

# 廈門大學



## 信息学院软件工程系

### 《计算机网络》实验报告

题 目 实验四 基于 PCAP 库侦听并分析网络流量

班 级 数字媒体技术 2022 级 1 班

姓 名 魏清晨

学 号 37220222203790

实验时间 2024 年 10 月 11 日

2024 年 10 月 11 日

# 填写说明

- 1、本文件为 Word 模板文件，建议使用 Microsoft Word 2021 打开，在可填写的区域中如实填写；
- 2、填表时勿改变字体字号，保持排版工整，打印为 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：Win11 编程语言：C++ 调试软件：CLion2023.2

## 3 实验结果

### 1. 用侦听解析软件观察数据格式

#### （1）网络协议层次嵌套

对于一段数据帧，它包含多个部分，包括上层数据、TCP 头部、IP 头部、MAC 头部，在每一个层级中，每个头部和剩余部分为一个整体，如在应用层只有上层数据，在传输层由 TCP 头部和上层数据组成**数据段**，在网络层由 IP 头部和数据段组成**数据包**，在数据链路层由 MAC 头部和数据包组成**数据帧**。

#### （2）TCP 段格式

```

Transmission Control Protocol, Src Port: 51213, Dst Port: 53, Seq: 1, Ack: 1, Len: 0
  Source Port: 51213
  Destination Port: 53
  [Stream index: 169]
  [Stream Packet Number: 3]
  > [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 0]
  Sequence Number: 1 (relative sequence number)
  Sequence Number (raw): 4144285289
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 1 (relative ack number)
  Acknowledgment number (raw): 1432934802
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
  Window: 64240
  [Calculated window size: 64240]
  [Window size scaling factor: -2 (no window scaling used)]
  Checksum: 0x0010 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]

```

应用层数据前就是 TCP，这里 TCP 头部有 20 位，具体来讲

源 端 口 号	目标 端口 号	序 列 号	确认 序列 号	TCP 头 部长度	标志位（此 处有效的为 9 个）	窗 口	校 验 和	紧 急 指 针
16 位	16 位	32 位	32 位	4 位	12 位	16 位	16 位	16 位

### (3) IP 报文格式

```

Internet Protocol Version 4, Src: 10.30.35.217, Dst: 210.34.0.14
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 40
  Identification: 0x6563 (25955)
  > 010. .... = Flags: 0x2, Don't fragment
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 128
  Protocol: TCP (6)
  Header Checksum: 0x9545 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 10.30.35.217
  Destination Address: 210.34.0.14
  [Stream index: 0]

```

TCP 头部前就是 IP 头部，有 20 个字节，具体来讲如下

版本	IP 包头长度	差分服务字段	IP 包总长度	标识字段	标记字段	分片偏移	存活时间	协议	校验和	目的地址	源地址
4 位	4 位	8 位	16 位	16 位	3 位	13 位	8 位	8 位	16 位	32 位	32 位

## (4) 帧格式

```
Ethernet II, Src: CloudNetwork_60:9c:87 (d8:80:83:60:9c:87), Dst: NewH3CTechno_fe:80:01 (40:fe:95:fe:80:01)
> Destination: NewH3CTechno_fe:80:01 (40:fe:95:fe:80:01)
> Source: CloudNetwork_60:9c:87 (d8:80:83:60:9c:87)
  Type: IPv4 (0x0800)
  [Stream index: 0]
```

这部分是以太网帧格式中前头部部分，此处可以看到目的地址

Destination: NewH3CTechno\_fe:80:01 (40:fe:95:fe:80:01)，源地址

Source: CloudNetwork\_60:9c:87 (d8:80:83:60:9c:87)，类型 IPv4。

## (5) MAC 地址、IP 地址、TCP 端口

由上述头部可以得到，源 MAC 地址为 NewH3CTechno\_fe:80:01 (40:fe:95:fe:80:01)，目的 MAC 地址为 CloudNetwork\_60:9c:87 (d8:80:83:60:9c:87)

源 IP 地址为 Source Address: 10.30.35.217，目的 IP 地址为

Destination Address: 210.34.0.14

源 TCP 端口为：51212，目的 TCP 端口为 53

## (6) FTP 协议

此处搭建了一个 127.0.0.1 的 ftp 网站

使用 cd 1 命令后，发送了

```
7388 2499.855799 127.0.0.1 127.0.0.1 FTP 51 Request: CWD 1
```

响应为

7390	2499.856202	127.0.0.1	127.0.0.1	FTP	73	Response: 250 CWD command successful.
------	-------------	-----------	-----------	-----	----	---------------------------------------

响应格式就是一个响应代码，250 就是文件行为完成，后面的 info 都会被输出到控制台中

此外还有其他响应：

8191	2779.668934	127.0.0.1	127.0.0.1	FTP	67	Response: 331 Password required.
------	-------------	-----------	-----------	-----	----	----------------------------------

8207	2783.270006	127.0.0.1	127.0.0.1	FTP	65	Response: 230 User logged in.
------	-------------	-----------	-----------	-----	----	-------------------------------

8351	2833.313160	127.0.0.1	127.0.0.1	FTP	98	Response: 125 Data connection already open; Transfer starting.
------	-------------	-----------	-----------	-----	----	--

8550	2902.047519	127.0.0.1	127.0.0.1	FTP	58	Response: 221 Goodbye.
------	-------------	-----------	-----------	-----	----	------------------------

## 2. 用侦听解析软件观察 TCP 机制

1424	390.449131	192.168.31.25	36.189.214.9	TCP	74	53144 → 80 [SYN] Seq=0 Win=65160 Len=0 MSS=1460 WS=256 SACK_PERM TSval=13177303 TSecr=0
1425	390.451066	192.168.31.25	121.204.230.169	TCP	66	53145 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1426	390.455952	36.189.214.9	192.168.31.25	TCP	54	80 → 53141 [ACK] Seq=1 Ack=428 Win=30720 Len=0
1427	390.456981	36.189.214.9	192.168.31.25	HTTP	530	HTTP/1.1 403 Forbidden (text/html)
1428	390.458776	121.204.230.169	192.168.31.25	TCP	66	80 → 53145 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1416 SACK_PERM WS=128
1429	390.458824	192.168.31.25	121.204.230.169	TCP	54	53145 → 80 [ACK] Seq=1 Ack=1 Win=263168 Len=0
1430	390.458895	192.168.31.25	121.204.230.169	HTTP	368	GET /filestreamingservice/files/3d3c4265-57fd-450e-9bda-9fb5f4612029/pieceshash HTTP/1.1
1431	390.459412	192.168.31.25	36.189.214.9	TCP	54	53141 → 80 [FIN, ACK] Seq=428 Ack=477 Win=262912 Len=0
1432	390.468613	121.204.230.169	192.168.31.25	TCP	54	80 → 53145 [ACK] Seq=1 Ack=315 Win=523648 Len=0
1433	390.468613	36.189.214.9	192.168.31.25	TCP	54	80 → 53142 [ACK] Seq=1 Ack=428 Win=30720 Len=0
1434	390.468613	121.204.230.169	192.168.31.25	TCP	766	80 → 53145 [PSH, ACK] Seq=1 Ack=315 Win=523648 Len=712 [TCP PDU reassembled in 1435]
1435	390.468613	121.204.230.169	192.168.31.25	HTTP	252	HTTP/1.1 200 OK
1436	390.468692	192.168.31.25	121.204.230.169	TCP	54	53145 → 80 [ACK] Seq=315 Ack=911 Win=262400 Len=0
1437	390.470382	36.189.214.9	192.168.31.25	HTTP	530	HTTP/1.1 403 Forbidden (text/html)
1438	390.471898	192.168.31.25	36.189.214.9	TCP	54	53142 → 80 [FIN, ACK] Seq=428 Ack=477 Win=262912 Len=0

## 关于段 ID，涉及到三次握手

TCP 的三次握手，1425、1428、1429 就是一次握手的过程，第一次 seq=0 表示是新的开始，之后第二次收到 seq=0，ack=1 表示对方要这方从 seq=1 开始传数据，第三次 seq=1 响应刚才的包，ack=1 表示收到。

然后，1430 这一行是客户端发起 http 的请求，这一帧的长度为 314

1432 这里服务端发回 Ack=314+1=315 表示接收成功，但没有数据传回

1434 带上了 PSH 标志字，表示有 DATA 数据传输，由 ACK 可知还是对 1430 的回复，这时数据长度为 712

1435 服务端又发了一个 http，而且 PSH 也是 1，把剩下的数据发过来，长度为 198

1436 表示收到了服务端的数据，应答 ACK=911=1+712+198 表示前两个数据都收到了

## 关于窗口机制

可以看到 WIN 字段，这里有的包使用了窗口大小因子对窗口大小进行了放缩，所以 WIN 的数值很大。

WIN 表示发送该 TCP 报文的接受窗口还可以接受多少字节的数据量，如果已满，就不会再发送，而且就算窗口中间有发送的包获得了回复，也不能移动窗口，只能从两端进行操作

## 关于拥塞控制机制

发送方维护 cwnd，cwnd 不出现在报文中，使得 swnd（发送窗口）等于 cwnd，cwnd 的维护有两个算法，当  $cwnd < ssthresh$ ，使用慢开始算法，发送方每收到一个对新报文的确认，使 cwnd 加一，1 到 2，2 到 4，4 到 8 指数增长；当  $cwnd > ssthresh$ ，使用拥塞避免，限制每个传输轮次，一次只能至多增长 1，当发生超时重传时，判断拥塞，令 ssthresh 的值变为 cwnd 的一半，所以这个拥塞是动态的拥塞

## 3. 用 Libpcap 或 WinPcap 库侦听网络数据

主要处理侦听到的数据的函数部分：

```
// 回调函数：处理每个捕获的包
void packet_handler(u_char *param, const pcap_pkthdr *header, const u_char *pkt_data)
{
    ethernet_header *eth_header;
    ip_header *ip_header;
    char srcMac[18], dstMac[18];
    char srcIp[16], dstIp[16];
    char timestamp[80];

    // 获取时间戳
    get_current_time(timestamp);

    // 解析以太网头
    eth_header = (ethernet_header *) (pkt_data);
    mac_to_str(eth_header->srcMac, srcMac);
    mac_to_str(eth_header->dstMac, dstMac);

    // 仅处理 IP 数据包
    if (ntohs(eth_header->type) == 0x0800)
    {
        ip_header = (struct ip_header *) (pkt_data + sizeof(struct ethernet_header));

        // 获取源和目标 IP 地址
        inet_ntop(Family: AF_INET, &(ip_header->srcAddr), srcIp, StringBufSize: sizeof(srcIp));
        inet_ntop(Family: AF_INET, &(ip_header->dstAddr), dstIp, StringBufSize: sizeof(dstIp));

        printf(format: "%s,%s,%s,%s,%s,%d\n", timestamp, srcMac, srcIp, dstMac, dstIp, header->len);
        fflush(stdout);
        // 记录到 CSV 日志，另外在运行时不要开着 excel，会写失败
        fprintf(logfile, format: "%s,%s,%s,%s,%s,%d\n", timestamp, srcMac, srcIp, dstMac, dstIp, header->len);

        // 统计数据长度
        data_from_mac[eth_header->srcMac] += header->len;
        data_to_mac[eth_header->dstMac] += header->len;

        data_from_ip[ip_header->srcAddr] += header->len;
        data_to_ip[ip_header->dstAddr] += header->len;

        // 定时输出来自/发至不同 MAC 和 IP 地址的通信数据长度
        auto currentTime:DWORD = GetTickCount();
        if (currentTime - lastPrintTime > 10000){...}
    }
}
```

此处设置 10 秒输出一次统计的收发数据长度

```
网卡：
#1: \Device\NPF_{8DAE8EE6-AD96-453B-BA67-79B1A4B23D06} - WAN Miniport (Network Monitor)
#2: \Device\NPF_{97C38C07-9231-4967-812B-E8D7B9EB3B84} - WAN Miniport (IPv6)
#3: \Device\NPF_{960BD1DC-AAF3-4DDA-B70F-2CF154D27A5F} - WAN Miniport (IP)
#4: \Device\NPF_{426D356D-0481-499C-A95A-9F192F802714} - Bluetooth Device (Personal Area Network)
#5: \Device\NPF_{033523CB-A3E6-40CA-AEA3-33D60B84CEEF} - OrayBoxVPN Virtual Ethernet Adapter
#6: \Device\NPF_{47414042-AEE7-46E5-943B-E391DB36A622} - VMware Virtual Ethernet Adapter for VMnet8
#7: \Device\NPF_{8C7838C8-F1DD-4D27-8BE1-A384758960E3} - VMware Virtual Ethernet Adapter for VMnet1
#8: \Device\NPF_{B0359F03-5C67-4A91-8CDF-8EEAC707865B} - MediaTek Wi-Fi 6 MT7921 Wireless LAN Card
#9: \Device\NPF_{21EB4869-EF98-48BD-A2AA-3F50BBEB0729} - Microsoft Wi-Fi Direct Virtual Adapter #2
#10: \Device\NPF_{E1DE4647-E2C1-47AE-8878-8ADE0922CC58} - Microsoft Wi-Fi Direct Virtual Adapter
#11: \Device\NPF_{7DBDAB04-77FE-47D6-B52B-F4057A483DF2} - VirtualBox Host-Only Ethernet Adapter
#12: \Device\NPF_{Loopback} - Adapter for loopback traffic capture
#13: \Device\NPF_{42D94E45-138A-4DB6-9851-6EFE4DC59731} - Sangfor SSL VPN CS Support System VNIC
8
侦听数据流 \Device\NPF_{B0359F03-5C67-4A91-8CDF-8EEAC707865B}...
2024-10-22 16:36:49, D8-80-83-60-9C-87, 192.168.31.25, 4C-C6-4C-58-5A-5C, 1.12.12.12, 55
来自 D8-80-83-60-9C-87 的数据长度为 55
发至 4C-C6-4C-58-5A-5C 的数据长度为 55
来自 192.168.31.25 的数据长度为 55
发至 1.12.12.12 的数据长度为 55
2024-10-22 16:36:49, 4C-C6-4C-58-5A-5C, 1.12.12.12, D8-80-83-60-9C-87, 192.168.31.25, 66
2024-10-22 16:36:49, D8-80-83-60-9C-87, 192.168.31.25, 4C-C6-4C-58-5A-5C, 150.138.235.209, 68
2024-10-22 16:36:49, 4C-C6-4C-58-5A-5C, 150.138.235.209, D8-80-83-60-9C-87, 192.168.31.25, 68
2024-10-22 16:36:50, D8-80-83-60-9C-87, 192.168.31.25, 01-00-5E-7F-FF-FA, 239.255.255.250, 331
2024-10-22 16:36:50, D8-80-83-60-9C-87, 192.168.31.25, 01-00-5E-7F-FF-FA, 239.255.255.250, 332
2024-10-22 16:36:55, D8-80-83-60-9C-87, 192.168.31.25, 01-00-5E-7F-FF-FA, 239.255.255.250, 331
2024-10-22 16:36:55, D8-80-83-60-9C-87, 192.168.31.25, 01-00-5E-7F-FF-FA, 239.255.255.250, 332
2024-10-22 16:36:59, D8-80-83-60-9C-87, 192.168.31.25, 4C-C6-4C-58-5A-5C, 150.138.235.209, 68
来自 4C-C6-4C-58-5A-5C 的数据长度为 134
来自 D8-80-83-60-9C-87 的数据长度为 1517
发至 01-00-5E-7F-FF-FA 的数据长度为 1326
发至 4C-C6-4C-58-5A-5C 的数据长度为 191
发至 D8-80-83-60-9C-87 的数据长度为 134
来自 1.12.12.12 的数据长度为 66
来自 150.138.235.209 的数据长度为 68
来自 192.168.31.25 的数据长度为 1517
发至 1.12.12.12 的数据长度为 55
发至 150.138.235.209 的数据长度为 136
发至 192.168.31.25 的数据长度为 134
发至 239.255.255.250 的数据长度为 1326
2024-10-22 16:36:59, 4C-C6-4C-58-5A-5C, 150.138.235.209, D8-80-83-60-9C-87, 192.168.31.25, 68
```

写入 csv 文件中的数据：

（虽然写入的日期是以“-”相隔，但在 csv 中不知为何变为了“/”



A	B	C	D	E	F
2024/10/22 16:36	D8-80-83-60-9C-87	192.168.31.25	4C-C6-4C-58-5A-5C	1.12.12.12	55
2024/10/22 16:36	4C-C6-4C-58-5A-5C	1.12.12.12	D8-80-83-60-9C-87	192.168.31.25	66
2024/10/22 16:36	D8-80-83-60-9C-87	192.168.31.25	4C-C6-4C-58-5A-5C	150.138.235.209	68
2024/10/22 16:36	4C-C6-4C-58-5A-5C	150.138.235.209	D8-80-83-60-9C-87	192.168.31.25	68
2024/10/22 16:36	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	331
2024/10/22 16:36	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	332
2024/10/22 16:36	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	331
2024/10/22 16:36	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	332
2024/10/22 16:36	D8-80-83-60-9C-87	192.168.31.25	4C-C6-4C-58-5A-5C	150.138.235.209	68
2024/10/22 16:36	4C-C6-4C-58-5A-5C	150.138.235.209	D8-80-83-60-9C-87	192.168.31.25	68
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	331
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	332
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	4C-C6-4C-58-5A-5C	1.12.12.12	55
2024/10/22 16:37	4C-C6-4C-58-5A-5C	1.12.12.12	D8-80-83-60-9C-87	192.168.31.25	66
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	331
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	332
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	4C-C6-4C-58-5A-5C	150.138.235.209	68
2024/10/22 16:37	4C-C6-4C-58-5A-5C	150.138.235.209	D8-80-83-60-9C-87	192.168.31.25	68
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	331
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	332
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	4C-C6-4C-58-5A-5C	52.55.118.121	171
2024/10/22 16:37	4C-C6-4C-58-5A-5C	52.55.118.121	D8-80-83-60-9C-87	192.168.31.25	54
2024/10/22 16:37	4C-C6-4C-58-5A-5C	52.55.118.121	D8-80-83-60-9C-87	192.168.31.25	106
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	4C-C6-4C-58-5A-5C	52.55.118.121	117
2024/10/22 16:37	4C-C6-4C-58-5A-5C	52.55.118.121	D8-80-83-60-9C-87	192.168.31.25	54
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	331
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	332
2024/10/22 16:37	4C-C6-4C-58-5A-5C	120.55.61.42	D8-80-83-60-9C-87	192.168.31.25	108
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	4C-C6-4C-58-5A-5C	120.55.61.42	128
2024/10/22 16:37	4C-C6-4C-58-5A-5C	120.55.61.42	D8-80-83-60-9C-87	192.168.31.25	54
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	4C-C6-4C-58-5A-5C	1.12.12.12	55
2024/10/22 16:37	4C-C6-4C-58-5A-5C	1.12.12.12	D8-80-83-60-9C-87	192.168.31.25	66
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	4C-C6-4C-58-5A-5C	150.138.235.209	68
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	4C-C6-4C-58-5A-5C	47.114.175.123	66
2024/10/22 16:37	4C-C6-4C-58-5A-5C	47.114.175.123	D8-80-83-60-9C-87	192.168.31.25	66
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	331
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	332
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	331
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	01-00-5E-7F-FF-FA	239.255.255.250	332
2024/10/22 16:37	D8-80-83-60-9C-87	192.168.31.25	4C-C6-4C-58-5A-5C	47.98.132.207	56
2024/10/22 16:37	4C-C6-4C-58-5A-5C	47.98.132.207	D8-80-83-60-9C-87	192.168.31.25	56

#### 4. 解析侦听到的网络数据

**ftp** 登录整个流程如下：

697 44.694029	127.0.0.1	127.0.0.1	FTP	71 Response: 220 Microsoft FTP Service
699 44.695617	127.0.0.1	127.0.0.1	FTP	58 Request: OPTS UTF8 ON
701 44.695659	127.0.0.1	127.0.0.1	FTP	102 Response: 200 OPTS UTF8 command successful - UTF8 encoding now ON.
735 47.284989	127.0.0.1	127.0.0.1	FTP	55 Request: USER test
737 47.285069	127.0.0.1	127.0.0.1	FTP	67 Response: 331 Password required
860 50.839577	127.0.0.1	127.0.0.1	FTP	57 Request: PASS 123456
862 50.840073	127.0.0.1	127.0.0.1	FTP	65 Response: 230 User logged in.

**实验登录 ftp** 流程如下：

```
E:\code\practice\Computer_Network\Experiment\cni-exp\E4_3790\b...
网卡:
#1: \Device\NPF_{8DAE8EE6-AD96-453B-BA67-7981A4B23D06} - WAN M...
#2: \Device\NPF_{97C38C07-9231-4967-812B-E8D7B9EB3B84} - WAN M...
#3: \Device\NPF_{960B01DC-AAF3-4DDA-B70F-2CF154D27A5F} - WAN M...
#4: \Device\NPF_{4260356D-0481-499C-A95A-9F192F802714} - Bluet...
#5: \Device\NPF_{033523CB-A3E6-40CA-AEA3-33D60B84CEEF} - OrayB...
#6: \Device\NPF_{47414042-AEE7-46E5-943B-E391DB36A622} - VMwar...
#7: \Device\NPF_{8C7838C8-F1D0-4D27-8BE1-A384758960E3} - VMwar...
#8: \Device\NPF_{B0359F03-5C67-4A91-8CDF-8EEAC707865B} - Media...
#9: \Device\NPF_{21EB4869-EF98-48BD-A2AA-3F50BBEB0729} - Micro...
#10: \Device\NPF_{E1DE4647-E2C1-47AE-8878-8ADE0922CC58} - Micro...
#11: \Device\NPF_{7DBDAB04-77FE-47D6-B52B-F4057A483DF2} - Virtu...
#12: \Device\NPF_Loopback - Adapter for loopback traffic captur...
#13: \Device\NPF_{42D94E45-138A-4DB6-9851-6EFE4DC59731} - Sangt...
12
侦听数据流 \Device\NPF_Loopback...
test
123456

进程已结束, 退出代码为 1
```

```
Microsoft Windows [版本 10.0.22631.4317]
(c) Microsoft Corporation。保留所有权利。

C:\Users\WQC20>ftp 127.0.0.1
连接到 127.0.0.1。
220 Microsoft FTP Service
200 OPTS UTF8 command successful - UTF8 encoding now ON.
用户(127.0.0.1:(none)): test
331 Password required
密码:
230 User logged in.
ftp> |
```

表格数据:

ip.cpp   ftp.cpp   logfile.csv								
C1	C2	C3	C4	C5	C6	C7	C8	
2024-10-22 20:02:02	00-30-02-07-40-00	127.0.0.1	02-00-00-00-45-00	127.0.0.1	test	123456	SUCCESS	

主要代码:

这里以太网帧格式的头部经过 wireshark 验证只有四个字节, 可能是回环数据的原因

```

void packet_handler(u_char *param, const pcap_pkthdr *header, const u_char *pkt_data)
{
    ethernet_header *eth_header = (ethernet_header *) (pkt_data);
    ip_header *ip_header = (struct ip_header *) (pkt_data + 4);
    const char *payload = (char *) (pkt_data + 44);
    char srcMac[18], dstMac[18];
    char srcIp[16], dstIp[16];
    char timestamp[80];

    // 获取时间戳
    get_current_time(timestamp);

    // 解析以太网头
    mac_to_str(eth_header->srcMac, srcMac);
    mac_to_str(eth_header->dstMac, dstMac);

    // 获取源和目标 IP 地址
    inet_ntop(Family: AF_INET, &(ip_header->srcAddr), srcIp, (StringBufSize: sizeof(srcIp)));
    inet_ntop(Family: AF_INET, &(ip_header->dstAddr), dstIp, (StringBufSize: sizeof(dstIp)));

    // 解析 FTP 数据
    std::string ftp_data(payload);
    // std::cout << "Captured FTP Data: " << ftp_data << std::endl;

    // 解析 FTP 命令
    int res = parse_ftp_commands(ftp_data);

    if (res)
    {
        if (res == 1)
        {
            // 记录到 CSV 日志, 另外在运行时不要开着excel, 会写失败
            fprintf(logfile, format:"%s,%s,%s,%s,%s,%s,SUCCESS\n", timestamp, srcMac, srcIp, dstMac, dstIp, user.c_str(), password.c_str());
        }
        if (res == -1)
        {
            // 记录到 CSV 日志, 另外在运行时不要开着excel, 会写失败
            fprintf(logfile, format:"%s,%s,%s,%s,%s,%s,FAILED\n", timestamp, srcMac, srcIp, dstMac, dstIp, user.c_str(), password.c_str());
        }

        fflush(logfile);
    }
}

```

```

int parse_ftp_commands(const std::string &ftp_data)
{
    std::istringstream stream(ftp_data);
    std::string command;

    while (stream >> command)
    {
        if (command == "USER")
        {
            stream >> user;
            std::cout << user << std::endl;
        }
        else if (command == "PASS")
        {
            stream >> password;
            std::cout << password << std::endl;
        }
        else if (command == "230")
        {
            return 1;
        }
        else if (command == "530")
        {
            user.clear();
            password.clear();
            return -1;
        }
    }

    return 0;
}

```

## 4 实验代码

本次实验的代码已上传于以下代码仓库：[CNI-Exp: 厦门大学计算机网络课程实验项目集 \(gitee.com\)](#)

## 5 课后思考题

无课后思考题

## 6 实验总结

本次实验通过 wireshark 的使用，一对网络层次协议嵌套有了更好的理解，二对 tcp 的数据收发机制有了大概的了解，三对 ftp 的命令格式有了一定的了解

另外，通过 npcap 库对网络数据的解析，对于接收方解析数据、分析数据有了一定的了解

最后，对于使用本机进行网络上的一些设置，如 ftp、防火墙设置等也有了一定的了解。