

****

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** 袁佳哲

**学　　号** 11920192203642

**实验时间 2021年**5**月**14**日**

**2021 年** 5 **月** 14 **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. 材料清单上传在代码托管平台上；
5. 在学期末将报告按“CNI-E1-00020190000000-张三”的命名后（其中E1是Experiment 1的缩写），压缩为zip文件，作为附件，以“计算机网络-实验报告-00020190000000-张三”发送至cni21@qq.com。

# 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。

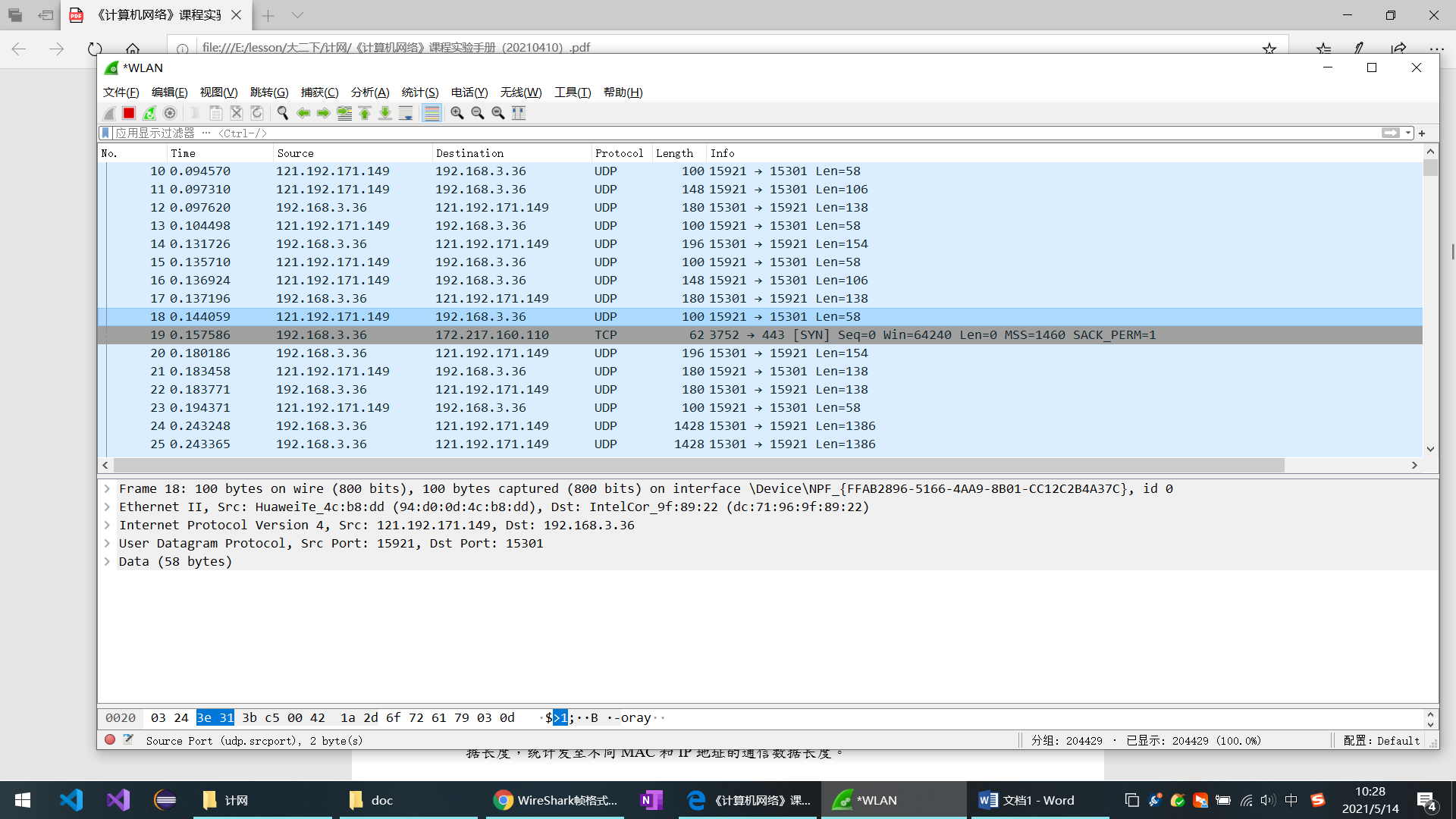
# 实验环境

Windows10、C++

# 实验结果

1、用侦听解析软件观察数据格式

用 Wireshark 或 Omnipeek 等网络侦听软件网络上的数据流，验证理论课讲 授的网络协议层次嵌套，验证帧格式、IP 报文格式、TCP 段格式和 FTP 协议命 令和响应的格式，验证 MAC 地址、IP 地址、TCP 端口等协议地址格式

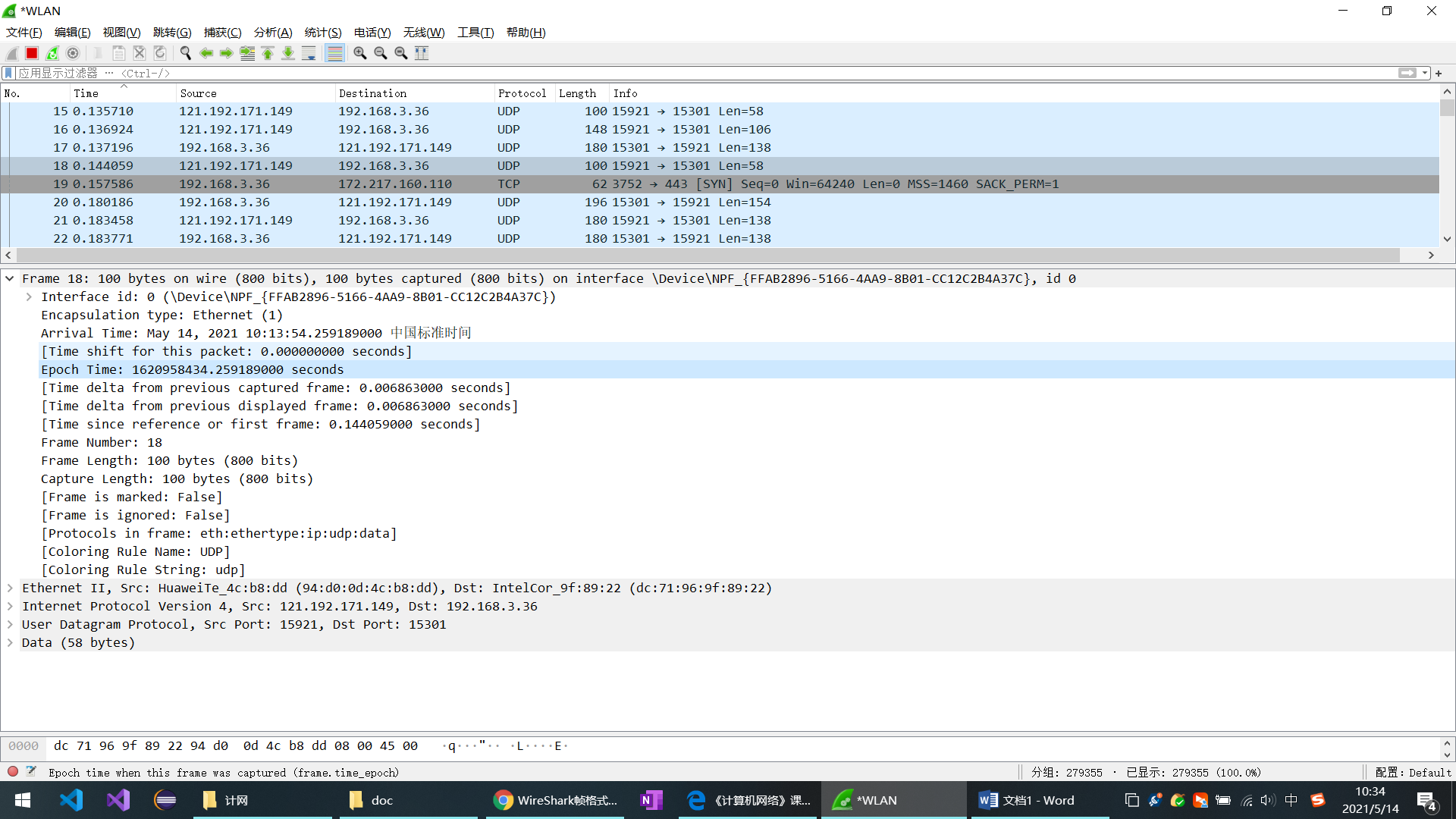


我们选取图中深蓝色的一条作为观察对象

No. Time Source Destination Protocol Length Info

18 0.144059 121.192.171.149 192.168.3.36 UDP 100 略

Frame栏为物理层数据帧概况：



其中重要信息有：

Frame 18: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface \Device\NPF\_{FFAB2896-5166-4AA9-8B01-CC12C2B4A37C}, id 0

解释：第18号帧，线路上有100字节的数据，实际捕获100字节

Arrival Time: May 14, 2021 10:13:54.259189000 中国标准时间

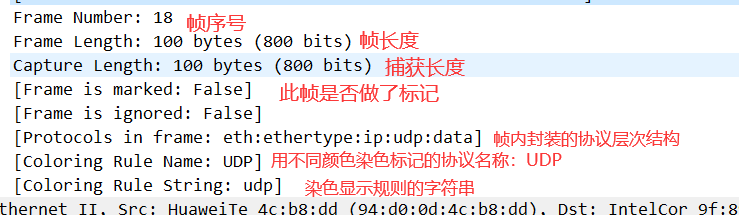
解释：捕获日期与时间

Time delta from previous captured frame: 0.006863000 seconds

解释：与上一帧被捕获的时间间隔

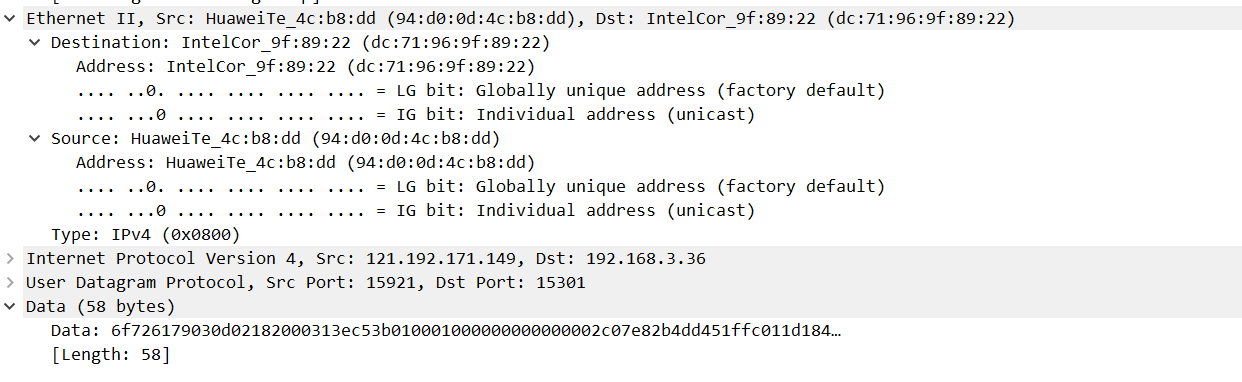
Time since reference or first frame: 0.144059000 seconds

解释：与第一帧的时间间隔



其中，通过[Protocols in frame]一栏，我们可以看到网络协议层次的嵌套

以此帧为例：data->udp->ip->ethertype->eth，层层封装

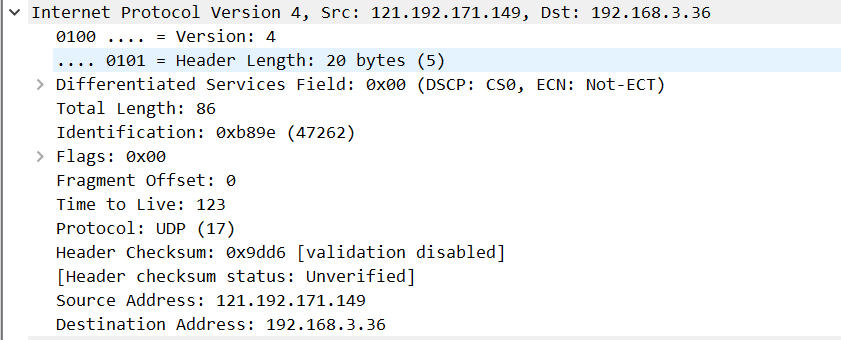


其中，帧格式中目的地址和源地址如图中第一下拉栏的Destination与Source所示，展示目的物理地址和源物理地址

类型如图中的Type：IPv4(0x0800)

数据部分在下面的Data内

然后是IP报文格式：



Internet Protocol Version 4, Src: 121.192.171.149, Dst: 192.168.3.36

解释：互联网协议，源IP地址，目的IP地址

报文组成如下：

Version: 4 版本号

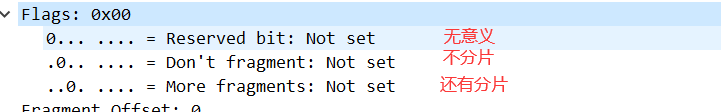
Header length: 20 bytes 报头长度

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) 服务类型

Total Length: 86 报文总长度，单位为字节

Identification: 0xb89e(47262) 标识

Flags: 0x00 分片标志



Fragment Offset: 0 片偏移

Time to Live: 123 生存时间TTL

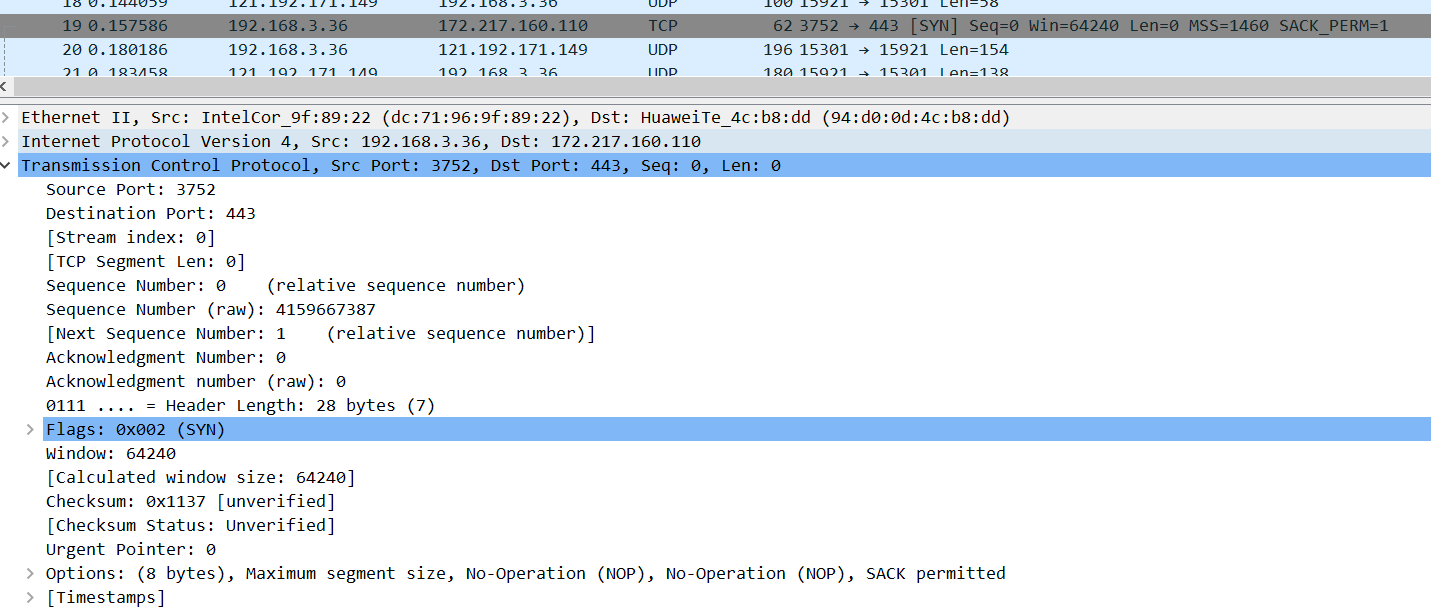
Protocol：UDP(17) 协议类型

Header Checksum：0x9dd6 报头校验和

Source Address：121.192.171.149 源IP地址

Destination Address：192.168.3.36 目的IP地址

接下来是TCP段格式：



源端口号：3752

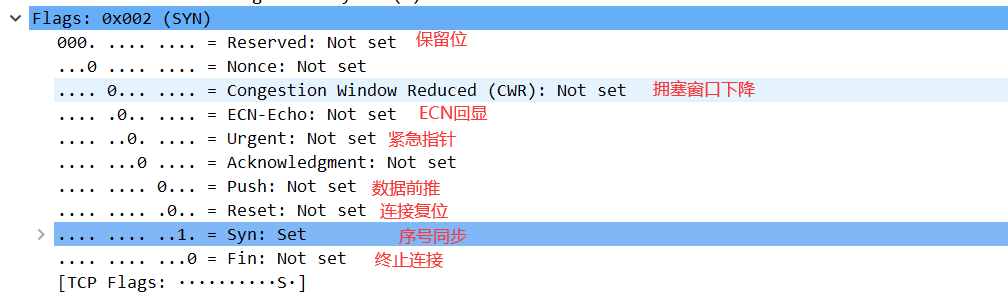
目的端口号：443

报文段序列号：0

确认序列号0

报头长度：28 Bytes（没有用）

标识符：code bits



滑动窗口缓冲区大小：64240

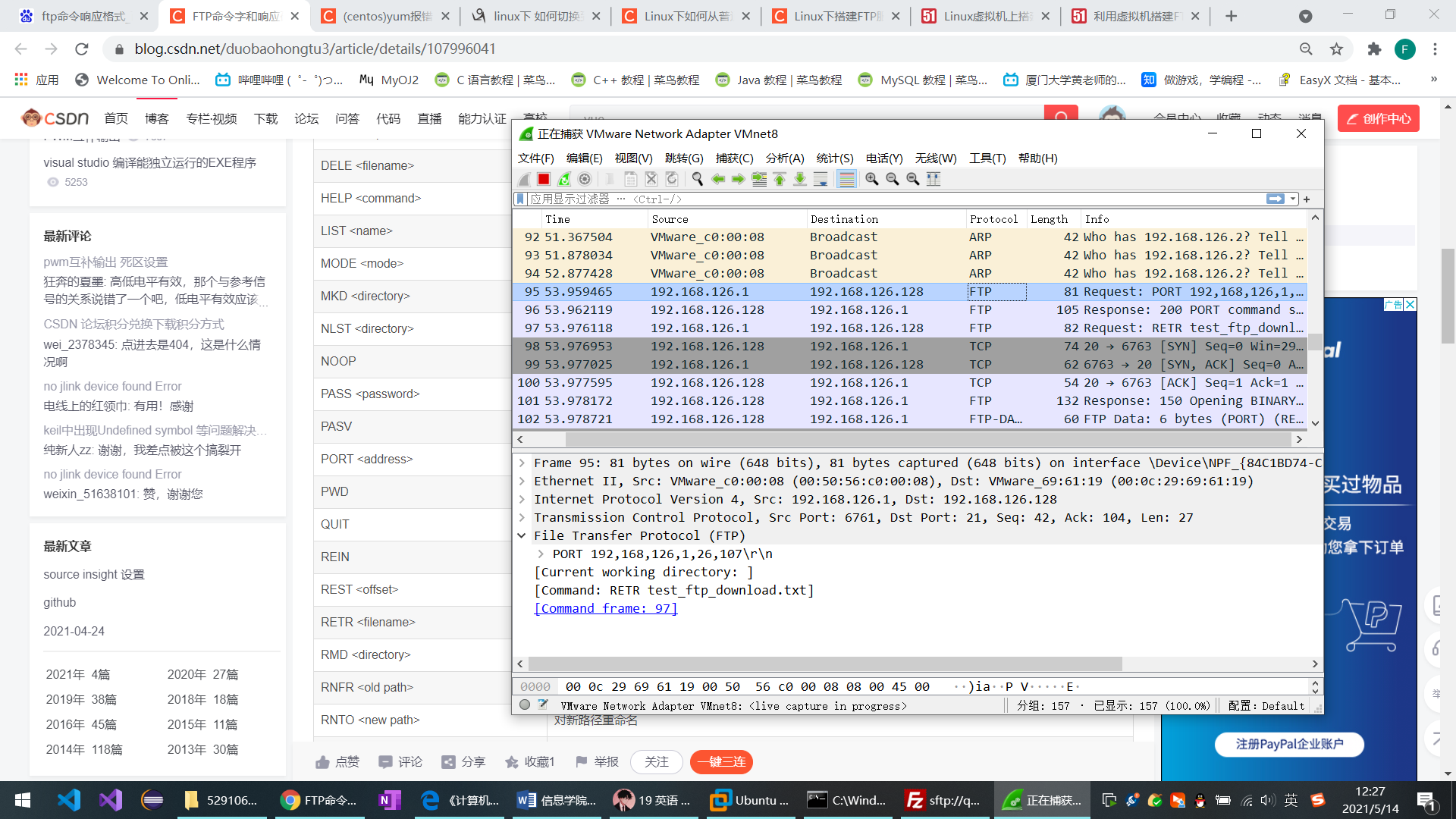
校验和：0x1137

紧急指针：0

选项字段：Options

时间戳：Timestamps

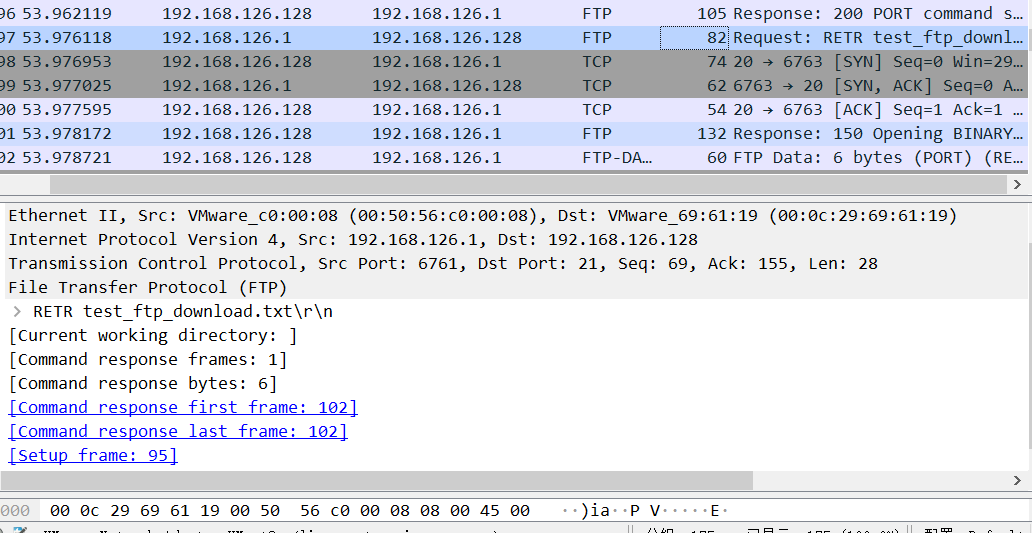
下面是FTP命令码



如图中所示

PORT 192,168,126,1,26,107\r\n

其中的PORT表示IP地址和两字节的端口ID

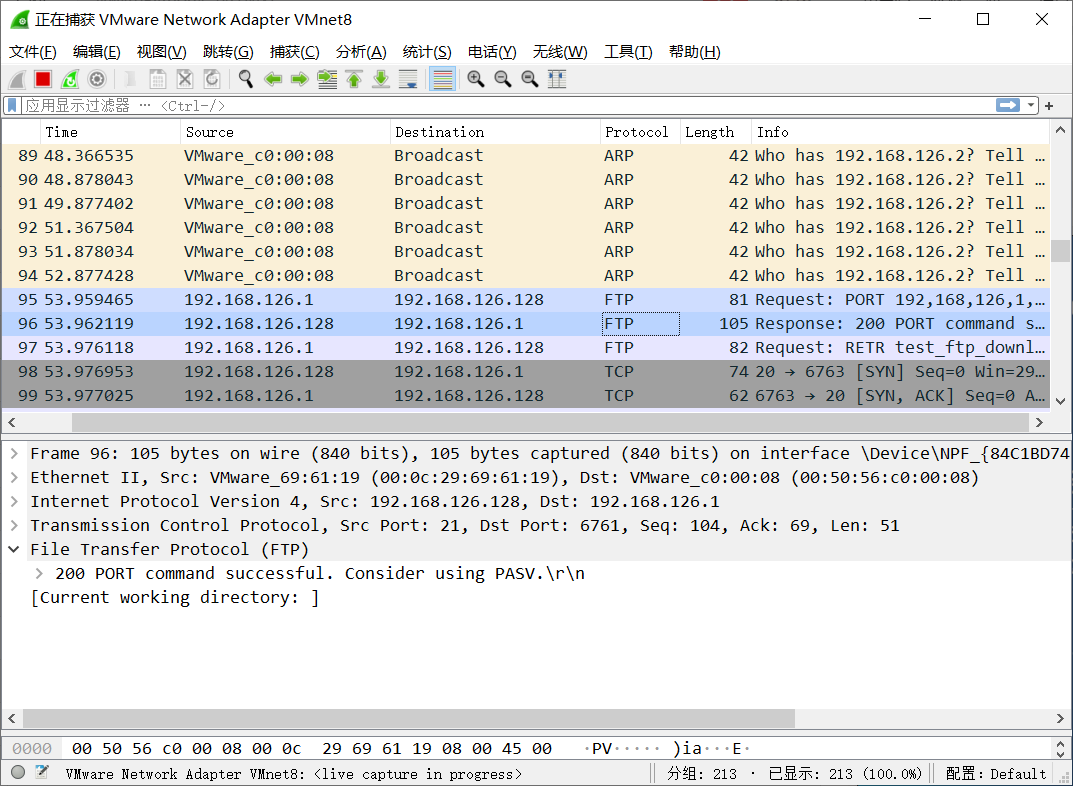


RETR 从服务上获取文件

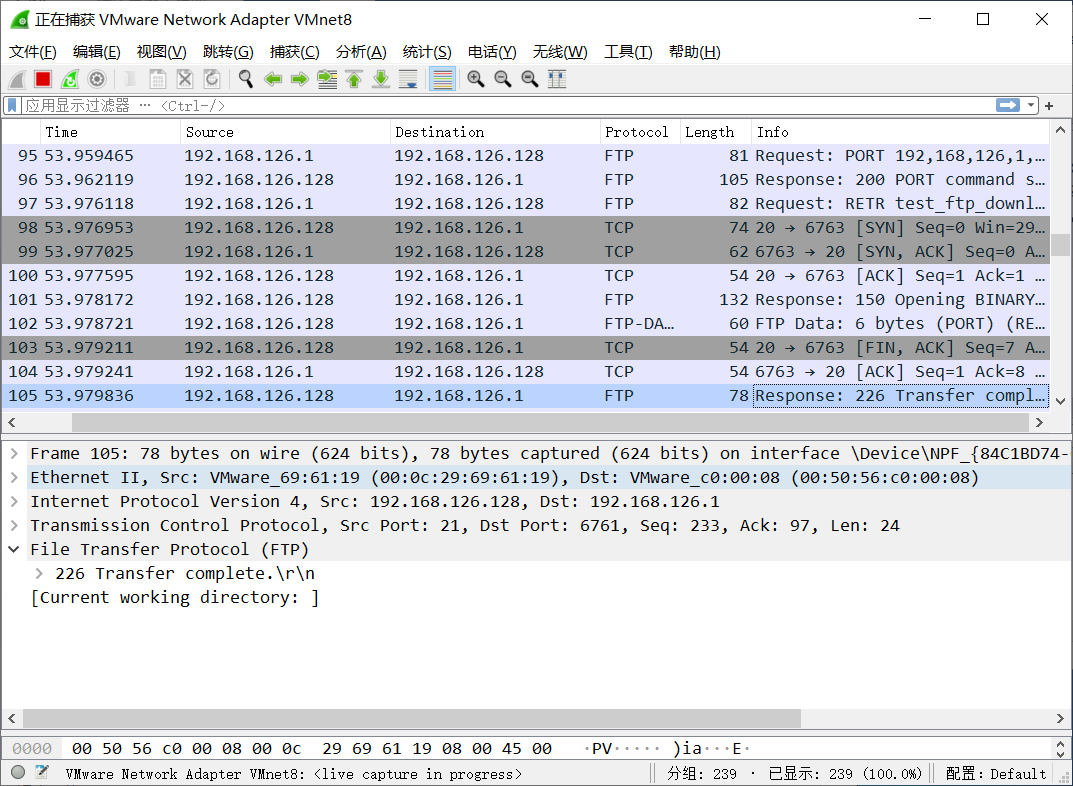
下面是两个响应码



150 打开连接

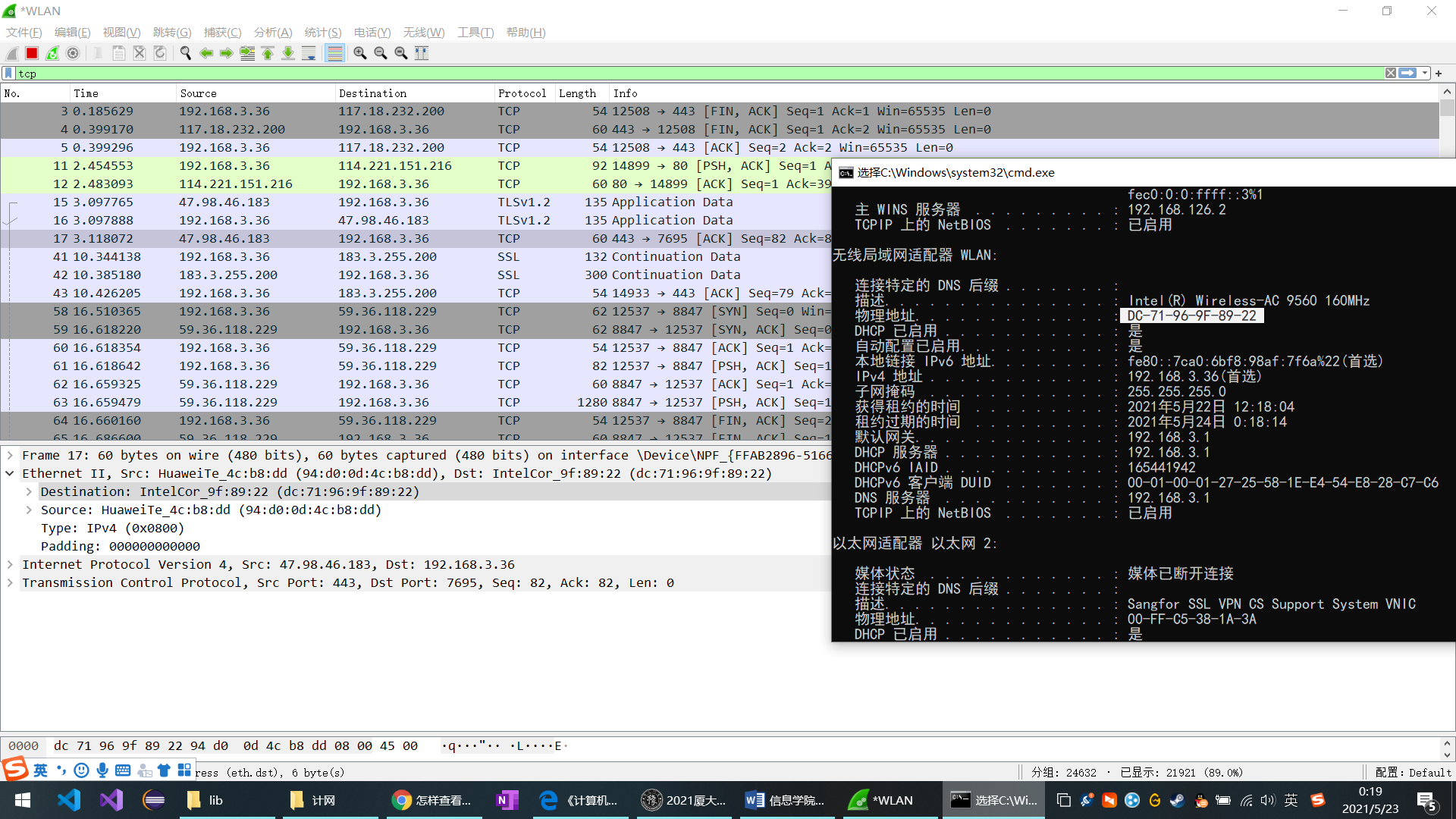


200 成功



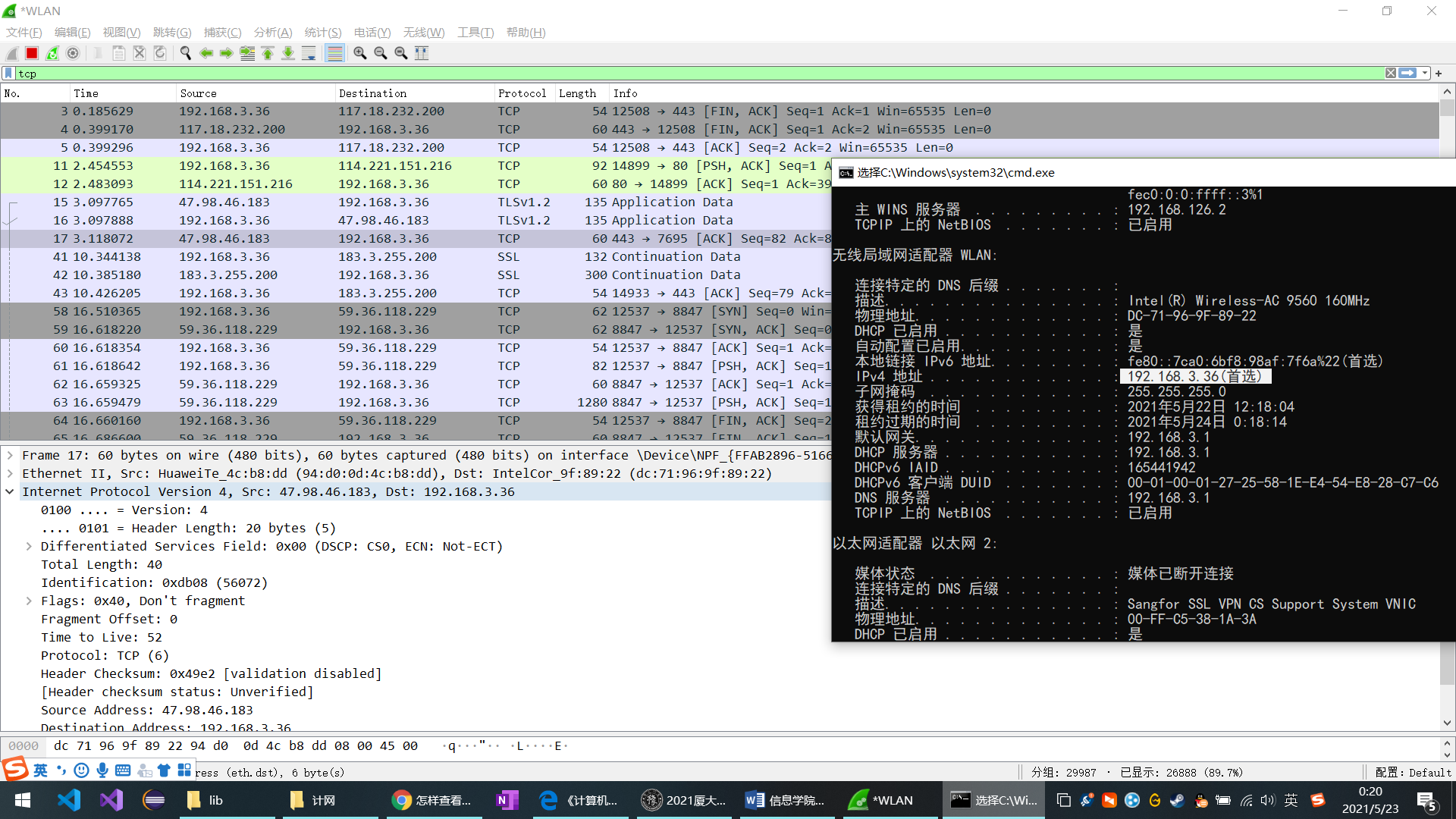
226 结束数据连接

验证Mac地址



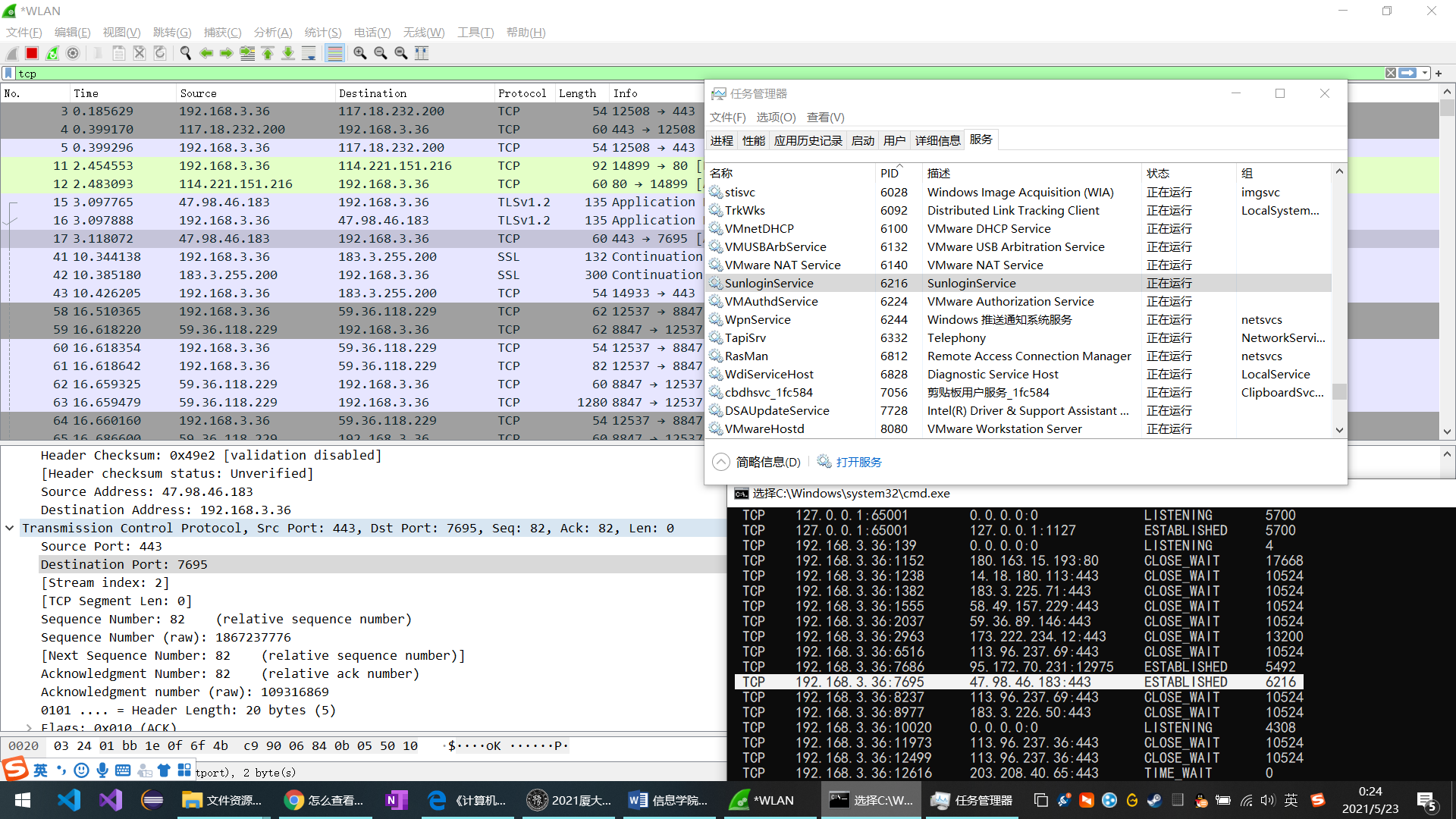
从图中可以看出，wireshark中选中的目标mac地址与本机网卡mac地址一致

验证IP地址



从图中可以看出，目标ip与本机的ip一致

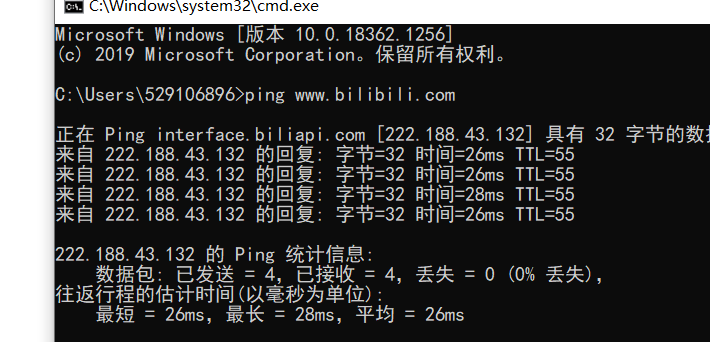
验证TCP端口



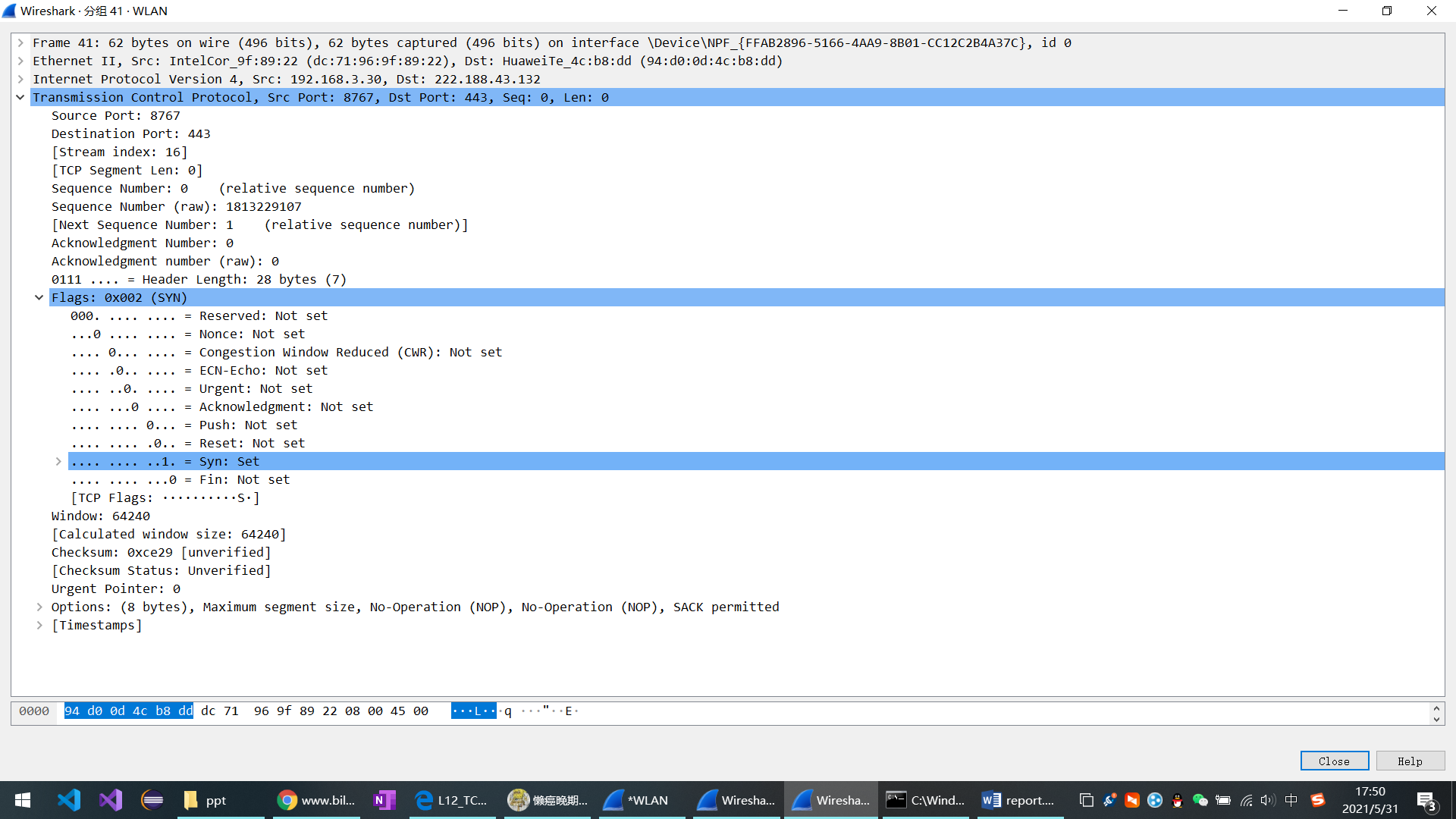
从图中可以看出，某进程端口为7695，是某程序在利用TCP进行通讯

TCP建立连接

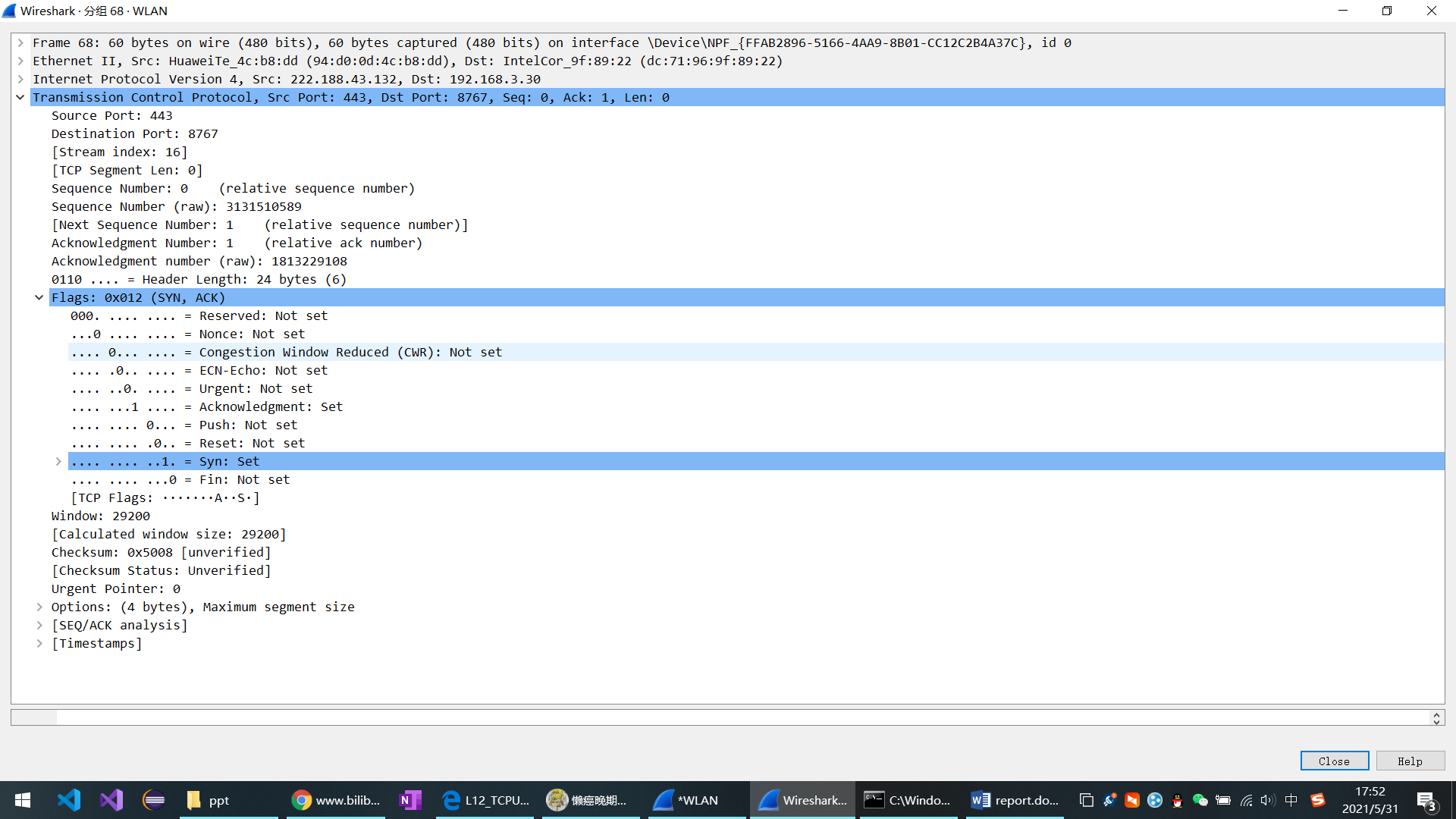
这里我们以B站为例，B站目前ip是222.188.43.132



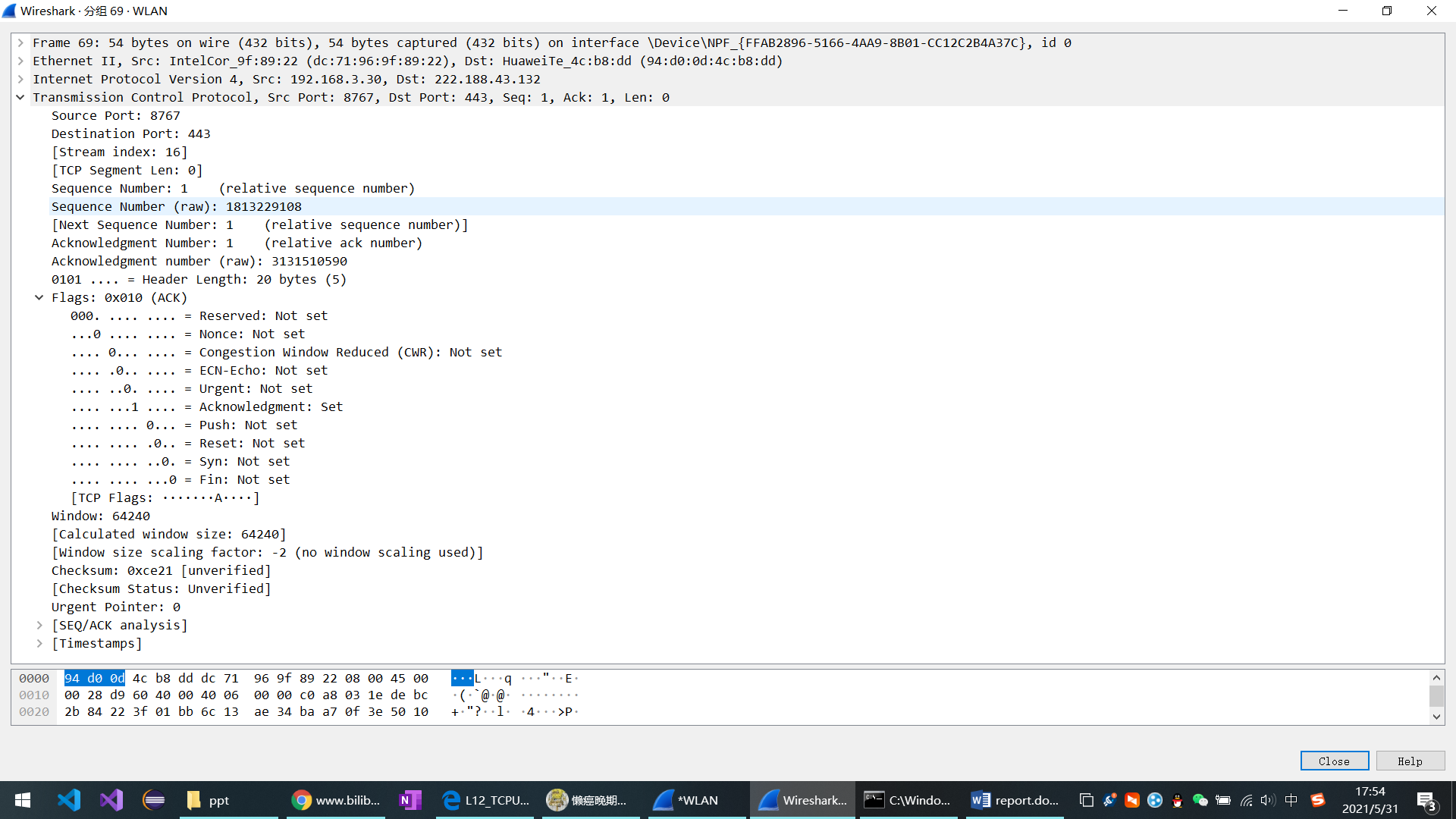
这是第一次握手，我们可以看到，从客户端到服务器发送连接请求报文，首部同部位SYN=1，并选择序号seq=0，表明传送数据时第一个数据字节的序号为0。



这是第二次握手，服务器的TCP收到了客户端发起的连接请求报文，同意后发回确认。在确认报文段中，SYN为1，ACK为1，确认号ack为连接请求报文的seq+1，即1，自己选择的序号seq为0

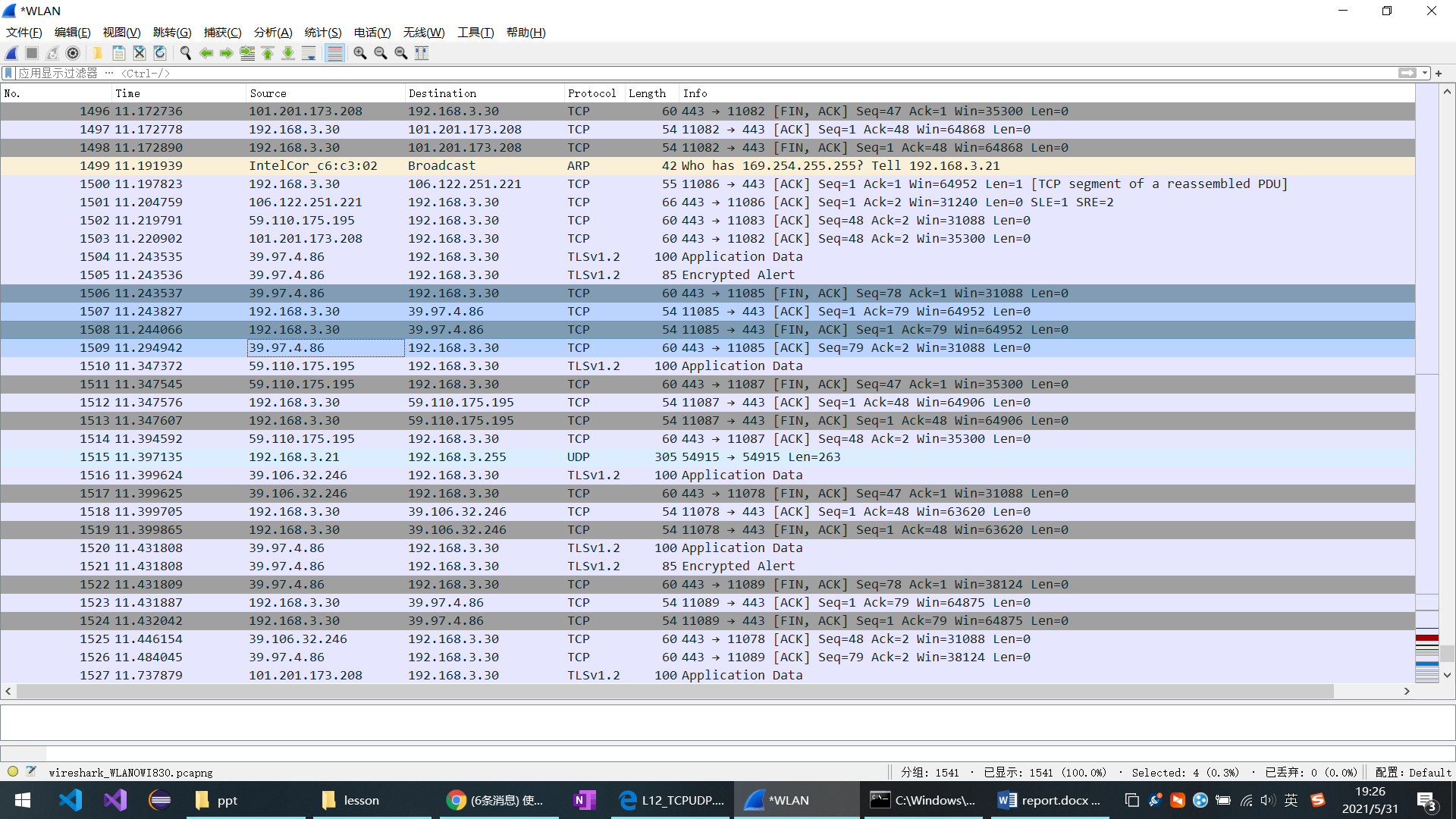


这是第三次握手，客户端收到了服务器发回的确认报文，还需要再次向服务器确认，在这次的确认报文中，ACK=1，序号seq为之前第一次的seq+1，即1，确认号ack为第二次握手的确认报文的序号+1，即1



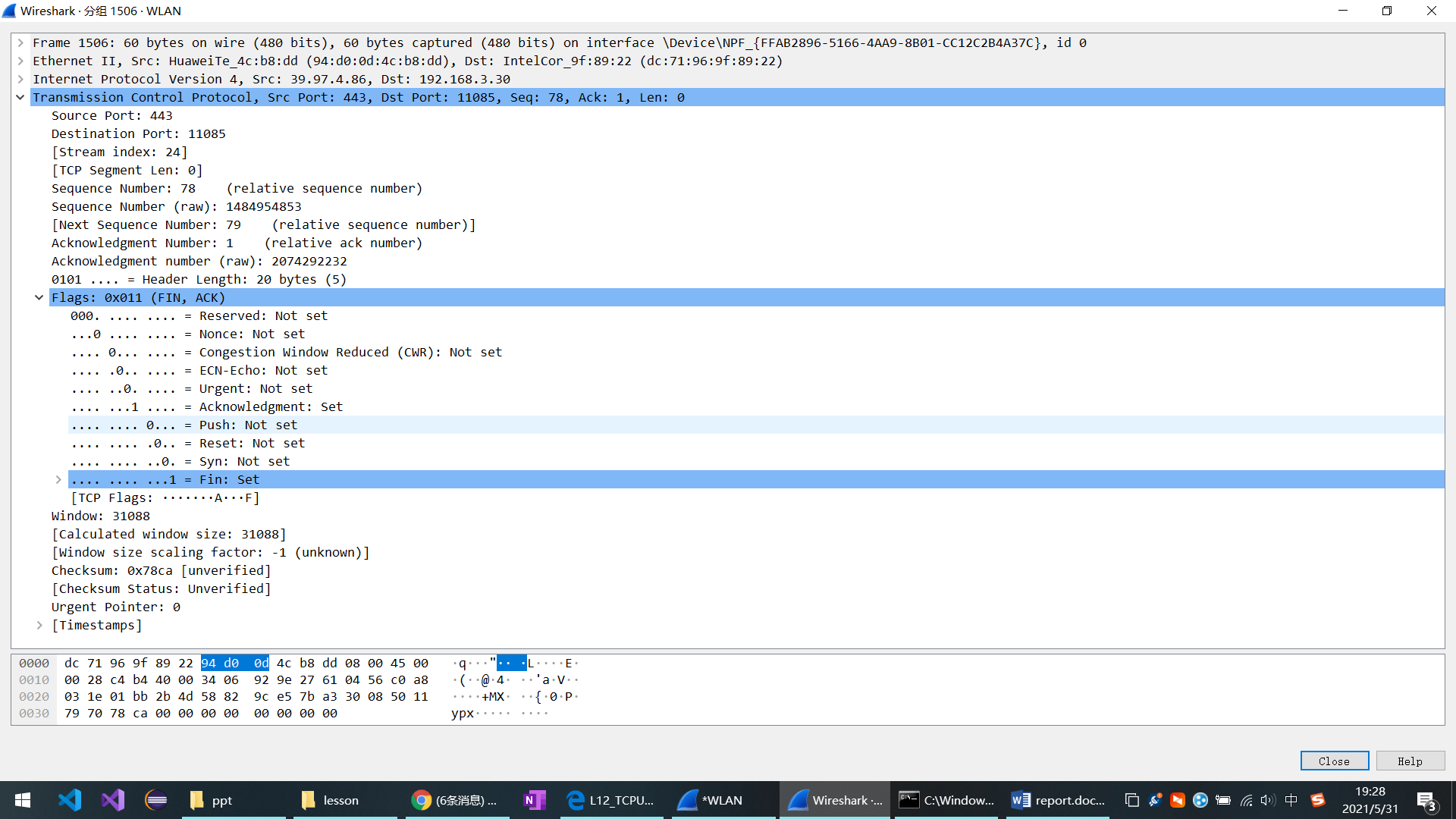
至此，服务器收到客户端的确认报文后，连接建立

TCP撤销连接

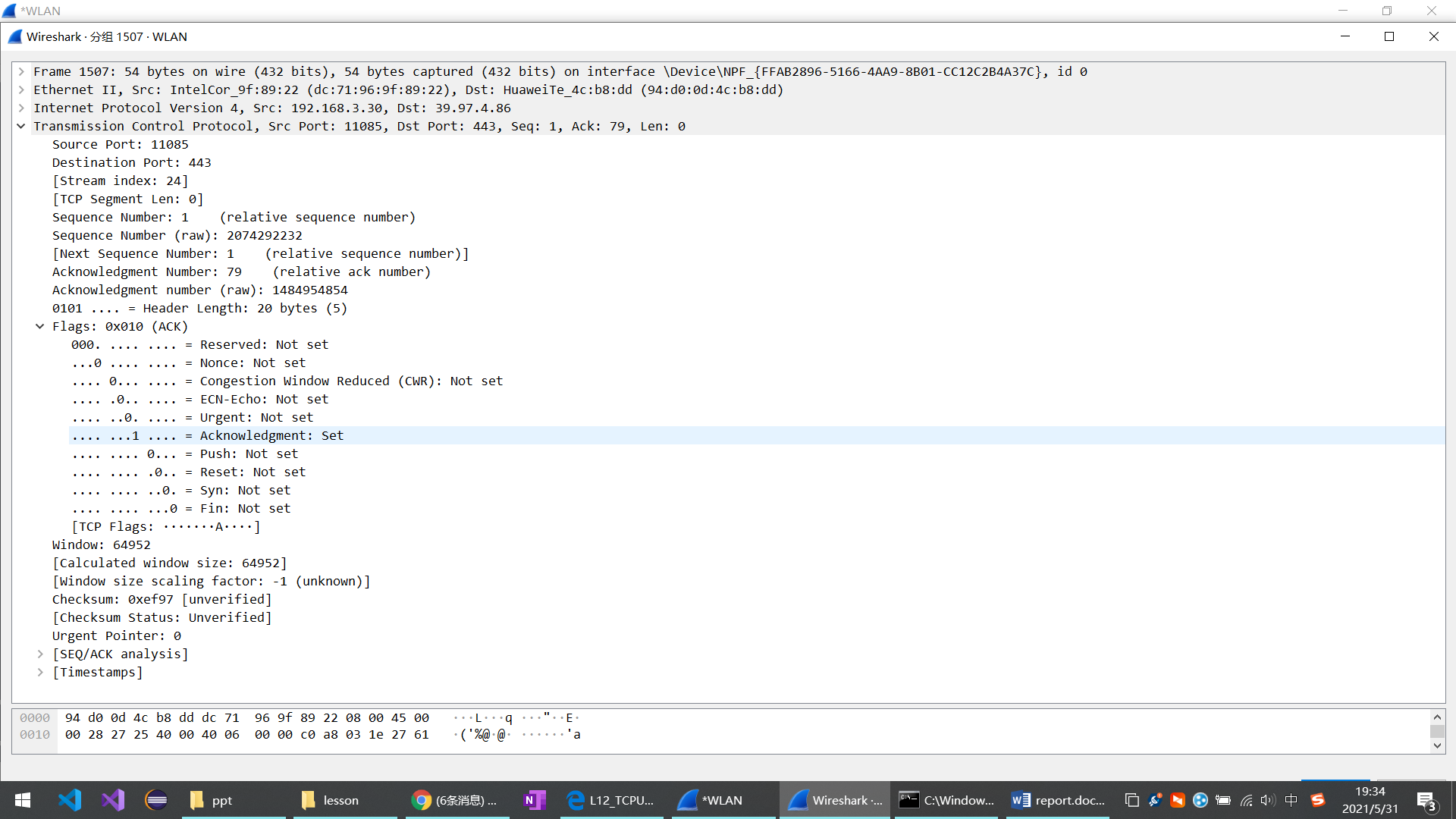


四次挥手完成过程如上图中1506~1509四条所示

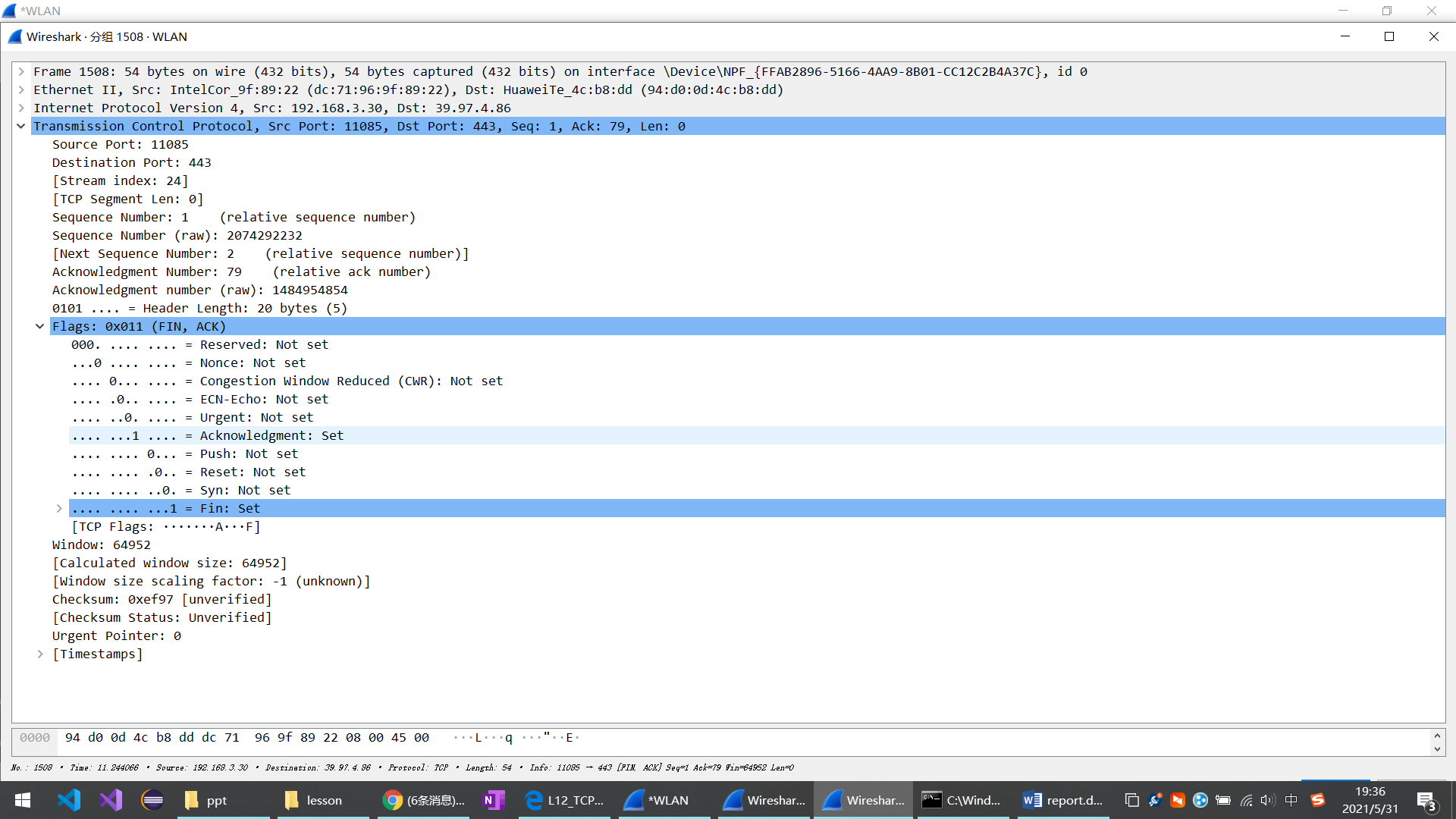
这是第一次挥手，由39.97.4.86主动发起，我们称为A，FIN=1，ACK=1，序号seq=78，确认序号ack=1，等候对方B的确认



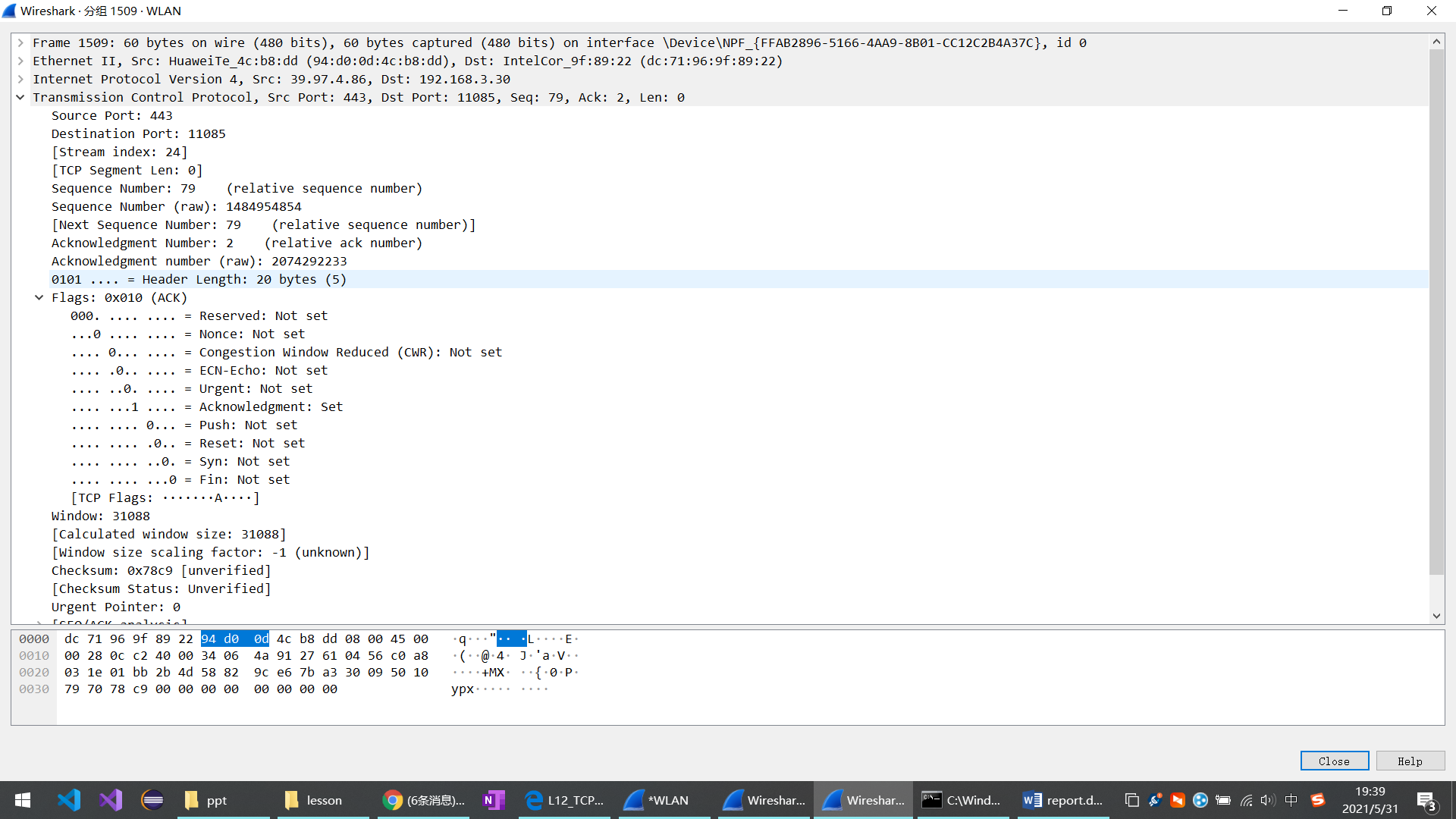
这是第二次挥手，是B的确认报，ACK=1，报文序号seq=1，确认好ack为第一次挥手的seq+1，即ack=79。至此，从A到B的方向的连接就被释放了，TCP处于半关闭状态



这是第三次挥手，从B向A发起，因为B已经没有要向A发送的数据，所以B的应用进程就通知TCP去释放连接。FIN=1，ACK=1，序号seq为1，确认号ack同第二次挥手，等于第一次挥手的seq+1，即ack=79



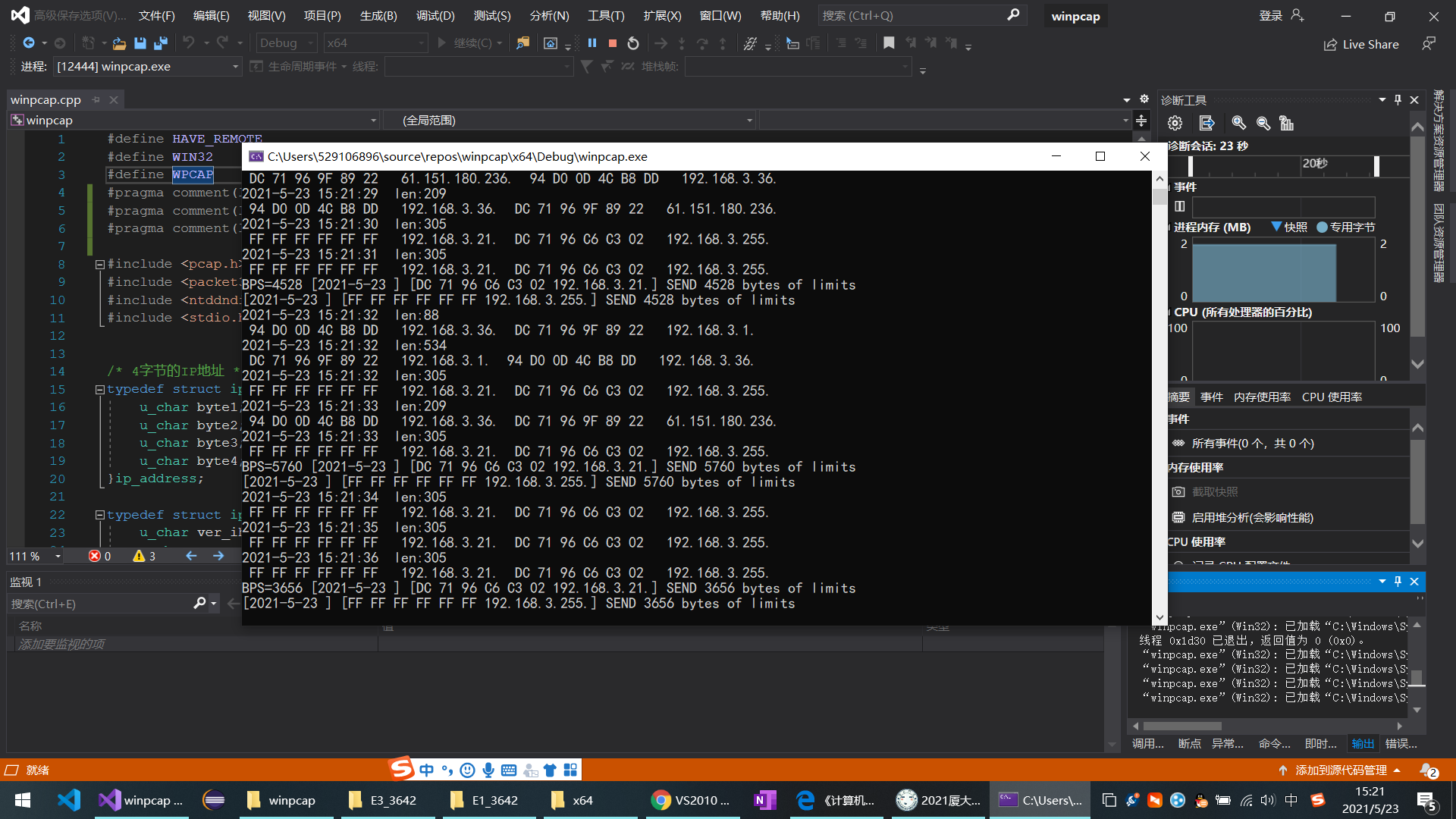
这是第四次挥手，从A向B发回，ACK=1，序号seq为第三次挥手的ack，即seq=79，确认号ack为第三次挥手的seq+1，即ack=2



用 Libpcap 或 WinPcap 库侦听网络数据

程序运行截图如下：

对应src下winpcap.cpp与bin下的winpcap.exe





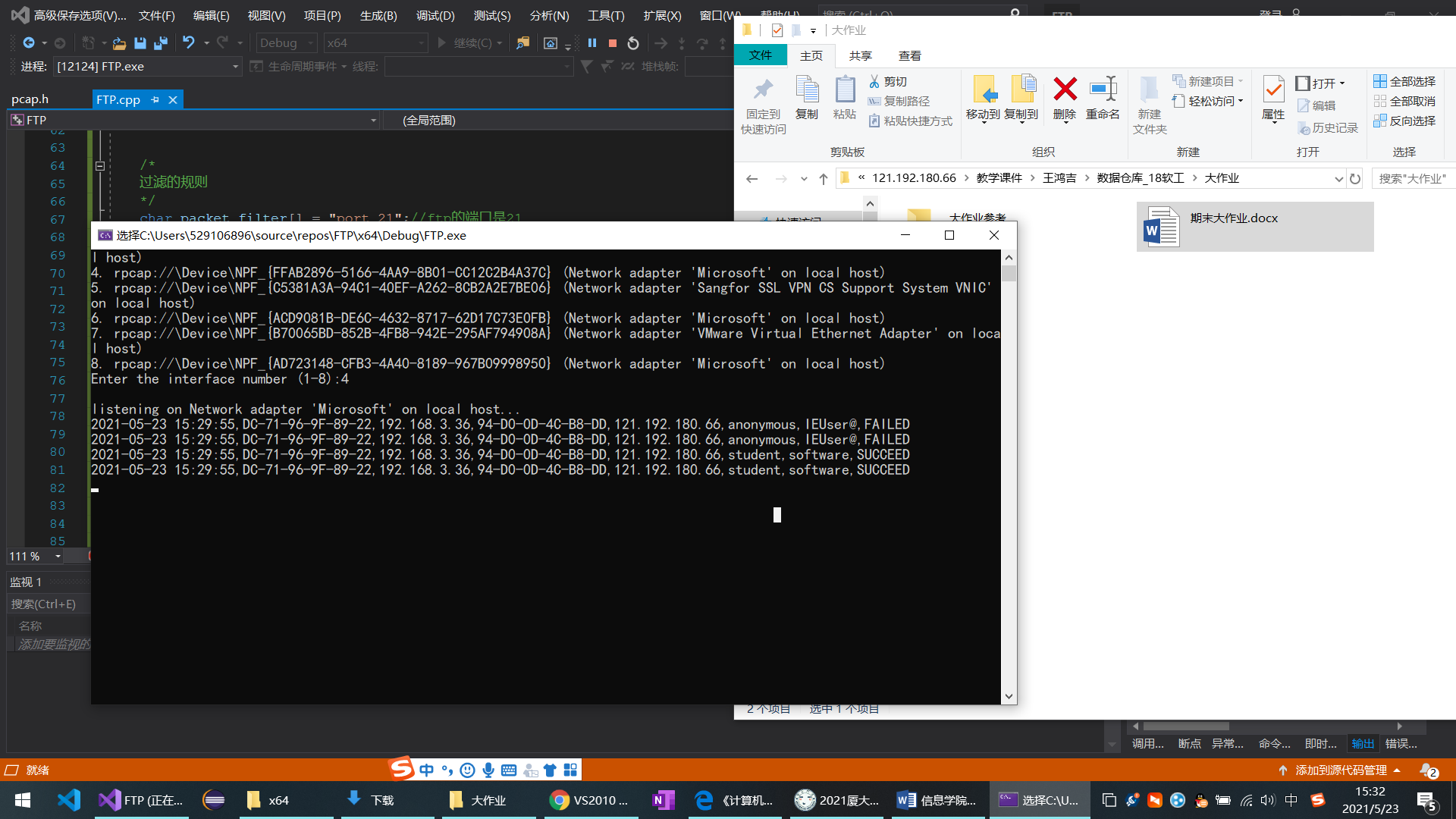
输出CSV文件例：



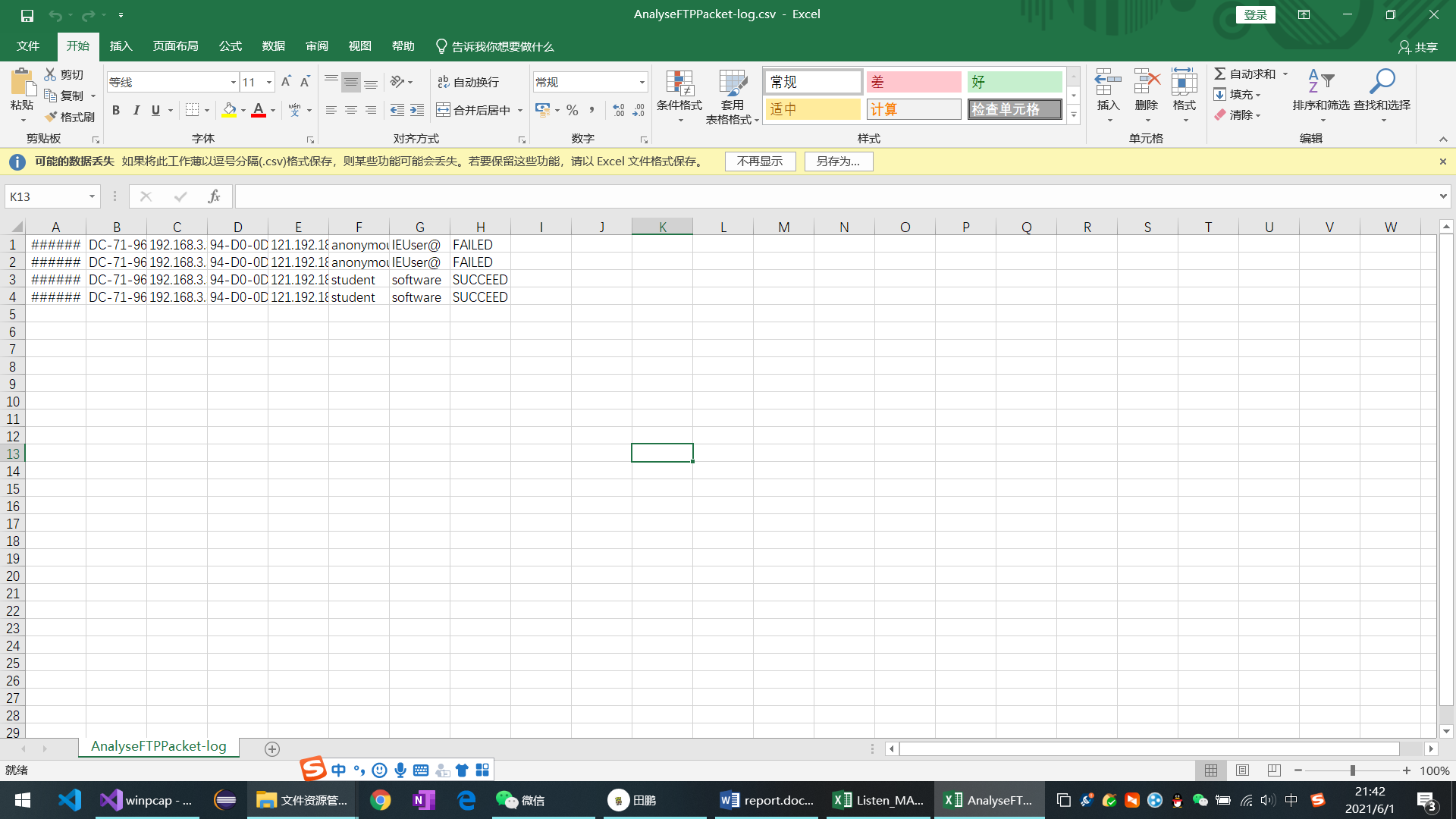
解析侦听到的网络数据

如图，登录学院ftp即可有以下结果

对应src下FTP.cpp与bin下的FTP.exe



输出CSV例：



# 实验代码

本次实验的代码已上传于以下代码仓库：https://gitee.com/yjz6666774/e3642/tree/master/E3\_3642

# 实验总结

1. TCP三次握手四次挥手，真正看到比只背诵和看书印象深刻的多

2. 期间把宿舍路由器搞崩了一次，学会了一些修路由器的方法（？）

3. 学会了在虚拟机的Ubuntu上搭ftp，但只是刚搭的那次能用，回到宿舍虚拟机关机重启后就用不了了。原因未知，还需要加强linux和虚拟机的学习