实验3：通过编程获取IP地址与MAC地址的对应关系

学号：2112066 姓名：于成俊 专业：密码科学与技术

一、实验要求：

（1）在IP数据报捕获与分析编程实验的基础上，学习NPcap的数据包发送方法。

（2）通过NPcap编程，获取IP地址与MAC地址的映射关系。

（3）程序要具有输入IP地址，显示输入IP地址与获取的MAC地址对应关系界面。界面可以是命令行界面，也可以是图形界面，但应以简单明了的方式在屏幕上显示。

（4）编写的程序应结构清晰，具有较好的可读性。

二、实验原理：

1.Npcap捕获数据包

- 设备列表获取方法：

  NpCap 提供了pcap_findalldevs 函数来获取计算机上的网络接口设备的列表；此函数会为传入的 pcap_if_t 赋值（该类型是一个表示了设备列表的链表头；每一个这样的节点都包含了 name 和 description 域来描述设备）。

- 打开网络接口方法：

  Npcap提供了pcap_open_live函数。该函数用于获取数据包捕获句柄以查看网络上的数据包。它接收五个参数：

  - const char *device：网络设备的名字。
  - int snaplen：定义了 pcap 捕获的最大字节数。
  - int promisc：指定是否将接口置于混杂模式；一般情况下，适配器只接收发给它自己的数据包， 而那些在其他机器之间通讯的数据包，将会被丢弃。但混杂模式将会捕获所有的数据包（因为我们需要捕获其他适配器的数据包，所以需要打开这个开关，即设置为1）。
  - int to_ms：指定读取数据的超时时间，以毫秒为单位；设置为 0 说明没有超时（如果没有数据包到达，则永远不返回）；对应的还有 -1：读操作后立刻返回。
  - char *errbuf：用于存储错误信息字符串的缓冲区

  该函数成功时返回 pcap_t *类型的handle，失败时返回 NULL。如果返回 NULL，则 errbuf 会填充适当的错误消息。

- 数据报捕获方法：

  Npcap提供了 pcap_next_ex 函数。它接收三个参数：

  - pcap_t* p：已打开的捕捉实例的描述符。
  - struct pcap_pkthdr** pkt_header：报文头

    - const u_char** pkt_data：报文内容
- 数据报发送方法：

  Npcap提供了pcap_sendpacket函数。它接收三个参数：

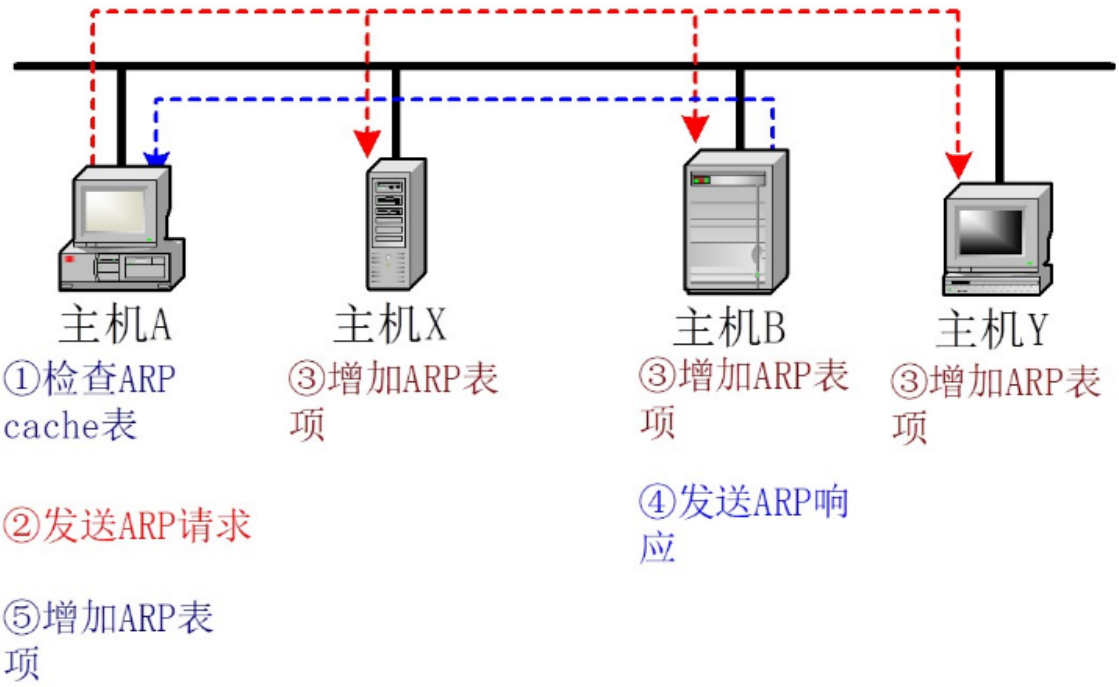
    - pcap_t *：一个适配器
    - const u_char *：一个装有要发送数据的缓冲区
    - int：要发送的长度

2.ARP协议

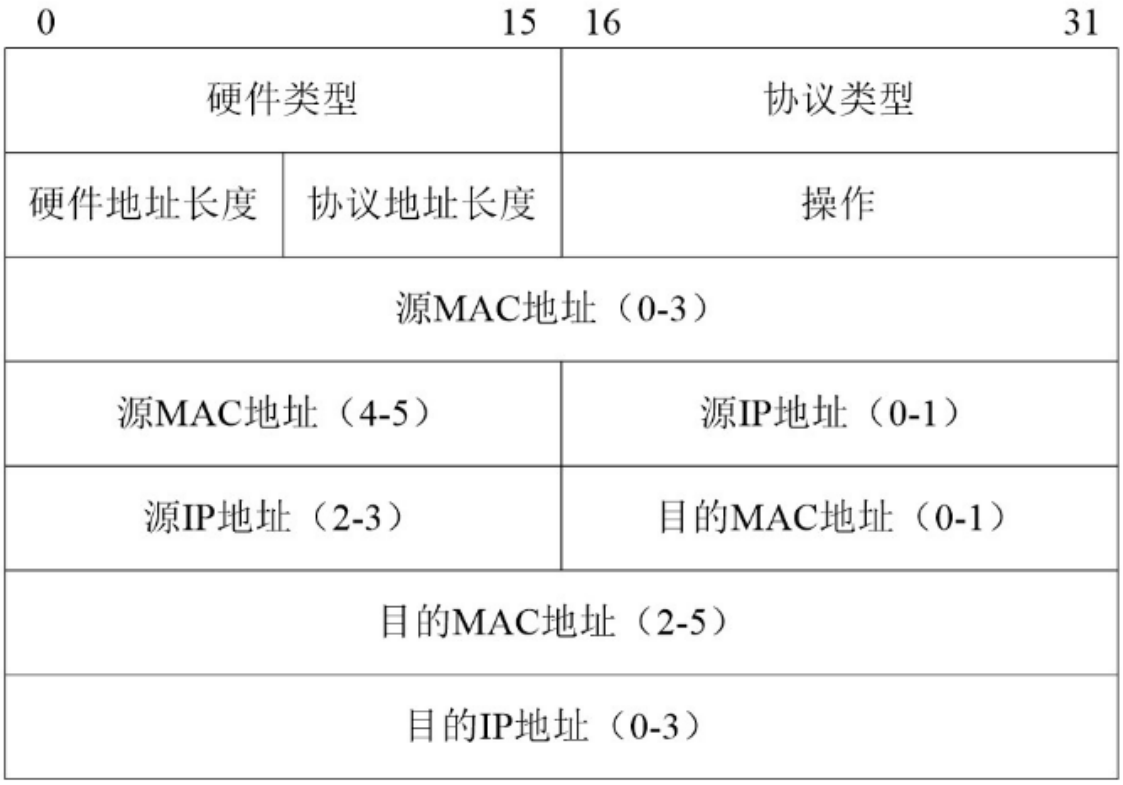
①ARP：

地址解析协议，即ARP（Address Resolution Protocol），是根据IP地址获取物理地址的一个TCP/IP协议。主机发送信息时将包含目标IP地址的ARP请求**广播**到局域网络上的所有主机，并接收返回消息，以此确定目标的物理地址；收到返回消息后将该IP地址和物理地址**存入本机ARP缓存**中并保留一定时间，下次请求时直接查询ARP缓存以节约资源。地址解析协议是建立在网络中各个主机互相信任的基础上的，局域网络上的主机可以自主发送ARP应答消息，其他主机收到应答报文时不会检测该报文的真实性就会将其记入本机ARP缓存；

②ARP的完整工作过程



③ARP的报文格式：



具体如下：

- **硬件类型：** 16位字段，用来定义运行ARP的网络类型。每个局域网基于其类型被指派一个整数。例如：以太网的类型为1。ARP可用在任何物理网络上。
- **协议类型：** 16位字段，用来定义使用的协议。例如：对IPv4协议这个字段是0800。ARP可用于任何高层协议
- **硬件地址长度：** 8位字段，用来定义MAC地址的长度，以字节为单位。例如：对于以太网的值为6。
- **协议地址长度：** 8位字段，用来定义IP地址的长度，以字节为单位。例如：对于IPv4协议的值为4。
- **操作：** 16位字段，用来定义报文的类型。已定义的分组类型有两种：ARP请求（0x0001），ARP响应（0x0002）。
- **源MAC地址：** 这是一个可变长度字段，用来定义发送方的MAC地址。例如：对于以太网这个字段的长度是6字节。
- **源IP地址：** 这是一个可变长度字段，用来定义发送方的IP地址。例如：对于IP协议这个字段的长度是4字节。
- **目的MAC地址：**这是一个可变长度字段。ARP请求中该字段没有意义；ARP响应中为接收方的MAC地址。例如，对以太网来说这个字段为6字节。对于ARP请求报文，这个字段为全0，因为发送方并不知道目标的MAC地址。全1时，代表广播地址。
- **目的IP地址：** ARP请求中为请求解析的IP地址；ARP响应中为接收方的IP地址。

④静态映射：

静态映射就是要手动创建一张ARP表，把逻辑`IP`地址和物理地址关联起来。这个ARP表储存在网络中的每一台机器上。例如，知道其机器的IP地址但不知道其物理地址的机器就可以通过查ARP表找出对应的物理地址。这样做有一定的局限性，因为物理地址可能发生变化：

1. 机器可能更换NIC（网络适配器），结果变成一个新的物理地址。
2. 在某些局域网中，每当计算机加电时，他的物理地址都要改变一次。
3. 移动电脑可以从一个物理网络转移到另一个物理网络，这样会时物理地址改变。

要避免这些问题出现，必须定期维护更新ARP表，此类比较麻烦而且会影响网络性能。

⑤动态映射：

动态映射时，每次只要机器知道另一台机器的逻辑（IP）地址，就可以使用协议找出相对应的物理地址。已经设计出的实现了动态映射协议的有ARP和RARP两种。ARP把逻辑（IP）地址映射为物理地址。RARP把物理地址映射为逻辑（IP）地址。

三、实验过程

1.大致流程

（1）定义帧首部和ARP帧结构体

（2）使用`pcap_findalldevs()`函数，获取可用的网络设备列表。

（3）遍历设备列表并打印设备信息，包括设备名字、设备描述以及IP地址类型，若为IPv4，再打印它的IP地址。

（4）输入要打开的网络接口号，并用`pcap_open_live()`函数打开网络接口。

（5）设置ARP帧的内容，包括帧类型、目的MAC地址、目的IP地址、源MAC地址、源IP地址等。

（6）使用`pcap_sendpacket()`函数发送设置好的ARP帧。

（7）使用`pcap_next_ex()`捕获ARP数据报，以此获得IP地址与MAC地址的对应关系。

（8）**输入目的IP地址**，切换到打开的网络接口所在的网段，重复（5）（6）（7）步骤，来获得指定IP地址与MAC地址的对应关系。

2.实现细节

定义结构体

根据雨课堂的介绍，我们需要将捕获的的结构体强制转化成标准数据报所具有的格式。我们使用pack
（1）关闭填充字节，使得结构体中的成员变量按照1字节的倍数进行对齐，这意味着每个成员变量都会紧密相邻。

结构体定义如下：

```
// 字节对齐方式
#pragma pack(1)
// 帧首部
typedef struct FrameHeader_t {
    // 目的地址
    BYTE DesMAC[6];
    // 源地址
    BYTE SrcMAC[6];
    // 帧类型
    WORD FrameType;
}FrameHeader_t;
// ARP帧
typedef struct ARPFrame_t {
    // 帧首部
    FrameHeader_t FrameHeader;
    // 硬件类型
    WORD HardwareType;
    // 协议类型（本次实验应该为ARP）
    WORD ProtocolType;
```

```
    // 硬件地址长度
    BYTE HLen;
    // 协议地址长度
    BYTE PLen;
    // 操作类型（比如ARP的请求或应答）
    WORD Operation;
    // 发送方MAC地址(源MAC地址)
    BYTE SendHa[6];
    // 发送方IP地址(源IP地址)
    DWORD SendIP;
    // 接收方MAC地址(目的MAC地址)
    BYTE RecvHa[6];
    // 接收方IP地址(目的IP地址)
    DWORD RecvIP;
}ARPFrame_t;
// 结束字节对齐方式
# pragma pack()
```

获取本机的设备列表

```
char errbuf[PCAP_ERRBUF_SIZE];  //用于存储错误信息
    pcap_if_t* alldevs;     //pcap_if_t 用于表示网络设备的信息
    pcap_if_t* dev;
    // 获取可用的网络设备列表
    if (pcap_findalldevs(&alldevs, errbuf) == -1) {
        cerr << "Error in pcap_findalldevs: " << errbuf << endl;
        return 1;
    }
```

`pcap_findalldevs()` 函数若成功执行，它将返回一个指向包含所有可用的网络设备信息的链表的头节点的指针。这就是 `alldevs` 所指向的值。使用alldevs即可遍历设备列表。

打印设备信息

将刚刚捕获到的网络接口的名字、描述和IP地址打印在屏幕上。

```
// 遍历设备列表并打印设备信息
    int i = 1;
    for (dev = alldevs; dev; dev = dev->next) {
        //打印网络接口名字
        cout << "Device " << i++ << ": " << dev->name << endl;
        //打印网络接口描述
        if (dev->description) {
            cout << "   Description: " << dev->description << endl;
        }
        else {
            cout << "   Description: None" << endl;
        }
        // 打印网络接口的IP地址
        pcap_addr_t* address;
        for (address = dev->addresses; address != NULL; address = address->next)
        {
            switch (address->addr->sa_family)
            {
                // IPV4类型地址
            case AF_INET:
                cout << "地址类型为IPv4 ";
```

```
                if (address->addr != NULL)
                {
                    // 打印IP地址
                    cout << "IP地址:  " << inet_ntoa(((struct
sockaddr_in*)address->addr)->sin_addr) << endl;
                }
                break;
                // IPV6类型地址
            case AF_INET6:
                cout << "地址类型为IPV6" << endl;
                break;
            default:
                break;
            }
        }
        cout << endl;
    }
```

选择要打开的网络接口

```
// 打开网络接口
    pcap_t* handle = NULL;
    int number;
    cout << "请输入要打开的网络接口号: " << endl;
    cin >> number;
    i = 0;
    for (dev = alldevs; dev; dev = dev->next) {
        i++;
        if (i == number) {
            // 打开网络接口以进行数据包捕获
            handle = pcap_open_live(dev->name, BUFSIZ, 1, 1000, errbuf);
            break;
        }
    }
    if (handle == NULL) {
        cerr << "Error opening network interface: " << errbuf << endl;
        return 1;
    }
```

发送ARP请求，获得指定网络接口上绑定的IP地址与MAC地址的对应关系

流程如下：

1.获取接口上绑定的IP地址

```
char* IP = new char[40];
    // 将设备的IP地址赋值给IP数组
    strcpy(IP, inet_ntoa(((struct sockaddr_in*)(dev->addresses)->addr)-
>sin_addr));
```

2.组装报文：

①使用`55-55-55-55-55-55`作为源MAC地址。

②使用`110.110.110.110`作为源IP地址。

③参考雨课堂，其他字段按照ARP请求要求的格式设置。

```
// ARP初始帧
    ARPFrame_t ARPFrame;
    //设置目的地址为广播地址
    for (int i = 0; i < 6; i++)
    {
        ARPFrame.FrameHeader.DesMAC[i] = 0xff;
    }
    // 设置为本机网卡的MAC地址（非真实）
    for (int i = 0; i < 6; i++)
    {
        ARPFrame.FrameHeader.SrcMAC[i] = 0x55;
    }
    // 帧类型为ARP
    ARPFrame.FrameHeader.FrameType = htons(0x0806);
    // 硬件类型为以太网
    ARPFrame.HardwareType = htons(0x0001);
    // 协议类型为IP
    ARPFrame.ProtocolType = htons(0x0800);
    // 硬件地址长度为6
    ARPFrame.HLen = 6;
    // 协议地址长为4
    ARPFrame.PLen = 4;
    // 操作为ARP请求
    ARPFrame.Operation = htons(0x0001);
    // 设置为本机网卡的MAC地址（非真实）
    for (int i = 0; i < 6; i++)
    {
        ARPFrame.SendHa[i] = 0x55;
    }
    // 设置为本机网卡上绑定的IP地址（非真实）
    ARPFrame.SendIP = inet_addr("110.110.110.110");
    // 设置目的MAC地址
    for (int i = 0; i < 6; i++)
    {
        ARPFrame.RecvHa[i] = 0x00;
    }
    // 设置为请求的IP地址
    ARPFrame.RecvIP = inet_addr(IP);
```

3.发送ARP请求，请求本机网络接口上绑定的IP地址与MAC地址的对应关系：本地主机模拟一个远端主机，发送一个ARP请求报文，该请求报文请求本机网络接口上绑定的IP地址与MAC地址的对应关系。

```
// 发送设置好的帧内容，如果发送失败直接退出
    if (pcap_sendpacket(handle, (u_char*)&ARPFrame, sizeof(ARPFrame_t)) != 0)
    {
        cout << "发送失败" << endl;
        return -1;
    }
```

4.接下来捕获ARP响应，打印绑定的IP地址与MAC地址的对应关系。

```
// 声明即将捕获的ARP帧
    ARPFrame_t* ARPPacket;

    // 开始捕获数据报
    while (true)
```

```
    {
        pcap_pkthdr* pkt_header;
        const u_char* pkt_data;
        int result = pcap_next_ex(handle, &pkt_header, &pkt_data);
        // 捕获到相应的信息
        if (result == 1)
        {
            ARPPacket = (ARPFrame_t*)pkt_data;
            if ((ntohs(ARPPacket->FrameHeader.FrameType) == 0x0806) &&
(ntohs(ARPPacket->Operation) == 0x0002))
                //如果帧类型为ARP并且操作为ARP应答
            {
                // 打印本机的MAC地址和IP地址
                cout << "IP地址: " << IP << endl;
                cout << "MAC地址: ";
                printf("%02x:%02x:%02x:%02x:%02x:%02x\n",
                    ARPPacket->FrameHeader.SrcMAC[0],
                    ARPPacket->FrameHeader.SrcMAC[1],
                    ARPPacket->FrameHeader.SrcMAC[2],
                    ARPPacket->FrameHeader.SrcMAC[3],
                    ARPPacket->FrameHeader.SrcMAC[4],
                    ARPPacket->FrameHeader.SrcMAC[5]
                );
                break;
            }
        }
    }
```

5.**切换到这个网络接口的网段**:

```
     //切换到这个网络接口的网段
     for (int i = 0; i < 6; i++){
         ARPFrame.FrameHeader.SrcMAC[i] = ARPPacket->FrameHeader.SrcMAC[i];
         ARPFrame.SendHa[i] = ARPPacket->FrameHeader.SrcMAC[i];
     }
     ARPFrame.SendIP = ARPPacket->SendIP;
```

6.接着**输入想要发送的目的IP地址**，请求以太网中其他主机的IP地址和MAC地址的对应关系。发送步骤与之前是**一样的**。依然可以直接使用非真实的IP和MAC地址进行发送，虽然本机网卡发送时需用的是虚拟MAC和IP地址，但是网关接收到组建的ARP请求后会由网关发出一个ARP请求，找到本机发送网卡的真实IP和MAC地址，从而进一步获取远程主机的MAC地址。（注意，目的IP地址要和本机IP地址在同一个网段，否则发送不成功）

```
// 目的IP地址
    char* DestIP = new char[40];

    cout << "请输入你想发送到的目的IP地址:" << endl;
    cin >> DestIP;

    // 设置目的IP地址
    ARPFrame.RecvIP = inet_addr(DestIP);
    // 发送设置好的帧内容，如果发送失败直接退出
    if (pcap_sendpacket(handle, (u_char*)&ARPFrame, sizeof(ARPFrame_t)) != 0)
    {
        cout << "发送失败" << endl;
        return -1;
```

```
    }
    else
    {
        cout << "发送成功" << endl;
    }
    ARPFrame_t* NewARPPacket;

    //开始捕获数据报
    while (true)
    {
        pcap_pkthdr* pkt_headerNew;
        const u_char* pkt_dataNew;
        int result = pcap_next_ex(handle, &pkt_headerNew, &pkt_dataNew);
        if (result == 1)
        {
            NewARPPacket = (ARPFrame_t*)pkt_dataNew;
            if ((ntohs(NewARPPacket->FrameHeader.FrameType) == 0x0806) &&
(ntohs(NewARPPacket->Operation) == 0x0002))
                //如果帧类型为ARP并且操作为ARP应答
            {
                // 输出其对应的MAC地址
                printf("Mac地址：\n");
                printf("%02x:%02x:%02x:%02x:%02x:%02x\n",
                    NewARPPacket->FrameHeader.SrcMAC[0],
                    NewARPPacket->FrameHeader.SrcMAC[1],
                    NewARPPacket->FrameHeader.SrcMAC[2],
                    NewARPPacket->FrameHeader.SrcMAC[3],
                    NewARPPacket->FrameHeader.SrcMAC[4],
                    NewARPPacket->FrameHeader.SrcMAC[5]
                );
                break;
            }
        }
    }
```

最后，释放设备列表并关闭捕获点，实验结束。

```
    // 释放设备列表
    pcap_freealldevs(alldevs);
    // 关闭捕获点
    pcap_close(handle);
```

四、实验结果展示

①打印设备信息：

```
Device 1: \Device\NPF_{0AE101E0-A85A-4345-BE36-9485E105D46A}
   Description: WAN Miniport (Network Monitor)

Device 2: \Device\NPF_{090DCC41-7FC2-4672-A0C5-BA9AEC6D1D06}
   Description: WAN Miniport (IPv6)

Device 3: \Device\NPF_{7D2A8917-35AC-495F-91B2-78F265C37837}
   Description: WAN Miniport (IP)

Device 4: \Device\NPF_{37BB740B-E0FA-4E3C-9427-3EAE72999625}
   Description: Intel(R) Wi-Fi 6 AX201 160MHz
地址类型为IPv4 IP地址:  10.136.17.63
地址类型为IPV6
地址类型为IPV6
地址类型为IPV6

Device 5: \Device\NPF_{D2ACC933-3637-4AFE-8653-CE2FC59A1752}
   Description: VMware Virtual Ethernet Adapter for VMnet8
地址类型为IPv4 IP地址:  192.168.163.1
地址类型为IPV6

Device 6: \Device\NPF_{BDB875A1-AA5A-4719-8561-7056CFF78223}
   Description: VMware Virtual Ethernet Adapter for VMnet1
地址类型为IPv4 IP地址:  192.168.246.1
地址类型为IPV6
```

②选择网络接口，获取本机网络接口的`MAC`地址和`IP`地址。

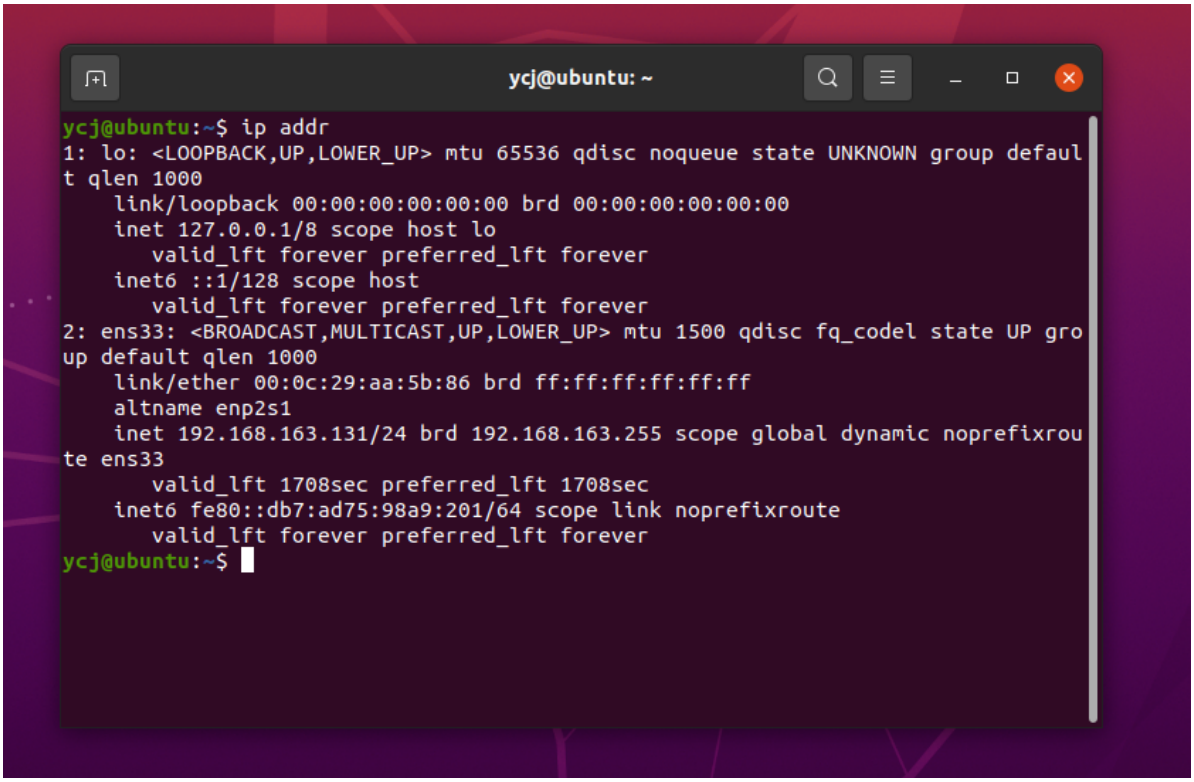
```
请输入要打开的网络接口号:
5
IP地址: 192.168.163.1
MAC地址: 00:50:56:c0:00:08
```

```
  描述. . . . . . . . . . . . . . . : VMware Virtual Ethernet Adapter for VMnet8
  物理地址. . . . . . . . . . . . . : 00-50-56-C0-00-08
```

结果一致。

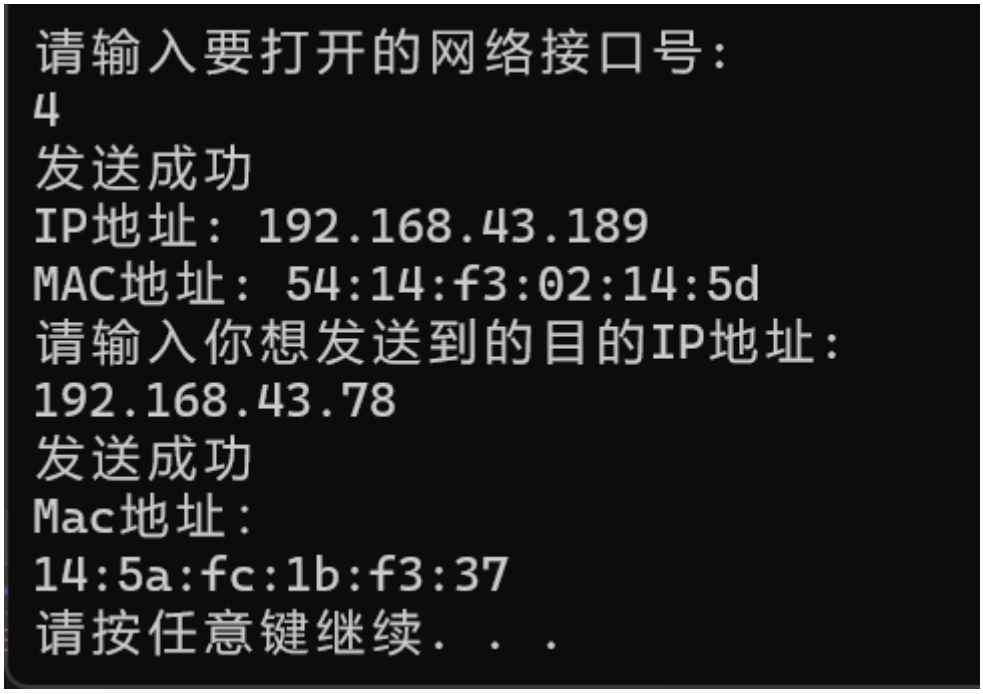
③输入虚拟机的IP地址进行测试：

```
请输入你想发送到的目的IP地址:
192.168.163.131
发送成功
Mac地址:
00:0c:29:aa:5b:86
请按任意键继续. . .
```



结果一致。

④通过连一个手机热点，输入同学的IP地址，进行测试：



代码链接：https://github.com/happy206/Network-Technology-and-Applications