

廈門大學



软件学院

《计算机网络》实验报告

题 目 用 WinPCAP 监听并分析以太网的帧

姓 名 黄勳

学 号 22920212204392

班 级 21 级计算机网络 2 班

实验时间 2023. 4. 7

2023 年 4 月 7 日

1 实验目的

- 通过捕获并分析以太网帧，分析常见数据包的帧格式，熟悉以太网中常用协议及其报文格式，如 ARP、ICMP、IP 协议。
- 学会对捕获到的数据帧按指定的条件进行过滤，为网络流量深入分析做基础。
所谓的指定条件可包含：指定的目的 IP 地址、指定的源 IP 地址、指定的协议类型等（参考 Wireshark 的过滤条件），比如当指定协议类型为 IP 时，其它类型的数据帧将被丢弃，仅留下 IP 数据帧。

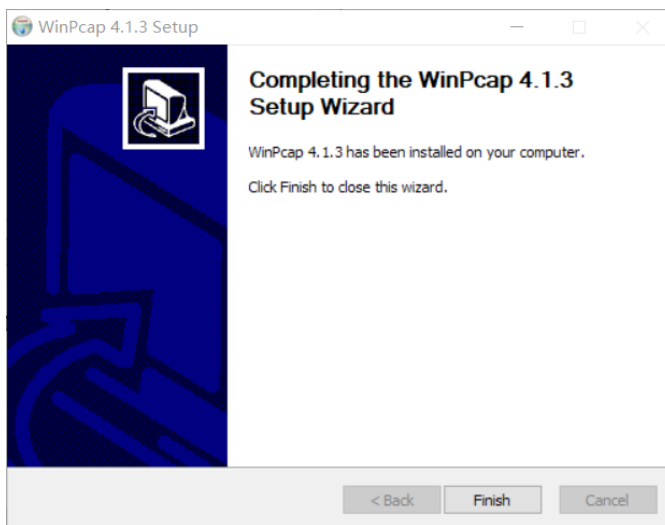
2 实验环境

操作系统：Windows10 21H2

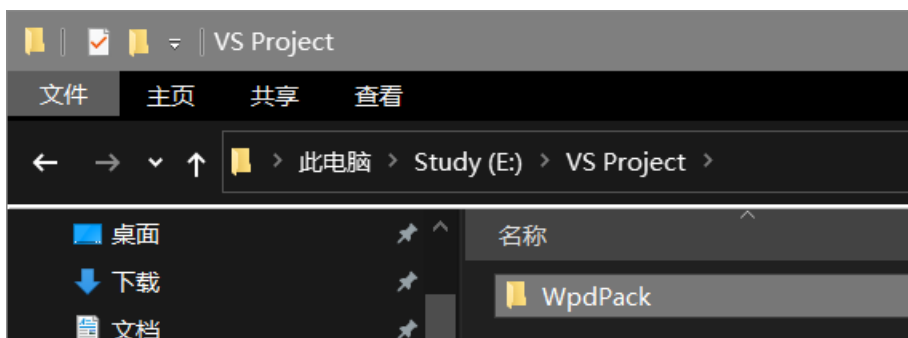
编程语言：C++ with Visual Studio 2022 Community

3 实验结果

[1] 事前准备 1：安装 WinPCAP

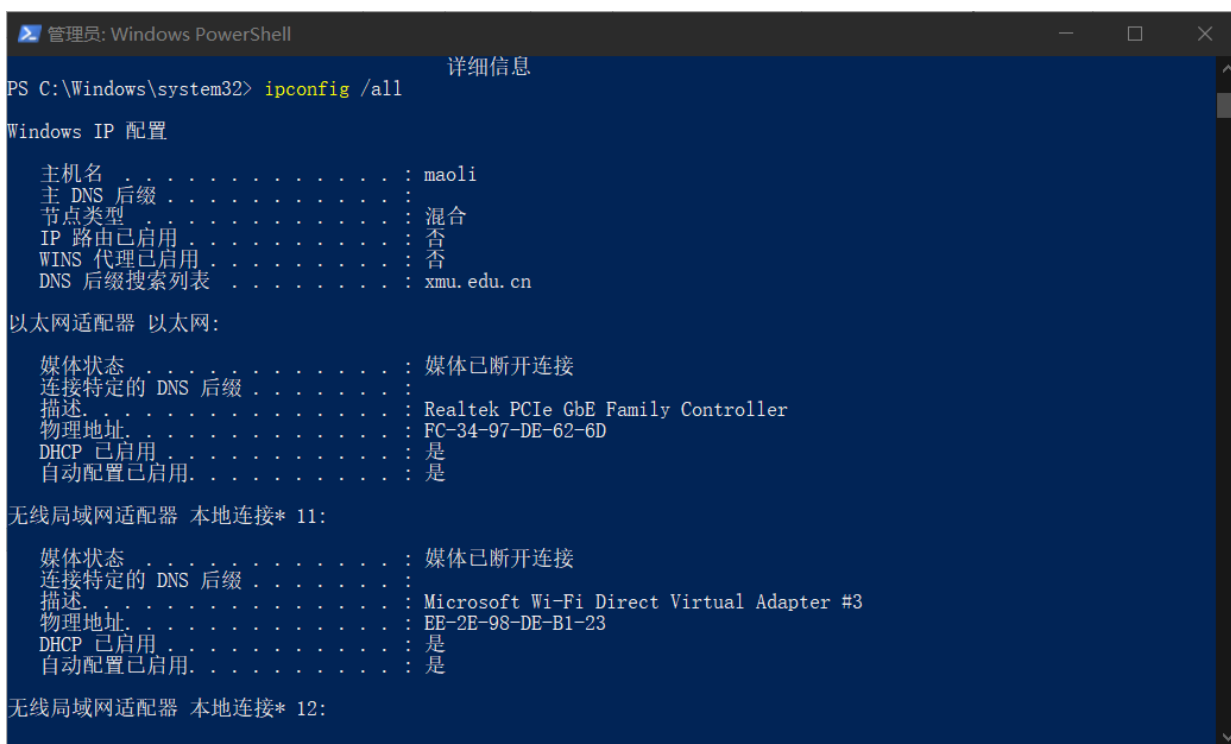


[2] 事前准备 2：解压缩 WpdPack.zip 将 Include、Lib 文件夹拷出备用



[3] 实验项-2: 使用 Windows 自带的“命令提示符”或“PowerShell”完成本机 IP、MAC 地址等信息的查询工作 (ipconfig 命令的使用)

a) 输入 ipconfig/all 查看本机 IP、MAC 地址 (物理地址) 等信息



```
管理员: Windows PowerShell

无线局域网适配器 WLAN:

连接特定的 DNS 后缀 . . . . . : xmu.edu.cn
描述 . . . . . : MediaTek Wi-Fi 6 MT7921 Wireless LAN Card
物理地址. . . . . : 28-EA-1F-A5-7F-92
DHCP 已启用 . . . . . : 是
自动配置已启用. . . . . : 是
IPv6 地址 . . . . . : 2409:8734:1a70:7e4:50d7:15ba:158a:a15b(首选)
临时 IPv6 地址. . . . . : 2409:8734:1a70:7e4:b436:f465:65f3:90bd(首选)
本地链接 IPv6 地址. . . . . : fe80::997a:4801:15a4:136a%20(首选)
IPv4 地址 . . . . . : 10.30.86.185(首选)
子网掩码 . . . . . : 255.255.224.0
获得租约的时间 . . . . . : 2023年4月2日 20:08:54
租约过期的时间 . . . . . : 2023年4月2日 21:59:25
默认网关. . . . . : fe80::42fe:95ff:fe80:8001%20
                  10.30.64.1
DHCP 服务器 . . . . . : 172.18.0.12
DHCPv6 IAID . . . . . : 686567064
DHCPv6 客户端 DUID . . . . . : 00-01-00-01-28-52-9C-23-FC-34-97-DE-62-6D
DNS 服务器 . . . . . : 211.138.156.66
                  210.34.0.14
TCP/IP 上的 NetBIOS . . . . . : 已启用

以太网适配器 以太网 2:

媒体状态 . . . . . : 媒体已断开连接
连接特定的 DNS 后缀 . . . . . :
描述 . . . . . : Sangfor SSL VPN CS Support System VNIC
物理地址. . . . . : 00-FF-3D-6E-3E-F1
DHCP 已启用 . . . . . : 否
自动配置已启用. . . . . : 是
```

[4] 实验项-1: 使用 Windows 自带的“命令提示符”或“PowerShell”完成“本机与具有某个 IP 的主机是否连通的检测 (ping 命令的使用)

a) ping 本机测试是否连通

```
管理员: Windows PowerShell

Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> ping 10.30.86.185

正在 Ping 10.30.86.185 具有 32 字节的数据:
来自 10.30.86.185 的回复: 字节=32 时间<1ms TTL=64
来自 10.30.86.185 的回复: 字节=32 时间<1ms TTL=64
来自 10.30.86.185 的回复: 字节=32 时间<1ms TTL=64
来自 10.30.86.185 的回复: 字节=32 时间<1ms TTL=64

10.30.86.185 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 0ms, 平均 = 0ms
```

b) ping 另一台主机测试是否连通

这里以百度为例:

```
管理员: Windows PowerShell

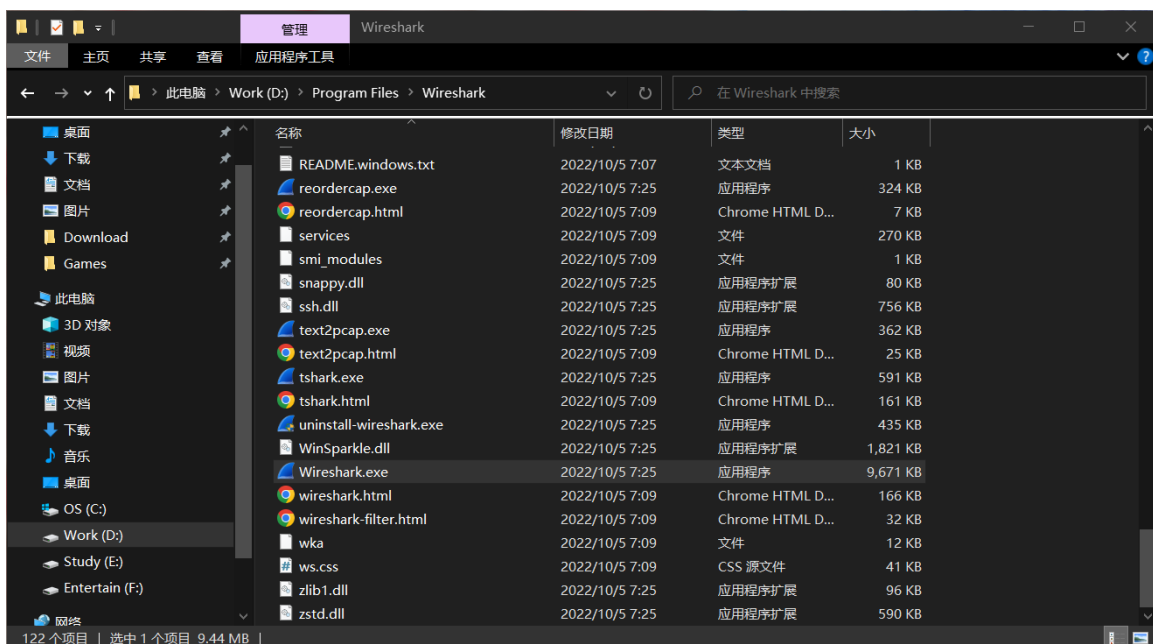
PS C:\Windows\system32> ping baidu.com

正在 Ping baidu.com [39.156.66.10] 具有 32 字节的数据:
来自 39.156.66.10 的回复: 字节=32 时间=50ms TTL=50
来自 39.156.66.10 的回复: 字节=32 时间=50ms TTL=50
来自 39.156.66.10 的回复: 字节=32 时间=51ms TTL=50
来自 39.156.66.10 的回复: 字节=32 时间=50ms TTL=50

39.156.66.10 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 50ms, 最长 = 51ms, 平均 = 50ms
PS C:\Windows\system32>
```

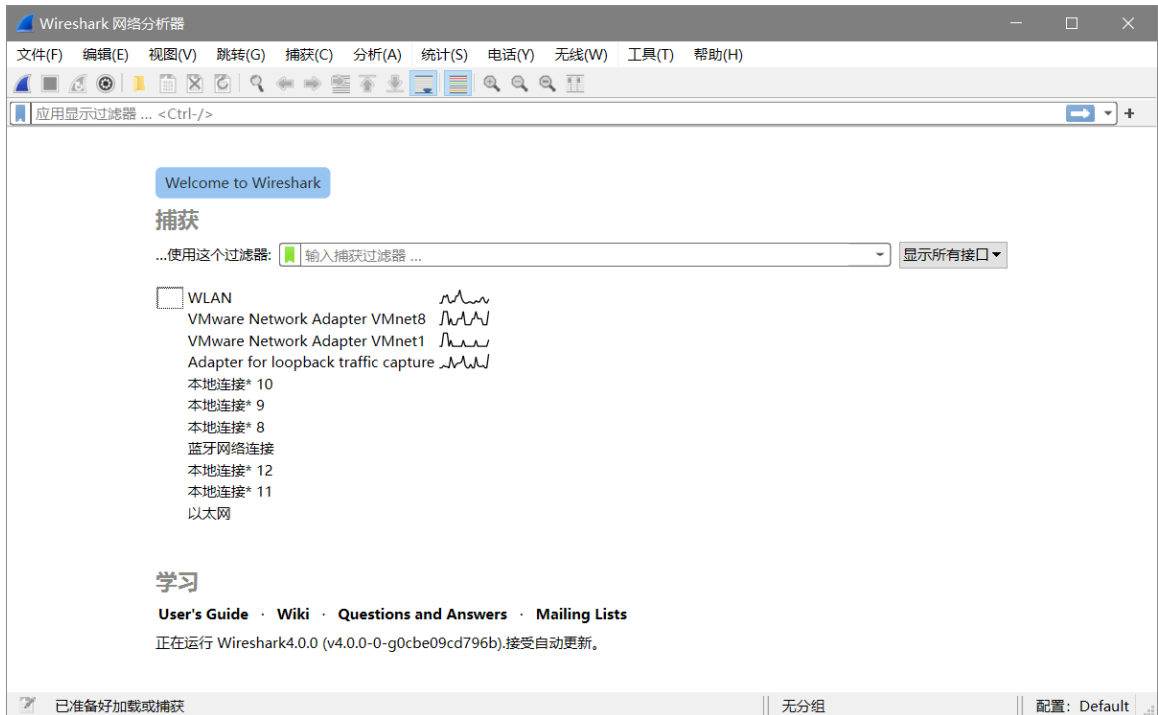
[5] 实验项 0: 熟悉 Wireshark 的使用, 会设置过滤条件, 如过滤出指定 IP 的数据帧 (熟悉抓包工具的使用)

a) 安装抓包工具 Wireshark



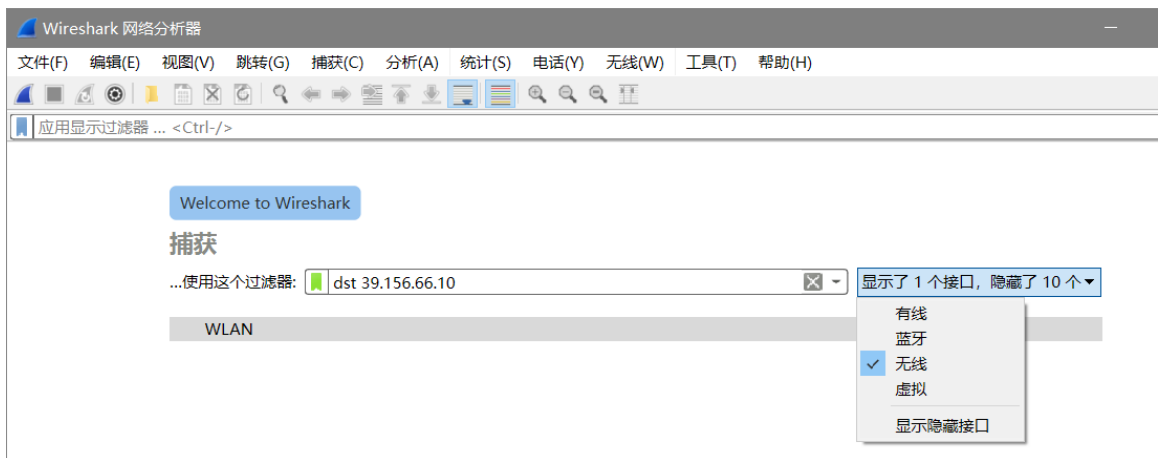
b) 熟悉 Wireshark 使用

打开软件, 选择 WLAN 网络:



添加过滤条件：

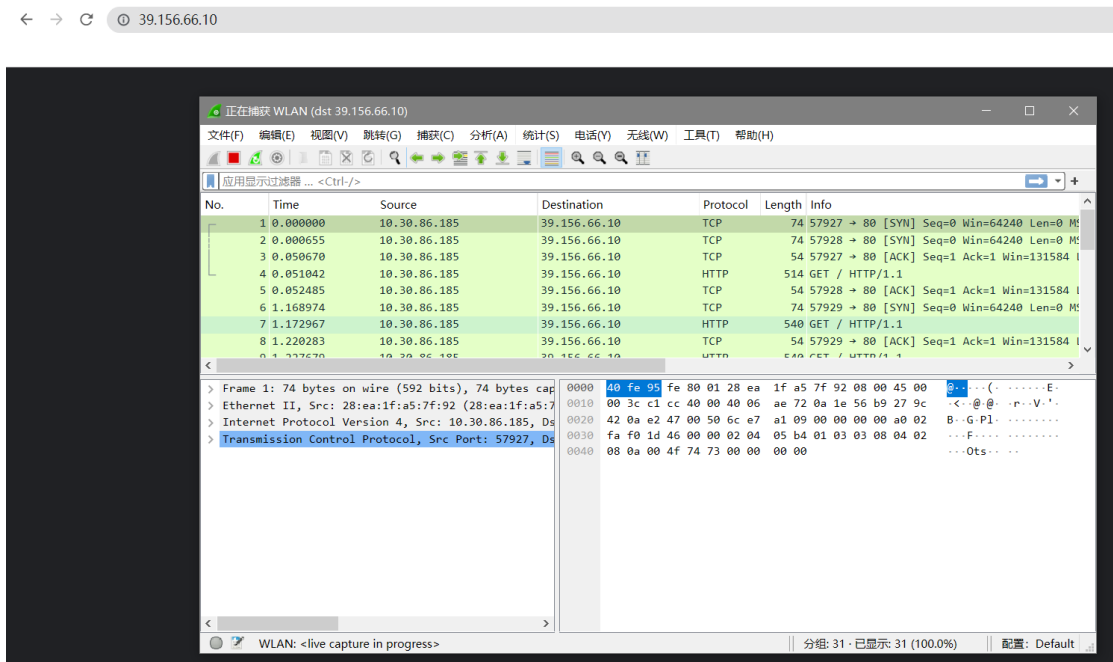
选择一个网络，选择管理捕获过滤器，新建过滤器进行添加



使用过滤器：src+ip：只过滤出 ip 地址作为源 ip 地址的数据包；

dst+ip：只过滤出 ip 地址作为目的 ip 地址的数据包；

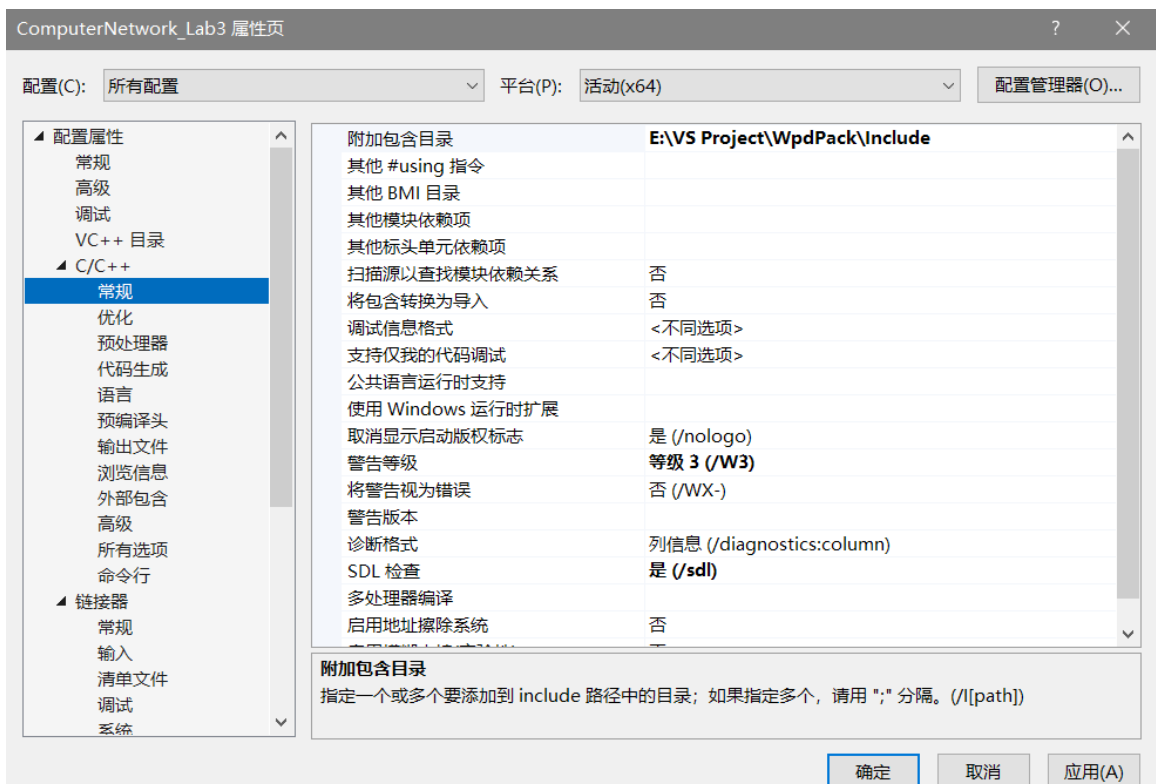
host+ip：只过滤出 ip 地址作为源 ip 地址或目的 ip 地址的数据包



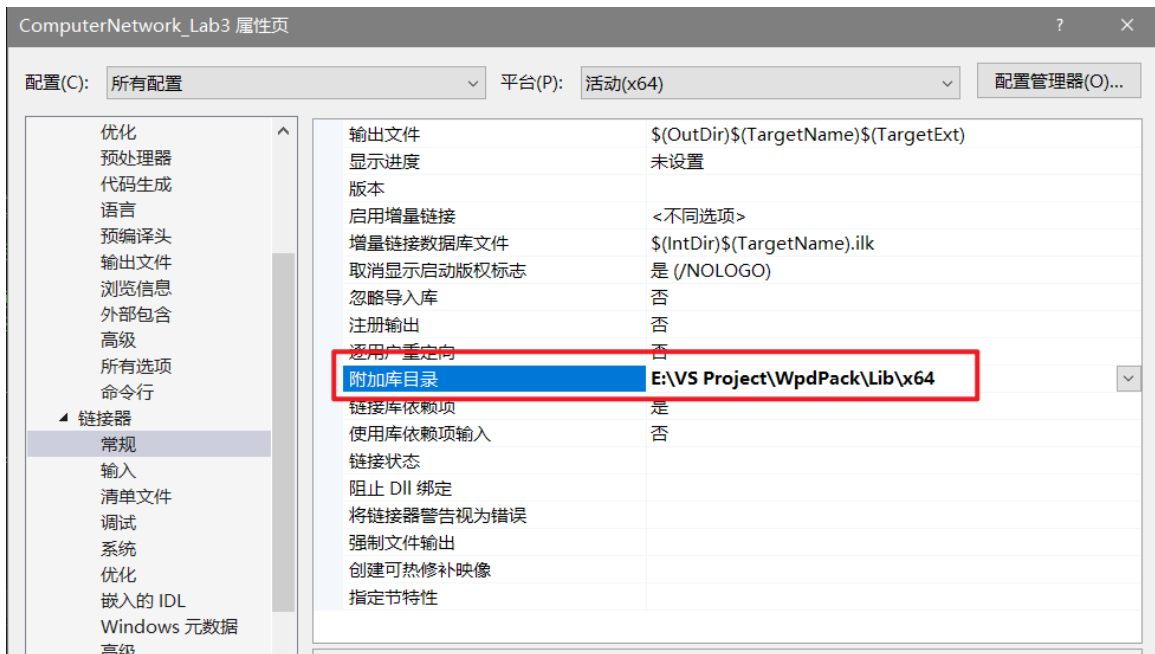
[6] 实验项 1：配置好实验环境，在控制台打印出网卡设备列表

在 VS 中创建项目，配置项目的相关属性

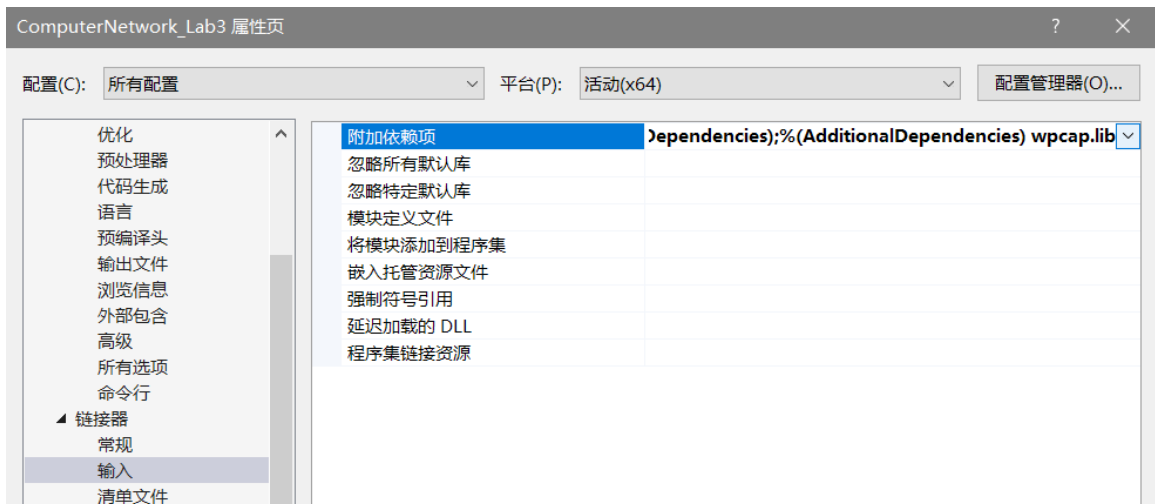
1) C/C++ - 常规 - 附加包含目录 - 添加前面准备的 Include 文件夹路径



2) 连接器 - 常规 - 附加库目录 - 准备的 Lib 文件夹路径



3) 连接器 - 附加依赖项 - 添加上空格+wpcap.lib



4) 在 VS 中编写 C 语言程序:


```
Lab3_getDevice.cpp  Lab3_catchMac.cpp  pcap.h
[+] 杂项文件  (全局范围)  main()
2  #define WIN32 //防止引用unix库
3  #include "pcap.h"
4
5  int main()
6  {
7      pcap_if_t* alldevs; //设备列表
8      pcap_if_t* d;
9
10     int i = 0;
11     char errbuf[PCAP_ERRBUF_SIZE]; //错误信息
12
13     /* 获取本地机器设备列表 */
14     if (pcap_findalldevs_ex((char*)PCAP_SRC_IF_STRING, NULL, &alldevs, errbuf) == -1) //强制类型转化, 将const char*类型转化为char*类型
15     {
16         fprintf(stderr, "Error in pcap_findalldevs_ex: %s\n", errbuf);
17         return 1;
18     }
19
20     /* 打印列表 */
21     for (d = alldevs; d != NULL; d = d->next)
22     {
23         printf("%d. %s", ++i, d->name);
24         if (d->description)
25             printf(" (%s)\n", d->description);
26         else
27             printf(" (No description available)\n");
28     }
29
30     if (i == 0)
31     {
32         printf("\nNo interfaces found! Make sure WinPcap is installed.\n");
33         return 0;
34     }
35
36     // 释放设备列表
37     pcap_freealldevs(alldevs);
38 }
```

5) 运行测试，编写成功

```
Microsoft Visual Studio 调试控制台
1. rpcap://\Device\NPF_{2F85E1B4-E950-4176-A2FF-A405A7EE5E52} (Network adapter 'VMware Virtual Ethernet Adapter' on local host)
2. rpcap://\Device\NPF_{3D6E3EF1-6FBA-4668-B77B-F63209DE8D86} (Network adapter 'Sangfor SSL VPN CS Support System VNIC' on local host)
3. rpcap://\Device\NPF_{13AE3657-617C-4C5A-9BDC-0572326D0D41} (Network adapter 'Microsoft' on local host)
4. rpcap://\Device\NPF_{D14B4F0D-8AFD-4AE4-A64F-92D5746E02C6} (Network adapter 'Realtek PCIe GbE Family Controller' on local host)
5. rpcap://\Device\NPF_{B817FC03-A1DB-4A83-A734-053F6350DA24} (Network adapter 'VMware Virtual Ethernet Adapter' on local host)
6. rpcap://\Device\NPF_{E4887CA5-3292-41F2-A836-D781314417A9} (Network adapter 'Microsoft' on local host)
7. rpcap://\Device\NPF_{C606816D-49CB-4901-B61F-4241A21E347C} (Network adapter 'Microsoft' on local host)

E:\VS Project\ComputerNetwork_Lab3\x64\Debug\ComputerNetwork_Lab3.exe (进程 25992) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .
```

[7] 实验项 2：捕获到以太网帧，并能够解析出目的 MAC、源 MAC

编写关键：

数据结构

```

12  typedef struct ip_header { //IP帧格式
13      u_char ver_ihl; // Version (4 bits) + Internet header length(4 bits)
14      u_char tos; // Type of service
15      u_short tlen; // Total length
16      u_short identification; // Identification
17      u_short flags_fo; // Flags (3 bits) + Fragment offset(13 bits)
18      u_char ttl; // Time to live
19      u_char proto; // Protocol
20      u_short crc; // Header checksum
21      u_char saddr[4]; // Source address
22      u_char daddr[4]; // Destination address
23      u_int op_pad; // Option + Padding
24  } ip_header;

```

抓包函数

```

123  pcap_loop(adhandle, 0, packet_handler, NULL); //捕获数据包, 捕获的数据包数, 回调函数, 回调函数的参数
124  }
125
126
127  void packet_handler(u_char* param, const struct pcap_pkthdr* header, const u_char* pkt_data) //通过libpcap
128  {
129      struct tm* ltime;
130      mac_header* mh;
131      char timestr[16];
132      ip_header* ih;
133      time_t local_tv_sec;
134
135      local_tv_sec = header->ts.tv_sec; //帧到达的时间
136      ltime = localtime(&local_tv_sec);
137      strftime(timestr, sizeof timestr, "%H:%M:%S", ltime);

```

确定包类型

```

149      //捕获后的处理
150      mh = (mac_header*)pkt_data;
151      ih = (ip_header*)(pkt_data + sizeof(mac_header)); //length of
152
153      //根据type储存第二位和第四位输出是什么类型的包
154      if (mh->type[1] == 0 && mh->type[0] == 8) //ip为: 0800 0001
155      {
156          printf("get an ip packet\n");
157          if (ih->proto == 0x0001) { //进一步确定是icmp
158              printf("get an icmp packet\n");
159          }
160      }
161      else if (mh->type[1] == 6 && mh->type[0] == 8) //arp为0806
162          printf("get an arp packet\n");
163

```

输出信息

```

163
164     printf("mac_header:\n");
165     printf("\tdest_addr: ");
166     for (int i = 0; i < 6; i++) { //输出mac地址
167         if (i != 5)
168             printf("%02x:", mh->dest_addr[i]);
169         else
170             printf("%02x\n", mh->dest_addr[i]);
171     }
172     printf("\tsrc_addr: ");
173     for (int i = 0; i < 6; i++) {
174         if (i != 5)
175             printf("%02x:", mh->src_addr[i]);
176         else
177             printf("%02x\n", mh->src_addr[i]);
178     }
179     printf("\ttype: %04X", ntohs((u_short)mh->type));
180     printf("\n");
181
182     printf("ip_header\n");
183     printf("\t%-10s: %02X\n", "ver_ihl", ih->ver_ihl);
184     printf("\t%-10s: %02X\n", "tos", ih->tos);
185     printf("\t%-10s: %04X\n", "tlen", ntohs(ih->tlen)); //网络端序转为主机端序ntohs
186     printf("\t%-10s: %04X\n", "identification", ntohs(ih->identification));
187     printf("\t%-10s: %04X\n", "flags_fo", ntohs(ih->flags_fo));
188     printf("\t%-10s: %02X\n", "ttl", ih->ttl);
189     printf("\t%-10s: %02X\n", "proto", ih->proto);
190     printf("\t%-10s: %04X\n", "crc", ntohs(ih->crc));
191     printf("\t%-10s: %08X\n", "op_pad", ntohs(ih->op_pad));
192     printf("\t%-10s: ", "saddr:"); //输出ip地址
193     for (int i = 0; i < 4; i++) {
194         if (i != 3)

```

Main 函数，仿照之前编写的网卡获取+处理过滤器即可

```

94     //预处理
95     if (pcap_datalink(adhandle) != DLT_EN10MB) { //检查链路层。只简单支持以太网。
96         fprintf(stderr, "\nThis program works only on Ethernet networks.\n");
97         pcap_freealldevs(alldevs);
98         return -1;
99     }
100     if (d->addresses != NULL)
101         netmask = ((struct sockaddr_in*)(d->addresses->netmask))->sin_addr.S_un.S_addr; //检索接口的第一个地址的掩码
102     else
103         netmask = 0xffffffff; //如果接口没有地址，假设在一个C类网
104
105     //编译和设置过滤器
106     //编译过滤器
107     if (pcap_compile(adhandle, &fcode, packet_filter, 1, netmask) < 0) {
108         fprintf(stderr, "\nUnable to compile the packet filter. Check the syntax.\n");
109         pcap_freealldevs(alldevs);
110         return -1;
111     }
112     //设置过滤器
113     if (pcap_setfilter(adhandle, &fcode) < 0) {
114         fprintf(stderr, "\nError setting the filter.\n");
115         pcap_freealldevs(alldevs);
116         return -1;
117     }

```

编译运行：可以查看到数据报中的 mac 和 ip 地址

```
E:\VS Project\ComputerNetwork_Lab3\x64\Debug\ComputerNetwork_Lab3.exe

tos      : 00
tlen     : 00A5
identification: E1B4
flags_fo : 0000
ttl      : 04
proto    : 11
crc      : 0425
op_pad   : 0000FA0C
saddr    : 192.168.31.204
daddr    : 239.255.255.250

get an ip packet
mac_header:
  dest_addr: ec:2e:98:de:b1:23
  src_addr: 5c:02:14:d1:c2:3e
  type: A829
ip_header
  ver_ihl : 45
  tos     : 00
  tlen    : 01C4
  identification: 0000
  flags_fo : 4000
  ttl     : 40
  proto   : 11
  crc     : 790B
  op_pad  : 0000076C
  saddr   : 192.168.31.1
  daddr   : 192.168.31.204
```

[8] 实验项 3：能够过滤出特定类型的数据包，指定类型的为 ARP，ICMP 等

修改对应过滤器即可，main 函数中定义的 char packet_filter[] 变量即为过滤帧的类型，所以将它的值修改为 arp、icmp 即可过滤出 ARP，ICMP 类型的数据包

Arp:

```
E:\VS Project\ComputerNetwork_Lab3\x64\Debug\ComputerNetwork_Lab3.exe

tos      : 01
tlen     : 0800
identification: 0604
flags_fo : 0001
ttl      : 5C
proto    : 02
crc      : 14D1
op_pad   : 00000000
saddr    : 194.62.192.168
daddr    : 31.1.0.0

get an arp packet
mac_header:
  dest_addr: 5c:02:14:d1:c2:3e
  src_addr: ec:2e:98:de:b1:23
  type: 2029
ip_header
  ver_ihl : 00
  tos     : 01
  tlen    : 0800
  identification: 0604
  flags_fo : 0002
  ttl     : EC
  proto   : 2E
  crc     : 98DE
  op_pad  : 000014D1
  saddr   : 177.35.192.168
  daddr   : 31.204.92.2
```

Icmp:

```
E:\VS Project\ComputerNetwork_Lab3\x64\Debug\ComputerNetwork_Lab3.exe

tlen      : 0020
identification: 3495
flags_fo  : 0000
ttl       : 05
proto     : 01
crc       : DA70
op_pad    : 00000800
saddr:    : 192.168.31.204
daddr:    : 104.21.94.78

get an ip packet
get an icmp packet
mac_header:
  dest_addr: ec:2e:98:de:b1:23
  src_addr: 5c:02:14:d1:c2:3e
  type: 144B
ip_header
  ver_ihl  : 45
  tos      : 88
  tlen     : 0038
  identification: 0361
  flags_fo : 0000
  ttl      : FB
  proto    : 01
  crc      : 6112
  op_pad   : 00000B00
  saddr:   : 111.24.11.61
  daddr:   : 192.168.31.204
```

[9] 实验项 4（附加）：能够将捕获到的帧保存到 CSV 文件中，包含：时间、源 MAC、源 IP、目标 MAC、目标 IP、帧长度（以逗号间隔）

编写代码，在 `void packet_handler` 函数中添加写入 csv 文件语句：
利用逗号分隔符，输出文件流

```
208 // 写文件
209 FILE* fp = fopen("data.csv", "a");//生成csv文件
210 fprintf(fp, "get a packet\n");
211 fprintf(fp, "time,%s\n", timestr);
212 fprintf(fp, "src_mac,");
213 for (int i = 0; i < 6; i++) { //写入mac源地址
214     if (i != 5)
215         fprintf(fp, "%02x:", mh->src_addr[i]);
216     else
217         fprintf(fp, "%02x\n", mh->src_addr[i]);
218 }
219 fprintf(fp, "dest_mac,");
220 for (int i = 0; i < 6; i++) { //写入mac目的地址
221     if (i != 5)
222         fprintf(fp, "%02x:", mh->dest_addr[i]);
223     else
224         fprintf(fp, "%02x\n", mh->dest_addr[i]);
225 }
226 fprintf(fp, "src_ip,");
227 for (int i = 0; i < 4; i++) { //写入ip源地址
228     if (i != 3)
229         fprintf(fp, "%02x:", ih->saddr[i]);
230     else
231         fprintf(fp, "%02x\n", ih->saddr[i]);
232 }
233 fprintf(fp, "dest_ip,");
234 for (int i = 0; i < 4; i++) { //写入ip目的地址
235     if (i != 3)
236         fprintf(fp, "%02x:", ih->daddr[i]);
237     else
238         fprintf(fp, "%02x\n", ih->daddr[i]);
239 }
240 fprintf(fp, "len,%d\n\n", header->len); //输出帧长度
```

« Study (E) » VS Project » ComputerNetwork_Lab3

在 ComputerNetwork_Lab3 中搜索

名称	修改日期	类型	大小
x64	2023/4/3 1:58	文件夹	
ComputerNetwork_Lab3.sln	2023/4/3 3:13	Visual Studio Sol...	2 KB
ComputerNetwork_Lab3.vcxproj	2023/4/3 3:13	VC++ Project	8 KB
ComputerNetwork_Lab3.vcxproj.filters	2023/4/3 3:13	VC++ Project Fil...	1 KB
ComputerNetwork_Lab3.vcxproj.user	2023/4/3 1:57	USER 文件	1 KB
data.csv	2023/4/6 10:55	Microsoft Excel ...	442 KB
Lab3_catchMac.cpp	2023/4/6 10:53	C++ Source File	8 KB
Lab3_getDevice.cpp	2023/4/6 10:44	C++ Source File	2 KB

data.csv - Excel

文件 开始 插入 页面布局 公式 数据 审阅 视图 帮助

A1 time:

	A	B	C	D	E
1	time:	3:17:49			
2	src_mac:	ec:2e:98:de:b1:23			
3	dest_mac:	5c:02:14:d1:c2:3e			
4	src_ip:	c0:a8:1f:cc			
5	dest_ip:	78:e9:14:c9			
6	len:	121			
7					
8					
9	time:	3:17:49			
10	src_mac:	5c:02:14:d1:c2:3e			
11	dest_mac:	ec:2e:98:de:b1:23			
12	src_ip:	78:e9:14:c9			
13	dest_ip:	c0:a8:1f:cc			
14	len:	201			
15					
16					
17	time:	3:17:50			
18	src_mac:	ec:2e:98:de:b1:23			
19	dest_mac:	5c:02:14:d1:c2:3e			
20	src_ip:	c0:a8:1f:cc			
21	dest_ip:	78:e9:14:c9			

data

就绪 辅助功能: 不可用

4 实验总结

1. WinPcap 网络编程入门——0. 环境配置及系列介绍

https://blog.csdn.net/weixin_46117139/article/details/122635095

2. VS2022 安装教程

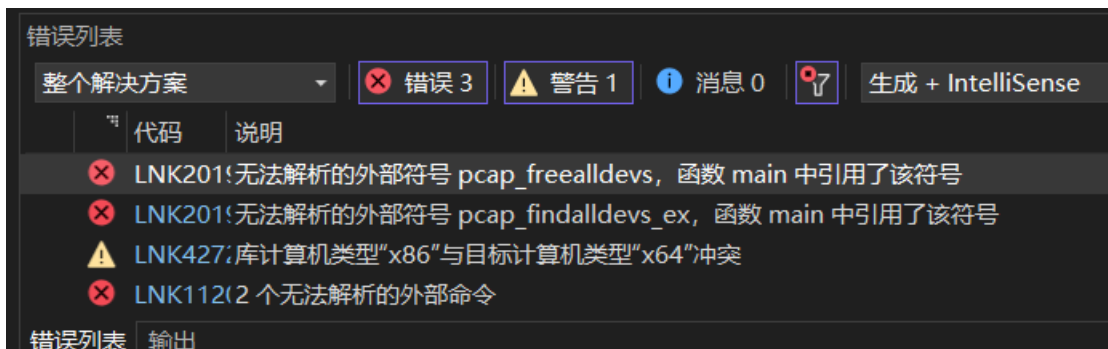
https://blog.csdn.net/qq_37444533/article/details/121401284

3. 添加目录出错



检查属性中添加的目录

4. 如下报错



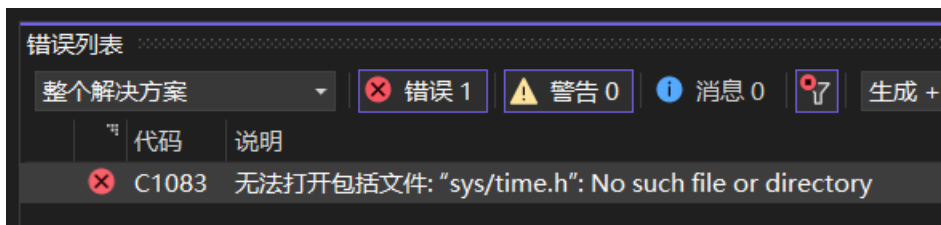
https://blog.csdn.net/weixin_44217239/article/details/127191823

5. 如下报错

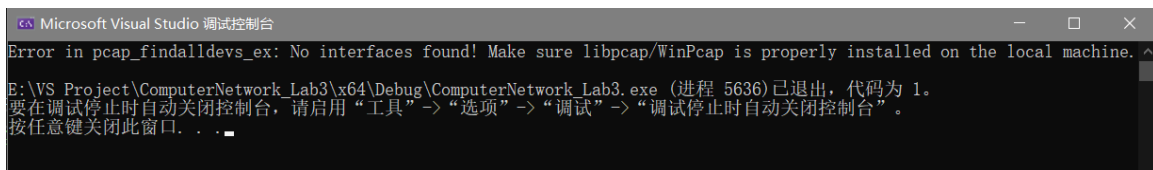
https://blog.csdn.net/C_to_OOP/article/details/77479282

6. 如下报错

https://blog.csdn.net/qq_37935909/article/details/109734328



7.如下报错



winPcap4.1.3 最后安装完成的时候有个开机启动 bootdriver 的选项这个记得要勾上, 然后就好使了。