实验3：获取IP地址与MAC地址的对应关系

物联网工程___2111194___胡博程

一、实验要求

摘自学院网站

通过编程获取IP地址与MAC地址的对应关系实验，要求如下：

（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。

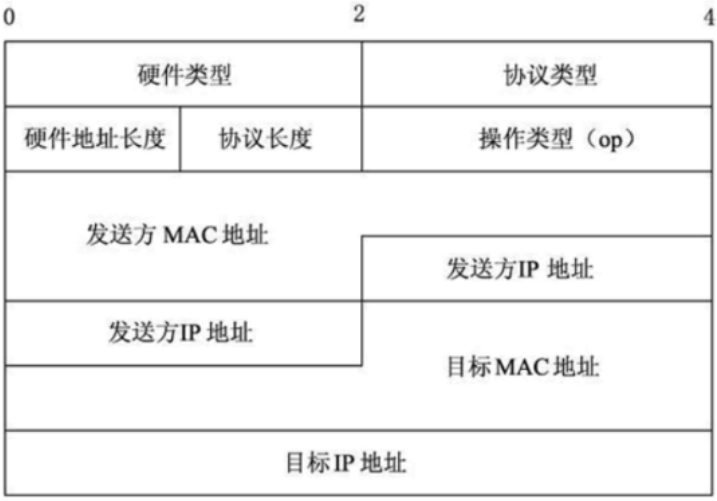
（4）编写的程序应结构清晰，具有较好的可读性。

二、实验原理与准备

1、ARP协议

ARP是一种用于将网络层地址（如IPv4地址）解析为链路层地址（如MAC地址）的网络协议。允许网络中的设备通过知道另一台设备的IP地址来发现其物理MAC地址。这是网络通信的基本要求，因为虽然IP地址用于网络层的通信，但实际的数据帧传输在链路层发生，需要使用MAC地址。

ARP报文包含以下主要字段：



1. **硬件类型（Hardware Type）**：指定网络接口的类型。对于以太网，通常是1。
2. **协议类型（Protocol Type）**：指定使用的协议类型。对于IPv4地址，通常是0x0800。
3. **硬件地址长度（Hardware Address Length）**：指定硬件地址（如MAC地址）的长度，以字节为单位。对于MAC地址，通常是6。
4. **协议地址长度（Protocol Address Length）**：指定协议地址（如IPv4地址）的长度，以字节为单位。对于IPv4地址，通常是4。
5. **操作（Operation）**：指定ARP包是请求（1）还是响应（2）。
6. **发送方MAC地址（Sender MAC Address）**：发送ARP请求或响应的设备的MAC地址。

7. **发送方IP地址（Sender IP Address）**：发送ARP请求或响应的设备的IP地址。
8. **目标MAC地址（Target MAC Address）**：ARP请求中的目标设备的MAC地址，响应中用于确认。
9. **目标IP地址（Target IP Address）**：ARP请求中要查询的目标设备的IP地址。

2、本实验的基本原理

通过ARP发包，广播一个ARP请求，它询问网络上的所有设备："拥有特定IP地址的设备，请告诉我你的MAC地址。"这个请求包括IP地址和MAC地址（作为请求者），以及想要查询的目标IP地址。

三、实验过程

1、代码总体流程

- 首先使用 `pcap_findalldevs` 函数查找所有网络适配器，并显示它们的列表。
- 用户从列表中选择一个网络适配器用于捕获和发送数据包。
- 使用 `pcap_open_live` 函数打开用户选择的网络适配器。
- 通过调用 `GetAdaptersInfo` 函数，获取系统网络适配器的信息，特别是本机的IP地址和MAC地址。
- 用户输入目标IP地址。
- 函数 `send_arp_request` 构建一个ARP请求包，其中包含了本机的IP和MAC地址，以及目标IP地址。这个请求使用广播MAC地址（FF:FF:FF:FF:FF:FF），以便在局域网内被所有设备接收。
- ARP请求通过 `pcap_sendpacket` 函数发送到网络上。
- 函数 `receive_arp_response` 循环监听网络上的ARP回复。
- 当接收到一个ARP响应时，该函数检查响应是否来自目标IP地址。如果是，它提取并显示相应的MAC地址。
- 最后，关闭打开的网络适配器并释放所有网络设备列表资源。

2、用到的数据结构

（1）结构体 in_addr

```
typedef struct in_addr {
  union {
    struct { u_char  s_b1, s_b2, s_b3, s_b4; } S_un_b;
    struct { u_short s_w1, s_w2; } S_un_w;
    u_long S_addr;
  } S_un;
} IN_ADDR, *PIN_ADDR, *LPIN_ADDR;
```

结构体 `in_addr` 用于表示IPv4网络地址，供了三种不同的方式来表示同一个IPv4地址：

1. **通过四个字节表示**：`struct { u_char s_b1, s_b2, s_b3, s_b4; } S_un_b;`
   - 这种方式允许你通过四个单独的字节（`s_b1` 到 `s_b4`）来访问IPv4地址的每一部分。这在处理IP地址的每个八位组（octet）时非常有用。
2. **通过两个短整型表示**：`struct { u_short s_w1, s_w2; } S_un_w;`
   - 这种方式允许你通过两个16位的值来访问IPv4地址。这在某些情况下可能更方便，尤其是在需要处理IP地址两个16位部分的操作中。
3. **通过一个长整型表示**：`u_long S_addr;`
   - 这种方式将整个IPv4地址视为一个32位的长整型（`u_long`）。这在需要将IP地址作为一个整体来处理时非常方便，例如在比较两个地址或将地址赋值给其他变量时。

(2) 结构体IP_ADAPTER_INFO

```
typedef struct _IP_ADAPTER_INFO {
    struct _IP_ADAPTER_INFO *Next;
    DWORD ComboIndex;
    char AdapterName[MAX_ADAPTER_NAME_LENGTH + 4];
    char Description[MAX_ADAPTER_DESCRIPTION_LENGTH + 4];
    UINT AddressLength;
    BYTE Address[MAX_ADAPTER_ADDRESS_LENGTH];
    DWORD Index;
    UINT Type;
    UINT DhcpEnabled;
    PIP_ADDR_STRING CurrentIpAddress;
    IP_ADDR_STRING IpAddressList;
    IP_ADDR_STRING GatewayList;
    IP_ADDR_STRING DhcpServer;
    WINBOOL HaveWins;
    IP_ADDR_STRING PrimaryWinsServer;
    IP_ADDR_STRING SecondaryWinsServer;
    time_t LeaseObtained;
    time_t LeaseExpires;
  } IP_ADAPTER_INFO,*PIP_ADAPTER_INFO;
```

`IP_ADAPTER_INFO` 用于存储网络适配器的信息。以下是各个字段的含义：

- `Next`：指向下一个 `IP_ADAPTER_INFO` 结构体的指针，用于实现链表结构。
- `ComboIndex`：适配器的组合索引。
- `AdapterName`：适配器的名称。
- `Description`：适配器的描述。
- `AddressLength`：适配器的物理地址的长度。
- `Address`：适配器的物理地址，通常是 MAC 地址。
- `Index`：适配器的索引。
- `Type`：适配器的类型，例如以太网适配器或令牌环适配器。
- `DhcpEnabled`：表示是否启用了 DHCP。
- `CurrentIpAddress`：指向适配器当前使用的 IP 地址的指针。
- `IpAddressList`：适配器的 IP 地址列表。

3、ARP包发送函数——send_arp_request详解

```
void send_arp_request(pcap_t *adhandle, in_addr local_ip, u_char *local_mac,
in_addr target_ip)
{
    u_char packet[sizeof(ethernet_header) + sizeof(arp_header)]; // 数据包内容，大
小为以太网帧头部 + ARP帧头部
    // 强转分离出以太网帧头部和ARP帧头部
    ethernet_header *eth = (ethernet_header *)packet;
    arp_header *arp = (arp_header *)(packet + sizeof(ethernet_header));

    // 填充以太网帧头部
    for (int i = 0; i < 6; i++)
    {
        eth->dest_mac[i] = 0xff; // 广播地址
```

```
        eth->src_mac[i] = local_mac[i]; // 本机MAC地址
    }
    eth->type = htons(ETHERNET_TYPE_ARP); // 以太网帧类型

    // 填充ARP帧头部
    arp->hardware_type = htons(1); // 硬件类型，1表示以太网
    arp->protocol_type = htons(0x0800); // 协议类型，0x0800表示IP协议
    arp->hardware_len = 6; // 硬件地址长度，6表示MAC地址长度
    arp->protocol_len = 4; // 协议地址长度，4表示IP地址长度
    arp->opcode = htons(ARP_OPCODE_REQUEST);// ARP操作码，1表示ARP请求
    // 复制到ARP帧头部各个字段
    memcpy(arp->sender_mac, local_mac, 6);
    memcpy(arp->sender_ip, &local_ip.S_un.S_addr, 4);
    memset(arp->target_mac, 0, 6);
    memcpy(arp->target_ip, &target_ip.S_un.S_addr, 4);

    if (pcap_sendpacket(adhandle, packet, sizeof(packet)) != 0) //
pcap_sendpacket函数发送数据包
    {
        std::cerr << "Error sending the packet: " << pcap_geterr(adhandle) <<
std::endl;
    }
}
```

- 函数参数：
  - `pcap_t  adhandle`：pcap 句柄，是一个指向 `pcap_t` 类型的指针，代表一个打开的网络设备（例如网卡）
  - `in_addr local_ip`：本地 IP 地址,`in_addr` 类型，用于填充ARP请求中的发送方IP地址字段。
  - `u_char local_mac`：本地 MAC 地址，用于填充ARP请求中的发送方MAC地址字段
  - `in_addr target_ip`：目标 IP 地址，发送ARP请求所查询的IP地址

首先创建了一个大小为以太网头部和 ARP 头部之和的数据包，并创建了两个指针 `eth` 和 `arp` 分别指向数据包的以太网头部和 ARP 头部。然后，填充以太网头部的信息——目标 MAC 地址被设置为广播地址（即所有位都为 1），源 MAC 地址被设置为本地 MAC 地址，类型被设置为 ARP。

接着填充 ARP 头部的信息。硬件类型被设置为 1（表示以太网），协议类型被设置为 0x0800（表示 IP），硬件地址长度被设置为 6（表示 MAC 地址的长度），协议地址长度被设置为 4（表示 IP 地址的长度），操作码被设置为 ARP 请求，发送者 MAC 地址被设置为本地 MAC 地址，发送者 IP 地址被设置为本地 IP 地址，目标 MAC 地址被设置为 0（表示未知），目标 IP 地址被设置为目标 IP 地址。

最后，它使用 `pcap_sendpacket` 函数发送数据包。如果发送失败，输出一条错误信息。

4、ARP包接收函数—— receive_arp_response详解

```
bool receive_arp_response(pcap_t *adhandle, in_addr target_ip, u_char
*target_mac)
{
    struct pcap_pkthdr *header; // 数据包头部
    const u_char *packet; // 数据包内容
    while (pcap_next_ex(adhandle, &header, &packet) >= 0) // 从 pcap 句柄中读取下一
个数据包
    {
        ethernet_header *eth = (ethernet_header *)packet; // 强转为以太网帧头部
```

```
        if (ntohs(eth->type) != ETHERNET_TYPE_ARP)
            continue;

        arp_header *arp = (arp_header *)(packet + sizeof(ethernet_header)); //
强转为ARP帧头部
        if (ntohs(arp->opcode) == 0x0002 && memcmp(arp->sender_ip,
&target_ip.S_un.S_addr, 4) == 0) // 如果是ARP响应包且源IP地址是目标IP地址
        {
            memcpy(target_mac, arp->sender_mac, 6); // 拷贝MAC地址到target_mac
            return true;
        }
    }
    return false;
}
```

函数 `receive_arp_response` 的目的是接收 ARP响应，并获取目标 IP 地址的 MAC 地址。

函数接受三个参数：

- `pcap_t *adhandle`：这是一个 pcap 句柄，用于捕获网络数据包。
- `in_addr target_ip`：这是目标 IP 地址，我们希望获取这个 IP 地址对应的 MAC 地址。
- `u_char *target_mac`：这是一个指针，用于存储获取到的 MAC 地址。

函数的主体是一个循环，使用 `pcap_next_ex` 函数从 pcap 句柄中读取下一个数据包。如果读取成功，它会检查数据包的类型是否为 ARP，如果不是，就跳过这个数据包。如果是 ARP 数据包，它会检查 ARP 操作码是否为 0x0002（表示 ARP 响应）以及发送者 IP 是否为目标 IP。如果都匹配，它会将发送者 MAC 地址复制到 `target_mac` 中，并返回 `true`。如果循环结束还没有找到匹配的 ARP 响应，函数会返回 `false`。

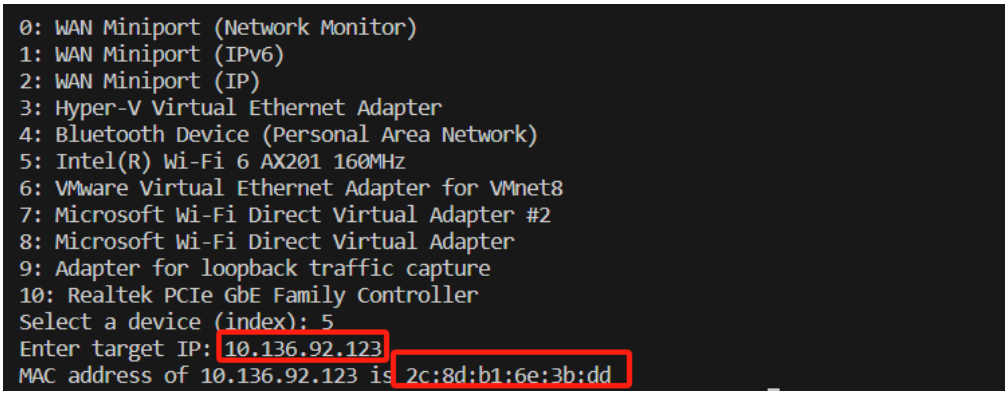
这个结构体通常用于 `GetAdaptersInfo` 函数，该函数获取系统中所有网络适配器的信息，并将信息存储在 `IP_ADAPTER_INFO` 结构体中

> 获取设备列表，设备打开等功能和之前实验1、2部分相差无几，这里不过多阐述

四、结果展示

1、查看本机的IP地址所映射的MAC地址

使用程序查看映射



打开命令行检查无误

```
无线局域网适配器 WLAN:

   连接特定的 DNS 后缀 . . . . . . . :
   描述. . . . . . . . . . . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz
   物理地址. . . . . . . . . . . . . : 2C-8D-B1-6E-3B-DD
   DHCP 已启用 . . . . . . . . . . . : 是
   自动配置已启用. . . . . . . . . . : 是
   IPv6 地址 . . . . . . . . . . . . : 2001:250:401:6576:2834:6007:ba94:b9b2(首选)
   临时 IPv6 地址. . . . . . . . . . : 2001:250:401:6576:807d:e624:4623:570d(首选)
   本地链接 IPv6 地址. . . . . . . . : fe80::ef9e:a169:d3c5:8f70%20(首选)
   IPv4 地址 . . . . . . . . . . . . : 10.136.92.123(首选)
   子网掩码 . . . . . . . . . . . . : 255.255.128.0
   获得租约的时间  . . . . . . . . . : 2023年11月22日 10:08:04
   租约过期的时间  . . . . . . . . . : 2023年11月22日 16:08:04
   默认网关. . . . . . . . . . . . . : fe80::865b:12ff:fe5e:3602%20
                                       10.136.0.1
   DHCP 服务器 . . . . . . . . . . . : 10.136.0.1
   DHCPv6 IAID . . . . . . . . . . . : 170692017
   DHCPv6 客户端 DUID  . . . . . . . : 00-01-00-01-28-5B-BE-75-7C-8A-E1-85-0C-90
   DNS 服务器  . . . . . . . . . . . : 222.30.45.41
                                       202.113.16.41
   TCPIP 上的 NetBIOS  . . . . . . . : 已启用
```

在wireshark中抓包

209	8.454873	IntelCor_6e:3b:dd	Broadcast	ARP	42	ARP Announcement for 10.136.92.123
454	19.185001	IETF-VRRP-VRID_08	IntelCor_6e:3b:dd	ARP	56	Who has 10.136.92.123? Tell 10.136.0.1
455	19.185023	IntelCor_6e:3b:dd	IETF-VRRP-VRID_08	ARP	42	10.136.92.123 is at 2c:8d:b1:6e:3b:dd

可以看到我们的程序可以成功广播ARP数据包，并实现定向应答

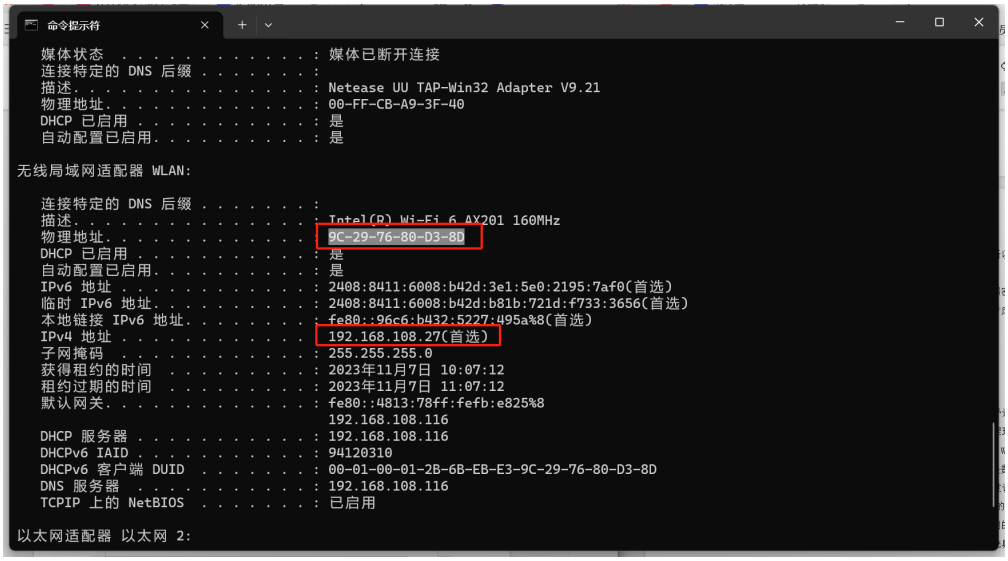
- **209数据包——ARP Announcement**：ARP公告，用于更新或确认网络上其他设备的MAC地址表，通常用于IP地址变更、设备启动或网络接口状态变更时，它是一种**特殊的ARP请求**，它的目标IP地址和发送者的IP地址相同。通过这种方式，设备可以通知网络上的其他设备其MAC地址
- **454数据包——ARP Request**：ARP请求，发送方将包含目标IP地址的ARP请求广播到网络上。网络上的所有设备都会收到这个请求，但只有IP地址与请求中的目标IP地址相匹配的设备会响应
- **455数据包——ARP reply**：ARP应答，对ARP请求的回应，用于告知请求方目标IP地址对应的MAC地址。当一个设备收到一个ARP请求，且发现请求中的目标IP地址与自己的IP地址匹配时，它会发送一个ARP应答给请求方。

解释reply包的来源

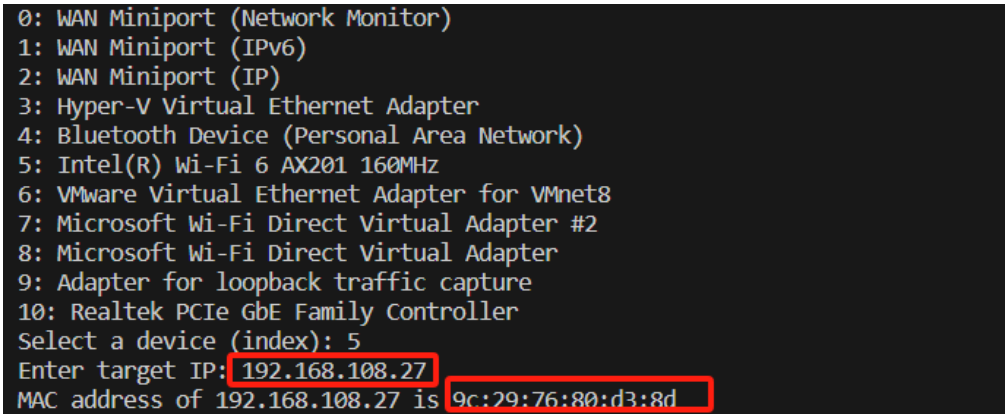
可以发现454数据包的发送方的IP地址为10.136.0.1，这个地址是校园网默认的网关地址和DHCP 服务器地址，网关和DHCP服务器是核心服务设备，负责管理网络流量和IP地址分配。它们知道网络上每个活动设备的MAC地址，所以我们发送的ARP数据包在局域网广播之后，返回ARP reply的设备就是网关和DHCP服务器。

2、查看非本机IP地址到MAC地址的映射

通过两台主机电脑连接在同一个无线局域网下，通过我的主机电脑查看对方主机电脑的IP到MAC地址的映射，下图为对方主机电脑的IP复制和其对应的物理地址



本机通过ARP发包查询，查找IP地址192.168.108.27对应的MAC地址



发现查询无误

五、实验中遇到的问题及其解决

1、通过设备名称匹配网络适配器部分出错

原先的代码是

```
 for (PIP_ADAPTER_INFO pAdapterInfo = AdapterInfo; pAdapterInfo; pAdapterInfo =
pAdapterInfo->Next)
    {
        // std::cout << pAdapterInfo->AdapterName << std::endl;
        if (strcmp(pAdapterInfo->AdapterName, d->name) == 0)
        {
            // 拷贝MAC地址
            for (i = 0; i < pAdapterInfo->AddressLength; i++)
            {
                local_mac[i] = pAdapterInfo->Address[i];
            }
            // 6字节的MAC地址后面跟着2字节的填充，用S_un.S_addr
            local_ip.S_un.S_addr = inet_addr(pAdapterInfo-
>IpAddressList.IpAddress.String); // 将点分十进制IP地址转换为网络字节序整数
            found = true;
            break;
        }
    }
```

但是由于每一个适配器的名字前都会有`\Device\NPF_`的字样，所以不能直接用`pAdapterInfo->AdapterName`与`d->name`做对比，需要跳过前面的提示信息用`pAdapterInfo->AdapterName`与`d->name+12`做比较

附录

github仓库链接：Network_technology(github.com)