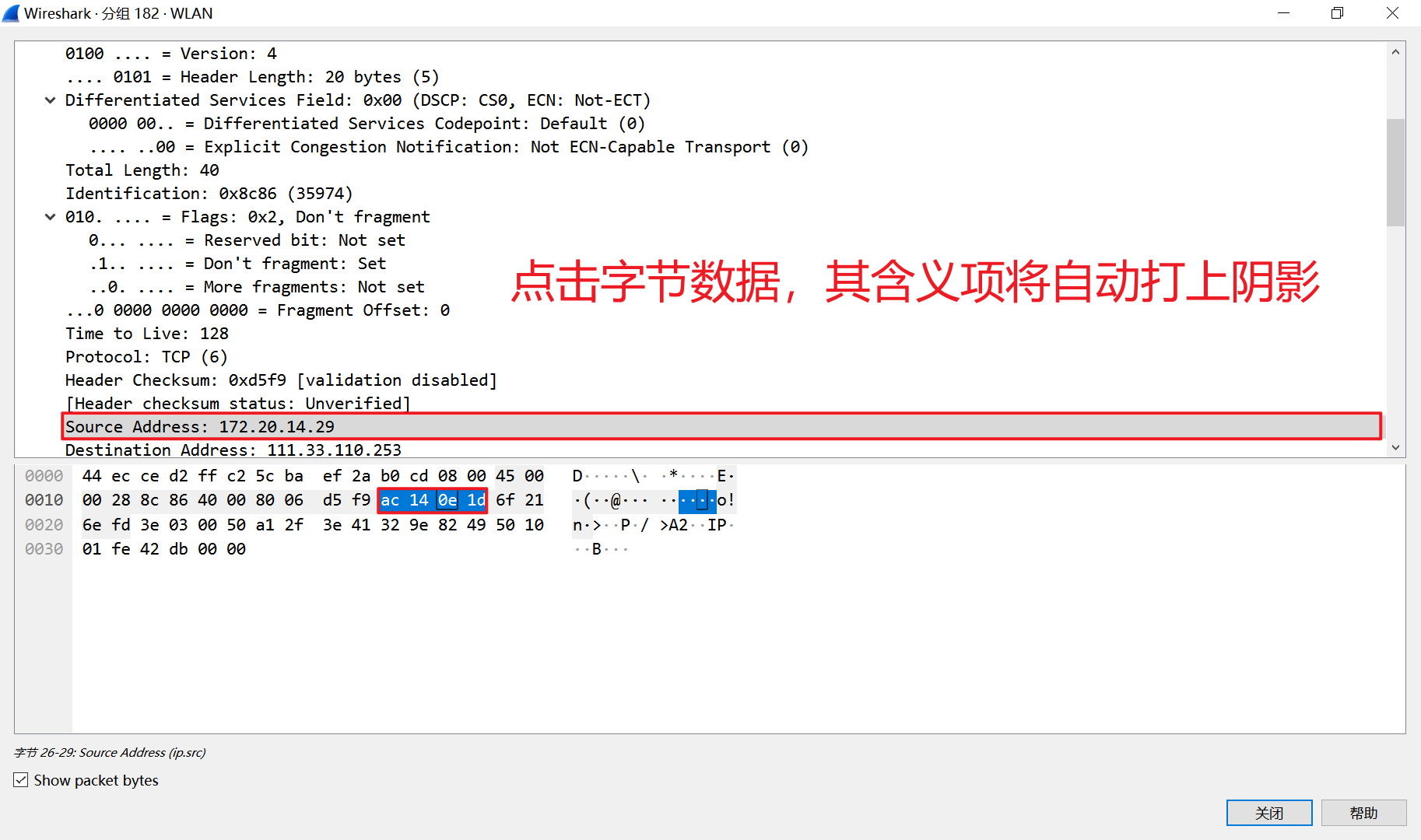
实验环境：Windows10 x64

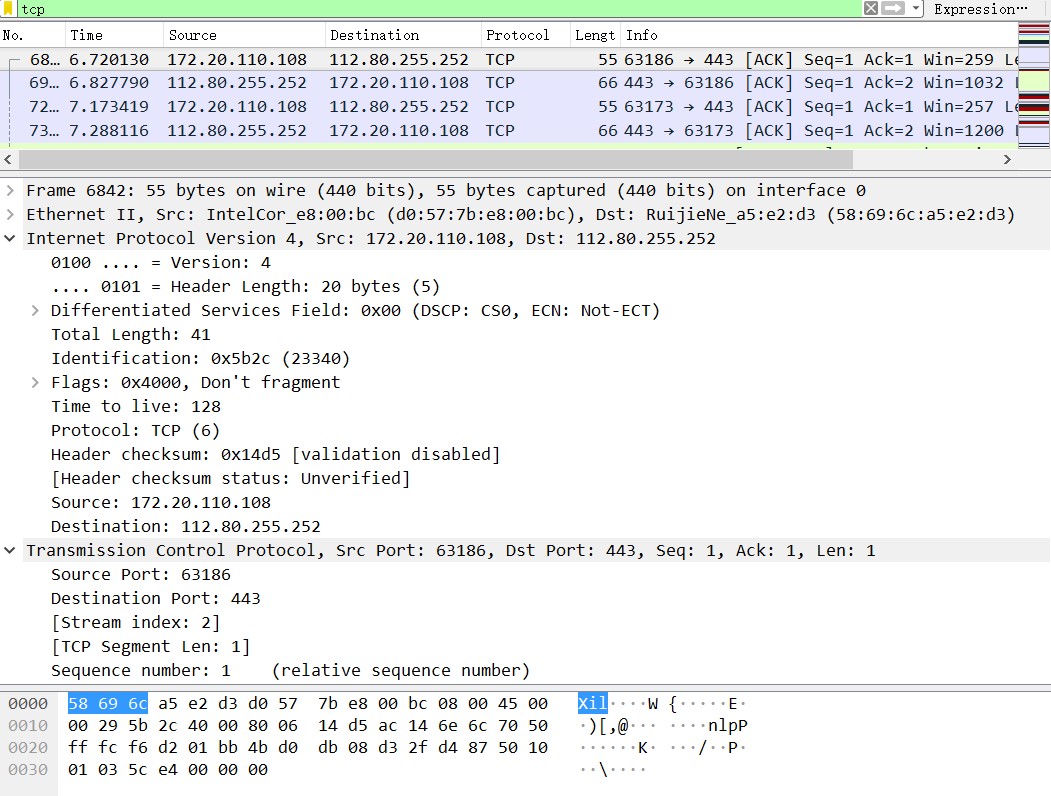
WireShark 4.0.0

## 1. 捕包并分析四元组

(1) TCP 分析

双击 TCP 数据包即可打开详情页，勾选 Show packet bytes ，点击对应的字节数据，窗口将自动给其代表的含义项打上阴影，一项一项点过去即可知道每一项所代表的含义。如图所示：



对字节数据逐项点击过去，可从IP数据包的数据获取相关信息如下：  


截图中这个 TCP 数据包，源 IP 为 172.20.110.108，目的 IP 为 112.80.255.252，源

端口为 63186，目的端口为 443。

分析：以太网头部：

前 6 个字节 58 69 6c a5 e2 d3 为目的主机 MAC，往后 6 个字节 d0 57 7b e8 00 bc 为源主机 MAC，

往后 2 个字节为上层协议，0x0800 表示 IPv4 协议；以太网头部结束，现在是 ip 头部：往后 1 个字节 0x45 表示 IP 版本为 4，头部长度为 5，往后 1 个字节为区分服务，0x00 表示默认，

往后 2 个字节为总长度，0x0029 = 41，往后 2 个字节为 id，值为 0x5b2c（23340），

往后 2 个字节为标志位+片偏移，值为 0x4000，

往后 1 个字节为 ttl，值为 0x80 = 128，

往后 1 个字节为上层协议，0x06 表示 TCP 协议，往后 2 个字节为头部校验和，值为 0x14d5，

往后 4 个字节为源 ip 地址 0xac146e6c，转换为 10 进制就是 172.20.110.108，往后4个字节为目的ip地址0x7050fffc，转换为10进制就是112.80.255.252；

ip 头部结束：现在是 TCP 头部：

往后 2 个字节为源端口 0xf6d2 = 63186，往后 2 个字节为目的端口 0x01bb = 443，往后 4 个字节为 seq 0x4bd0db08，往后 4 个字节为 ack 0xd32fd487，

往后 1 个字节为头部长度 0x50 = 80（其实是 4 位），

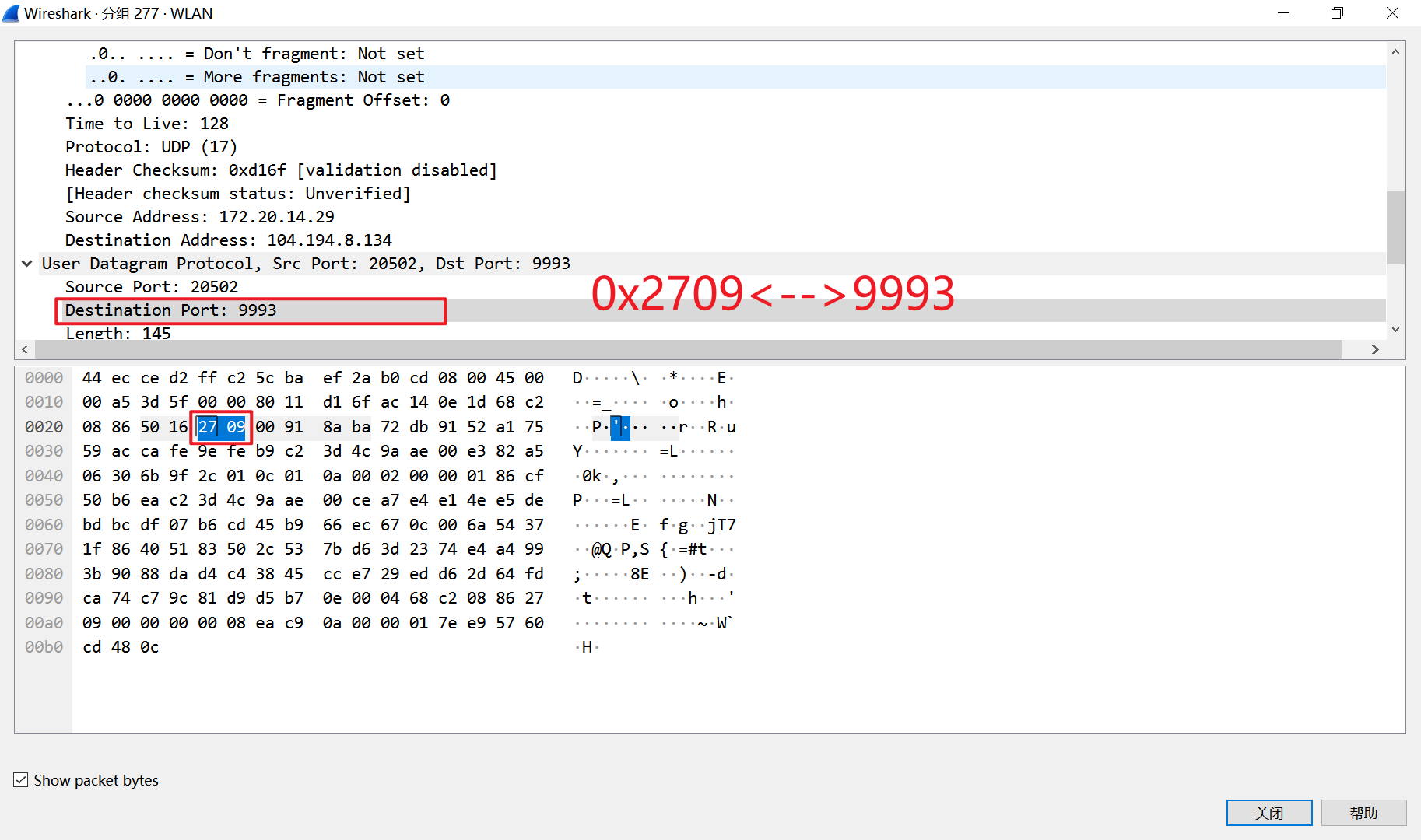
往后 1 个字节为标志位（其实是 6 位），

往后 2 个字节为窗口大小 0x0103 = 259，往后 2 个字节为校验和 0x5ce4，

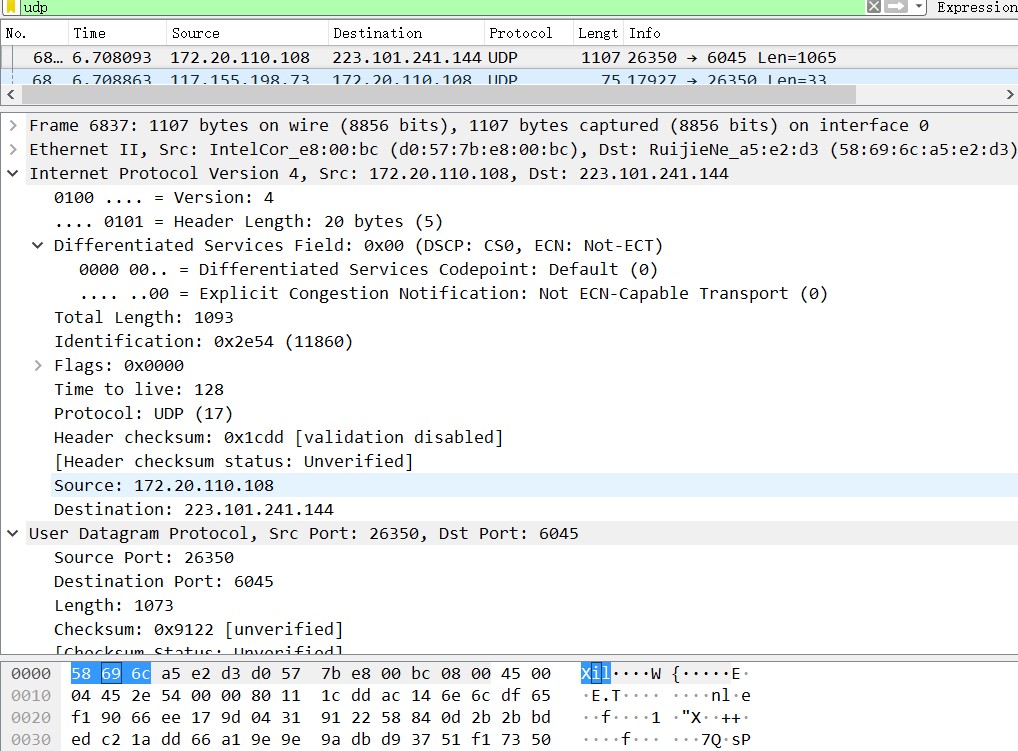
往后 2 个字节为紧急指针 0x0000； TCP 头部结束，接下来是数据。

(2) UDP 分析

UDP分析类似TCP分析，逐项点击字节数据即可知道对应的信息。



分析结果如下：



截图中这个 UDP 数据包，源 IP 为 172.20.110.108，目的 IP 为 223.101.241.144，

源端口为 26350，目的端口为 6045。

分析：

以太网头部和 ip 头部与 TCP 协议基本相同，直接分析 UDP 头部，从第三行第三个字节开始：

2 个字节源端口 0x66ee = 26350，

2 个字节目的端口 0x179d = 6045，

2 个字节总长度 0x0431 = 1073， 2 个字节校验值：0x9122；

后面全都是数据。

**（二）利用 libpcap 编写捕包软件**

**实验基本信息：**

实验环境：Ubuntu 20.04 x64

编程语言：C 语言

## 1. 需求分析

本程序需要运用 libpcap 来捕获本机数据包，并获取数据包中的四元组，将其展示给

用户。

程序功能：

(1)捕获本机数据包（可以自定义过滤条件）；

(2)逐层解析数据包，获得 IPv4 数据包的源 ip、目的 ip、源端口、目的端口；

(3)将上述四元组写入文件（每次运行程序都新生成一个文件）。

## 2. 环境配置

终端运行sudo apt-get install libpcap-dev 即可安装 libpcap 库。

## 3. 回调函数设计

void packet\_handler(u\_char \*user, const struct pcap\_pkthdr \*header, const u\_char \*packet);

回调函数是由主程序（在这个例子中是 main 函数）注册的，当有数据包到达时，pcap 库会调用该函数。

函数的参数列表包括：

user：指向传递给 pcap\_loop 函数的用户数据，这里是输出文件指针。

header：指向数据包的 pcap 头。

packet：指向数据包的实际内容。

回调函数首先解析数据包的以太网头部，然后判断协议类型是 IP 协议。如果是 IP 协议，回调函数再解析 IP 头部，然后根据协议类型（TCP 或 UDP）输出四元组和协议类型。输出使用 fprintf 函数，第一个参数是输出文件指针，后面是要输出的内容，格式类似于 printf 函数。

在这个程序中，回调函数的主要作用是从数据包中解析出四元组和协议类型，并将这些信息输出到指定的文件中。回调函数使用了标准的网络协议结构体，如 ether\_header、ip、tcphdr 和 udphdr，并使用 ntohs 和 inet\_ntoa 函数将网络字节序转换为主机字节序。回调函数还使用了标准的时间函数 time 和 localtime，以及 strftime 函数将当前时间格式化为字符串，并将其用作输出文件名的一部分。

## 5. 主函数设计

## 这个主函数主要分为以下几个步骤：

## 1. 定义一些常量和变量，包括最大数据包大小、网络设备名、用于存储错误信息的缓冲区、用于存储过滤器程序的结构体、过滤器表达式、网络地址和掩码、输出文件名等。

## 2. 打开网络接口，使用pcap\_open\_live函数打开网络设备，并指定数据包的最大大小、是否开启混杂模式、超时时间等参数。

## 3. 获取网络接口地址和掩码，使用pcap\_lookupnet函数获取网络地址和掩码。

## 4. 编译和设置过滤器，使用pcap\_compile和pcap\_setfilter函数编译并设置过滤器。

## 5. 创建输出文件，使用time函数获取当前时间，并使用strftime函数将时间转换为文件名格式，然后使用fopen函数创建输出文件。

## 6. 开始捕获数据包，使用pcap\_loop函数循环捕获数据包，并使用packet\_handler函数处理捕获到的每个数据包。在调用pcap\_loop函数时，需要将输出文件指针作为第四个参数传递给packet\_handler函数。

## 7. 关闭输出文件和网络接口，使用fclose和pcap\_close函数分别关闭输出文件和网络接口。

## 总体来说，这个主函数的设计比较简单，主要是利用pcap库提供的函数和结构体完成网络数据包捕获和处理的功能。其中，pcap\_loop函数和packet\_handler函数是关键，它们实现了数据包捕获和处理的核心逻辑。

## 6. 编译运行

按照以下步骤编译并运行这个main.c文件：

1. 在终端中进入main.c所在的目录。
2. 使用以下命令编译main.c文件：

gcc -o main main.c -lpcap

该命令将生成一个可执行文件main。

1. 运行可执行文件：

sudo ./main

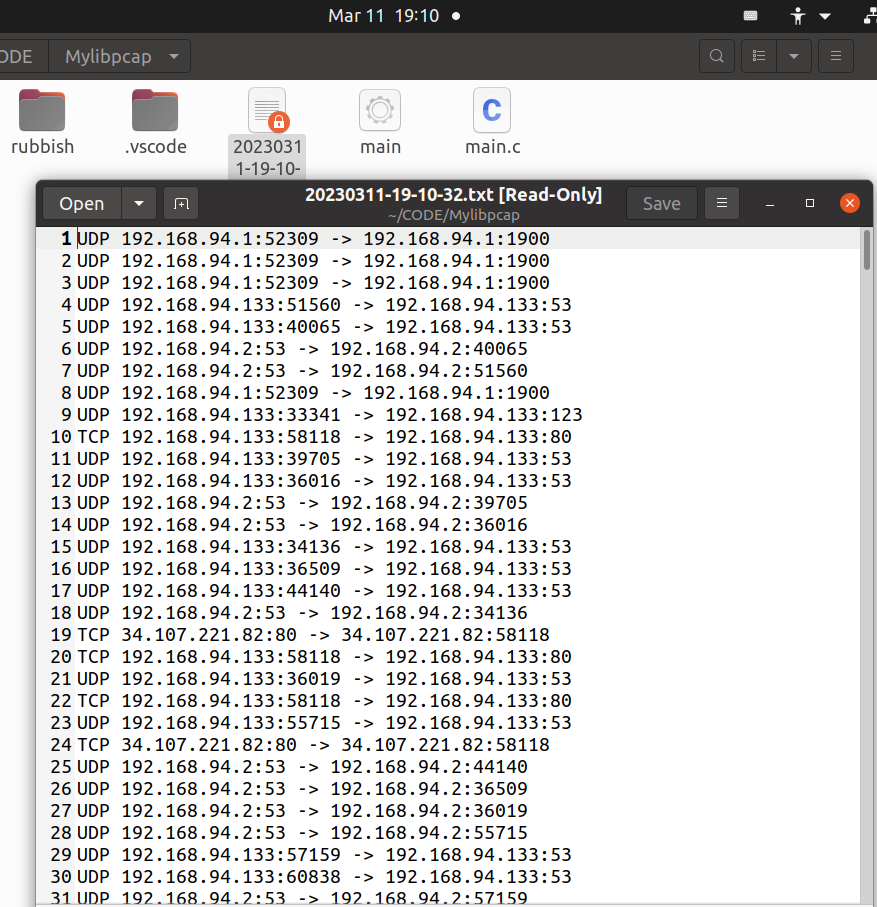
注意：使用libpcap库需要管理员权限。

运行后，程序将捕获本机数据包，并将这些信息写入txt文件中。

# 四、实验结果

1. **使用 wireshark 软件对协议进行还原**
2. **利用 libpcap 编写捕包软件**

以下为捕包时文件中部分输出：



# 

# 五、心得体会

利用libpcap库编写捕包软件的实验，让我深刻认识到网络数据包的构成和传输过程，并通过编写程序实现了网络数据包的捕获和分析。

在编写程序的过程中，我学习了如何使用libpcap库来操作网络接口，以及如何编写回调函数来对捕获到的数据包进行处理和分析。同时，我也学会了如何编写过滤器来过滤出符合条件的数据包，以及如何将捕获到的数据包写入到文件中。

通过这次实验，我对网络数据包的结构和传输过程有了更深入的了解，同时也掌握了一些常用的网络编程技巧和工具。这对我今后从事网络编程和安全相关工作都会有很大的帮助。