信息学院软件工程系

《计算机网络》实验报告

题　　目 实验三　基于 PCAP 库侦听并分析网络流量

班　　级 软件工程 2019 级 2 班

姓　　名 李世豪

学　　号 22920192204229

实验时间 2021 年 4 月 03 日

2021 年 6 月 03 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，勿超过 5MB；

4、应将材料清单上传在代码托管平台上；

5、在学期最后一节课前按要求打包发送至 cni21@qq.com。

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。

实验任务：

1、用侦听解析软件观察数据格式

2、用侦听解析软件观察 TCP 机制

3、用 Libpcap 或 WinPcap 库侦听网络数据

4、解析侦听到的网络数据

2 实验环境

操作系统：Mac Big Sur 11.3（类 Unix 系统）

编程语言：C/C++，libpcap 库

3 实验结果

。(1) 网络协议层嵌套

```
> Frame 5: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface en0, id 0
> Ethernet II, Src: Apple_d5:e9:6a (18:3e:ef:d5:e9:6a), Dst: Hangzhou_3c:d7:5d (c4:ca:d9:3c
> Internet Protocol Version 4, Src: 10.30.82.220, Dst: 182.61.200.6
> Transmission Control Protocol, Src Port: 50606, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
```

如图，在 wireshark 中捕捉到的数据包 packet 中：

第 1 行描述的含义：

Frame 5: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface en0, id 0

即数据包在本机 pc 网络接口处被捕获的记录，是网络层次中物理层的表现，表达含义即共 54bytes 在网络线路中传输，共 54bytes 在网络接口 en0 中被捕捉。

第 2 行到第 4 行描述：
以太网帧层次报文头，网络层 ip 报文头，传输层 tcp 报文头，层层嵌套，每层都包含下一层的所有信息作为其负载 data

以上即计算机网络中的网络嵌套机制模型。

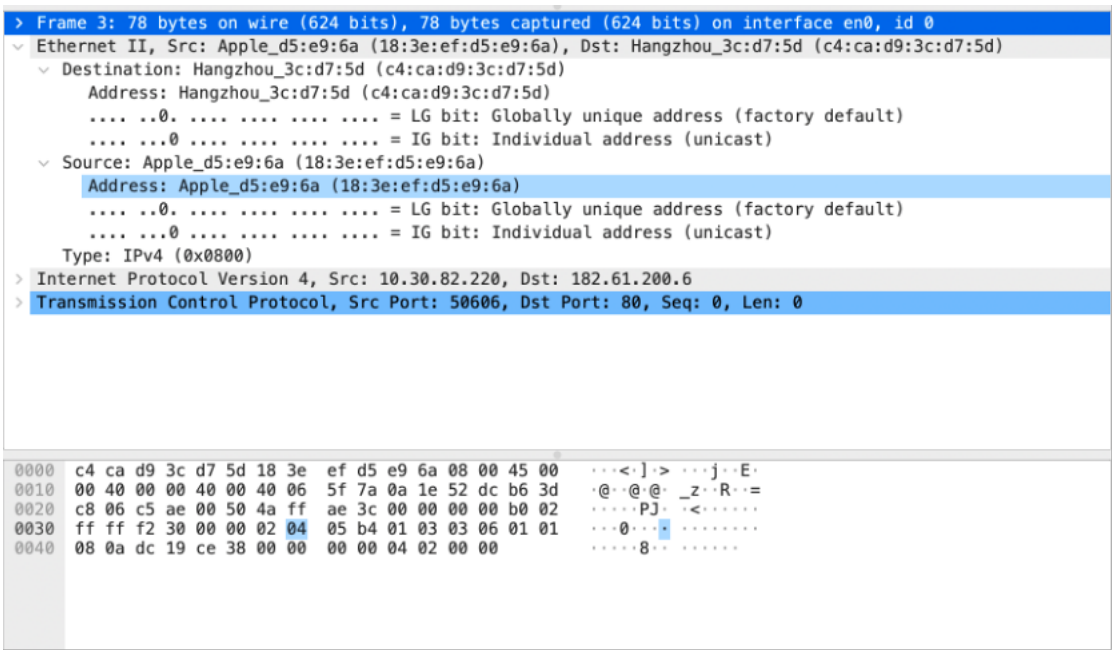
以太网帧格式：

802.3 以太网帧结构

前导码	帧开始符	MAC 目标地址	MAC 源地址	802.1Q 标签 (可选)	以太类型	负载	冗余校验	帧间距
10101010 7个octet	10101011 1个octet	6 octets	6 octets	(4 octets)	2 octets	46—1500 octets	4 octets	12 octets
		64—1522 octets						
72—1530 octets								
84—1542 octets								

图源：维基百科-《以太网帧格式》

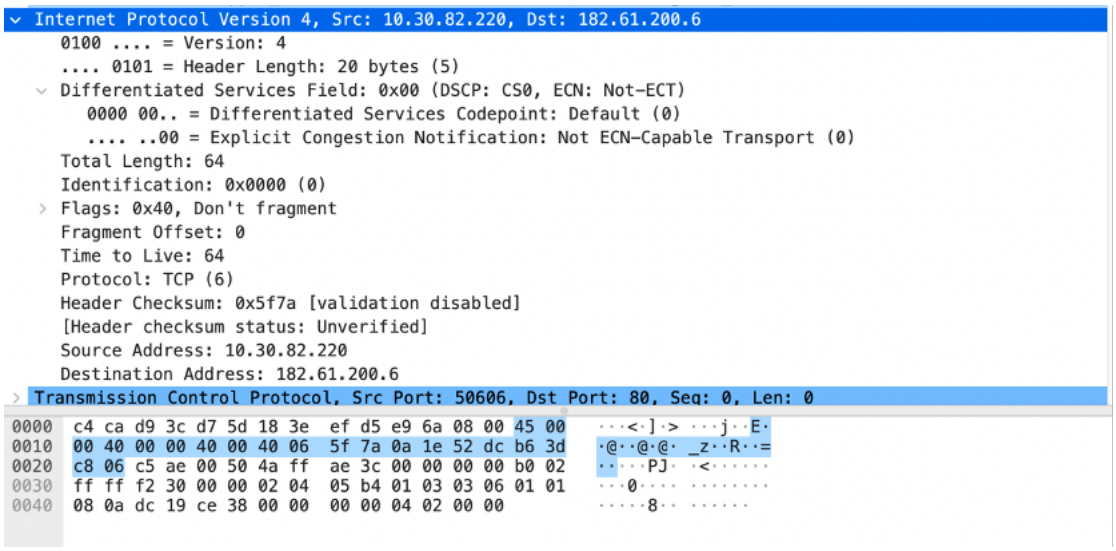
Wireshark 捕捉 packet 的以太网帧格式见下图：
以太网帧的前导码和帧首定位符在链路传输中，在接口处被过滤，因此以太网帧以 MAC 地址开始；
如图，前 6bytes 为 MAC 目的地址，紧接 6bytes 为 MAC 源地址，再接 2bytes 为下一层的协议类型，这里为 0x0800，代表是 IPv4 协议。在这之后的数据即为负载。



IP 报文格式：

位偏移	0—3	4—7	8—13	14—15	16—18	19—31
0	版本	首部长度	区分服务	显式拥塞通告	全长	
32	标识符				标志	分片偏移
64	存活时间		协议		首部检验和	
96	源IP地址					
128	目的IP地址					
160	选项（如首部长度>5）					
160 or 192+	数据					

图源：维基百科—《IPv4》

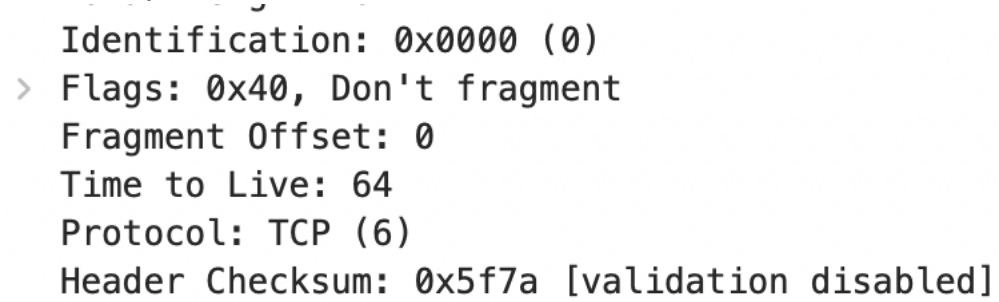


第 1 个 bytes 对应版本控制和 IP 报文长度，前 4bits 为版本，后 4bits 为长度。

第 2 个 byte 对应区分服务领域，包括区分服务和拥塞通告等

第 3-4 个 bytes 对应 IP 数据包的长度，即 IP 报文长度+数据长度

之后 2bytes 为标识符，4bits 标志号，12bits 分片偏移，1byte 存活时间，2byte 传输层协议类型，2bytes 校验和 checksum，如图



TCP 段格式：

图源：维基百科-《传输控制协议》

前 4bytes 分别表示源端口和目的端口，之后 4bytes 序列号，4bytes ACKNumber，后 2bytes 为标志等，包括段长度，保留符，标志符等，后 2bytes 为窗口大小，如下图：

```
Source Port: 50606
Destination Port: 80
[Stream index: 0]
[TCP Segment Len: 0]
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 1258270268
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
1011 .... = Header Length: 44 bytes (11)
Flags: 0x002 (SYN)
Window: 65535
[Calculated window size: 65535]
```

紧接的 4bytes 分别为 checksum 和 urgent pointer；最后为可选项：

```
[Calculated window size: 65535]
Checksum: 0xf230 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (24 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-
[Timestamps]
```

FTP 命令和响应格式：

FTP 中数据的传输分两种模式，一是主动模式，一是被动模式。两种模式的主要区别在于发起连接方的不同，主动模式中是由 Client 主动发送 PORT 地址端口到 Server，发出 TCP 连接请求，Server 接收请求后进行确认后发起 TCP 连接；而被动模式中，有 Server 确认到 Client 发出的 PASV 命令请求后，Server 发送地址端口到 Client 中，由 Client 发起 TCP 连接。但这里不对这两种数据流传输方式进行验证说明。

详细可见 https://blog.csdn.net/hhd1988/article/details/114885609

服务器准备响应： 状态符号（220，530）+响应参数

```
File Transfer Protocol (FTP)
220 Serv-U FTP Server v6.2 for WinSock ready...\r\n
    Response code: Service ready for new user (220)
    Response arg: Serv-U FTP Server v6.2 for WinSock ready...
[Current working directory: ]
```

用户标识认证：

```
File Transfer Protocol (FTP)
USER student\r\n
    Request command: USER
    Request arg: student
[Current working directory: ]
```

```
331 User name okay, need password.\r\n
    Response code: User name okay, need password (331)
    Response arg: User name okay, need password.
[Current working directory: ]
```

```
File Transfer Protocol (FTP)
    PASS [redacted]\r\n
        Request command: PASS
        Request arg: [redacted]
    [Current working directory: ]
```

```
File Transfer Protocol (FTP)
    230 User logged in, proceed.\r\n
        Response code: User logged in, proceed (230)
        Response arg: User logged in, proceed.
    [Current working directory: ]
```

其他命令还有：更换目录，要求文件等等，这里不再展示

```
File Transfer Protocol (FTP)
    SYST\r\n
        Request command: SYST
    [Current working directory: ]
```

2、用侦听解析软件观察 TCP 机制

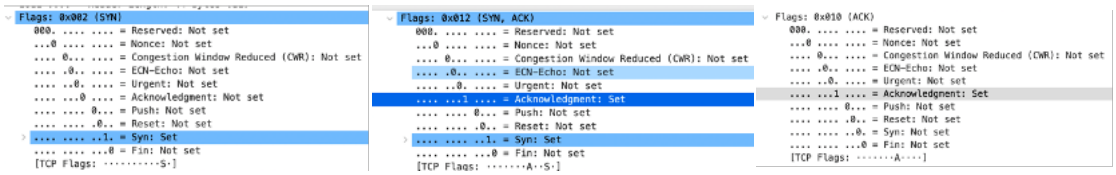
用 Wireshark 侦听并观察 TCP 数据段。观察其建立和撤除连接的过程，观察 段 ID、窗口机制和拥塞控制机制等。将该过程截图在报告中。

建立：

TCP 三次握手：

	Time	Source	Destination	Protocol	Length	Info
3	2.506825	10.30.82.220	182.61.200.6	TCP	78	50606 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3692678712 TSec
4	2.578997	182.61.200.6	10.30.82.220	TCP	78	80 → 50606 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1386 WS=32 SACK_PERM=1
5	2.579131	10.30.82.220	182.61.200.6	TCP	54	50606 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
6	2.579162	10.30.82.220	182.61.200.6	HTTP	131	GET / HTTP/1.1

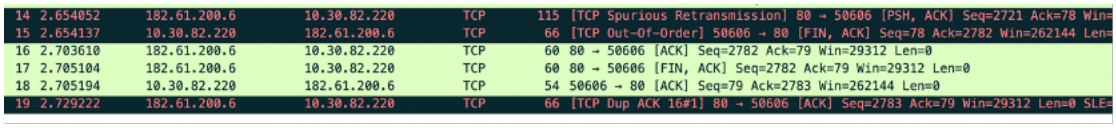
如图，首先原目的地址的 client 发送连接请求，标志符置 SYN=1，序列号 Seq=0；Server 端收到连接请求后返回确认响应，标志符置 SYN=1，ACK=1，置序列号 Seq=0，Ack=1；Client 端接收到确认响应后发送确认连接请求，建立连接，标志符置 ACK=1，序列号置 Seq=1，Ack=1。



撤除连接：

TCP 三次握手撤除连接：

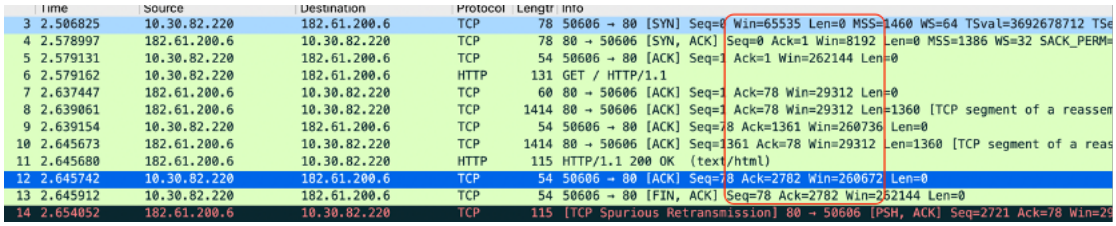
Client 发送关闭 Close 请求，标志符置 FIN，ACK；Server 收到 close 请求，检查 ACK 序列号是否=client 发送的 Seq+1，若是则标志符置 FIN，ACK，序列号 Ack+1，返回响应信息；Client 收到确认关闭信息，置 ACK+1，注意此时 Seq 不变，发送确认关闭数据段包，关闭连接。Server 端收到信息，检查出现 DUP 错误，说明连接结束，断开连接。



窗口机制：

即在 TCP 数据段中，Client 使用 2bytes 窗口大小字节来表达当前可接收数据的最大字节数，Server 接收到信息后，不能响应发送大于 client 窗口大小的数据包；Client 根据当前剩余的可接收量，不断修改窗口大小，当窗口大小为 0 时，说明 Client 无法接受字节，此时 server 端开始一个计时器，停止发送数据并进行一定时间的计时，计时结束时重新进行数据发送，并再次开启窗口机制来控制流量发送。注意：窗口流量控制机制时发送在发送端和接受端两侧的终端流量控制，而不是在链路中的流量控制。

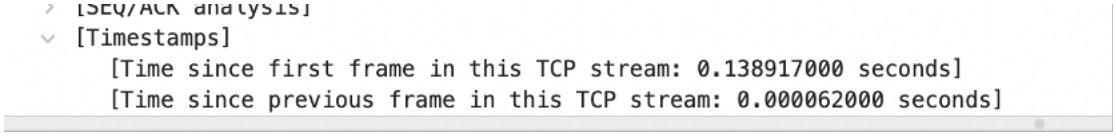
如图：接收端窗口大小在不断的变化，确认可接收数据包大小。



拥塞控制：

拥塞控制主要是处理链路层中数据传输发生的拥塞崩溃等，主要是使用 RTT，来回通信时延，计时器，来确定网络链路的通信状况，若 RTT 高，即网络拥堵，则减小发送量，降低发送速率；若 RTT 小，则增大发送量，提高发送速率。

如图为 TCP 中 RTT 处理信息：



3、用 Libpcap 或 WinPcap 库侦听网络数据

用 Libpcap 或 WinPcap 库侦听网络上的数据流，解析发送方与接收方的 MAC 和 IP 地址，

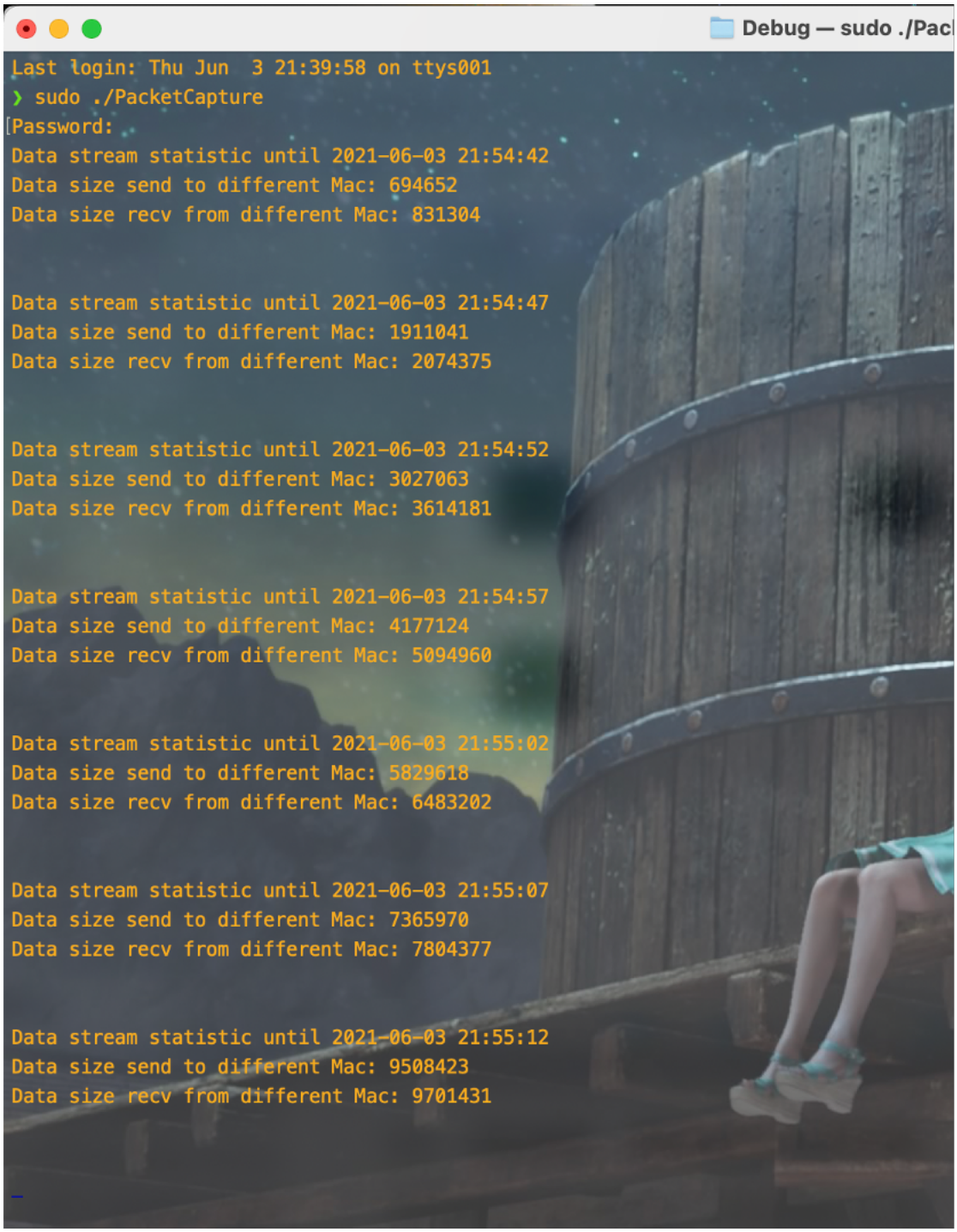
并作记录与统计。程序在文件上输出形如下列 CSV 格式的日志：

时间、源 MAC、源 IP、目标 MAC、目标 IP、帧长度（以逗号间隔）

2015-03-14 13:05:16,60-36-DD-7D-D5-21,192.168.33.1,60-36DD-7D-D5-72,192.168.33.2,1536

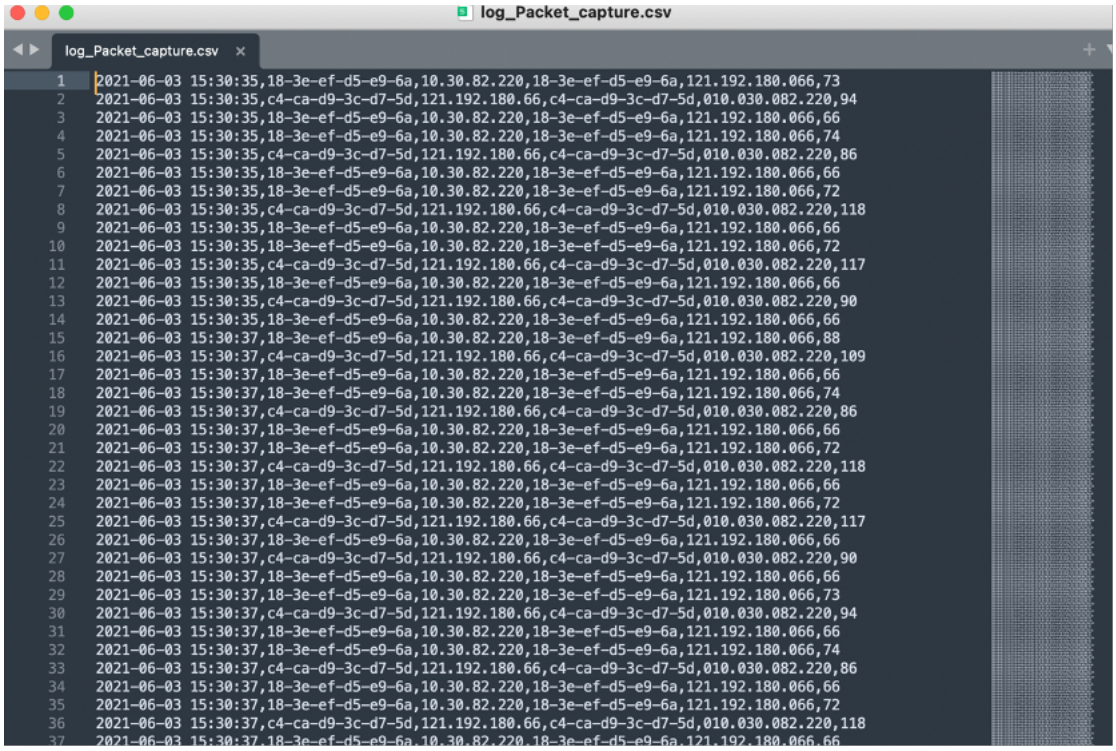
每隔一段时间（如 1 分钟），程序统计来自不同 MAC 和 IP 地址的通信数 据长度，统计发至不同 MAC 和 IP 地址的通信数据长度。

程序运行如下：



程序在运行开始时，对本地网络接口 en0（一般 WIFI 网络接口），进行一个数据统计，每个 5sec 进行一次新的统计，统计来自不同 MAC 地址和 IP 地址的数据包总长度以及发送到不同 MAC 和 IP 的总数据包大小。

记录每一条记录的 CSV 文件如图：



代码核心部分：主要是对 libpcap 捕捉到的数据包的以太网报文，IP 报文，TCP 报文头进行解析，抓取其中的源 MAC，目标 MAC，源 IP，目标 IP 等等字节：如下示：

```
void handle_packets(u_char* arg,const struct pcap_pkthdr* pkthdr,const u_char* packet_data){
    //int *count=(int*)arg;
    char l_time[20];
    strftime(l_time, 20, "%Y-%m-%d %H:%M:%S", localtime(&pkthdr->ts.tv_sec));
    csv_record_form record;
    bcopy(l_time, record.local_time, sizeof(l_time));
    bcopy(packet_data,record.des_mac, sizeof(record.des_mac));
    bcopy(packet_data+6,record.src_mac, sizeof(record.src_mac));
    bcopy(packet_data+30,record.des_ip , sizeof(record.des_ip));
    bcopy(packet_data+26,record.src_ip , sizeof(record.src_ip));
    record.length=pkthdr->caplen;
    log_record(record);
#ifdef DISPLAY_MODE
    printf("Packet num:%d\n",++(*count));
    printf("Packet captured size:%d\n",record.length);
    printf("Packet size:%d\n",pkthdr->len);
    printf("Packet Arrival time:%s\n",record.local_time);
    printf("Packet
        Src_Mac:%02x-%02x-%02x-%02x-%02x-%02x\n",record.src_mac[0],record.src_mac[1],record.src_mac
        [2],record.src_mac[3],record.src_mac[4],record.src_mac[5]);
    printf("Packet
        Src_IP:%d.%d.%d.%d\n",record.src_ip[0],record.src_ip[1],record.src_ip[2],record.src_ip[3]);
    printf("Packet
        Des_Mac:%02x-%02x-%02x-%02x-%02x-%02x\n",record.des_mac[0],record.des_mac[1],record.des_mac
        [2],record.des_mac[3],record.des_mac[4],record.des_mac[5]);
    printf("Packet
        Des_IP:%d.%d.%d.%d\n",record.des_ip[0],record.des_ip[1],record.des_ip[2],record.des_ip[3]);
    printf("\n\n");
#endif
```

4、解析侦听到的网络数据

用 Wireshark 侦听并观察 FTP 数据，分析其用户名密码所在报文的上下文特 征，再总结出提取用户名密码的有效方法。解析协议内容，并作记录与统计。对 用户登录行为进行记

录。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、登录名、口令、成功与否

2015-03-14 13:05:16,60-36-DD-7D-D5-21,192.168.33.1,60-36DD-7D-D5-72,192.168.33.2,student,software,SUCCEED

2015-03-14 13:05:16,60-36-DD-7D-D5-21,192.168.33.1,60-36DD-7D-D5-72,192.168.33.2,student,software1,FAILED

题目简析：使用 wireshark 对 FTP 的命令通信进行学习，并解析 FTP 的用户登陆机制，获取用户的账号，密码，并获取成功失败状态。

FTP 数据包抓包分析：
首先要过滤是否是 FTP 协议，
//判断方法一：对是否包含命令字段进行判断，如客户端请求一般包含字段（SYST，USER，PASS，CWD，PCWD，FEAT，PASV，PORT，MKD，EPRT，DELET，EPSV，LIST 等等），服务器端返回码 3 字节（200：成功，202:命令未执行，220:服务准备就绪，227:进入被动模式，331，332:账号密码，421:服务器不可用，450:文件不可用，500：无效命令，等等）
//判断方法二：对端口进行一个过滤选择，命令控制端口为 21，数据传输端口为 20
//这里选择使用方法二，设置 pcap 过滤器对端口进行过滤，这里只保留端口 21，进行一个命令控制学习
//FTP 命令报文格式，客户端：CMD 参数　　服务器：状态码　参数
//因为 FTP 是基于 TCP 协议，是 TCP 协议的一部分，而一般包含 FTP 的 TCP 协议的报文头长度为 32bytes，新增了 12bytes 可选项描述 tcp 传输中的状态
//因此可以认为 TCP 报文开始后 32bytes 之后的数据即为 FTP 命令请求报文
//判断 FTP 报文结束，即 FTP 固定在 CMD 模式中以\t\n 结束，hex 分别为 0x0d,0x0a;

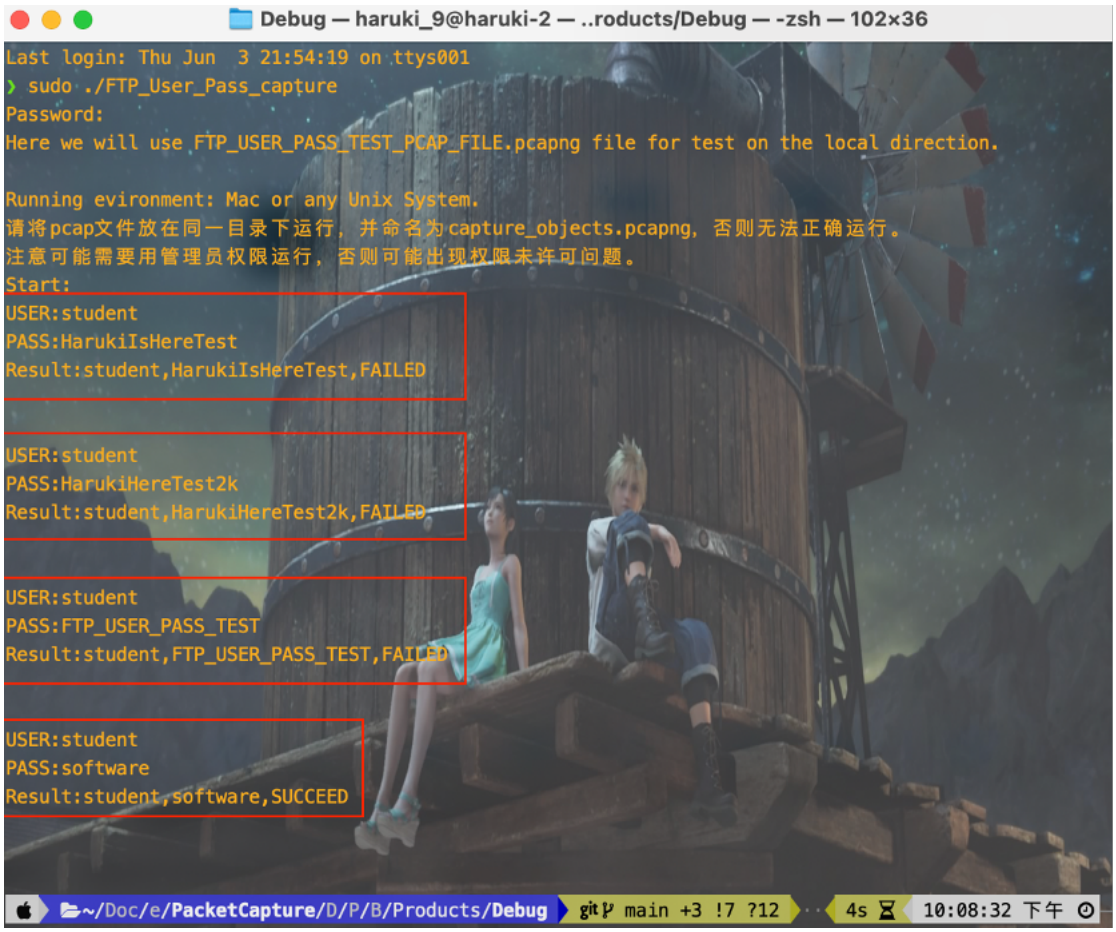
核心代码如下：
解析 FTP 命令数据包中的用户密码信息：

```
bool user_ready=false;
    u_char USER[20];
    bzero(USER, 20);
    if (memcmp((u_char*)"USER", FTP_stream,4)==0) {
        pcap_pkthdr pk;
        const u_char* data=pcap_next(device_fp, &pk);
        if(data==NULL){
            printf("not capture.");
            exit(-1);
        }
        if (memcmp(data+14+20+32, (u_char*)"331", 3)==0) {
            user_ready=true;
            int i=5;
            while
(FTP_stream[i]!=0x0d&&FTP_stream[i+1]!=0x0a) {
                USER[i-5]=FTP_stream[i];
```

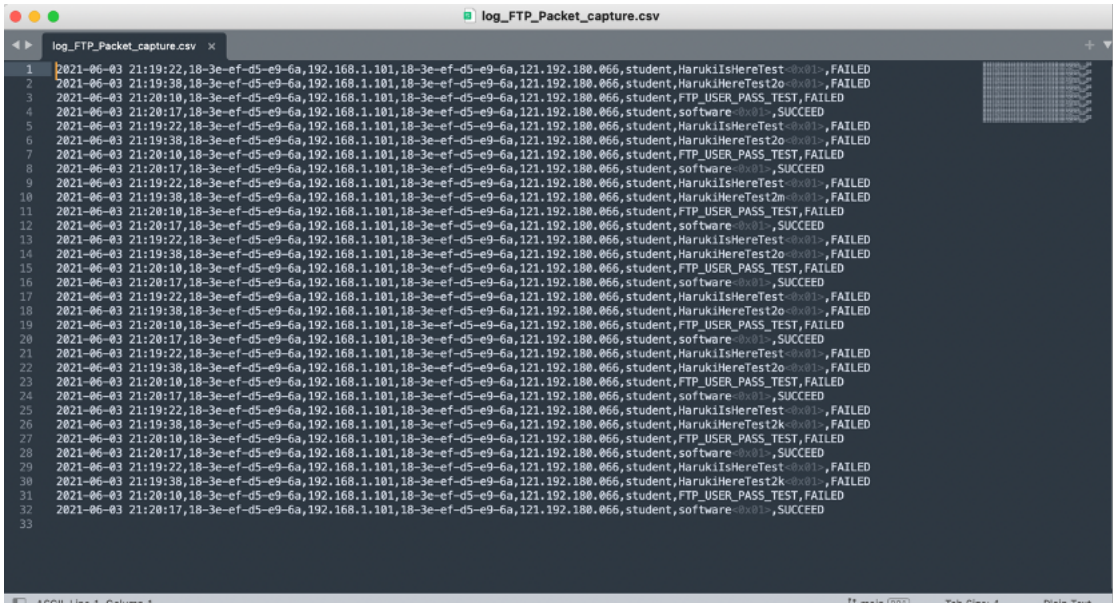
```
                i++;
            }
            printf("USER:%s\n",USER);
        }
    }

    bool pass_ready=false;
    u_char PASS[20];
    pcap_pkthdr pass_pk;//过滤每次server返回状态后client发送确认
tcp协议数据包
    pcap_next(device_fp, &pass_pk);
    const u_char* send_pass=pcap_next(device_fp, &pass_pk);
    if (memcmp(send_pass+14+20+32, (u_char*)"PASS", 4)==0)
{
        pcap_pkthdr ret_pk;
        int pass_len=pass_pk.caplen-2-5-14-20-32;
        memcpy(PASS, send_pass+14+20+32+5, pass_len);
        printf("PASS:%s\n",PASS);
        const u_char* ret=pcap_next(device_fp, &ret_pk);
        if (memcmp("230", ret+14+20+32, 3)==0) {
            pass_ready=true;
        }
}
```

程序运行效果如图：



CSV 文件记录如下图示：



4 实验代码

本次实验的代码已上传于以下代码仓库：代码放置于 Github，地址如下：

Github：https://github.com/Haruki9/Computer-Network_Labs/tree/main

Gitee：https://gitee.com/haruki9/computer-network_-labs

5 实验总结

对 TCP，IP，以太网帧格式有了进一步了解，并加深记忆，对于 FTP 的工作方式也有了更深的了解，对于 FTP 的两个工作模式也终于明白了。