IP数据报捕获与分析（利用NPcap编程捕获数据包）

# 实验要求

IP数据报捕获与分析编程实验，要求如下：（1）了解NPcap的架构。（2）学习NPcap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）通过NPcap编程，实现本机的IP数据报捕获，显示捕获数据帧的源MAC地址和目的MAC地址，以及类型/长度字段的值。（4）捕获的数据报不要求硬盘存储，但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地址和类型/长度字段的值。（5）编写的程序应结构清晰，具有较好的可读性。

# 实验内容

## Npcap了解

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；在 Linux 上也有对应的 LibPcap；目前 WinPcap 已经处于无人维护的状态，对于 Windows 10 有更新的且目前有人维护的开源项目 NpCap。

### 架构

WinPcap/NpCap 则为 Win32 应用程序提供了这样的接口：

* 捕获原始数据包；无论它是发往某台机器的，还是在其他设备（共享媒介）上进行交换的
* 在数据包发送给某应用程序前，根据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

SDK 提供的这些功能需要借助运行在 Win32 内核中的网络设备驱动程序来实现；在安装完成驱动之后，SDK 将这些功能作为一个接口表现出来以供使用。

### 设备列表获取（pcap\_findalldevs­\_ex）

NpCap 提供了 pcap\_findalldevs\_ex 和 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。

除此之外，pcap\_if\_t 结构体还包含了一个 pcap\_addr 结构体；后者包含了一个地址列表、一个掩码列表、一个广播地址列表和一个目的地址的列表；此外，pcap\_findalldevs\_ex 还能返回远程适配器信息和一个位于所给的本地文件夹的 pcap 文件列表。

### 网卡设备打开（pcap\_open）

用来打开一个适配器，实际调用的是 pcap\_open\_live；它接受五个参数：

* name：适配器的名称（GUID）
* snaplen：制定要捕获数据包中的哪些部分。在一些操作系统中 (比如 xBSD 和 Win32)，驱动可以被配置成只捕获数据包的初始化部分：这样可以减少应用程序间复制数据的量，从而提高捕获效率；本次实验中，将值定为 65535，比能遇到的最大的 MTU 还要大，因此总能收到完整的数据包。
* flags：主要的意义是其中包含的混杂模式开关；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。
* to\_ms：指定读取数据的超时时间，以毫秒计；在适配器上使用其他 API 进行读取操作的时候，这些函数会在这里设定的时间内响应——即使没有数据包或者捕获失败了；在统计模式下，to\_ms 还可以用来定义统计的时间间隔：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：读操作立刻返回。
* errbuf：用于存储错误信息字符串的缓冲区

该函数返回一个 pcap\_t 类型的 handle。

### 数据包捕获方法（pcap\_loop）

API 函数 pcap\_loop 和 pcap\_dispatch 都用来在打开的适配器中捕获数据包；但是前者会已知捕获直到捕获到的数据包数量达到要求数量，而后者在到达了前面 API 设定的超时时间之后就会返回（尽管这得不到保证）；前者会在一小段时间内阻塞网络的应用，故一般项目都会使用后者作为读取数据包的函数；虽然在本次实验中，使用前者就够了。

这两个函数都有一个回调函数；这个回调函数会在这两个函数捕获到数据包的时候被调用，用来处理捕获到的数据包；这个回调函数需要遵顼特定的格式。但是需要注意的是我们无法发现 CRC 冗余校验码——因为帧到达适配器之后，会经过校验确认的过程；这个过程成功，则适配器会删除 CRC；否则，大多数适配器会删除整个包，因此无法被 NpCap 确认到。

int pcap\_loop(pcap\_t \* p,int cnt, pcap\_handler callback, uchar \* user);

参数说明：

* p是由pcap\_open\_live()返回的所打的网卡的指针；
* cnt用于设置所捕获数据包的个数，如果为-1则无限循环捕获；
* pcap\_handler是void packet\_handler()使用的一个参数，即回调函数的名称；
* user值一般为NULL

## Wireshark使用

安装过程在写实验报告前已经完成了，报告中就忽略了具体内容。

## NPcap编程

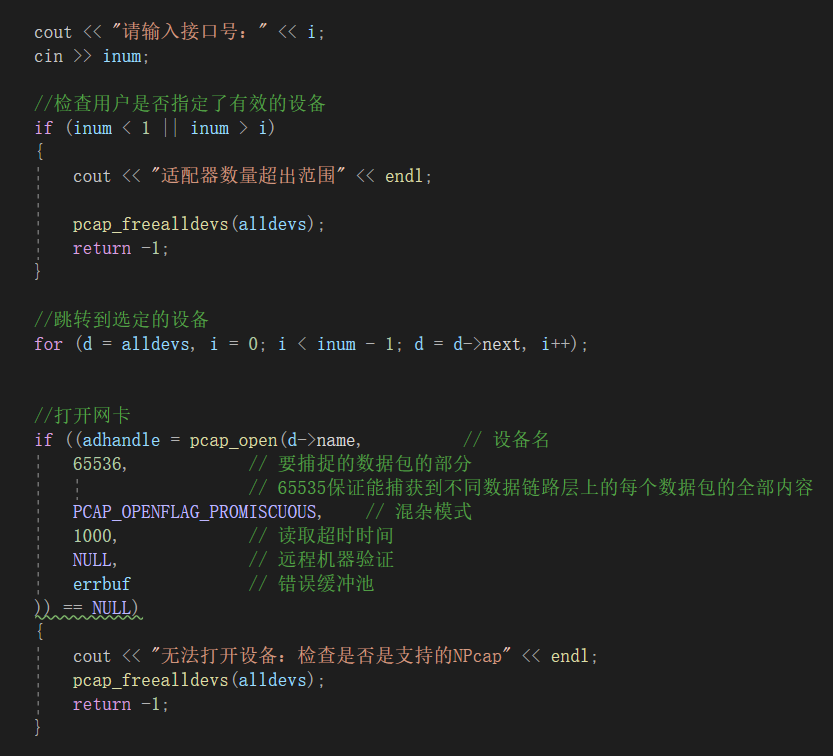
### 获得本机的设备列表



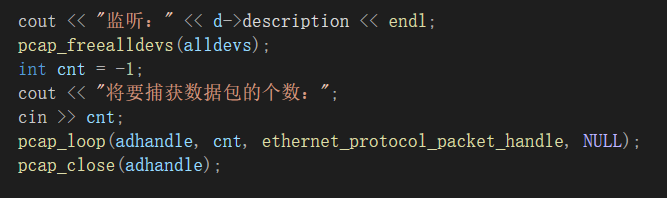
### 打印列表



### 打开设备



### 获取数据包



问题：error C4996: 'strcpy': This function or variable may be unsafe. Consider using strcpy\_s instead.

解决：找到【项目属性】，点击【C++】里的【预处理器】，对【预处理器】进行编辑，在里面加入一段代码：\_CRT\_SECURE\_NO\_WARNINGS。

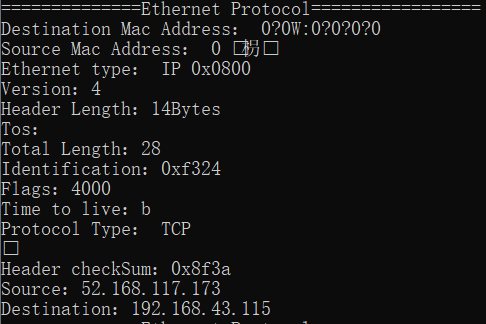
问题：错误 C4996 ‘inet\_ntoa‘: Use inet\_ntop() or InetNtop() instead or define \_WINS

解决：改使用inet\_ntop()函数 inet\_ntop(AF\_INET,&source.sin\_addr,str,16)

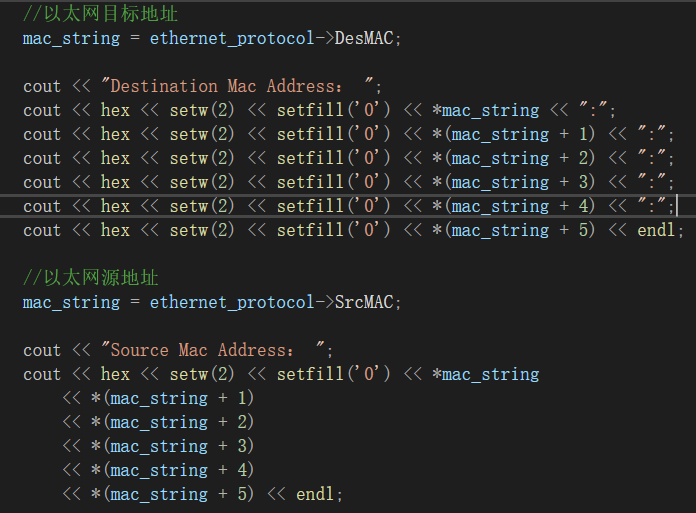
问题：error LNK2019: 无法解析的外部符号

解决：添加#pragma comment(lib,"ws2\_32.lib")

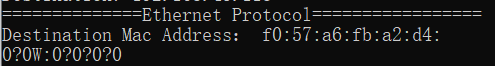
问题：ip地址打印错误



代码如下图：



解决：网上所给的实例使用的都是printf（）函数，没有对应的cout输出示例。

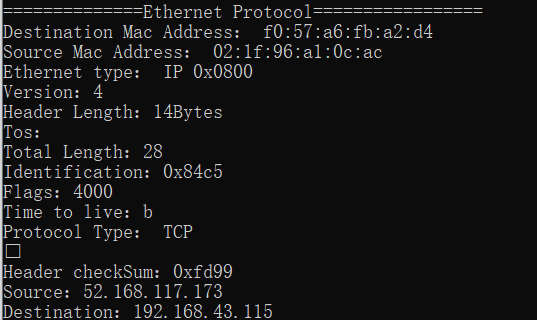


这是printf（）函数和cout输出的结果对比、

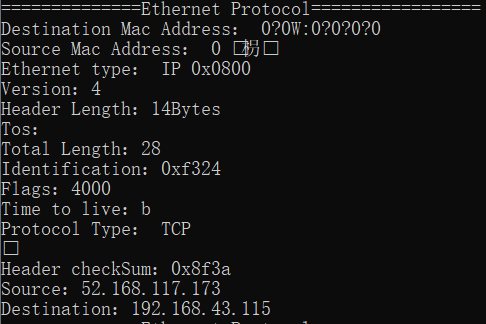
在ASCII中，一共定义了128个字符，其中33个无法显示，为0～31和127，32到126为可显示字符，当使用cout输出一个char型字符时，如果是可显示范围内，则输出相应可显示字符，其余的都输出乱码。

如果我们要使字符按十六进制输出，可以使用hex，但是发现cout << hex << data[0];没有输出十六进制，因为hex只对整数起作用，将data[0]转换为int，cout << hex << int(data[0])； 发现输出的结果前面带了很多f。因为data[0]是有符号数，最高位为1时，转换为int时，其余位都为1，cout << hex << (unsigned int) (unsigned char)data[0];结果正确。

此时输出结果格式正确。



问题：数据等结果不正确



解决：在c++中，首先按照几进制输出得在前面写出来一个 cout << hex <<(十六进制)

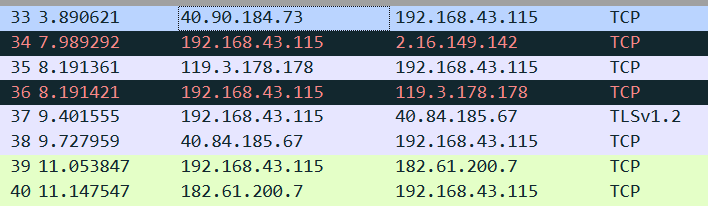
然后因为hex 只对整形数据起作用，所以还得强制类型转换

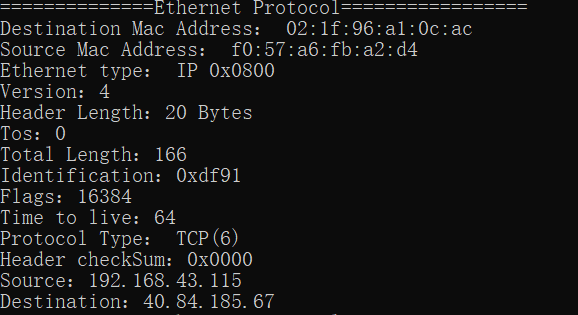
修改后代码如下：



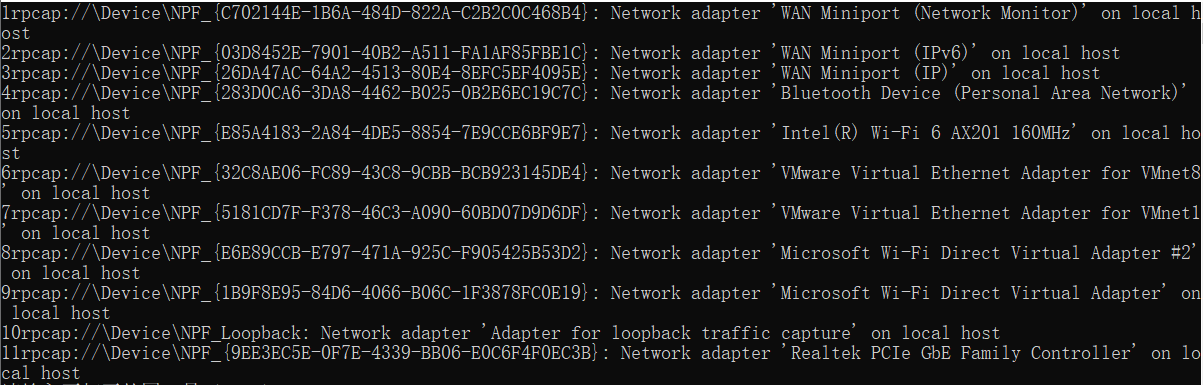
此时输出结果正确

问题：某一条输出输出的协议类型和wireshark中的不一样





解决：TLSv1.2是基于TCP协议，而且查询对应的ip协议表，并没有TSLv1.2这一项



1-3（WAN Miniport ）：wan微型端口协议，安装了VPN或者PPPOE就会出现，使用V-P-N等拨号连接产生的虚拟网卡

4（Bluetooth 个人区域网 (PAN)）：Bluetooth 个人区域网 (PAN) 是一种可利用便携式计算机、移动电话和手持设备之间的无线链接创建以太网网络的技术。可以连接到以下类型的启用 Bluetooth 的设备（这些设备都使用 PAN）：个人区域网用户 (PANU) 设备、组式临时网络 (GN) 设备或网络访问点 (NAP) 设备。

下面是有关这些设备中每种设备的功能的详细信息：

* PANU 设备。 连接到启用 Bluetooth 的 PANU 设备可创建一个包含您的计算机和设备的临时网络。
* GN 设备。 连接到启用 Bluetooth 的 GN 设备可创建一个包含您的计算机、GN 设备和其他任何与同一 GN 设备连接的 PANU 设备的临时网络。
* NAP 设备。 连接到启用 Bluetooth 的 NAP 设备可让您将计算机连接到更大的网络，如家庭网络、企业网络或 Internet。：

8（Microsoft virtual wifi adapter）：用来无线显示时使用的，当你的电脑和其他设备在同一个网络环境（例连接同一WiFi）时其他设备可以将其显示内容和声音投放到你的电脑上，而不使用你原来的WiFi连接（用不同的网络适配器接受数据）

经实践，1-11中，5能快速捕获到大量的数据包；10能捕获到数据包，但速度较慢；1-4基本无法捕获到数据包；6-7能够捕获到数据包，但速度较慢