计算机网络实验

3. 捕获并分析帧和IP报文

厦门大学软件学院

黄炜 助理教授

实验目的

- 捕获并分析以太网的帧，获取目标与源网卡的MAC和IP地址
- 获取本机地址
  - IPCONFIG.EXE
  - 通过WinSock的GetAddress命令
- 获取远端MAC地址
  - ARP
  - WinPCAP

只获取本机MAC地址

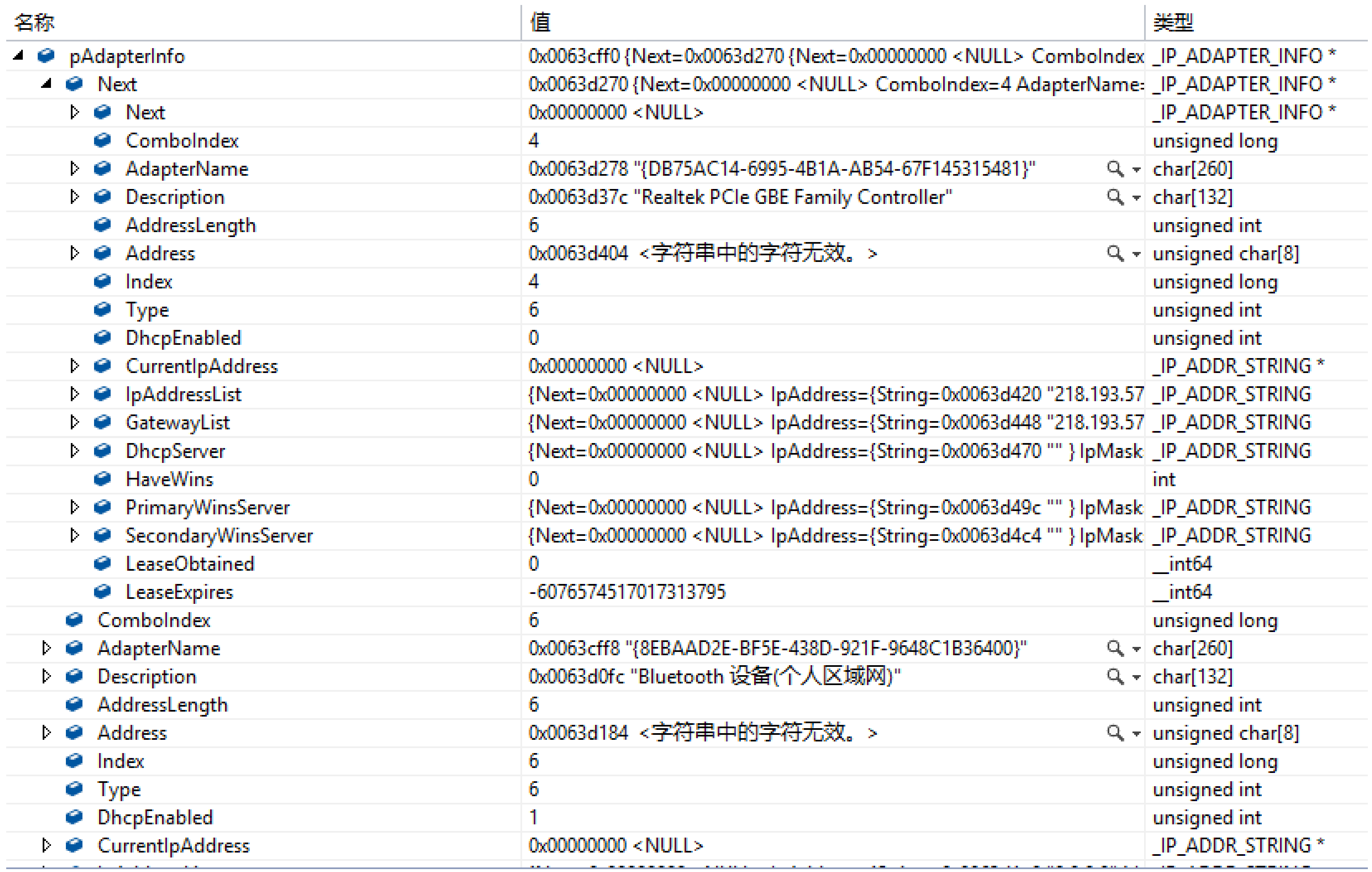
包含文件头和库

- `#include <Iphlpapi.h>`
- `#include <windows.h>`
- `#pragma comment(lib,"Iphlpapi")`
- `#pragma comment(lib,"WS2_32")`
- `using namespace std;`

包含文件头和库

- 1、申请内存
  - `::GetAdaptersInfo(pAdapterInfo, &ulLen);`
  - `pAdapterInfo=(PIP_ADAPTER_INFO)::malloc(ulLen);`
  - 此时，**ulLen**为网卡个数（禁用除外）乘以640。
- 2、获取本地适配器结构信息
  - `::GetAdaptersInfo(pAdapterInfo, &ulLen);`
- 3、**malloc**新建的变量都要**free**

适配器信息



名称	值	类型
◢ pAdapterInfo	0x0063cff0 {Next=0x0063d270 {Next=0x00000000 <NULL> ComboIndex	_IP_ADAPTER_INFO *
◢ Next	0x0063d270 {Next=0x00000000 <NULL> ComboIndex=4 AdapterName=	_IP_ADAPTER_INFO *
▷ Next	0x00000000 <NULL>	_IP_ADAPTER_INFO *
ComboIndex	4	unsigned long
▷ AdapterName	0x0063d278 "{DB75AC14-6995-4B1A-AB54-67F145315481}"	char[260]
▷ Description	0x0063d37c "Realtek PCIe GBE Family Controller"	char[132]
AddressLength	6	unsigned int
▷ Address	0x0063d404 <字符串中的字符无效。>	unsigned char[8]
Index	4	unsigned long
Type	6	unsigned int
DhcpEnabled	0	unsigned int
▷ CurrentIpAddress	0x00000000 <NULL>	_IP_ADDR_STRING *
▷ IpAddressList	{Next=0x00000000 <NULL> IpAddress={String=0x0063d420 "218.193.57	_IP_ADDR_STRING
▷ GatewayList	{Next=0x00000000 <NULL> IpAddress={String=0x0063d448 "218.193.57	_IP_ADDR_STRING
▷ DhcpServer	{Next=0x00000000 <NULL> IpAddress={String=0x0063d470 "" } IpMask	_IP_ADDR_STRING
HaveWins	0	int
▷ PrimaryWinsServer	{Next=0x00000000 <NULL> IpAddress={String=0x0063d49c "" } IpMask	_IP_ADDR_STRING
▷ SecondaryWinsServer	{Next=0x00000000 <NULL> IpAddress={String=0x0063d4c4 "" } IpMask	_IP_ADDR_STRING
LeaseObtained	0	__int64
LeaseExpires	-6076574517017313795	__int64
ComboIndex	6	unsigned long
▷ AdapterName	0x0063cff8 "{8EBAAD2E-BF5E-438D-921F-9648C1B36400}"	char[260]
▷ Description	0x0063d0fc "Bluetooth 设备(个人区域网)"	char[132]
AddressLength	6	unsigned int
▷ Address	0x0063d184 <字符串中的字符无效。>	unsigned char[8]
Index	6	unsigned long
Type	6	unsigned int
DhcpEnabled	1	unsigned int
▷ CurrentIpAddress	0x00000000 <NULL>	_IP_ADDR_STRING *

运行结果

```
NIC 1:

        IP: 0.0.0.0; Mask: 0.0.0.0; Gateway: 0.0.0.0

        MAC: C4D987******

NIC 2:

        IP: 218.193.57.***; Mask: 255.255.255.***; Gateway:
218.193.57.***

        MAC: F8B156******
```

代码示例

```
PIP_ADAPTER_INFO pAdapterInfo = NULL;
ULONG ulLen = 0;
::GetAdaptersInfo(pAdapterInfo, &ulLen);
pAdapterInfo = (PIP_ADAPTER_INFO)::malloc(ulLen);
::GetAdaptersInfo(pAdapterInfo, &ulLen);
int count = 0;
while (pAdapterInfo) {
        printf("NIC %d: \n", ++count);
        printf("\tIP: %s; Mask: %s; Gateway: %s\n", pAdapterInfo-
>IpAddressList.IpAddress.String, pAdapterInfo->IpAddressList.IpMask.String,
pAdapterInfo->GatewayList.IpAddress.String);
        printf("\tName: %s; Desc: %s\n", pAdapterInfo->AdapterName, pAdapterInfo-
>Description);
        printf("\tMAC: ");
        for (size_t i = 0; i < pAdapterInfo->AddressLength; i++) {
                printf("%02X", pAdapterInfo->Address[i]);
        }
        printf("\n");
        pAdapterInfo = pAdapterInfo->Next;
}
system("pause");
if (pAdapterInfo) {
        free(pAdapterInfo);
}
```

Windows环境

事前准备

- 安装WinPCAP
  - WinPcap_4_1_3.exe
- 解压缩WpdPack将文件夹拷出备用
  - 4.1.1-WpdPack.zip
  - Include、Lib
- 正确解析MAC和IP地址是本节课第一要务，也是基本功

包含头文件和库

```
#define HAVE_REMOTE
#include <pcap.h>
#include <Packet32.h>
#include <ntddndis.h>
#pragma comment(lib, "Packet")
#pragma comment(lib, "wpcap")
#pragma comment(lib, "WS2_32")
```

数据结构定义

IP帧格式

```
typedef struct ip_header {
        u_char ver_ihl;                // Version (4 bits) +
Internet header length (4 bits)
        u_char tos;                    // Type of service
        u_short tlen;                  // Total length
        u_short identification; // Identification
        u_short flags_fo;              // Flags (3 bits) + Fragment
offset (13 bits)
        u_char ttl;                    // Time to live
        u_char proto;                  // Protocol
        u_short crc;                   // Header checksum
        u_char saddr[4];               // Source address
        u_char daddr[4];               // Destination address
        u_int  op_pad;                         // Option + Padding
} ip_header;
```

数据结构定义

```
typedef struct mac_header {
        u_char dest_addr[6];
        u_char src_addr[6];
        u_char type[2];
} mac_header;
```

以太网数据链路层的帧格式

用回调方法捕获数据包

```
/* prototype of the packet handler */
void packet_handler(u_char *param, const struct pcap_pkthdr
*header, const u_char *pkt_data);
```

packet_handler指向一个可以接收数据包的函数。这个函数会在收到每个新的数据包并收到一个通用状态时被libpcap所调用。

函数主体

- 1、获取本地适配器结构信息，打开适配器
  - **pcap_findalldevs_ex; pcap_freealldevs**
- 2、循环编译过滤器
  - **pcap_compile**
  - **pcap_setfilter**
- 3、开始循环捕获
  - **pcap_loop**

获得设备列表

```
/* Retrieve the device list */
if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL, &alldevs,
errbuf) == -1) {
        fprintf(stderr, "Error in pcap_findalldevs: %s\n",
errbuf);
        exit(1);
}
/* Print the list */
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");
}
```

选择设备

```
if (i == 0) {
        printf("\nNo interfaces found! Make sure WinPcap is
installed.\n");
        return -1;
}
printf("Enter the interface number (1-%d):", i);
scanf_s("%d", &inum);
if (inum < 1 || inum > i) {
        printf("\nInterface number out of range.\n");
        /* Free the device list */
        pcap_freealldevs(alldevs);
        return -1;
}
/* Jump to the selected adapter */
for (d = alldevs, i = 0; i< inum - 1; d = d->next, i++);
```

选择设备

设备名

捕获包，混杂模式
65536为所有包

```c
/* Open the adapter */
if ((adhandle = pcap_open(d->name, 65536,
PCAP_OPENFLAG_PROMISCUOUS,  1000, NULL, errbuf)) == NULL) {
```

混杂模式

时延

远程验证

错误缓冲

```c
        fprintf(stderr, "\nUnable to open the adapter. %s is
not supported by WinPcap\n");
        pcap_freealldevs(alldevs);
        return -1;
}
```

释放设备列表，最后一步都要释放

预处理

```
if (pcap_datalink(adhandle) != DLT_EN10MB) {
        fprintf(stderr, "\nThis program works only on Ethernet
networks.\n");
        pcap_freealldevs(alldevs);
        return -1;
}
if (d->addresses != NULL)
        netmask = ((struct sockaddr_in *)(d->addresses-
>netmask))->sin_addr.S_un.S_addr;
else
        netmask = 0xffffff;
```

检查链路层。只简单支持以太网。

检索接口的第一个地址的掩码

如果接口没有地址，假设在一个C类网络

编译和设置过滤器

编译过滤器

char packet_filter[] = "ip and udp";

```
//compile the filter
if (pcap_compile(adhandle, &fcode, packet_filter, 1, netmask)
<0) {
      fprintf(stderr, "\nUnable to compile the packet filter.
Check the syntax.\n");
      pcap_freealldevs(alldevs);
      return -1;
}
//set the filter
if (pcap_setfilter(adhandle, &fcode)<0) {
      fprintf(stderr, "\nError setting the filter.\n");
      pcap_freealldevs(alldevs);
      return -1;
}
```

设置过滤器

开始捕获

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


/* At this point, we don't need any more the device list.
Free it */
pcap_freealldevs(alldevs);


/* start the capture */
pcap_loop(adhandle, 0, packet_handler, NULL);
```

开始捕获

开始捕获

```
void packet_handler(u_char *param, const struct pcap_pkthdr
*header, const u_char *pkt_data)
{
...
       mac_header *mh;
       ip_header *ih;
...
       int length = sizeof(mac_header)+sizeof(ip_header);
       for (int i = 0; i<length; i++) {
              printf("%02X ", pkt_data[i]);
              if ((i & 0xF) == 0xF)
                     printf("\n");
       }
       printf("\n");
```

通过libpcap的每一个传入的数据包调用回调函数

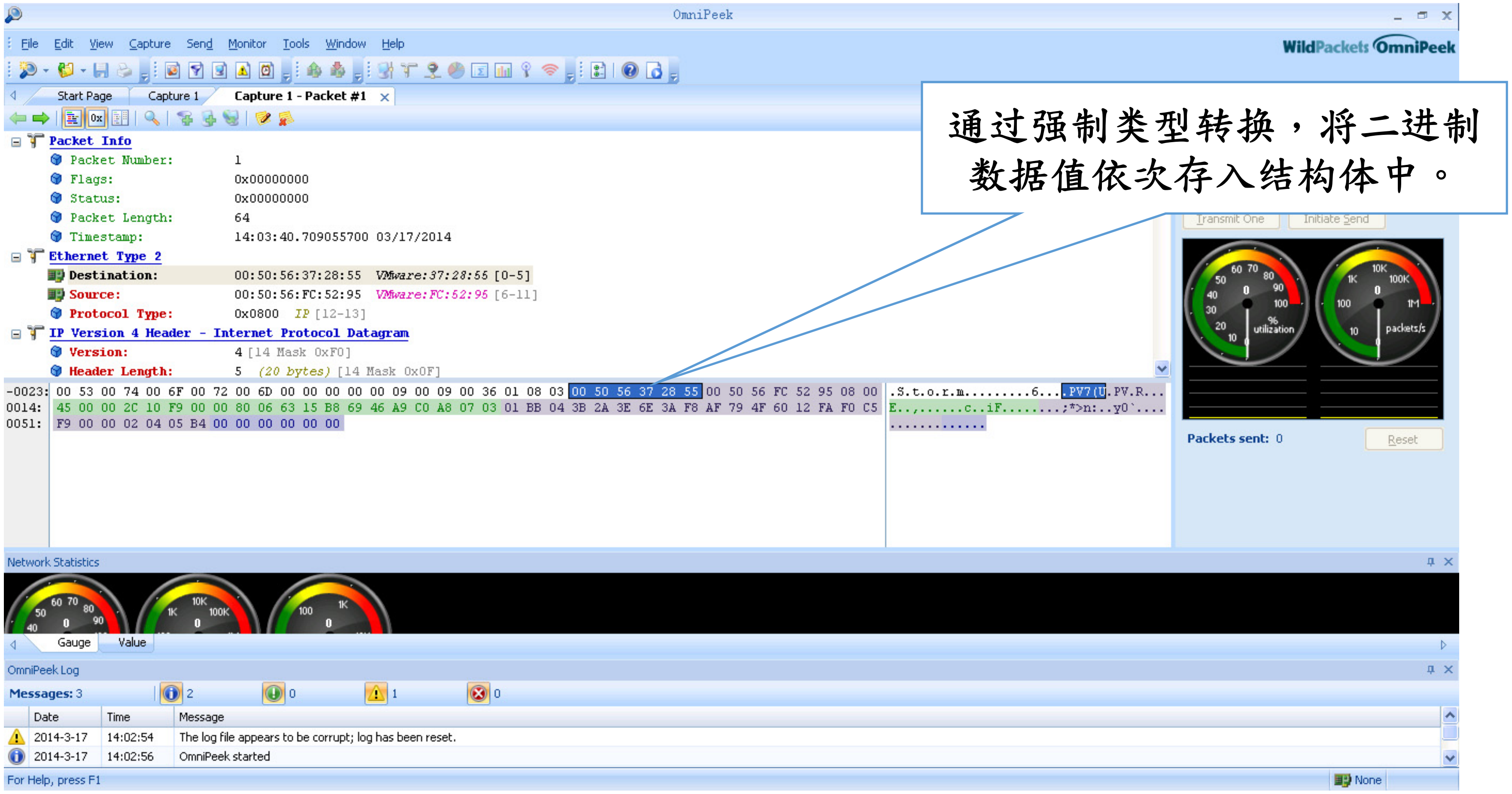
按二进制输出数据

捕获后的处理

```
mh = (mac_header*)pkt_data;
printf("mac_header:\n");
printf("\tdest_addr: ");
for (int i = 0; i<6; i++) {
      printf("%02X ", mh->dest_addr[i]);
}
printf("\n");
printf("\tsrc_addr: ");
for (int i = 0; i<6; i++) {
      printf("%02X ", mh->src_addr[i]);
}
printf("\n");
printf("\ttype: %04X", ntohs(mh->type));
printf("\n");
```

通过强制类型转换，将二进制数据值依次存入结构体中。

编程时注意对照Omnipeek工具



通过强制类型转换，将二进制数据值依次存入结构体中。

```
Packet Info
    Packet Number:        1
    Flags:                0x00000000
    Status:               0x00000000
    Packet Length:        64
    Timestamp:            14:03:40.709055700 03/17/2014
Ethernet Type 2
    Destination:          00:50:56:37:28:55  VMware:37:28:55 [0-5]
    Source:               00:50:56:FC:52:95  VMware:FC:52:95 [6-11]
    Protocol Type:        0x0800  IP [12-13]
IP Version 4 Header - Internet Protocol Datagram
    Version:              4 [14 Mask 0xF0]
    Header Length:        5  (20 bytes) [14 Mask 0x0F]

-0023: 00 53 00 74 00 6F 00 72 00 6D 00 00 00 00 00 09 00 09 00 36 01 08 03 00 50 56 37 28 55 00 50 56 FC 52 95 08 00  .S.t.o.r.m.........6....PV7(U.PV.R...
0014: 45 00 00 2C 10 F9 00 00 80 06 63 15 B8 69 46 A9 C0 A8 07 03 01 BB 04 3B 2A 3E 6E 3A F8 AF 79 4F 60 12 FA F0 C5  E..,......c..iF........;*>n:..yO`....
0051: F9 00 00 02 04 05 B4 00 00 00 00 00 00  .............
```

Messages: 3 2 0 1 0

```
Date       Time      Message
2014-3-17  14:02:54  The log file appears to be corrupt; log has been reset.
2014-3-17  14:02:56  OmniPeek started
```

For Help, press F1

捕获后的处理

```
/* retireve the position of the ip header */
ih = (ip_header *)(pkt_data + sizeof(mac_header)); //length of
ethernet header

printf("ip_header\n");
printf("\t%-10s: %02X\n", "ver_ihl", ih->ver_ihl);
printf("\t%-10s: %02X\n", "tos", ih->tos);
printf("\t%-10s: %04X\n", "tlen", ntohs(ih->tlen));
printf("\t%-10s: %04X\n", "identification", ntohs(ih-
>identification));
printf("\t%-10s: %04X\n", "flags_fo", ntohs(ih->flags_fo));
printf("\t%-10s: %02X\n", "ttl", ih->ttl);
printf("\t%-10s: %02X\n", "proto", ih->proto);
printf("\t%-10s: %04X\n", "crc", ntohs(ih->crc));
printf("\t%-10s: %08X\n", "op_pad", ntohs(ih->op_pad));
printf("\t%-10s: ", "saddr:");
```

通过强制类型转换，将二进制数据值依次存入结构体中。

网络端序转为主机端序ntohs

捕获后的处理

```
for (int i = 0; i<4; i++) {
        printf("%02X ", ih->saddr[i]);
}
printf(" ");
for (int i = 0; i<4; i++) {
        printf("%d.", ih->saddr[i]);
}
printf("\n");
printf("\t%-10s: ", "daddr");
for (int i = 0; i<4; i++) {
        printf("%02X ", ih->daddr[i]);
}
printf(" ");
for (int i = 0; i<4; i++) {
        printf("%d.", ih->daddr[i]);
}
printf("\n");
```

运行结果

```
14:30:38.511184 len:339
00 0C 29 73 69 8A 00 50 56 FC 52 95 08 00 45 00
01 45 1D FB 00 00 80 11 8C 56 C0 A8 07 02 C0 A8
07 04 00 35 CB 42
mac_header:
        dest_addr : 00 0C 29 73 69 8A
        src_addr  : 00 50 56 FC 52 95
        type      : 0800
ip_header
        ver_ihl   : 45
        tos       : 00
        tlen      : 0145
        identification: 1DFB
        flags_fo  : 0000
        ttl       : 80
        proto     : 11
        crc       : 8C56
        op_pad    : 0035CB42
        saddr     : C0 A8 07 02   192.168.7.2
        daddr     : C0 A8 07 04   192.168.7.4
```

有时候在这里还有PPPoE头

注意：MAC地址合理。

注意：IP Ver应为4

注意：IP地址合理。

Linux环境

包含头文件

```
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <net/if.h>
#include <netinet/if_ether.h>
#include <linux/sockios.h>
#include <stdio.h>
#include <string.h>
```

强制类型转换

```
int main()
{
...
        eh = (struct ethhdr *)ep;

        fd = socket(AF_INET, SOCK_PACKET, htons(0x0003));
        strcpy(ifr.ifr_name, "eth0");
        i = ioctl(fd, SIOCGIFFLAGS, &ifr);
        ifr.ifr_flags |= IFF_PROMISC;
        i = ioctl(fd, SIOCSIFFLAGS, &ifr);
...
```

输出源地址和目的地址

```
        while (1) {
                fl = read(fd, ep, sizeof(ep));
                if (fl > ETH_HLEN) {
                        printf("Packet is from ");
                        for (i = 0; i < 6; i++)
                                printf("%x-", eh->h_source[i]);
                        printf(" to ");
                        for (i = 0; i < 6; i++)
                                printf("%x-", eh->h_dest[i]);
                        printf("\n");
                } // End of if
        } // End of While
}
```

计算机网络实验

3.

THANK YOU.

厦门大学软件学院

黄炜 助理教授