厦門大學

信息学院软件工程系

《计算机网络》实验报告

题　　目 实验三　用 PCAP 库侦听并分析网络流量

班　　级 软件工程 2020 级卓越班

姓　　名 庾晓萍

学　　号 20420192201952

实验时间 2022 年 3 月 23 日

2022 年 3 月 23 日

填写说明

1、本文件为 Word 模板文件，建议使用 Microsoft Word 2019 打开，在可填写的区域中如实填写；

2、填表时勿破坏排版，勿修改字体字号，打印成 PDF 文件提交；

3、文件总大小尽量控制在 1MB 以下，最大勿超过 5MB；

4、应将材料清单上传在代码托管平台上；

5、在实验课结束 14 天内，按原文件发送至课程 FTP 指定位置。

1 实验目的

通过完成实验，理解数据链路层、网络层、传输层和应用层的基本原理。掌握用 Wireshark 观察网络流量并辅助网络侦听相关的编程；掌握用 Libpcap 或 WinPcap 库侦听并处理以太网帧和 IP 报文的方法；熟悉以太网帧、IP 报文、TCP 段和 FTP 命令的格式概念，掌握 TCP 协议的基本机制；熟悉帧头部或 IP 报文头部各字段的含义。熟悉 TCP 段和 FTP 数据协议的概念，熟悉段头部各字段和 FTP 控制命令的指令和数据的含义。

2 实验环境

操作系统：Windows 10；

IDE：Visual Studio 2019

编程语言：C++；

3 实验结果

1、用侦听解析软件 Wireshark 观察数据格式。

① 进入命令窗口之后，输入：ipconfig/all 回车，查看电脑 IP 详细地址。

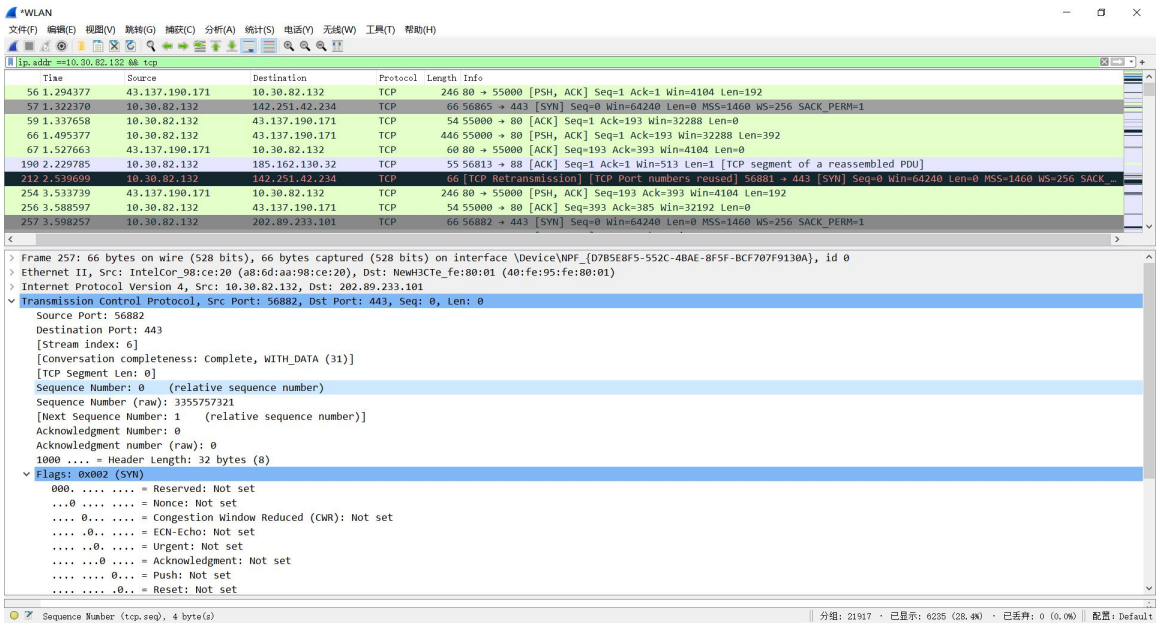
```
无线局域网适配器 WLAN:

   连接特定的 DNS 后缀 . . . . . . . : xmu.edu.cn
   描述. . . . . . . . . . . . . . . : Intel(R) Wireless-AC 9560 160MHz
   物理地址. . . . . . . . . . . . . : A8-6D-AA-98-CE-20
   DHCP 已启用 . . . . . . . . . . . : 是
   自动配置已启用. . . . . . . . . . : 是
   IPv6 地址 . . . . . . . . . . . . : 2001:da8:e800:71e4:59fa:5623:4902:2d30(首选)
   临时 IPv6 地址. . . . . . . . . . : 2001:da8:e800:71e4:4d7:8204:54fb:865b(首选)
   本地链接 IPv6 地址. . . . . . . . : fe80::59fa:5623:4902:2d30%20(首选)
   IPv4 地址 . . . . . . . . . . . . : 10.30.82.132(首选)
   子网掩码 . . . . . . . . . . . . : 255.255.224.0
   获得租约的时间  . . . . . . . . . : 2022年3月23日 18:22:58
   租约过期的时间  . . . . . . . . . : 2022年3月23日 20:15:40
   默认网关. . . . . . . . . . . . . : fe80::42fe:95ff:fefe:8001%20
                                       10.30.64.1
   DHCP 服务器 . . . . . . . . . . . : 172.18.0.12
   DHCPv6 IAID . . . . . . . . . . . : 145255850
   DHCPv6 客户端 DUID  . . . . . . . : 00-01-00-01-24-40-9C-86-C4-65-16-A9-CA-4B
   DNS 服务器  . . . . . . . . . . . : 121.192.181.18
                                       210.34.0.14
   TCPIP 上的 NetBIOS  . . . . . . . : 已启用
```

② 将过滤器设为 TCP 与本地 ip，根据数据包依次获取了帧数、IP 地址、MAC 地址。


```
> Frame 7376: 1294 bytes on wire (10352 bits), 1294 bytes captured (10352 bits) on interface \Device\NPF_{D7B5E8F5-552C-4BAE-8F5F-BCF707F9130A}, id 0

> Ethernet II, Src: NewH3CTe_fe:80:01 (40:fe:95:fe:80:01), Dst: IntelCor_98:ce:20 (a8:6d:aa:98:ce:20)
> Internet Protocol Version 4, Src: 58.205.220.42, Dst: 10.30.82.132
```

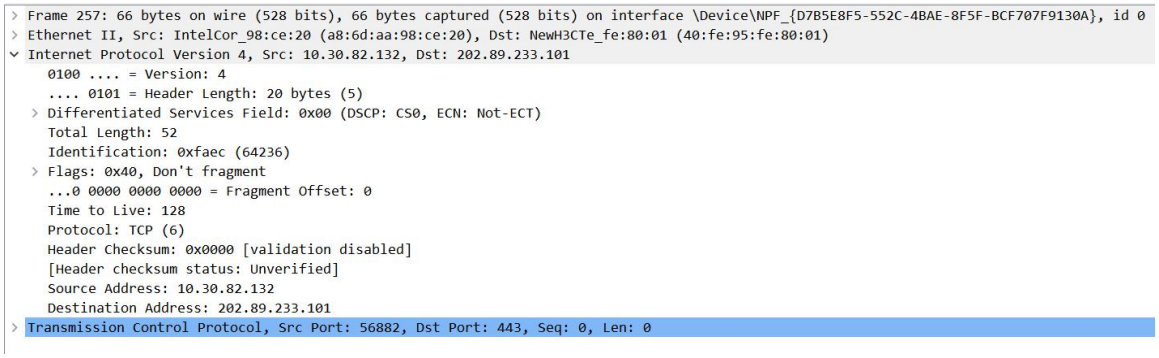
```
0000  a8 6d aa 98 ce 20 40 fe  95 fe 80 01 08 00 45 00   ·m··· @· ·····E·
0010  05 00 b9 0c 40 00 33 06  16 52 3a cd dc 2a 0a 1e   ····@·3· ·R:··*··
0020  52 84 01 bb de 7b e7 85  92 67 30 f0 0f c3 50 10   R····{·· ·g0···P·
0030  04 9c 31 88 00 00 23 4c  ca 32 76 67 00 d6 72 c6   ··1···#L ·2vg··r·
0040  c5 bf a1 69 8c 53 4b ac  34 b5 f5 87 2e 41 f9 c4   ···i·SK· 4····A··
0050  85 33 b0 ab 54 58 7a a4  3b 09 27 75 71 79 2d 7d   ·3··TXz· ;·'uqy-}
0060  c5 53 54 73 15 f4 3a 04  e3 d0 26 81 63 f8 5d 62   ·STs··:· ··&·c·]b
0070  51 fa ad 1f ca 49 03 37  4e bd 8d e0 da 12 76 d7   Q····I·7 N·····v·
0080  04 18 ea dd 2f 8d b7 82  a2 00 cf e3 d7 83 d9 0a   ····/··· ········
0090  98 90 28 d8 d4 c7 32 1c  82 62 98 2a 57 f3 5c 05   ··(···2· ·b·*W·\·
00a0  5c 21 95 8c a7 b8 91 8c  51 5b dc fa 5c ef 8e 3d   \!····· Q[··\··=
00b0  80 96 29 35 57 d3 8f 84  96 e2 a0 3f c8 69 a4 45   ··)5W··· ···?·i·E
00c0  0b 62 61 25 7b 18 a5 97  0d 1a 4b 9a 71 34 da 13   ·ba%{··· ··K·q4··
```

③ 使用 wireshark 分析 TCP 协议的数据包，其中包含物理层（frame），数据链路层（Ethernet），网络层（Internet），传输层（Transmission）的信息。



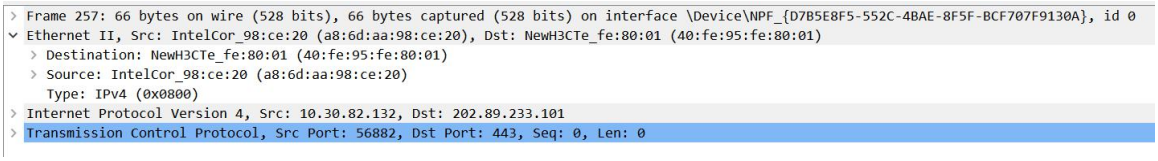
A. 观察 IP 报文格式（网络层）：

（1）IP 版本号 Version：4；（2）：首部长度 Header Length：20 字节；（3）服务类型 Differentiated Service Field：0x00；（4）数据包总长度 Total Length：52 字节；（5）标识符 Identification：0xfaec；（6）标识 Flags：不允许分片；（7）分片偏移 Fragment offset：0；（8）生存时间 Time to Live：数据报的生存周期，可以经过路由器的 128 条；（9）上层协议 Protocol：TCP；（10）校验和：0x0000，未校验；（11）源地址：10.30.82.132；（12）目的地址：202.89.233.101。



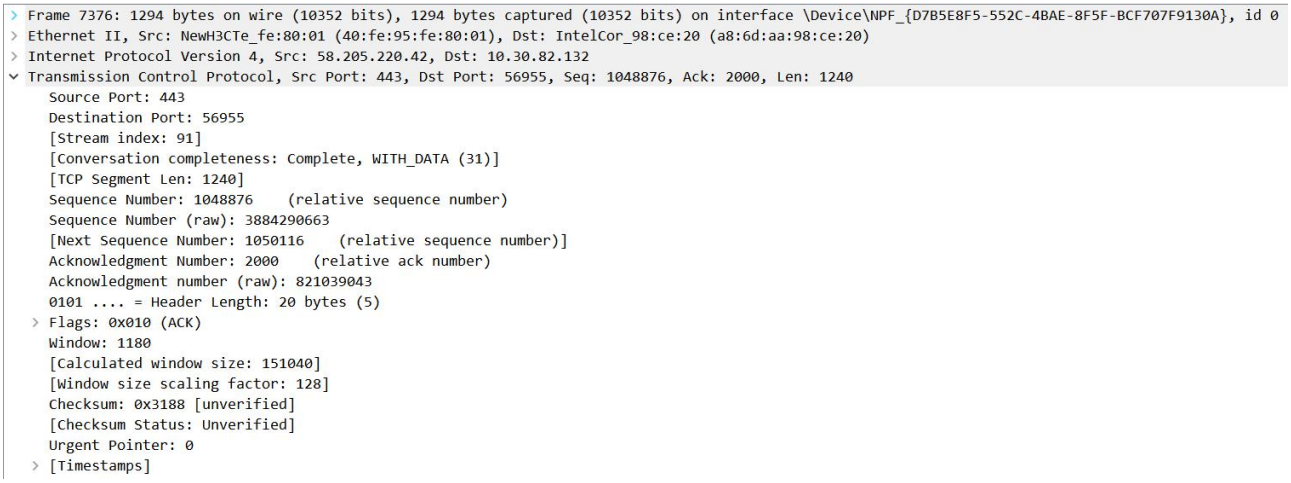
B. 观察帧格式（数据链路层）：

（1）目的地址：40:fe:95:fe:80:01；（2）源地址：a8:6d:aa:98:ce:20；（3）类型：IPv4。



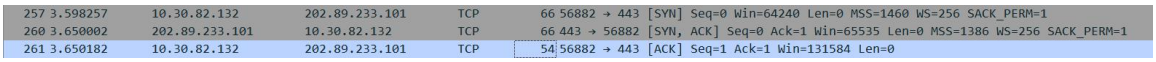
C. 观察 TCP 格式（传输层）：

（1）源端口:443； （2）目的端口：56955；（3）序号表示发送的字节流序号是 3884290663；（4） 确认号期待对方发送的字节序号是 821039043；（5）头部长度：20 字节；（6）标志位 Flags：确认位 ACK=1，表示已连接，在连接建立后所有传送的报文段都必须把 ACK 置为 1。（7）窗口大小：1180；（8）校验和：0x3188;（9）紧急指针：0。



2、用侦听解析软件观察 TCP 机制（TCP 的三次握手、四次挥手）

A. TCP 三次握手



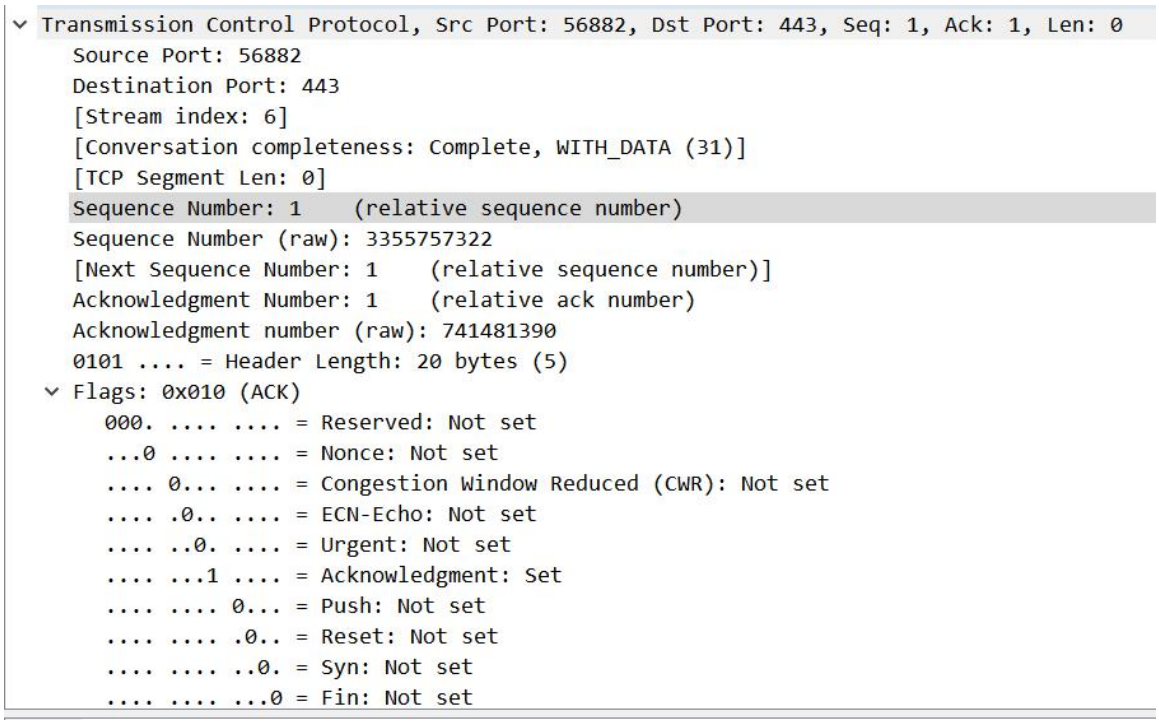
① 第一个握手数据包：客户端发送一个连接请求报文段，无应用层数据，标志位为同步比特 SYN，用来同步序号。序列号 seq 为 0，代表客户端请求建立连接。

```
Transmission Control Protocol, Src Port: 56882, Dst Port: 443, Seq: 0, Len: 0
    Source Port: 56882
    Destination Port: 443
    [Stream index: 6]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 3355757321
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  Flags: 0x002 (SYN)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    > .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
```

② 第二个握手数据包：服务器端为该 TCP 连接分配缓存和变量，并向客户端返回确认报文段，允许连接，无应用层数据。标志位为 SYN=1，ACK=1。将确认序号 ack 设置为 1。

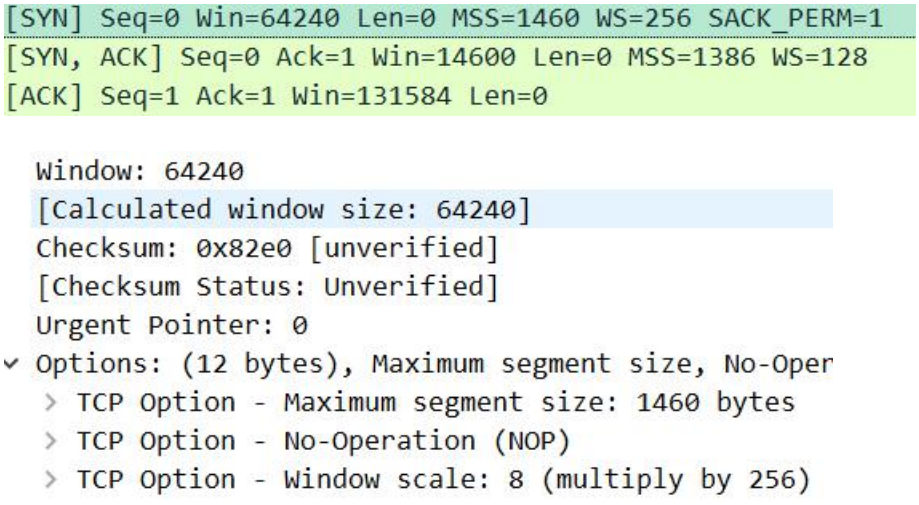
```
Transmission Control Protocol, Src Port: 443, Dst Port: 56882, Seq: 0, Ack: 1, Len: 0
    Source Port: 443
    Destination Port: 56882
    [Stream index: 6]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 741481389
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 3355757322
    1000 .... = Header Length: 32 bytes (8)
  Flags: 0x012 (SYN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    > .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
```

③ 第三个握手数据包：客户端为该 TCP 连接分配缓存和变量，并向服务器端再次发送确认包(ACK) ，可以携带数据。SYN 标志位为 0，ACK 标志位为 1。并且把服务器发来 ACK 的序号字段+1，放在确定字段中发送给对方。并且在数据段放写 ISN（Sequence Number）+1。



④ 三次握手时窗口滑动

用 wireshark 抓取 TCP 连接时的报文发现客户端的 Win 变大了，这里是使用了 Window Scale 来扩张 TCP 接收窗口，使得接收窗口可以大于 131584 字节。



首先 1 号包是 TCP 第一次握手连接时客户端的请求包，客户端如果窗口大于 64240，那么就先将 Window size 设置为 64240，表示客户端窗口将大于 64240，具体多大，在 3 号数据包会给出窗口的基数。

```
> Flags: 0x010 (ACK)
  Window: 514
  [Calculated window size: 131584]
  [Window size scaling factor: 256]
  Checksum: 0x82d4 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
> [Timestamps]
```

在三号包中，可以看到窗口的基数 Window 是 514，窗口的倍数是 2^8=256，那么实际的窗口大小为 514*256=131584Byte。

B. 四次挥手

7	0.101989	10.30.82.132	202.89.233.101	TCP	54 56876 → 443 [FIN, ACK] Seq=1 Ack=1 Win=510 Len=0
9	0.147712	202.89.233.101	10.30.82.132	TCP	60 443 → 56876 [ACK] Seq=1 Ack=2 Win=2051 Len=0
10	0.147712	202.89.233.101	10.30.82.132	TCP	60 443 → 56876 [FIN, ACK] Seq=1 Ack=2 Win=2051 Len=0
11	0.147862	10.30.82.132	202.89.233.101	TCP	54 56876 → 443 [ACK] Seq=2 Ack=2 Win=510 Len=0

① 第一次挥手：客户端发送连接释放报文段，停止发送数据，主动关闭 TCP 连接。FIN = 1，ACK=1。

```
v Transmission Control Protocol, Src Port: 56876, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
    Source Port: 56876
    Destination Port: 443
    [Stream index: 0]
    [Conversation completeness: Incomplete (20)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2403157568
    [Next Sequence Number: 2    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 2108969738
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x011 (FIN, ACK)
    Window: 510
    [Calculated window size: 510]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x107c [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
```

② 第二次挥手：服务器端回送一个确认报文段，客户到服务器这个方向的连接释放，就是半关闭状态。ACK = 1。序号为 1，确认序号为 2。

```
Transmission Control Protocol, Src Port: 443, Dst Port: 56876, Seq: 1, Ack: 2, Len: 0
    Source Port: 443
    Destination Port: 56876
    [Stream index: 0]
    [Conversation completeness: Incomplete (20)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2108969738
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 2    (relative ack number)
    Acknowledgment number (raw): 2403157569
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 2051
    [Calculated window size: 2051]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x114b [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
```

③ 第三次挥手：服务器端发送完数据，就发送连接释放报文段，主动关闭 TCP 连接。FIN = 1，ACK = 1。序号为 1，确认序号为 2。

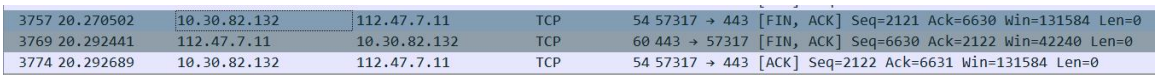
7	0.101989	10.30.82.132	202.89.233.101	TCP	54 56876 → 443 [FIN, ACK] Seq=1 Ack=1 Win=510 Len=0
9	0.147712	202.89.233.101	10.30.82.132	TCP	60 443 → 56876 [ACK] Seq=1 Ack=2 Win=2051 Len=0
10	0.147712	202.89.233.101	10.30.82.132	TCP	60 443 → 56876 [FIN, ACK] Seq=1 Ack=2 Win=2051 Len=0
11	0.147862	10.30.82.132	202.89.233.101	TCP	54 56876 → 443 [ACK] Seq=2 Ack=2 Win=510 Len=0

④ 第四次挥手：客户端回送一个确认报文段，再等到时间等待计时器设置的最长报文段寿命后，连接彻底关闭。ACK=1，序号为 2，确认序号为 2。

```
Transmission Control Protocol, Src Port: 56827, Dst Port: 443, Seq: 2, Ack: 2, Len: 0
    Source Port: 56827
    Destination Port: 443
    [Stream index: 62]
    [Conversation completeness: Incomplete (20)]
    [TCP Segment Len: 0]
    Sequence Number: 2    (relative sequence number)
    Sequence Number (raw): 2427486286
    [Next Sequence Number: 2    (relative sequence number)]
    Acknowledgment Number: 2    (relative ack number)
    Acknowledgment number (raw): 514432221
    0101 .... = Header Length: 20 bytes (5)
```

⑤ 问题讨论：执行程序时，发现很多抓到的 TCP 挥手是三次，而不是四次。查阅资料是因为服务器端收到客户端的 FIN 后，服务器端同时也要关闭连接，

这样就可以把 ACK 和 FIN 合并到一起发送，节省了一个包，变成了“三次挥手”。而通常情况下，服务器端收到客户端的 FIN 后，很可能还没发送完数据，所以就会先回复客户端一个 ACK 包，稍等一会儿，完成所有数据包的发送后，才会发送 FIN 包，这也就是四次挥手了。

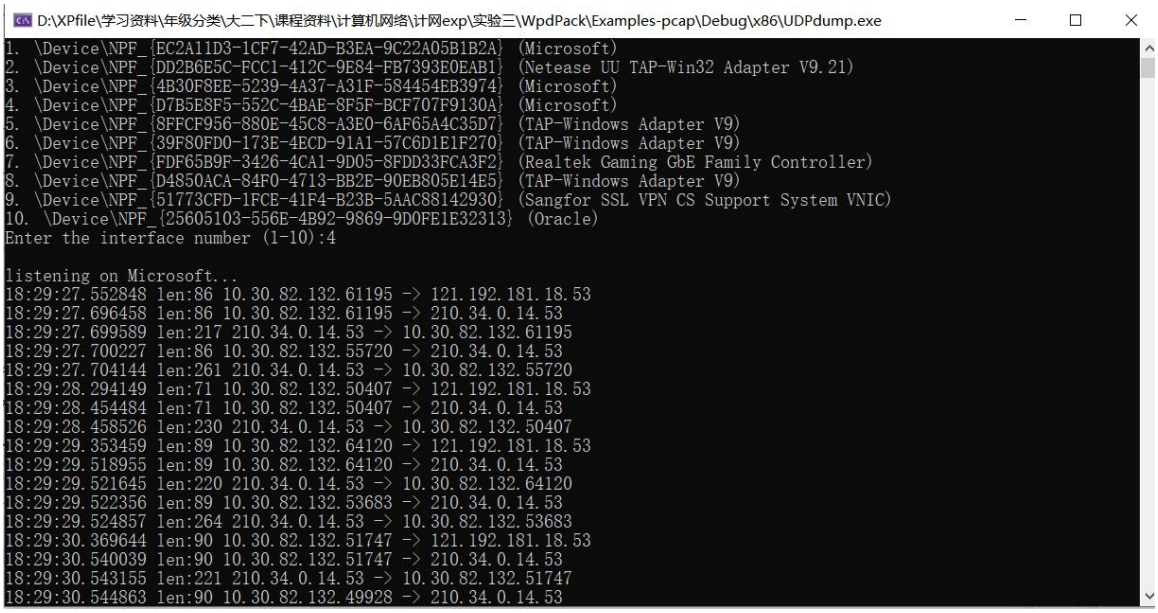


3757	20.270502	10.30.82.132	112.47.7.11	TCP	54 57317 → 443 [FIN, ACK] Seq=2121 Ack=6630 Win=131584 Len=0
3769	20.292441	112.47.7.11	10.30.82.132	TCP	60 443 → 57317 [FIN, ACK] Seq=6630 Ack=2122 Win=42240 Len=0
3774	20.292689	10.30.82.132	112.47.7.11	TCP	54 57317 → 443 [ACK] Seq=2122 Ack=6631 Win=131584 Len=0

3、用 WinPcap 库侦听网络数据

一、WinPcap 库侦听网络数据

① 下载 WinPcap 文件，卸载其他项目后成功运行。

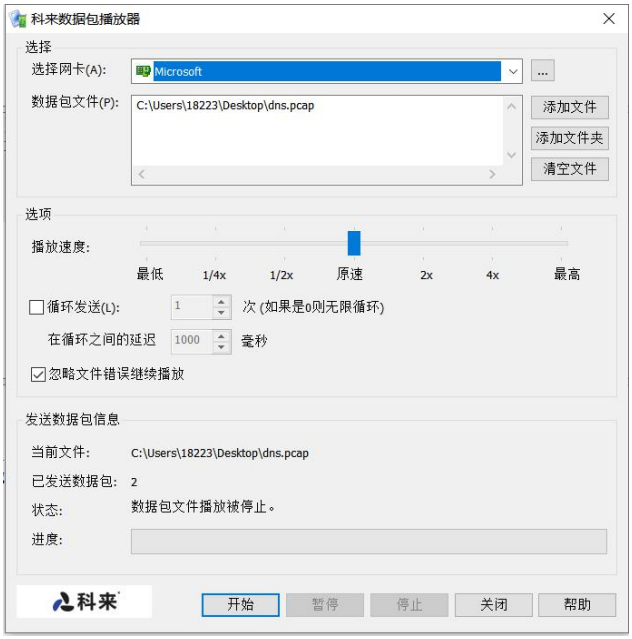


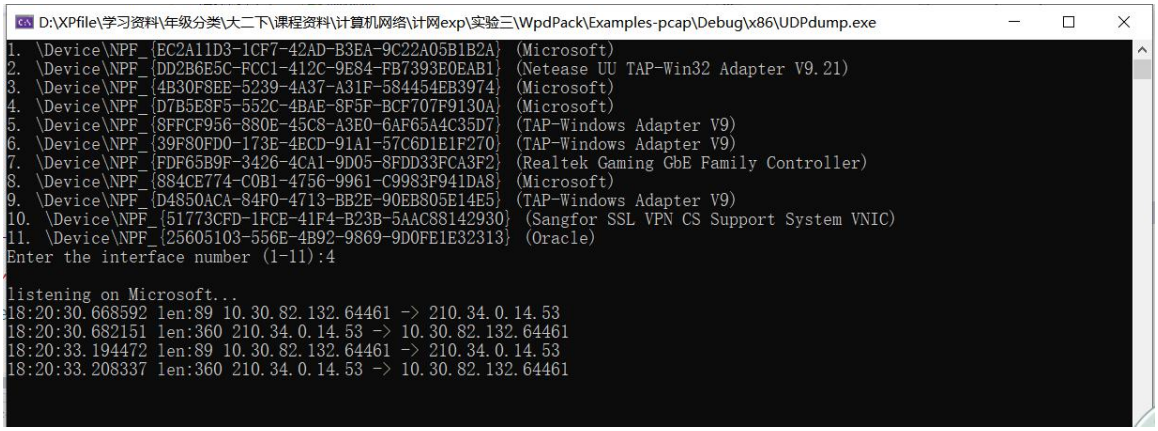
```
D:\XPfile\学习资料\年级分类\大二下\课程资料\计算机网络\计网exp\实验三\WpdPack\Examples-pcap\Debug\x86\UDPdump.exe
1. \Device\NPF_{EC2A11D3-1CF7-42AD-B3EA-9C22A05B1B2A} (Microsoft)
2. \Device\NPF_{DD2B6E5C-FCC1-412C-9E84-FB7393E0EAB1} (Netease UU TAP-Win32 Adapter V9.21)
3. \Device\NPF_{4B30F8EE-5239-4A37-A31F-584454EB3974} (Microsoft)
4. \Device\NPF_{D7B5E8F5-552C-4BAE-8F5F-BCF707F9130A} (Microsoft)
5. \Device\NPF_{8FFCF956-880E-45C8-A3E0-6AF65A4C35D7} (TAP-Windows Adapter V9)
6. \Device\NPF_{39F80FD0-173E-4ECD-91A1-57C6D1E1F270} (TAP-Windows Adapter V9)
7. \Device\NPF_{FDF65B9F-3426-4CA1-9D05-8FDD33FCA3F2} (Realtek Gaming GbE Family Controller)
8. \Device\NPF_{D4850ACA-84F0-4713-BB2E-90EB805E14E5} (TAP-Windows Adapter V9)
9. \Device\NPF_{51773CFD-1FCE-41F4-B23B-5AAC88142930} (Sangfor SSL VPN CS Support System VNIC)
10. \Device\NPF_{25605103-556E-4B92-9869-9D0FE1E32313} (Oracle)
Enter the interface number (1-10):4

listening on Microsoft...
18:29:27.552848 len:86 10.30.82.132.61195 -> 121.192.181.18.53
18:29:27.696458 len:86 10.30.82.132.61195 -> 210.34.0.14.53
18:29:27.699589 len:217 210.34.0.14.53 -> 10.30.82.132.61195
18:29:27.700227 len:86 10.30.82.132.55720 -> 210.34.0.14.53
18:29:27.704144 len:261 210.34.0.14.53 -> 10.30.82.132.55720
18:29:28.294149 len:71 10.30.82.132.50407 -> 121.192.181.18.53
18:29:28.454484 len:71 10.30.82.132.50407 -> 210.34.0.14.53
18:29:28.458526 len:230 210.34.0.14.53 -> 10.30.82.132.50407
18:29:29.353459 len:89 10.30.82.132.64120 -> 121.192.181.18.53
18:29:29.518955 len:89 10.30.82.132.64120 -> 210.34.0.14.53
18:29:29.521645 len:220 210.34.0.14.53 -> 10.30.82.132.64120
18:29:29.522356 len:89 10.30.82.132.53683 -> 210.34.0.14.53
18:29:29.524857 len:264 210.34.0.14.53 -> 10.30.82.132.53683
18:29:30.369644 len:90 10.30.82.132.51747 -> 121.192.181.18.53
18:29:30.540039 len:90 10.30.82.132.51747 -> 210.34.0.14.53
18:29:30.543155 len:221 210.34.0.14.53 -> 10.30.82.132.51747
18:29:30.544863 len:90 10.30.82.132.49928 -> 210.34.0.14.53
```

② WinPcap 库侦听数据

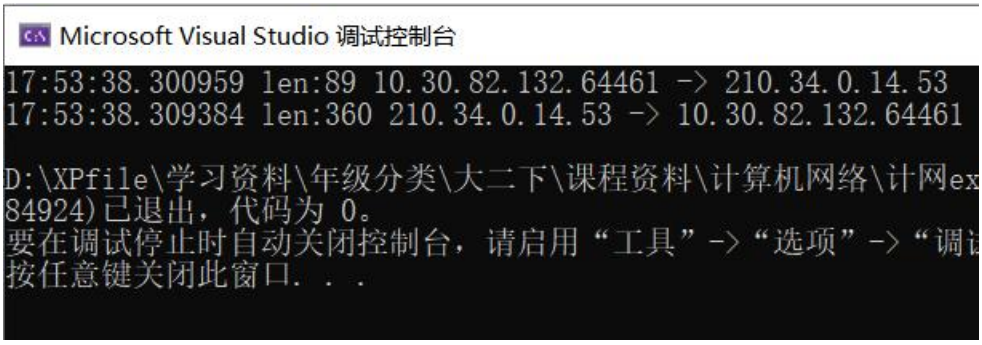
（1）使用 WirShark 捕获 WLAN 上的数据包。在左上角输入 dns 可以过滤出 dns。选中两个记录，导出特定分组，命名为 dns.pcap

（2）在高级防火墙中新建出站规则，阻止其他一切连接。使用科来数据包播放器单独播放 dns.pcap，调试成功。





（3）修改 UDPdump 项目的代码，根据同一文件目录下的 readfile 工程的文件的读取 pcap 文件的代码段，使得 UDPdump 项目可以读取 pcap 文件。



二、解析 MAC 和 IP 地址，记录统计（在 winpcap 自带的工程上进行更新）

（1）核心代码

① PART 1：修改输出到 csv 文件中的格式

（下图展示了修改源 MAC 地址和修改源 IP 地址的代码，修改目的地址也是类似的，不再贴出。）

```
//修改时间戳格式
strftime(timestr, sizeof timestr, "%Y-%m-%d %H:%M:%S", ltime);
mh = (mac_header*)pkt_data;


//打印源MAC地址
for (int i = 0; i < 6; i++)
{
    fprintf(file, "%02X", mh->src_addr[i]);
    printf("%02X", mh->src_addr[i]);
    if (i != 5) {
        fprintf(file, "-");printf("-");
    }
}
fprintf(file, ",");
printf(",");

//打印源ip地址
fprintf(file, "%d.%d.%d.%d,",ih->saddr.byte1,ih->saddr.byte2,
    ih->saddr.byte3,ih->saddr.byte4);
printf("%d.%d.%d.%d,",ih->saddr.byte1,ih->saddr.byte2,
    ih->saddr.byte3,ih->saddr.byte4);

//打印帧长度
fprintf(file, "%d\n", header->len);
printf("%d\n", header->len);
```

② PART 2：对每分钟数据统计分析

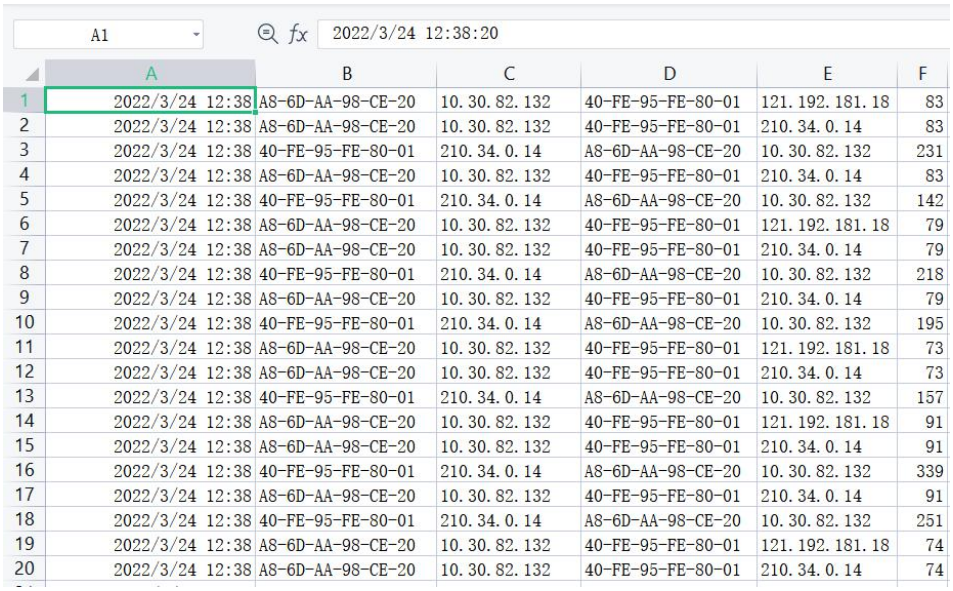
（下图展示了统计来自不同 MAC 和 IP 地址的通信数据长度的代码，统计发送到不同 MAC 和 IP 地址的通信数据长度的代码是类似的，不再展示。）

```
//程序统计来自不同 MAC 和 IP 地址的通信数据长度
int flag = 0;
for (int i = 0; i < src_length; i++)
{
    //如果src数组中第i+1个地址与saddr对应，则将length数组（存储数据长度）的第i+1个元素的值加上len
    if (src[i][0] == ih->saddr.byte1 && src[i][1] == ih->saddr.byte2 && src[i][2] == ih->saddr.byte3
        && src[i][3] == ih->saddr.byte4 && src[i][4] == mh->src_addr[0] && src[i][5] == mh->src_addr[1]
        && src[i][6] == mh->src_addr[2] && src[i][7] == mh->src_addr[3] && src[i][8] == mh->src_addr[4]
        && src[i][9] == mh->src_addr[5]){
        src_packet_length[i] += header->len;
        flag = 1;
        break;
    }
}
//如果上面的循环一次也没有进入，则将saddr直接赋给src第src_length个元素，并将相应length中的值+len
if (!flag){
    src[src_length][0] = ih->saddr.byte1;   src[src_length][1] = ih->saddr.byte2;
    src[src_length][2] = ih->saddr.byte3;   src[src_length][3] = ih->saddr.byte4;
    src[src_length][4] = mh->src_addr[0];   src[src_length][5] = mh->src_addr[1];
    src[src_length][6] = mh->src_addr[2];   src[src_length][7] = mh->src_addr[3];
    src[src_length][8] = mh->src_addr[4];   src[src_length][9] = mh->src_addr[5];
    src_packet_length[src_length] = header->len;
    src_length++;
}


//当时间到达一分钟
if (GetTickCount64() - preview >= 60000)
{
    fprintf(file, "\n\tSource Address and Packets(within 1 min):\t\n");
    for (int i = 0; i < src_length; i++)
    {
        fprintf(file, "Statistic%d: ,IP Address: %d.%d.%d.%d,MAC Address: %02X-%02X-%02X-%02X-%02X-%02X,Packets: %d\n",
            (i + 1), src[i][0], src[i][1], src[i][2], src[i][3], src[i][4], src[i][5], src[i][6], src[i][7],
            src[i][8], src[i][9], src_packet_length[i]);
    }
}
```

（2）实验结果

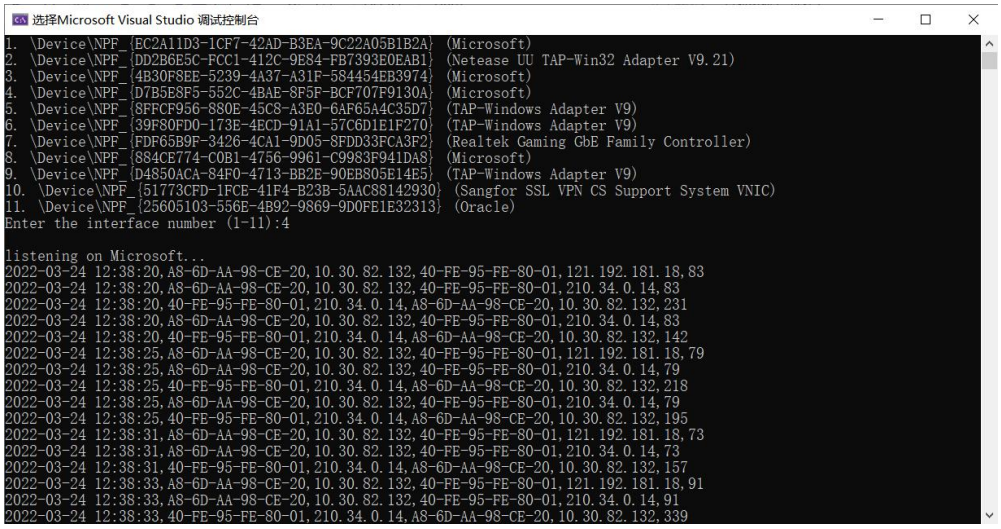
① 程序在文件上输出形如下列 CSV 格式的日志：

时间、源 MAC、源 IP、目标 MAC、目标 IP、帧长度（以逗号间隔）



A1 2022/3/24 12:38:20

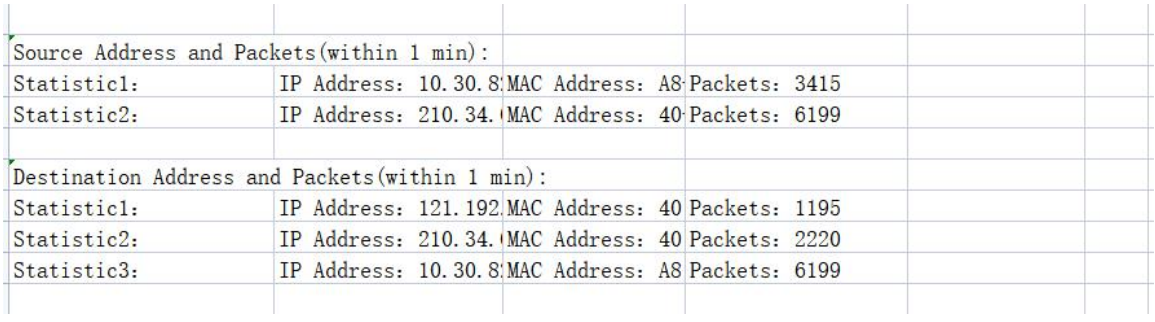
	A	B	C	D	E	F
1	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	121.192.181.18	83
2	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	83
3	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	231
4	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	83
5	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	142
6	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	121.192.181.18	79
7	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	79
8	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	218
9	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	79
10	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	195
11	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	121.192.181.18	73
12	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	73
13	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	157
14	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	121.192.181.18	91
15	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	91
16	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	339
17	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	91
18	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	251
19	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	121.192.181.18	74
20	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	74



选择Microsoft Visual Studio 调试控制台

```
1. \Device\NPF_{EC2A11D3-1CF7-42AD-B3EA-9C22A05B1B2A} (Microsoft)
2. \Device\NPF_{DD2B6E5C-FCC1-412C-9E84-FB7393E0EAB1} (Netease UU TAP-Win32 Adapter V9.21)
3. \Device\NPF_{4B30F8EE-5239-4A37-A31F-584454EB3974} (Microsoft)
4. \Device\NPF_{D7B5E8F5-552C-4BAE-8F5F-BCF707F9130A} (Microsoft)
5. \Device\NPF_{8FFCF956-880E-45C8-A3E0-6AF65A4C35D7} (TAP-Windows Adapter V9)
6. \Device\NPF_{39F80FD0-173E-4ECD-91A1-57C6D1E1F270} (TAP-Windows Adapter V9)
7. \Device\NPF_{FDF65B9F-3426-4CA1-9D05-8FDD33FCA3F2} (Realtek Gaming GbE Family Controller)
8. \Device\NPF_{884CE774-C0B1-4756-9961-C9983F941DA8} (Microsoft)
9. \Device\NPF_{D4850ACA-84F0-4713-BB2E-90EB805E14E5} (TAP-Windows Adapter V9)
10. \Device\NPF_{51773CFD-1FCE-41F4-B23B-5AAC88142930} (Sangfor SSL VPN CS Support System VNIC)
11. \Device\NPF_{25605103-556E-4B92-9869-9D0FE1E32313} (Oracle)
Enter the interface number (1-11):4

listening on Microsoft...
2022-03-24 12:38:20,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,121.192.181.18,83
2022-03-24 12:38:20,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,210.34.0.14,83
2022-03-24 12:38:20,40-FE-95-FE-80-01,210.34.0.14,A8-6D-AA-98-CE-20,10.30.82.132,231
2022-03-24 12:38:20,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,210.34.0.14,83
2022-03-24 12:38:20,40-FE-95-FE-80-01,210.34.0.14,A8-6D-AA-98-CE-20,10.30.82.132,142
2022-03-24 12:38:25,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,121.192.181.18,79
2022-03-24 12:38:25,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,210.34.0.14,79
2022-03-24 12:38:25,40-FE-95-FE-80-01,210.34.0.14,A8-6D-AA-98-CE-20,10.30.82.132,218
2022-03-24 12:38:25,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,210.34.0.14,79
2022-03-24 12:38:25,40-FE-95-FE-80-01,210.34.0.14,A8-6D-AA-98-CE-20,10.30.82.132,195
2022-03-24 12:38:31,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,121.192.181.18,73
2022-03-24 12:38:31,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,210.34.0.14,73
2022-03-24 12:38:31,40-FE-95-FE-80-01,210.34.0.14,A8-6D-AA-98-CE-20,10.30.82.132,157
2022-03-24 12:38:33,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,121.192.181.18,91
2022-03-24 12:38:33,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,210.34.0.14,91
2022-03-24 12:38:33,40-FE-95-FE-80-01,210.34.0.14,A8-6D-AA-98-CE-20,10.30.82.132,339
```

② 每隔一段时间（如 1 分钟），程序统计来自不同 MAC 和 IP 地址的通信数据长度，统计发至不同 MAC 和 IP 地址的通信数据长度。



Source Address and Packets(within 1 min):			
Statistic1:	IP Address: 10.30.8	MAC Address: A8	Packets: 3415
Statistic2:	IP Address: 210.34.(	MAC Address: 40	Packets: 6199
Destination Address and Packets(within 1 min):			
Statistic1:	IP Address: 121.192	MAC Address: 40	Packets: 1195
Statistic2:	IP Address: 210.34.(	MAC Address: 40	Packets: 2220
Statistic3:	IP Address: 10.30.8	MAC Address: A8	Packets: 6199

三、解析侦听到的网络数据（以 FTP 密码侦听为例）

（1）核心代码

① 找到标志性的信息。一般登录名以“USER”开头，口令以“PASS”开头，登录成功以“230”开头，失败以“530”开头。

```
for (head = 0; head < 60; head++)
{
    com.clear();
    for (int i = 0; i < 4; i++)  com += (char)pkt_data[head + i];
    //找到标志性的信息
    if (com == "USER" || com == "PASS" || com == "230 " || com == "530 ")
        break;
}
```

② 获取 USER 信息，获取 PASS 信息类似，不再重复。打印时间、源 MAC、源 IP、目标 MAC、目标 IP 与上一个实验项目中的代码相同。

```
//一般登录名以“USER”开头
if (com == "USER")
{
    std::ostringstream sout;
    //从第6位开始，第5位是空格，遇到回车(13)跳出循环
    for (int i = head + 5; pkt_data[i] != 13; i++){
        sout << pkt_data[i];
    }
    user = sout.str();//获取user
}
```

（2）实验结果：侦听得到系 FTP 的用户名和密码

A	B	C	D	E	F	G	H
2022/3/24 15:49	40-FE-95-F	121.192.18	A8-6D-AA-9	10.30.82.1	anonymous	IEUser@	FAILED
2022/3/24 15:49	A8-6D-AA-9	10.30.82.1	40-FE-95-F	121.192.180.66			FAILED
2022/3/24 15:49	A8-6D-AA-9	10.30.82.1	40-FE-95-F	121.192.180.66			FAILED
2022/3/24 15:49	40-FE-95-F	121.192.18	A8-6D-AA-9	10.30.82.1	student	software	SUCCEED
2022/3/24 15:49	A8-6D-AA-9	10.30.82.1	40-FE-95-F	121.192.180.66			SUCCEED
2022/3/24 15:49	40-FE-95-F	121.192.18	A8-6D-AA-9	10.30.82.1	student	software	SUCCEED
2022/3/24 15:49	A8-6D-AA-9	10.30.82.1	40-FE-95-F	121.192.180.66			SUCCEED

4 实验代码

本次实验的代码已上传于以下代码仓库：

https://github.com/ryanregal/Exp_ComputerNetwork

5 实验总结

通过这次实验，我对网络信息传输有了更直接的体验和了解，学习了 WINPCAP 和 WIRESHARK。学习了如何在调试中观察内存，查看有关 MAC 地址、IP 报文的内容，观察了 TCP 的三次握手，四次挥手和滑动窗口机制。我还学会了如何在已有的工程项目上进行更改、添加，调试等。过程中遇到了一些问题，比如一开始使用科来数据包生成器找不到网卡适配器，发现是软件本身与 Win10 系统的兼容性有问题，在更改电脑兼容性并以管理员身份运行后发现可以正常解决。还有比如用 Wireshark 抓挥手包的时候有的找不到，查阅资料才知道是因为服务器端收到客户端的 FIN 后，服务器端同时也要关闭连接，这样就可以把 ACK 和 FIN 合并到一起发送，节省了一个包，变成了“三次挥手”。另外在进行 FTP 实验时，出现了 The C++ Standard Library forbids macroizing keywords 的错误，这是自定义 inline 导致和系统文件发生冲突，于是预处理器定义中加入“_XKEYCHECK_H”，成功运行。