南開大學
Nankai University

南　开　大　学

计 算 机 学 院

网络技术与应用课程报告

编程实验（1）IP 数据报捕获与分析

学号：1913630

姓名：安祺

年级：2019

专业：计算机科学与技术

2021 年 11 月 11 日

目录

1 实验内容说明 **2**

1.1 实验题目 2

1.2 实验说明 2

2 实验准备 **2**

3 实验过程 **3**

3.1 项目设计思路 3

3.2 关键代码分析 4

3.2.1 选择设备 4

3.2.2 新建线程 5

3.2.3 监听设备 5

3.2.4 解析报文 6

4 特殊现象分析 **7**

4.1 回环网卡的帧段 7

1 实验内容说明

1.1 实验题目

编程实验（1）IP 数据报捕获与分析

1.2 实验说明

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。

2 实验准备

WinPcap 是一个数据包捕获体系框架，主要功能是进行数据包捕获和网络分析。包括了内核基本的包过滤、低层次的库 (packet.lib)、高级别系统无关的函数库 (wpcap.dll)。

目前，WinPcap 停止维护，故采用 NPcap 进行实验。运行相关程序，需要安装驱动程序。此外还需要配置开发工具包才能进行程序编写，在 Vscode 中使用 TDM-GCC 进行编译，使用静态连接库的编译器参数如下。

```
"args": [
    "-g",
    "${workspaceFolder}/src/*.c",
    "-o","${workspaceFolder}/outputBin",
    "-m64","-std=c17",
    "-Wall","-Wextra",
    "-lpthread","-lws2_32",
    "-D WPCAP","-D HAVE_REMOTE",
    "-I${workspaceFolder}/lib/Npcap/Include",
    "-L${workspaceFolder}/lib/Npcap/Lib",
    "-lwpcap","-lPacket"
],
```

3 实验过程

3.1 项目设计思路

为了利用 Npcap 捕获 IP 报文，需要能够获得本机网卡设备，能够捕获设备上的数据包，能够解析报文，能够新建一个工作线程进行数据包捕获。

调用 Npcap 接口 pcap_findalldevs_ex 可以获得设备列表，调用 Npcap 接口 pcap_open() 可以打开设备，调用 pcap_next_ex() 可以主动捕获数据。

在 C 程序中预定义结构体 (字节对齐) 并强转指针即可快速对数据进行解析，但是整数需要转换字节序。

可以使用 pthread 创建和管理线程，通过原子变量实现原子操作。

</> CODE 1: 报文格式定义

```
#pragma pack(1)
typedef struct frame_header
{
    uint8_t des_mac[6];  //目的MAC地址
    uint8_t src_mac[6];  //源MAC地址
    uint16_t frame_type; //帧类型
} frmhdr_s;

typedef struct ip_header
{
    uint8_t ver_hlen;
    uint8_t tos;
    uint16_t total_len;
    uint16_t id;
    uint16_t flags;
    uint8_t ttl;
    uint8_t protocol;
    uint16_t check_sum;
    uint32_t src_ip;
    uint32_t des_ip;
} iphdr_s;

typedef struct capture_data
{
    frmhdr_s frame_header;
    iphdr_s ip_header;
} capdata_s;
#pragma pack()
```

3.2 关键代码分析

程序运行主要流程为:

$$main \rightarrow user\_select\_device \rightarrow pthread\_create \rightarrow watch\_ippkt \rightarrow print\_hdr$$

user_select_device 函数用于打印所有设备信息，并获得所选设备的名称。watch_ippkt 用于打开并监听设备。print_hdr 用于解析和输出报文。

3.2.1 选择设备

</> CODE 2: 获得设备列表

```
/* 获取本地设备列表 */
pcap_if_t *alldevs;
char errbuf[PCAP_ERRBUF_SIZE];
if (-1 == pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL /* auth is
   not needed */, &alldevs, errbuf))
    PANIC("Error in pcap_findalldevs_ex: %s\n", errbuf);
if (NULL == alldevs)
    PANIC("No interfaces found! Make sure NPcap is installed.");

/* 打印列表 */
fputs(LIGHT_PURPLE "All device on local host:\n" NONE, out);
int no = 0;
for (pcap_if_t *d = alldevs; d != NULL; d = d->next)
{
    fprintf_s(out, LIGHT_GREEN "[%d]" NONE "%s\n", no++, d->
   description);
}

//选择设备，获取名字略......
```

alldevs是获得的指向设备列表的指针，其定义如下, 可以从中获得设备名称和描述。

</> CODE 3: pcap_if_t 定义

```
/* Item in a list of interfaces. */
struct pcap_if
{
    struct pcap_if *next;
    char *name;         /* name to hand to "pcap_open_live()" */
    char *description; /* textual description of interface, or NULL
   */
    struct pcap_addr *addresses;
    bpf_u_int32 flags; /* PCAP_IF_ interface flags */
};
```

3.2.2 新建线程

</> CODE 4: 创建线程

```
pthread_t work_thrd = 0;
/*此处为开始与终止控制代码，略.....*/
int ret = pthread_create(&work_thrd, NULL, work_thread, &arg);
/*此处为开始与终止控制代码，略.....*/
pthread_join(work_thrd, (void **)&thread_ret); /*等待并入*/
```

3.2.3 监听设备

选择设置等待时间为 1000ms, 最大捕获长度为 1024 字节。

</> CODE 5: 监听设备

```
/* 打开设备 */
int snaplen = 1024;
int read_timeout = 1000;
char errbuf[PCAP_ERRBUF_SIZE] = {0};
pcap_t *capture_device = pcap_open(device, /*设备名称*/
                                   snaplen, /*数据包的最大捕获长度*/
                                   PCAP_OPENFLAG_PROMISCUOUS,
                                   read_timeout, /*等待延时*/
                                   NULL, /*本机无需授权*/
                                   errbuf);
if (NULL == capture_device)
    PANIC("Can not open decive:%s", errbuf);

/* 开始捕获 */
struct pcap_pkthdr *pkthdr = NULL;
const u_char *rawdata = NULL;
while (!atomic_load(stop))
{
    int capret = pcap_next_ex(capture_device, &pkthdr, &rawdata);
    if (1 == capret)
    {
        fputs("----------------------------\n(", out);
        print_tv(out, &pkthdr->ts);
        fprintf_s(out, ") Capture: %d/%d\n",
                  pkthdr->caplen, pkthdr->len);
        print_hdr(out, (capdata_s *)rawdata);
    }
    else{/*错误处理略*/}
}
pcap_close(capture_device);
```

3.2.4 解析报文

用到了的大小端转换函数ntohs()

</> CODE 6: 解析报文节选代码

```
/* 输出帧头，节选MAC的解析输出 */
frmhdr_s *frame_header = &capdata->frame_header;
uint8_t *dm = frame_header->des_mac;
fprintf_s(out, "DestinationMAC: %02X:%02X:%02X:%02X:%02X:%02X\n",
          dm[0], dm[1], dm[2], dm[3], dm[4], dm[5]);
/* 输出IP头，节选IP版本和头长度的解析 */
iphdr_s *ip_header = &capdata->ip_header;
uint_fast8_t ver = (ip_header->ver_hlen) >> 4;
uint_fast8_t hlen = (ip_header->ver_hlen) & 0x0F;
fprintf_s(out, "version:        %u\n", ver);
fprintf_s(out, "header_length:  %u\n", hlen);
/* 输出IP头，节选IP地址的解析输出 */
uint8_t *sip = (uint8_t *)&ip_header->src_ip;
fprintf_s(out, "src_address:    %u.%u.%u.%u\n",
          sip[0], sip[1], sip[2], sip[3]);
```

此外还自行计算了校验和进行对比。思路是使用 32 位整数对 IP 头各 16 位进行加和，最后令结果的高 16 位于低 16 位相加。

> **Note**
>
> 可以证明，最多 2 次高低位相加后，高 16 位一定为 0，故高低位只需相加 2 次。

</> CODE 7: 校验和计算

```
uint16_t checksum_iphdr(const iphdr_s *hdr)
{
    /*复制一份IP头，并将校验位设为0*/
    iphdr_s header;
    memcpy_s(&header, sizeof(header), hdr, sizeof(iphdr_s));
    header.check_sum = 0;
    /*使用32为整数加法，对各16位求和，此处不考虑可选长度*/
    uint16_t *words = (uint16_t *)&header;
    uint32_t checksum = 0;
    int looplen = sizeof(iphdr_s) / sizeof(uint16_t);
    for (int i = 0; i < looplen; i++)
        checksum += ntohs(words[i]);
    /*进行两次高16位于低16位相加*/
    checksum = (checksum >> 16) + (checksum & 0x0000FFFF);
    checksum = (checksum >> 16) + checksum;
    return (uint16_t)~checksum;
}
```

4 特殊现象分析

4.1 回环网卡的帧段

对 windows 的 loopback adapter 进行监听时，发现其 frame 段不符合 Ethernet 协议格式。其 frame 段为 02 00 00 00 四个字节。

使用 wireshark 进行分析，发现这 Looback adaptor 的 frame 的封装格式。

此外，查询相关资料得到，该部分是用于回环的伪层，主要用于指示下一层即 IP。