

**网络技术与应用课程实验报告**

**实验二：数据包捕获与分析**

****

专 业 信息安全

学 号 2113662

姓 名 张丛

班 级 信息安全一班

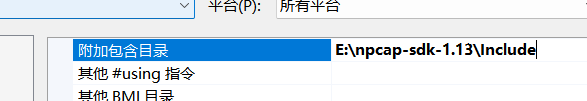
1. **实验目的**
2. 了解NPcap的架构。
3. 学习NPcap的设备列表获取方法、网卡设备打开方法，以及数据包捕获方法。
4. 通过NPcap编程，实现本机的数据包捕获，显示捕获数据帧的源MAC地址和目的MAC地址，以及类型/长度字段的值。
5. 捕获的数据报不要求硬盘存储，但应以简单明了的方式在屏幕上显示。必显字段包括源MAC地址、目的MAC地址和类型/长度字段的值。（5）编写的程序应结构清晰，具有较好的可读性。
6. **实验过程**

实验可大致分为以下几步：

* 获取设备列表打印设备信息
* 打开设备
* 对数据包实现嗅探，对数据包进行解析
* 输出捕获数据报的所有信息。

1. 编程环境：

实验中使用Visual Studio出现,无法打开源文件"pacp.h" 的问题。经过查阅，在安装Npcap后，还需要下载npcap-sdk，然后在项目中配置。





然后可以正常按照课上学的来进行实验。

1. 捕获和打印设备信息：

主要是通过pcap\_findalldevs\_ex(source, auth, alldevs, errbuf)函数来捕获设备信息。

调用成功时，pcap\_findalldevs\_ex返回0，并且alldevs参数指向网络接口链表的第一个元素。

在网络接口链表中，每一个元素都是一个pacp\_if\_t结构，如下图：

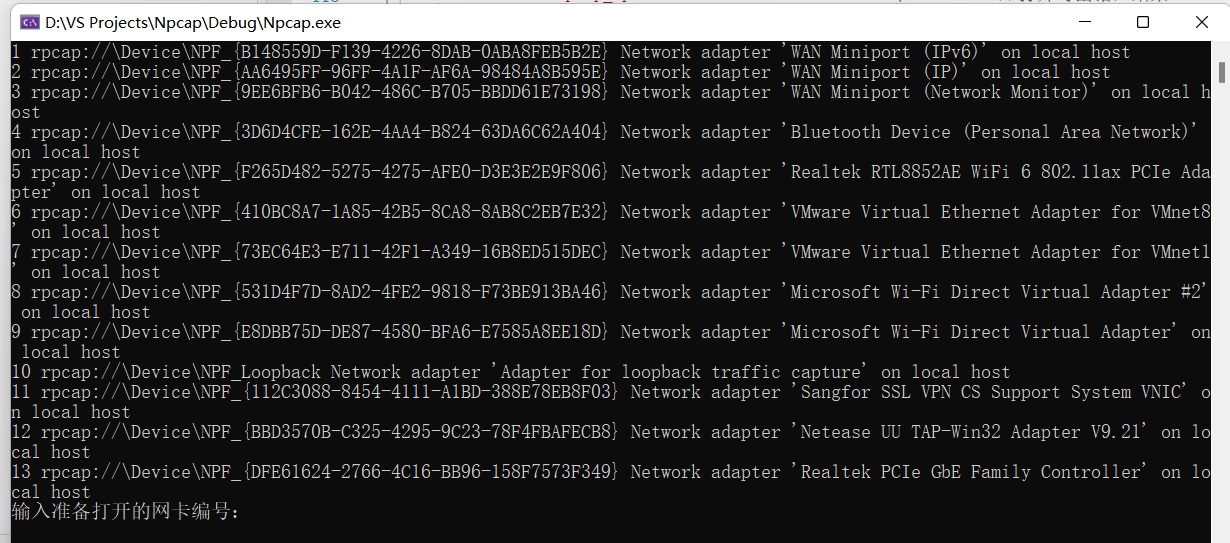


我们可以通过这个结构来打印设备信息，打印的同时为设备编号，方便后续的步骤。

代码如下：

1. pcap\_if\_t\* alldevs; *//指向设备链表*
2. pcap\_if\_t\* d;
3. int num = 0;        *//准备打开的网卡号*
4. int i = 0;          *//统计网卡，编号*
5. pcap\_t\* adhandle;
6. char errbuf[PCAP\_ERRBUF\_SIZE];
7. *//获取设备列表*
8. if (pcap\_findalldevs\_ex(PCAP\_SRC\_IF\_STRING,         *//本机接口*
9. NULL,
10. &alldevs,
11. errbuf) == -1)
12. {
13. cout << "find\_all\_dev error:" << errbuf << endl;
14. return 0;
15. }
16. for (d = alldevs; d != NULL; d = d->next)            *//打印设备列表*
17. {
18. cout << ++i << " " << d->name << " ";
19. if (d->description)
20. cout << d->description << endl;
21. else
22. cout << "No description……" << endl;
23. }
24. if (i == 0)
25. {
26. cout << "无网卡" << endl;
27. exit(1);
28. }

实现效果：



1. 打开设备：

在对某一个网络端口进行监听之前，都要将其打开，主要是使用pcap\_open（）函数。

代码如下：

1. cout << "输入准备打开的网卡编号：" << endl;
2. cin >> num;
3. if (num < 1 || num > i)
4. {
5. cout << "没有此网卡" << endl;
6. pcap\_freealldevs(alldevs);      *//释放*
7. exit(1);
8. }
9. for (d = alldevs, i = 0; i < num - 1; d = d->next, i++);   *// 迭代到选择的网卡*
10. if ((adhandle = pcap\_open(d->name,
11. 65536,                           *//常见的默认值，即64KB,决定了数据包捕获的缓冲区大小*
12. PCAP\_OPENFLAG\_PROMISCUOUS,
13. 1000,
14. NULL,
15. errbuf)) == NULL)
16. {
17. cout << " pacp\_open error:" << errbuf << endl;          *//打开时出错，结束*
18. pcap\_freealldevs(alldevs);
19. exit(1);
20. }
21. pcap\_freealldevs(alldevs);
22. 对数据包嗅探、解析

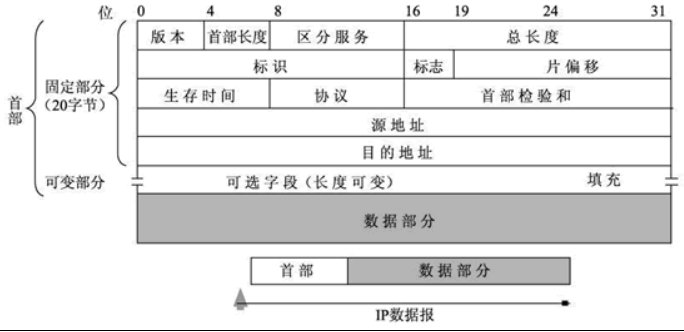
实验中我们使用 pcap\_loop 函数捕获数据包，函数会在捕获到数据包后调用回调函数 packet\_callback 来处理数据包。

捕获的数据包放在无结构缓冲区中，我们需要定义一些有关首部的数据结构来简化数据提取的过程。

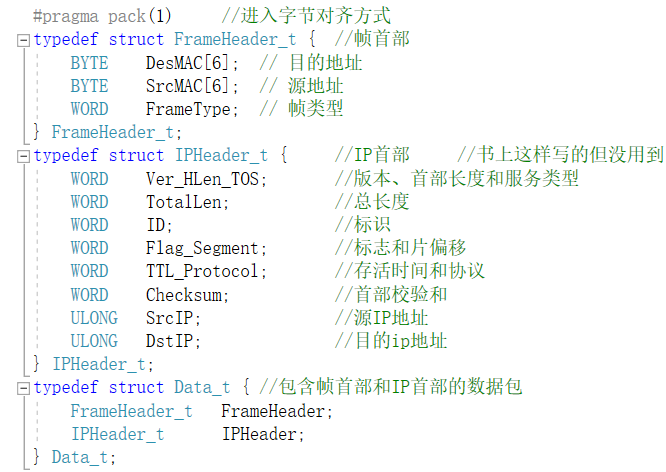
以太网帧结构：



IP数据报格式：



定义的结构体如下：



有了上述结构体后，在 packet\_callback 回调函数中，我们可以将 packet\_content 参数（指向数据包的字节数据）强制转换为 Data\_t 结构体，以便访问数据包的不同部分，如以太网帧头部和 IP 头部。

然后，我们可以从数据包中提取源MAC地址、目的MAC地址、IP地址等信息。

代码如下：

回调函数packet\_callback

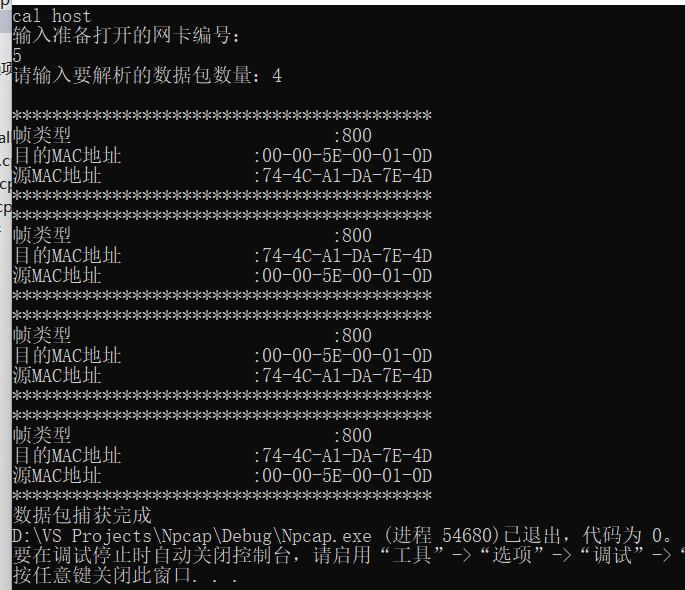
1. void packet\_callback(u\_char\* argument,
2. const struct pcap\_pkthdr\* packet\_header,
3. const u\_char\* packet\_content)
4. {
5. Data\_t\* IPPacket;
6. IPPacket = (Data\_t\*)packet\_content;                   *//将 packet\_content 强制类型转换为 Data\_t\* 类型，以访问字段*
7. WORD Kind = ntohs(IPPacket->FrameHeader.FrameType);   *//帧类型*
8. *//IPv4数据包：0x0800,  IPv6数据包：0x86DD,*
9. *//ARP数据包：0x0806,   IEEE 802.1Q VLAN 标记帧：0x8100*
11. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
12. cout << "帧类型\t\t\t\t:" << uppercase << hex << Kind << endl;
13. cout << "目的MAC地址\t\t:" << char2mac(IPPacket->FrameHeader.DesMAC) << endl;
14. cout << "源MAC地址\t\t:" << char2mac(IPPacket->FrameHeader.SrcMAC) << endl;
15. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
16. }

pacp\_loop捕获数据包：

1. int count = -1;
2. cout << "请输入要解析的数据包数量：";
3. cin >> count;
4. cout << endl;
5. pcap\_loop(adhandle, count, packet\_callback, NULL);
6. *//pcap\_loop 函数用于在捕获数据包的会话中循环处理每个捕获到的数据包*
7. 输出数据包信息

输出数据包信息代码也在回调函数内，不再赘述。

实现效果：



1. **总结**

1.熟悉了Npcap框架：

Npcap 允许用户在 Windows 平台上进行网络数据包捕获和分析。这个框架包括了一个设备驱动程序以及用户态库，使得网络应用程序可以轻松地与网络接口设备进行交互，捕获网络数据包，并对数据包进行分析。

1. 学习NPcap的设备列表获取方法、网卡设备打开方法，以及数据包捕获方法；实现了本机的数据包捕获。

实验中虽然定义了IP数据报的结构体但此次实验要求中没有让我们获取IP数据报的信息。

1. 学习到了mac的大小是14B，6B的目的地址，6B的源地址，2B的类型/长度。

其中类型字段标志了数据报的解析方式。