

# 南 开 大 学

## 计算机学院

网络技术与应用课程报告

---

### 第 4 次实验报告

---

学号：1911406

姓名：丁彦添

年级：2019 级

专业：计科

2021 年 11 月 9 日

## 第1节 实验内容说明

本次实验为 IP 数据报捕获与分析编程实验，具体要求如下：

- (1) 了解 WinPcap 的架构。
- (2) 学习 WinPcap 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。
- (3) 学习多线程程序编写方法。
- (4) 通过 WinPcap 编程，实现本机的 IP 数据报捕获。
- (5) 捕获的数据报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，源 IP、目的 IP 地址、校验和字段的数值。
- (6) 编写的程序应结构清晰，具有较好的可读性。

## 第2节 实验准备

在使用 WinPcap 之前，首先需要安装 WinPcap 驱动程序和 DLL 程序。这些程序可以从官网上下载获得。本次实验使用的是 WinPcap 的升级版 NPcap。

NPcap 编程环境配置。在官网上下载四个文件，安装 NPcap 并将 NPcap 的头文件解压到对应的目录。

在 Visual Studio Code 2019 中需要添加相应的包含目录、库目录；连接器输入添加附加依赖项 ws2\_32.lib、wpcap.lib。

## 第3节 实验过程

项目设计思路：

参考教材上的 WinPcap 编程各种函数如 pcap\_next\_ex, pcap\_open,

pcap\_findalldevs\_ex 等函数的功能和实例化方法。

代码思路具体为：

先利用 pcap\_findalldevs\_ex 打开本机所有的端口，让用户输入选择监听哪一个端口，并捕获多少个 IP 数据包。

利用 pcap\_open 函数打开网卡，利用 pcap\_loop 函数来实现本机的数据包的捕获，通过调用回调函数 packet\_handler 以简单明了的方式在命令行中显示源 MAC 地址、目的 MAC 地址，源 IP、目的 IP 地址、校验和字段的值。

关键代码分析：（完整代码在实验报告同一目录中）

```
//存储MAC地址的结构体
struct ethernet_header
{
    uint8_t ether_final[6]; //目的MAC地址
    uint8_t ether_from[6]; //源MAC地址
    uint16_t ether_type;
};
//存储IP地址和校验和的结构体
struct ip_header
{
    uint8_t ip_header_length : 4,
            ip_version : 4;

    uint8_t ip_tos;
    uint16_t ip_length;
    uint16_t ip_checksum; //校验和字段
    struct in_addr ip_source_address; //源地址
    struct in_addr ip_destination_address; //目的地址
};
/* packet handler 函数 */
/* 每次捕获到数据包时，libpcap都会自动调用这个回调函数 */
void packet_handler(u_char* param, const struct pcap_pkthdr* header, const
u_char* pkt_data)
{
    struct tm* ltime;
    struct ethernet_header* ethernet_protocol; /*以太网协议变量*/
```

```

    struct ip_header* ip_protocol; /*ip协议变量*/
    ip_protocol = (struct ip_header*)(pkt_data + 14); /*处理ip数据包的内容*/
    char timestr[16];
    time_t local_tv_sec;
    u_char* macsave;
    cout << "捕获到数据包!" << endl;
    /* 将时间戳转换成可识别的格式 */
    local_tv_sec = header->ts.tv_sec;
    ltime = localtime(&local_tv_sec);
    cout << "捕获时间:  ";
    strftime(timestr, sizeof timestr, "%H:%M:%S", ltime);
    cout << timestr << endl;
    cout << "数据包长度:  " << header->len << "字节" << endl;
    ethernet_protocol = (struct ethernet_header*)pkt_data;
    /*源MAC地址*/
    macsave = ethernet_protocol->ether_from;
    cout << "源MAC地址:  ";
    printf("%02x:%02x:%02x:%02x:%02x:%02x", *macsave, *(macsave + 1), *(macsave
+ 2), *(macsave + 3), *(macsave + 4), *(macsave + 5)); //经过测试, cout会产生奇怪的
bug, 故用也能支持的printf来表示
    /*目的MAC地址*/
    cout << endl;
    macsave = ethernet_protocol->ether_final;
    cout << "目的MAC地址:  ";
    printf("%02x:%02x:%02x:%02x:%02x:%02x", *macsave, *(macsave + 1), *(macsave
+ 2), *(macsave + 3), *(macsave + 4), *(macsave + 5));
    cout << endl;
    /*源ip地址*/
    cout << "源IP地址:  " << inet_ntoa(ip_protocol->ip_source_address) <<
endl;
    /*目的ip地址*/
    cout << "目的IP地址:  " << inet_ntoa(ip_protocol->ip_destination_address) <<
endl;
    /*校验和字段*/
    cout << "校验和字段:  " << ip_protocol->ip_checksum << endl;
    cout << endl << endl;
}

//main函数中扫描所有端口并展示
if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING,
    NULL,
    &alldevs,
    errbuf
) == -1)
{

```

```

        cout << "error!" << endl << "获取端口错误! ";
        exit(1);
    }
    for (d = alldevs; d != NULL; d = d->next)
    {
        cout << ++i << " " << d->name << endl;
        if (d->description)
        {
            cout << d->description << endl;
        }
        else
        {
            cout << "No description available" << endl << "没有可用的描述! " <<
endl;
        }
    }
    if (i == 0)
    {
        cout << "Check Winpcap" << endl << "没有找到端口! 请检查 Npcap! " <<
endl;
        return -1;
    }
    //main中通过调用pcap_loop来捕获pnum个数据包
    pcap_loop(adhandle, pnum, packet_handler, NULL);

```

程序测试过程:

对最终代码生成解决方案并运行。

扫描到本机的所有网络端口如下:

```

Start!
开始扫描端口!
1 rpcap://\Device\NPF_{2297DCDE-C90D-4AEF-AF06-B5BD580B1EB8}
Network adapter 'WAN Miniport (Network Monitor)' on local host
2 rpcap://\Device\NPF_{BABD972E-EE2B-4223-B0D8-5E6583E9E15D}
Network adapter 'WAN Miniport (IPv6)' on local host
3 rpcap://\Device\NPF_{0BC18797-99B9-4EB6-80D6-01129BA7160B}
Network adapter 'WAN Miniport (IP)' on local host
4 rpcap://\Device\NPF_{94BCEE79-5354-4109-8150-C5D7F55002AE}
Network adapter 'Microsoft Wi-Fi Direct Virtual Adapter' on local host
5 rpcap://\Device\NPF_{C78B61E6-1364-4AFD-8E27-FE68F47C19A1}
Network adapter 'Realtek 8822CE Wireless LAN 802.11ac PCI-E NIC' on local host
6 rpcap://\Device\NPF_{F653009C-4D70-4614-8D32-AB17789B9EBC}
Network adapter 'Microsoft Wi-Fi Direct Virtual Adapter #2' on local host
7 rpcap://\Device\NPF_{7C53B443-28DB-484D-AE46-0D2B135508DD}
Network adapter 'VirtualBox Host-Only Ethernet Adapter' on local host
8 rpcap://\Device\NPF_Loopback
Network adapter 'Adapter for loopback traffic capture' on local host
9 rpcap://\Device\NPF_{431C2C7A-077A-460B-9BD1-D712C7A4A7A3}
Network adapter 'Netaase UU TAP-Win32 Adapter V9.21' on local host
10 rpcap://\Device\NPF_{1D223EEF-EF26-4AE0-9E51-CA152363B6F6}
Network adapter 'TAP-Windows Adapter V9 #5' on local host
11 rpcap://\Device\NPF_{028F9220-3492-4510-AD32-9B440EE79ECC}
Network adapter 'TAP-Windows Adapter V9 #4' on local host
12 rpcap://\Device\NPF_{CE684674-ACA1-4AD0-A807-C7FE7A1877B5}
Network adapter 'TAP-Windows Adapter V9 #3' on local host
13 rpcap://\Device\NPF_{4F1812FC-A607-41AF-AD94-655D86D04178}
Network adapter 'TAP-Windows Adapter V9 #2' on local host
14 rpcap://\Device\NPF_{3F3AD08E-3759-4751-B6DC-55F0A8923639}
Network adapter 'TAP-Windows Adapter V9' on local host

```

可以看出一共有 14 个网络端口，选择第 5 个端口，即 “Realtek 8822CE Wireless LAN 802.11ac PCI-E NIC”。

捕获 10 个 IP 数据包。

```
Enter the interface number:(range:1-14)
请输入进入的端口号: (范围: 1-14)
5
listening on : Network adapter 'Realtek 8822CE Wireless LAN 802.11ac PCI-E NIC' on local host
Enter the number of data package needed!
输入捕获数据包数量
10
捕获到数据包!
捕获时间: 19:57:31
数据包长度: 66字节
源MAC地址: f2:ee:a7:f5:58:ff
目的MAC地址: 40:23:43:d7:66:4f
源IP地址: 220.6.61.22
目的IP地址: 35.165.130.171
校验和字段: 36046

捕获到数据包!
捕获时间: 19:57:34
数据包长度: 54字节
源MAC地址: 40:23:43:d7:66:4f
目的MAC地址: f2:ee:a7:f5:58:ff
源IP地址: 128.6.186.184
目的IP地址: 192.168.43.224
校验和字段: 8729

捕获到数据包!
捕获时间: 19:57:34
数据包长度: 54字节
源MAC地址: f2:ee:a7:f5:58:ff
目的MAC地址: 40:23:43:d7:66:4f
源IP地址: 101.6.229.65
目的IP地址: 52.109.6.0
校验和字段: 20745
```

如图，能正确得到 IP 数据包的长度、源 MAC 地址、目的 MAC 地址、源 IP 地址、目的 IP 地址、校验和字段。

## 第4节 特殊现象分析

在使用 C++ 的 cout 语句输出 MAC 地址时：

```
cout << hex << setw(2) << *macsave << *(macsave + 1) << *(macsave + 2) << *(macsave + 3) << *(macsave + 4) << *(macsave + 5) << endl;
```

无法正常输出 mac 地址，乱码输出，而使用 printf 则不会，可以正常得到正确输出。