



南開大學
Nankai University

南 开 大 学

计算机学院

网络技术与应用课程实验报告

实验2: IP数据报捕获与分析 (利用NPcap编程捕获数据包)

姓 名: 陈睿颖
学 号: 2013544
专 业: 计算机科学与技术
年 级: 2020
指导教师: 张建忠

一、 实验内容

1. 了解NPcap的架构。
2. 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。
3. 通过NPcap编程，实现本机的IP数据报捕获，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。
4. 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。

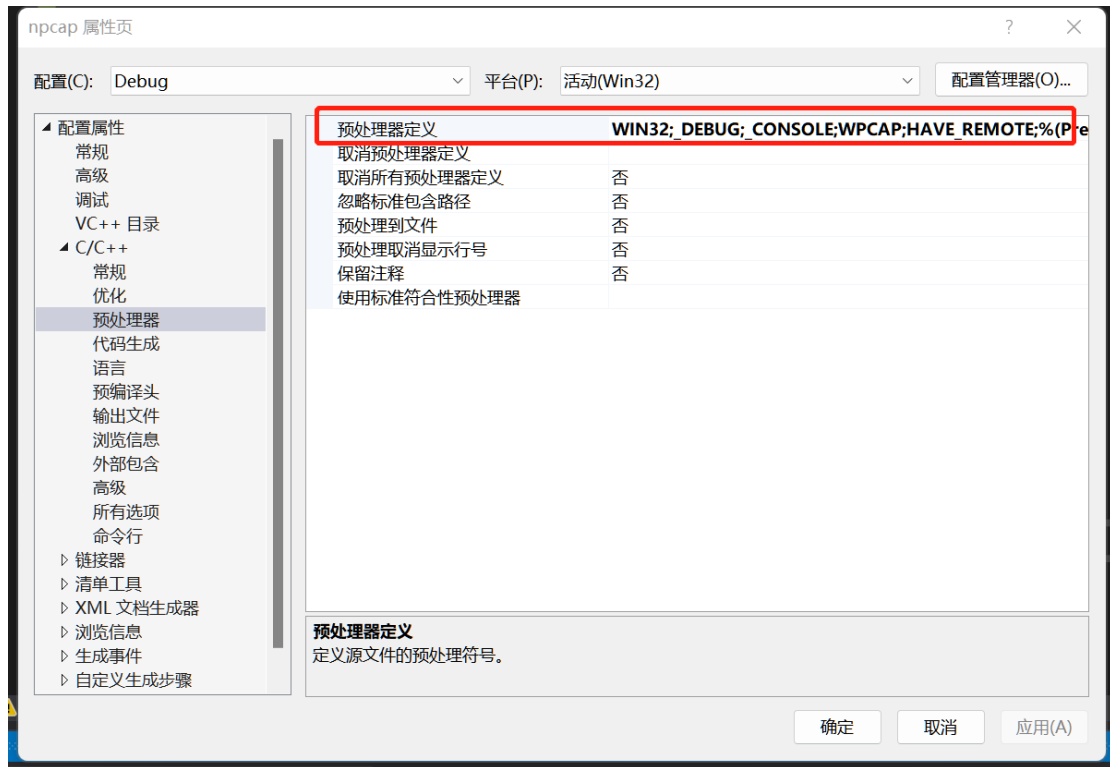
二、 实验步骤

1. 添加pcap.h包含文件：程序中使用了WinPcap提供的函数，所以需要包含头文件：

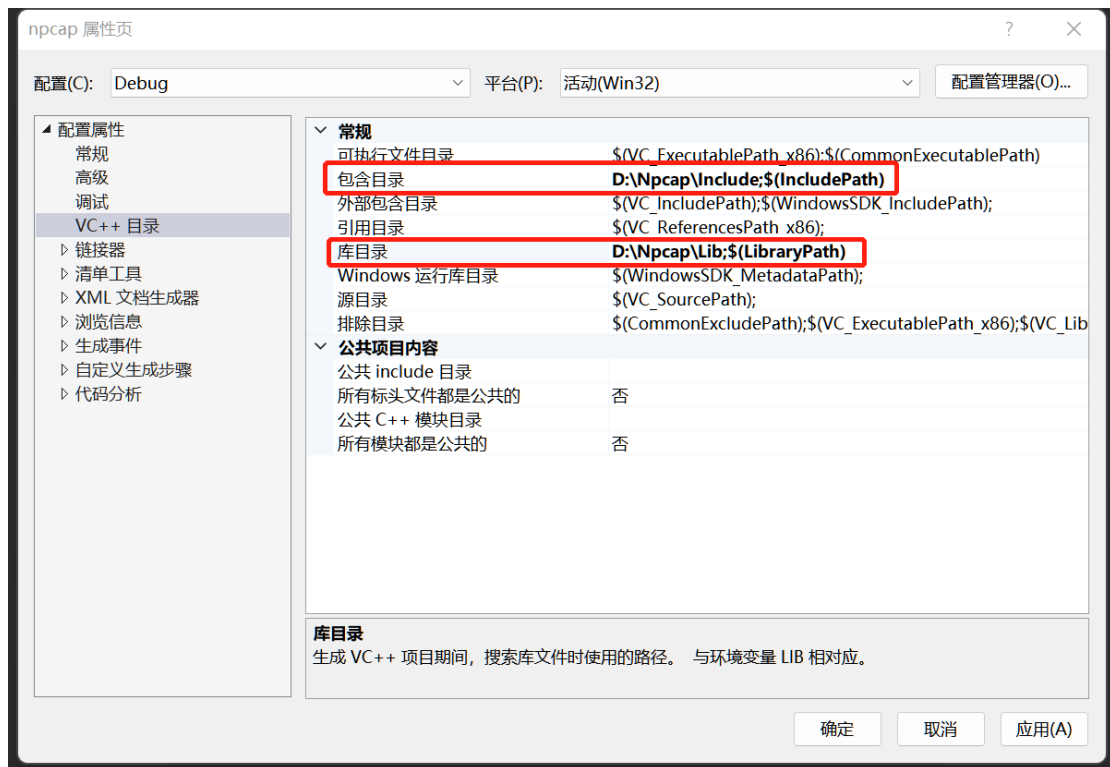
```
#include "pcap.h"
```

2. 增加与WinPcap有关的预处理器定义：

```
WPCAP  
HAVE_REMOTE
```



3. 添加包含文件目录与wpcap.lib文件目录:



4. 学习WinPcap的设备列表获取方法、网卡设备打开方法、数据包捕获方法以及多线程程序编写方法进行编程。

(1) 代码中用到的主要数据结构：

`FrameHeader_t`：帧首部，包含48位的源MAC地址、目的MAC地址和帧类型。

`IPHeader_t`：IP首部，包含校验和、源IP地址、目的IP地址等。

`Data_t`：数据包，包含帧首部和IP首部。

`string* ByteToHexStr(unsigned char byte_arr[], int arr_len)`：自定义函数，用于将Byte类型转换为十六进制的字符串，主要用来处理源MAC地址和目的MAC地址。

`string GetIp(unsigned long u)`：自定义函数，将源IP地址和目的IP地址转换成“.”点隔的字符串格式。

`void WordToBitStr(WORD b)`：自定义函数，将字格式的数据转化为二进制的位格式。

`void captureIP(void* a)`：用来捕获数据包的线程函数。

(2) 程序主体框架

1 用`pcap_findalldevs()`函数获取网络接口设备列表。如果获取成功，则继续向下进行；否则，打印`errbuf`里的错误信息，函数返回。

```
if (pcap_findalldevs(&allDevices, errbuf) == -1)
```

```

{
    cout << stderr << "寻找设备错误" << errbuf << endl;
    return 0;
}

```

2 上一个函数调用成功后，`allDevices`参数指向获取的网络接口列表的第一个元素，下面通过指针和循环，遍历所有的网络接口，并打印设备的名字和描述。

```

for (currentDevice = allDevices; currentDevice; currentDevice =
currentDevice->next)
{
    cout << ++i << ". " << currentDevice->name;
    if (currentDevice->description)
        cout << "(" << currentDevice->description << ")" << endl;
    else
        cout << "(无可用描述)\n";
}

```

3 由用户输入选择指定的设备，并调用函数`pcap_open`打开对应的网卡，成功则继续进行；否则，打印`errbuf`里的错误信息，调用`pcap_freealldevs()`释放该设备列表，函数返回。

```

targetDevice = pcap_open(currentDevice->name, 100,
PCAP_OPENFLAG_PROMISCUOUS, 1000, NULL, errbuf);
if (targetDevice == NULL) {
    cout << "打开设备错误：" << errbuf << endl;
    pcap_freealldevs(allDevices);
    return 0;
}

```

```
} }
```

4 通过 `_beginthread(Capturer, 0, NULL)` 打开 `captureIP` 线程，开始捕获数据包。

(a) 循环调用 `pcap_next_ex()` 函数，捕获数据包，如果返回值等于0，则进入下一次循环，继续捕获；如果返回值大于0，则在屏幕上打印想要的数椐；如果返回值小于0，则跳出循环，打印错误信息。

```
for(i= pcap_next_ex(targetDevice, &pkt_header, &pkt_data);i>=0;){  
    if (i == 0)continue;  
    else {  
        Data_t* IPPacket;  
        IPPacket = (Data_t*)pkt_data;  
        cout << "第" << num << "个数据包: " << endl;  
        cout << "源MAC地址: " <<  
        *(ByteToHexStr(IPPacket->FrameHeader.SrcMAC, 6)) << endl;  
        cout << "目的MAC地址: " <<  
        *(ByteToHexStr(IPPacket->FrameHeader.DesMAC, 6)) << endl;  
        cout << "源IP地址: " <<  
        GetIp(IPPacket->IPHeader.SrcIP) << endl;  
        cout << "目的IP地址: " <<  
        GetIp(IPPacket->IPHeader.DstIP) << endl;  
        cout << "类型: " <<  
        IPPacket->FrameHeader.FrameType << endl;  
        cout << "长度: " <<  
        GetIp(IPPacket->IPHeader.TotalLen) << endl;  
  
        num++;  
    }  
    if (num == 5)break;  
}  
if (i < 0) {  
    cout << "Error in pcap_next_ex." << endl;  
}
```

(b)捕获结束后，调用函数_endthread()关闭线程。

5 最终在主函数中，调用函数pcap_freealldevs(allDevices)释放所有的设备。

5. 运行并验证对比：

```
选择 命令提示符
DHCIPv6 Iaid . . . . . : 201347158
DHCIPv6 客户端 DUID . . . . . : 00-01-00-01-28-C8-D5-24-0C-7A-15-DC-4A-9C
TCP/IP 上的 NetBIOS . . . . . : 已启用

无线局域网适配器 WLAN:

  连接特定的 DNS 后缀 . . . . . :
  描述 . . . . . : Intel(R) Wireless-AC 9560 160MHz
  物理地址. . . . . : 0C-7A-15-DC-4A-9C
  DHCP 已启用 . . . . . : 是
  自动配置已启用 . . . . . : 是
  IPv6 地址 . . . . . : 2001:250:401:6576:d130:c5f5:cb3:49e0(首选)
  临时 IPv6 地址. . . . . : 2001:250:401:6576:a8d7:d64c:e86e:d706(首选)
  本地链接 IPv6 地址. . . . . : fe80::d130:c5f5:cb3:49e0%7(首选)
  IPv4 地址 . . . . . : 10.136.80.21(首选)
  子网掩码 . . . . . : 255.255.128.0
  获得租约的时间 . . . . . : 2022年10月10日 10:42:47
  租约过期的时间 . . . . . : 2022年10月10日 16:48:47
  默认网关 . . . . . : fe80::865b:12ff:fe5e:3602%7
  . . . . . : 10.136.0.1
  DHCP 服务器 . . . . . : 10.136.0.1
  DHCIPv6 Iaid . . . . . : 67926549
  DHCIPv6 客户端 DUID . . . . . : 00-01-00-01-28-C8-D5-24-0C-7A-15-DC-4A-9C

DNS 服务器 . . . . . : 222.30.45.41
. . . . . : 202.113.16.41
TCP/IP 上的 NetBIOS . . . . . : 已启用

以太网适配器 以太网:

  媒体状态 . . . . . : 媒体已断开连接
  连接特定的 DNS 后缀 . . . . . :
  描述 . . . . . : Sangfor SSL VPN CS Support System VNIC
  物理地址. . . . . : 00-FF-A0-85-05-7C
  DHCP 已启用 . . . . . : 否
  自动配置已启用 . . . . . : 是

以太网适配器 蓝牙网络连接:

  媒体状态 . . . . . : 媒体已断开连接
  连接特定的 DNS 后缀 . . . . . :
  描述 . . . . . : Bluetooth Device (Personal Area Network)
  物理地址. . . . . : 0C-7A-15-DC-4A-A0
  DHCP 已启用 . . . . . : 是
  自动配置已启用 . . . . . : 是

C:\Users\陈睿颖>

D:\311\programs\ncap\Debug\ncap.exe
1. \Device\NPF_{7B3530DE-9F19-4D65-8BC1-637A22654776} (WAN Miniport (Network M
2. \Device\NPF_{3D4577F8-D388-4149-AD57-D760C6564F79} (WAN Miniport (IPv6))
3. \Device\NPF_{65CC831E-6D65-4AC2-9FD9-7C7DD83A4741} (WAN Miniport (IP))
4. \Device\NPF_{4537A16E-6787-45AE-BBE1-D4FE5AFE07BD} (Bluetooth Device (Perso
network))
5. \Device\NPF_{3337A2ED-C76E-4AD7-91EA-461B663C2C70} (Intel(R) Wireless-AC 95
6. \Device\NPF_{358B86BA-B0DF-4975-A253-820493E87C5A} (VMware Virtual Ethernet
or VMnet8)
7. \Device\NPF_{FA75D958-50AF-48DC-A7CC-717CCC52E5F0} (VMware Virtual Ethernet
or VMnet1)
8. \Device\NPF_{37E15D16-0C40-434A-BFCE-556285100DF3} (Microsoft Wi-Fi Direct
apter #2)
9. \Device\NPF_{80B2AA34-1FB6-4A82-AC18-FDC6CBDE6FA0} (Microsoft Wi-Fi Direct
apter)
10. \Device\NPF_{Loopback (Adapter for loopback traffic capture)}
11. \Device\NPF_{A085057C-BD85-437E-BB8E-C30696F49140} (Sangfor SSL VPN CS Sup
am VNIC)
请选择目标设备: 5
第1个数据包:
源MAC地址: 00-00-5E-00-01-08
目的MAC地址: 0C-7A-15-DC-4A-9C
源IP地址: 20.72.226.48
目的IP地址: 10.136.80.21
类型: 8
长度: 0.52.0.0
第2个数据包:
源MAC地址: 00-00-5E-00-01-08
目的MAC地址: 0C-7A-15-DC-4A-9C
源IP地址: 20.72.226.48
目的IP地址: 10.136.80.21
类型: 8
长度: 0.52.0.0
第3个数据包:
源MAC地址: 00-00-5E-00-01-08
目的MAC地址: 0C-7A-15-DC-4A-9C
源IP地址: 20.72.226.48
目的IP地址: 10.136.80.21
类型: 8
长度: 0.52.0.0
第4个数据包:
源MAC地址: 00-00-5E-00-01-08
目的MAC地址: 0C-7A-15-DC-4A-9C
源IP地址: 20.72.226.48
目的IP地址: 10.136.80.21
类型: 8
长度: 0.52.0.0
```

三、 实验代码

```
#define WIN32
#define WPCAP
#define HAVE_REMOTE
#include "pcap.h"
#include <iostream>
#include<WinSock2.h>
#include<bitset>
#include <process.h>
using namespace std;
#pragma comment(lib, "wpcap.lib")
#pragma comment(lib, "packet.lib")
```

```

#pragma comment(lib, "ws2_32.lib")
#pragma warning(disable:4996)
#pragma pack(1)
#define BYTE unsigned char
pcap_t* targetDevice;
typedef struct FrameHeader_t {
    BYTE DesMAC[6];
    BYTE SrcMAC[6];
    WORD FrameType;
}FrameHeader_t;
typedef struct IPHeader_t {
    BYTE Ver_HLen;
    BYTE TOS;
    WORD TotalLen;
    WORD ID;
    WORD Flag_Segment;
    BYTE TTL;
    BYTE Protocol;
    WORD Checksum;
    ULONG SrcIP;
    ULONG DstIP;
}IPHeader_t;
typedef struct Data_t {
    FrameHeader_t FrameHeader;
    IPHeader_t IPHeader;
}Data_t;
#pragma pack()
string* ByteToHexStr(unsigned char byte_arr[], int arr_len)
{
    string* hexstr = new string();
    for (int i = 0; i < arr_len; i++)
    {
        char hex_1;
        char hex_2;
        int value = byte_arr[i];
        int x = value / 16;
        int y = value % 16;
        if (x >= 0 && x <= 9)
            hex_1 = (char)(48 + x);
        else
            hex_1 = (char)(55 + x);
        if (y >= 0 && y <= 9)
            hex_2 = (char)(48 + y);
        else

```



```

        hex_2 = (char)(55 + y);
    if (i != arr_len - 1) {
        *hexstr = *hexstr + hex_1 + hex_2 + "-";
    }
    else
        *hexstr = *hexstr + hex_1 + hex_2;
}
return hexstr;
}
string GetIp(unsigned long u) {
    in_addr addr;
    memcpy(&addr, &u, sizeof(u));
    return inet_ntoa(addr);
}
void WordToBitStr(WORD b) {
    cout << bitset<16>(b);
}
void captureIP(void* a) {
    struct pcap_pkthdr* pkt_header;
    const u_char* pkt_data;
    pkt_data = NULL;
    int i;
    int num = 1;
    for(i= pcap_next_ex(targetDevice, &pkt_header, &pkt_data);i>=0;){
        if (i == 0)continue;
        else {
            Data_t* IPPacket;
            IPPacket = (Data_t*)pkt_data;
            cout << "第" << num << "个数据包: " << endl;
            cout << "源MAC地址:  " <<
            *(ByteToHexStr(IPPacket->FrameHeader.SrcMAC, 6)) << endl;
            cout << "目的MAC地址: " <<
            *(ByteToHexStr(IPPacket->FrameHeader.DesMAC, 6)) << endl;
            cout << "源IP地址:  " << GetIp(IPPacket->IPHeader.SrcIP) <<
            endl;
            cout << "目的IP地址: " << GetIp(IPPacket->IPHeader.DstIP) <<
            endl;
            cout << "类型:          " << IPPacket->FrameHeader.FrameType <<
            endl;
            cout << "长度:          " << IPPacket->IPHeader.TotalLen <<
            endl;

            num++;
        }
    }
}

```

```

        if (num == 5)break;
    }
    if (i < 0) {
        cout << "Error in pcap_next_ex." << endl;
    }
    _endthread();
}
//将Byte类型转化为十六进制字符串以便与主机信息进行验证

int main() {
    pcap_if_t* allDevices, * currentDevice;
    int i = 0;
    char errbuf[PCAP_ERRBUF_SIZE];
    if (pcap_findalldevs(&allDevices, errbuf) == -1)
    {
        cout << stderr << "寻找设备错误" << errbuf << endl;
        return 0;
    }
    for (currentDevice = allDevices; currentDevice; currentDevice =
currentDevice->next)
    {
        cout << ++i << ". " << currentDevice->name;
        if (currentDevice->description)
            cout << "(" << currentDevice->description << ")" << endl;
        else
            cout << "(无可用描述)\n";
    }
    if (i == 0)
    {
        cout << "\n 未找到接口；请确认WinPcap已安装！\n";
        return 0;
    }
    currentDevice = allDevices;
    int j;
    cout << "请选择目标设备： ";
    cin >> j;

    for (i = 0; i < j - 1; i++) {
        currentDevice = currentDevice->next;
    }
    targetDevice = pcap_open(currentDevice->name, 100,
PCAP_OPENFLAG_PROMISCUOUS, 1000, NULL, errbuf);
    if (targetDevice == NULL) {
        cout << "打开设备错误： " << errbuf << endl;
    }
}

```

```
        pcap_freealldevs(allDevices);
        return 0;
    }
    _beginthread(captureIP, 0, NULL);
    cin.ignore();
    getchar();
    pcap_freealldevs(allDevices);
}
```