计算机网络编程
实验报告

班级：07111707

组长：1120171189 崔程远

成员：1120172149 吴沁璇

1120172153 张澈

1120172163 王晓媛

1120172736 张鉴昊

1120172765 曾煜瑾

1120173326 曾紫飞

北京理工大学

计算机学院

2020 年 4 月

第四章 实验 1　链路层抓包及协议分析

1. 实验目的

利用 WinPap 实现网络数据链路层帧捕获，显示分析帧和上层包结构。

2. 实验内容

程序运行屏幕输出要点：
首先屏幕显示当前配置的网络适配器，并要求选择捕获适配器编号
按照捕获帧的层次关系显示以下信息：
数据链路层（MAC 子层）层结构及各个字段的值
网络层分组的格式及各个字段的值
运输层报文段的格式及各个字段的值
应用层报文格式及各个字段的值

3. 实验原理

WinPcap 是一个基于 Win32 平台的，用于捕获网络数据包并进行分析的开源库。它提供了以下功能：捕获原始数据包；在数据包发送给某应用程序前，根据用户指定的规则过滤数据包；将原始数据包通过网络发送出去；收集并统计网络流量信息。

4. 实验环境

操作系统：Windows 10
编译器：Visual Studio 2017
环境：WinPcap4.1.3　WpdPack

5. 实验步骤

以下是 C++代码和具体思路：

1) VS 中环境的配置

首先下载 WinPcap 和 WpdPack，然后在 vs 的项目中添加包含目录和库目录，修改预处理器，添加依赖项。然后就可以进行代码的编写。

2) 定义数据报头结构

如下图所示，分别定义了以太网帧头、帧尾，IP 数据包、IP 数据包首部、IP 地址、MAC 地址，UDP 和 TCP 的数据包和首部，还有应用层的 DHCP 的报文。根据各个字段的大小用相应的数据类型进行定义。

```c
/* 6字节的MAC地址 */
typedef struct mac_address {
    u_char byte1; //地址第一个字节 8位
    u_char byte2;
    u_char byte3;
    u_char byte4;
    u_char byte5;
    u_char byte6;
}mac_address;

/* 14字节的以太网帧头 */
typedef struct mac_header {
    mac_address dmac;    //目的mac地址 6字节
    mac_address smac;    //源mac地址 6字节
    u_short type;    //类型 2字节
    //string
}mac_header;

/* 4字节的以太网帧尾 */
typedef struct mac_tail {
    u_int fcs;  //4字节
}mac_tail;
```

```c
/* 最多1500字节最少为46字节的ip数据包 */
typedef struct ip_data {
    u_int byte1;    //4字节
    u_int byte2;
    u_int byte3;
    u_int byte4;
    u_int byte5;
    u_int byte6;
    u_int byte7;
    u_int byte8;
    u_int byte9;
    u_int byte10;
    u_int byte11;
    u_int byte12;
}ip_data;

/* 4字节的IP地址 */
typedef struct ip_address {
    u_char byte1; //地址第一个字节 8位
    u_char byte2;
    u_char byte3;
    u_char byte4;
}ip_address;
```

```c
/* IPv4 数据包首部 */
typedef struct ip_header {
    u_char  ver_ihl;        // 版本 (4 bits) + 首部长度 (4 bits) 共1字节
    u_char  tos;            // 服务类型(Type of service)
    u_short tlen;           // 总长(Total length)  16位 单位一个字节 2字节
    u_short identification; // 标识(Identification)
    u_short flags_fo;       // 标志位(Flags) (3 bits) + 段偏移量(Fragment offset) (13 bits)
    u_char  ttl;            // 存活时间(Time to live) 8位
    u_char  proto;          // 协议(Protocol) 8位
    u_short crc;            // 首部校验和(Header checksum) 16位
    ip_address  saddr;      // 源地址(Source address) 32位 4字节
    ip_address  daddr;      // 目的地址(Destination address) 32位
    u_int   op_pad;         // 选项与填充(Option + Padding)  32 位 4字节
}ip_header;

/* UDP 首部*/
typedef struct udp_header {
    u_short sport;          // 源端口(Source port) 2字节
    u_short dport;          // 目的端口(Destination port) 2字节
    u_short len;            // UDP数据包长度(Datagram length) 2字节
    u_short crc;            // 校验和(Checksum) 2字节
}udp_header;
```

```c
/* 10字节的udp数据包 */
typedef struct udp_data {
    u_int byte1;    //4字节
    u_int byte2;
    u_int byte3;
    u_int byte4;
    u_int byte5;
    u_int byte6;
    u_int byte7;
    u_int byte8;
    u_int byte9;
    u_int byte10;
}udp_data;
```

```c
/* TCP 首部*/
typedef struct tcp_header {
    u_short sport;          // 源端口(Source port) 2字节 16bit
    u_short dport;          // 目的端口(Destination port) 2字节
    u_int shunxu;           //  4字节
    u_int queren;           //  4字节
    u_char len;             // 偏移+保留 10位 算8位 当成一个字节
    u_char control;         // 控制 6位算一个字节
    u_short window;         //窗口 2字节
    u_short crc;            //校验和 2字节
    u_short jinji;          //紧急指针 2字节
    u_int   op_pad;         // 选项与填充(Option + Padding) 4字节 32位
}tcp_header;
```

```c
/* 10字节的tcp数据包 */
typedef struct tcp_data {
    u_int byte1;     //4字节
    u_int byte2;
    u_int byte3;
    u_int byte4;
    u_int byte5;
    u_int byte6;
    u_int byte7;
    u_int byte8;
    u_int byte9;
    u_int byte10;
}tcp_data;
```

```c
/* UDP专用的DHCP报文格式*/
typedef struct dhcp {
    u_char op;
    u_char htype;
    u_char hlen;
    u_char hops;
    u_int  xid;
    u_short secs;
    u_short flags;
    u_int ciaddr;
    u_int yiaddr;
    u_int siaddr;
    u_int giaddr;
}dhcp;
```

3) 获得设备并打印设备列表

如下图，使用了在 WinPacp 中文文档给出的函数，来获得设备列表和打印列表设备具体信息。

```c
/* 获得设备列表 */
if (pcap_findalldevs(&alldevs, errbuf) == -1)
{
    fprintf(stderr, "Error in pcap_findalldevs: %s\n", errbuf);
    exit(1);
}

/* 打印列表 */
for (d = alldevs; d; d = d->next)
{
    printf("%d. %s", ++i, d->name);
    if (d->description)
        printf(" (%s)\n", d->description);
    else
        printf(" (No description available)\n");
}

if (i == 0)
{
    printf("\nNo interfaces found! Make sure WinPcap is installed.\n");
    return -1;
}

printf("Enter the interface number (1-%d):", i);
scanf("%d", &inum);

if (inum < 1 || inum > i)
{
    printf("\nInterface number out of range.\n");
    /* 释放设备列表 */
    pcap_freealldevs(alldevs);
    return -1;
}
```

```
/* 打开适配器 */
if ((adhandle = pcap_open(d->name,  // 设备名
    65536,      // 要捕捉的数据包的部分
                // 65535保证能捕获到不同数据链路层上的每个数据包的全部内容
    PCAP_OPENFLAG_PROMISCUOUS,          // 混杂模式
    1000,       // 读取超时时间
    NULL,       // 远程机器验证
    errbuf      // 错误缓冲池
)) == NULL)
{
    fprintf(stderr, "\nUnable to open the adapter. %s is not supported by WinPcap\n");
    /* 释放设备列表 */
    pcap_freealldevs(alldevs);
    return -1;
}
```

4） 设置过滤器并调用回调函数

由于捕获的报文信息太多，所以需要设置一个过滤器，用于只处理 tcp 和 udp 的报文信息，其他信息都过滤。具体过滤器的设置如下。

```
u_int netmask;
char packet_filter[] = "ip and udp or ip and tcp";
```

```
//编译过滤器
if (pcap_compile(adhandle, &fcode, packet_filter, 1, netmask) < 0)
{
    fprintf(stderr, "\nUnable to compile the packet filter. Check the syntax.\n");
    /* 释放设备列表 */
    pcap_freealldevs(alldevs);
    return -1;
}

//设置过滤器
if (pcap_setfilter(adhandle, &fcode) < 0)
{
    fprintf(stderr, "\nError setting the filter.\n");
    /* 释放设备列表 */
    pcap_freealldevs(alldevs);
    return -1;
}

printf("\nlistening on %s...\n", d->description);

/* 释放设备列表 */
pcap_freealldevs(alldevs);

/* 开始捕捉 */
pcap_loop(adhandle, 0, packet_handler, NULL);
```

5） 获得各报文具体字段的值

要输出各层结构的值，就要使用之前定义的数据结构用来表示适配器的各个字段。首先将时间戳转换为可识别的格式，然后打印数据包的时间戳和长度。然后根据报文的字段划分将定义的数据结构和获取到的报文的各个字节一一对应，就可以得各字段具体的值。对于上下层结构之间，因为上层的报文首部包含在下层的数据包当中，所以可以通过报文中的数据包长度这一属性和字段的固定长度进行简单的计算，再进行格式的转化，就可以得到上一层结构的报文的首部位置。

```
struct tm *ltime;
char timestr[16];
ip_header *ih;
udp_header *uh;
u_int ip_len,ip_tlen,udp_len,tcp_len;
u_short sport, dport, sport2, dport2;
time_t local_tv_sec;

/* 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);
strftime(timestr, sizeof timestr, "%H:%M:%S", ltime);

/* 打印数据包的时间戳和长度 */
printf("%s.%.6d len:%d\n ", timestr, header->ts.tv_usec, header->len);
```

```
/* 获得以太网帧头部的位置 */
mac_header *mh;
mh = (mac_header*)pkt_data;

/* 获得IP数据包头部的位置 */
ih = (ip_header *)(pkt_data + 14); //以太网头部长度

/* 获得UDP首部的位置 */
ip_len = (ih->ver_ihl & 0xf) * 4;
uh = (udp_header *)((u_char*)ih + ip_len);

/* 获得TCP首部的位置 */
tcp_header *th;
th = (tcp_header *)((u_char*)ih + ip_len);

/* 获得ip数据包的位置 */
ip_data *id;
id = (ip_data *)((u_char*)ih + ip_len);

/* 获得以太网帧尾部的位置 */
mac_tail *mt;
ip_tlen = (ih->tlen & 0xf) * 1;
mt = (mac_tail*)((u_char*)ih + ip_tlen);

/* 获得dhcp的位置 */
dhcp *d;
udp_len = (uh->len & 0xf) * 1;
d = (dhcp*)((u_char*)uh + udp_len);

/* 获得udp数据包的位置 */
udp_data *ud;
ud = (udp_data *)((u_char*)uh + udp_len);

/* 获得tcp数据包的位置 */
tcp_data *td;
tcp_len = (th->len & 0xf) * 4;
td = (tcp_data *)((u_char*)uh + tcp_len);
```

```
/* 将网络字节序列转换成主机字节序列 */
sport = ntohs(uh->sport);
dport = ntohs(uh->dport);
sport2 = ntohs(th->sport);
dport2 = ntohs(th->dport);
```

6) 打印各结构的各字段值

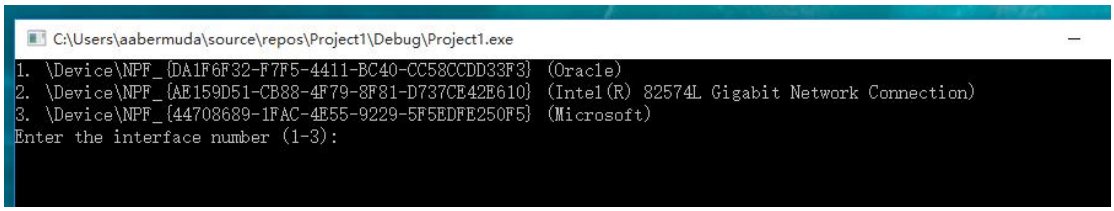
最后是打印各个报文的各个字段的具体值。由于在代码编译时，程序必须包含的头文件“pcap.h”和 string、ostream 之间存在问题导致编译失败，大概是里面有一些重复定义，然而这个问题没能得到解决，所以就只能用 printf 对结构体的每一项一个个打印出来。

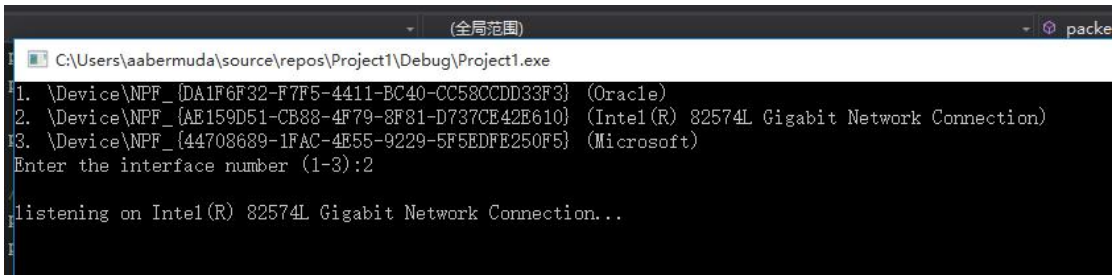
```
/* 打印网络层结构 */
printf("打印网络层结构\n");
printf("ip首部:%d 版本号和首部长度:%d 服务类型:%d 协议:%d 源ip地址:%d.%d.%d.%d 目标ip地址:%d.%d.%d.%d",
    ih,
    ih->ver_ihl,
    ih->tos,
    ih->proto,
    ih->saddr.byte1,
    ih->saddr.byte2,
    ih->saddr.byte3,
    ih->saddr.byte4,
    ih->daddr.byte1,
    ih->daddr.byte2,
    ih->daddr.byte3,
    ih->daddr.byte4);
printf("\n");
printf("前46个字节的数据包:%d%d%d%d%d%d%d%d%d%d",
    id->byte1,
    id->byte2,
    id->byte3,
    id->byte4,
    id->byte5,
    id->byte6,
    id->byte7,
    id->byte8,
    id->byte9,
    id->byte10,
    id->byte11,
    id->byte12);
printf("\n\n");
```

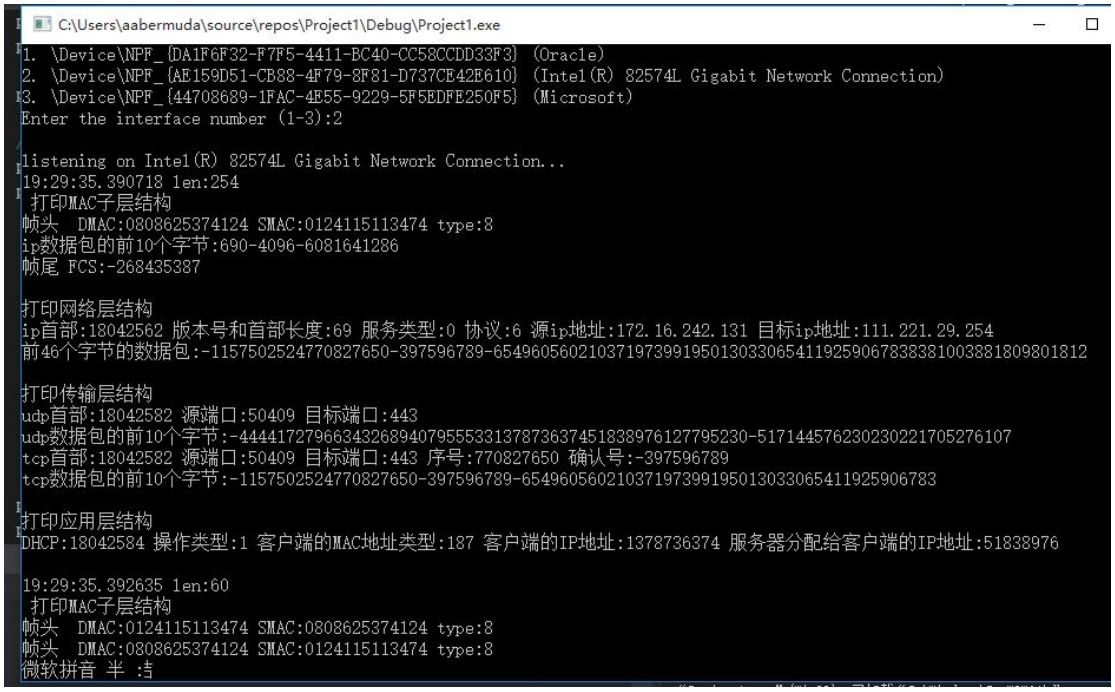
```
/* 打印传输层结构 */
printf("打印传输层结构\n");
printf("udp首部:%d 源端口:%d 目标端口:%d",
    uh,
    sport,
    dport);
printf("\n");
printf("udp数据包的前10个字节:%d%d%d%d%d%d%d%d%d%d",
    ud->byte1,
    ud->byte2,
    ud->byte3,
    ud->byte4,
    ud->byte5,
    ud->byte6,
    ud->byte7,
    ud->byte8,
    ud->byte9,
    ud->byte10);
printf("\n");
printf("tcp首部:%d 源端口:%d 目标端口:%d 序号:%d 确认号:%d",
    th,
    sport2,
    dport2,
    th->shunxu,
    th->queren);
printf("\n");
printf("tcp数据包的前10个字节:%d%d%d%d%d%d%d%d%d%d",
    td->byte1,
    td->byte2,
    td->byte3,
    td->byte4,
    td->byte5,
    td->byte6,
    td->byte7,
    td->byte8,
    td->byte9,
    td->byte10);
printf("\n\n");
```

```
/* 打印应用层结构 */
printf("打印应用层结构\n");
printf("DHCP:%d 操作类型:%d 客户端的MAC地址类型:%d 客户端的IP地址:%d 服务器分配给客户端的IP地址:%d",
    d,
    d->op,
    d->htype,
    d->ciaddr,
    d->yiaddr);
printf("\n\n");
/*
```

运行结果截图：







如上图所示，选定了 2 号适配器后，依次打印了各层的结构和格式，以及各个字

段的值。由于应用层的协议相对比较复杂，在此只选了 DHCP 协议进行打印，所得的结果如上图所示。

6. 实验总结

本次实验其实说起来并不是很难，主要在于环境的配置和对文档的熟悉，配置 winpacp 环境就花了很久。好在具体的函数和代码在文档中都给出了示例，所以只需理解了文档中的示例代码，再对对给出的函数加以运用就可以完成这个实验。

本次实验我也遇到了很多的问题，在配置环境上花了不少时间，对报文的各个字段也不熟悉，头文件和源文件的互相包含问题等等。最后还是解决了大部分问题，完成了实验。通过本实验，我更加熟悉了各层之间存在的结构关系，对如何获取高层的报文信息，如何得到每个字段具体值都有了更深的理解，对 WinPacp 编程也有了一定的认识和了解。