

# 中山大学计算机学院计算机网络本科生实验报告

---

课程名称：计算机网络

教学班级	专业（方向）	学号	姓名
2班	计算机科学与技术	21307174	刘俊杰

## 一、实验题目

计算机网络 6.1.2

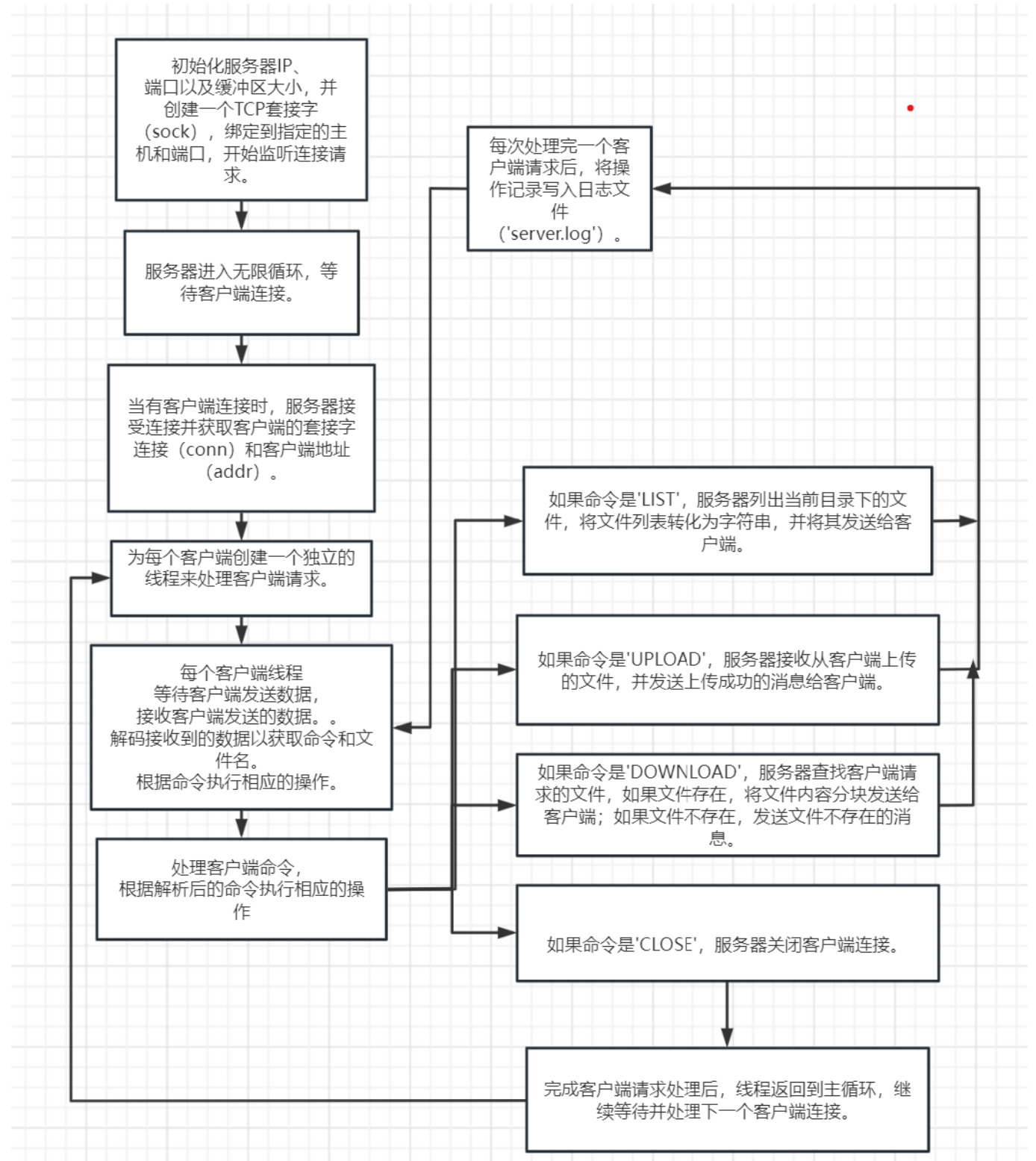
## 二、实验内容

- 1) 在支持5.2实验的2个机器上都启动wireshark 抓包。
- 2) 在客户端、服务器端运行 实验5.2的程序；
- 3) 截图显示传输层相关的连接建立及关闭的过程；观察期间数据传输；

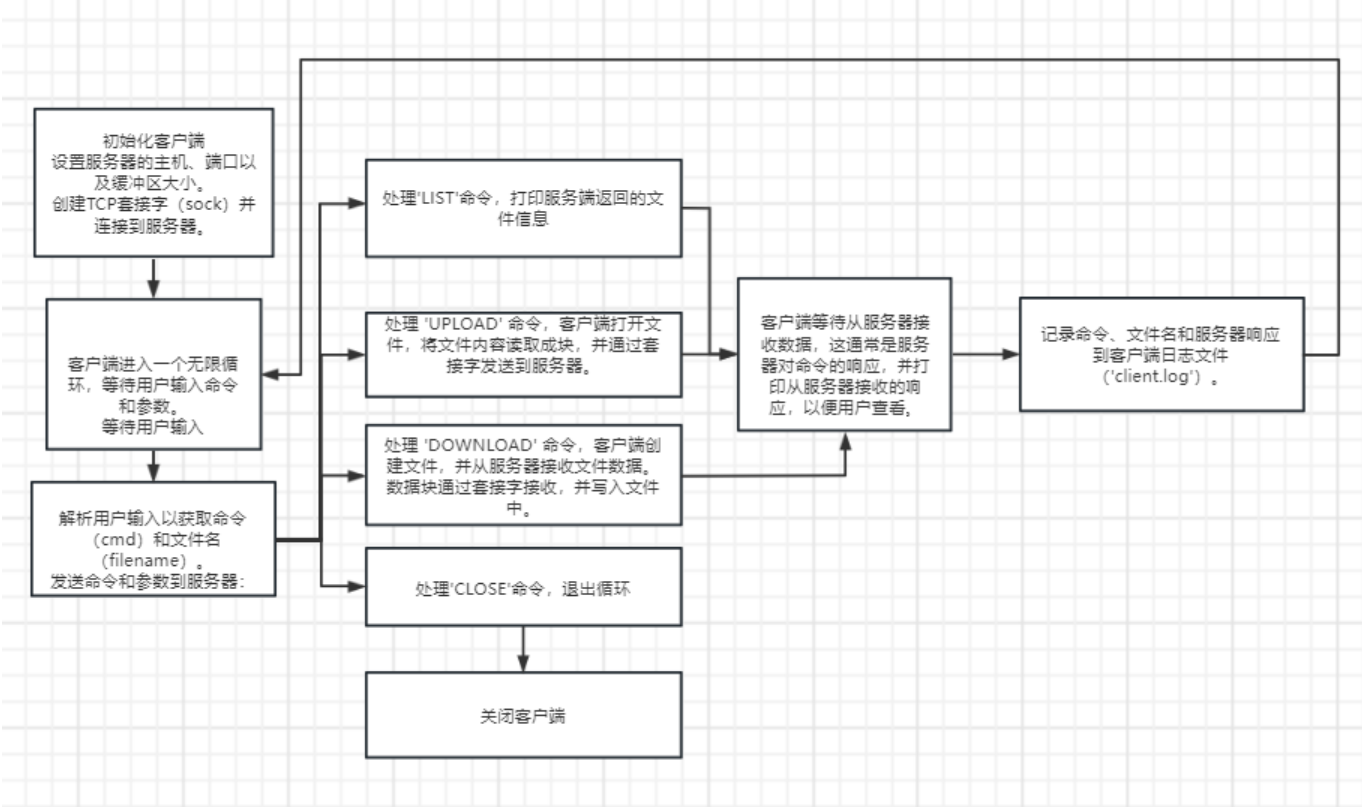
分析并解释以上实验结果

## 三、流程图

服务端:



客户端:



## 四、源代码

服务器端:

```
import socket
import os
import threading
import time
# 设置服务器主机、端口和缓冲区大小
HOST = '172.19.19.113'
PORT = 8000
BUFSIZE = 2048

# 创建TCP Socket对象并绑定IP地址和端口号
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind((HOST, PORT))
sock.listen(5)

print('Server listening on {}:{}'.format(HOST, PORT))

# 处理客户端请求的函数
def serve_client(conn, addr):
    print('Client connected from', addr)
    while True:
        # 等待客户端连接请求
        # 接收客户端发来的数据
        data = conn.recv(BUFSIZE).decode()

        if not data:
            conn.close()
```

```
        return

# 处理客户端命令
cmd, filename = data.split()
if cmd == 'LIST':
    # 列出当前目录下的文件列表
    files = os.listdir('.')
    filelist = '\n'.join(files)
    conn.send(filelist.encode())
elif cmd == 'UPLOAD':
    # 接收上传的文件
    with open(filename, 'wb') as f:
        while True:
            data = conn.recv(BUFSIZE)
            if data == b'end':
                break
            f.write(data)
        conn.send('Upload success'.encode())
elif cmd == 'DOWNLOAD':
    # 发送请求的文件
    if os.path.exists(filename):
        with open(filename, 'rb') as f:
            while True:
                data = f.read(BUFSIZE)
                if not data:
                    conn.sendall(b'end')
                    break
                time.sleep(0.8)
                conn.sendall(data)
            conn.sendall('Download success'.encode())
    else:
        conn.sendall('File not found'.encode())
        with open('server.log', 'a') as f: #写服务器日志
            f.write('The downloading file not found\n\n')
        continue
elif cmd == 'CLOSE':
    #关闭服务器
    time.sleep(1)
    with open('server.log', 'a') as f: #写服务器日志
        f.write('server is close!\n\n')
    conn.close()
# 记录操作日志
else:
    continue

print('{} {} {} {}\n\n'.format(addr, cmd, filename, 'success'))
with open('server.log', 'a') as f: #写服务器日志
    f.write('{} {} {} {}\n\n'.format(cmd, filename, 'success'))

# 启动多个线程来处理客户端连接
while True:
    conn, addr = sock.accept()
```

```
client_handler = threading.Thread(target=serve_client, args=(conn, addr))
client_handler.start()
```

### 客户机端:

```
import socket
import time
# 设置服务器主机、端口和缓冲区大小
HOST = '172.19.19.113'
PORT = 8000
BUFSIZE = 1024

# 创建TCP Socket对象并连接服务器
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((HOST, PORT))

while True:
    # 输入命令和参数
    cmd = input('> ').strip()
    if not cmd:
        continue
    items = cmd.split()
    if len(items) == 1:
        cmd, filename = items[0], None
    elif len(items) == 2:
        cmd, filename = items

    # 发送命令和参数到服务器
    sock.send('{} {}'.format(cmd, filename).encode())
    if cmd == "UPLOAD":#如果执行的为上传命令,将该文件内容发送给服务器
        with open(filename, 'rb') as f:
            while True:
                data = f.read(BUFSIZE)
                if not data:
                    sock.sendall(b'end')
                    break
                time.sleep(0.8)
                sock.sendall(data)
    if cmd=="DOWNLOAD":
        # 接收上传的文件
        with open(filename, 'wb') as f:
            while True:
                data = sock.recv(BUFSIZE)
                if data == b'end':
                    break
                f.write(data)
    if cmd=="CLOSE":
        break
    # 接收服务器返回的数据
    data = sock.recv(BUFSIZE).decode()
    print("-----")
```

```
print("RESPONSE FROM SERVER:")
print(data)
print("-----")
# 记录操作日志
with open('client.log', 'a') as f:
    f.write('{} {} \n{}\n'.format(cmd, filename, data))

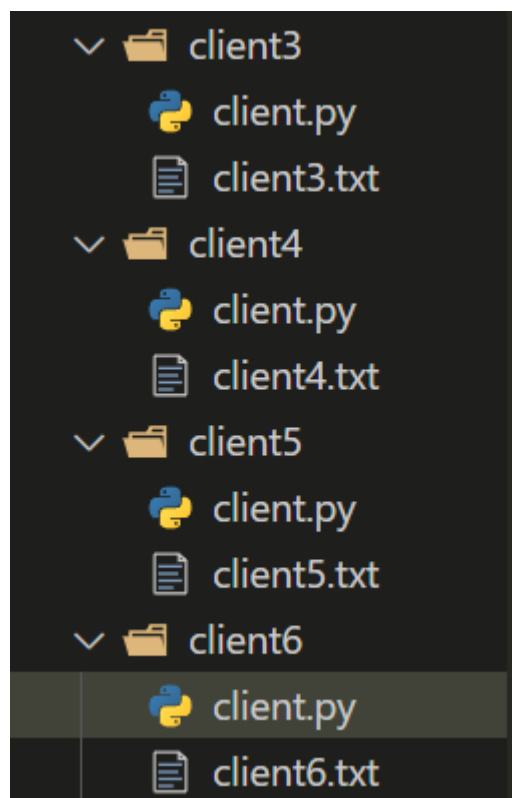
sock.close()
```

## 五、测试运行

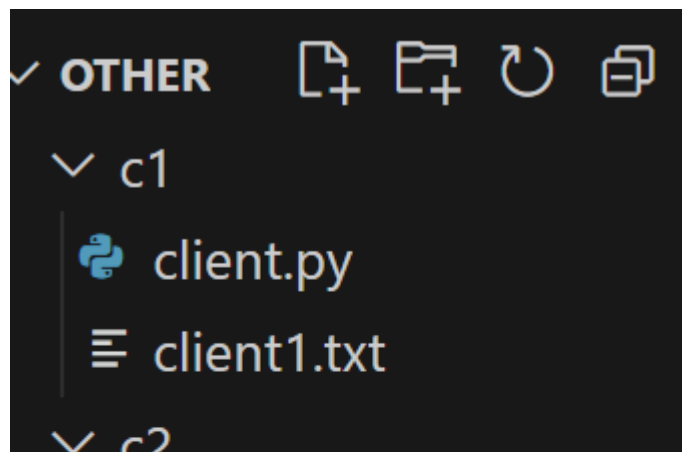
A 机器IP:192.168.43.227

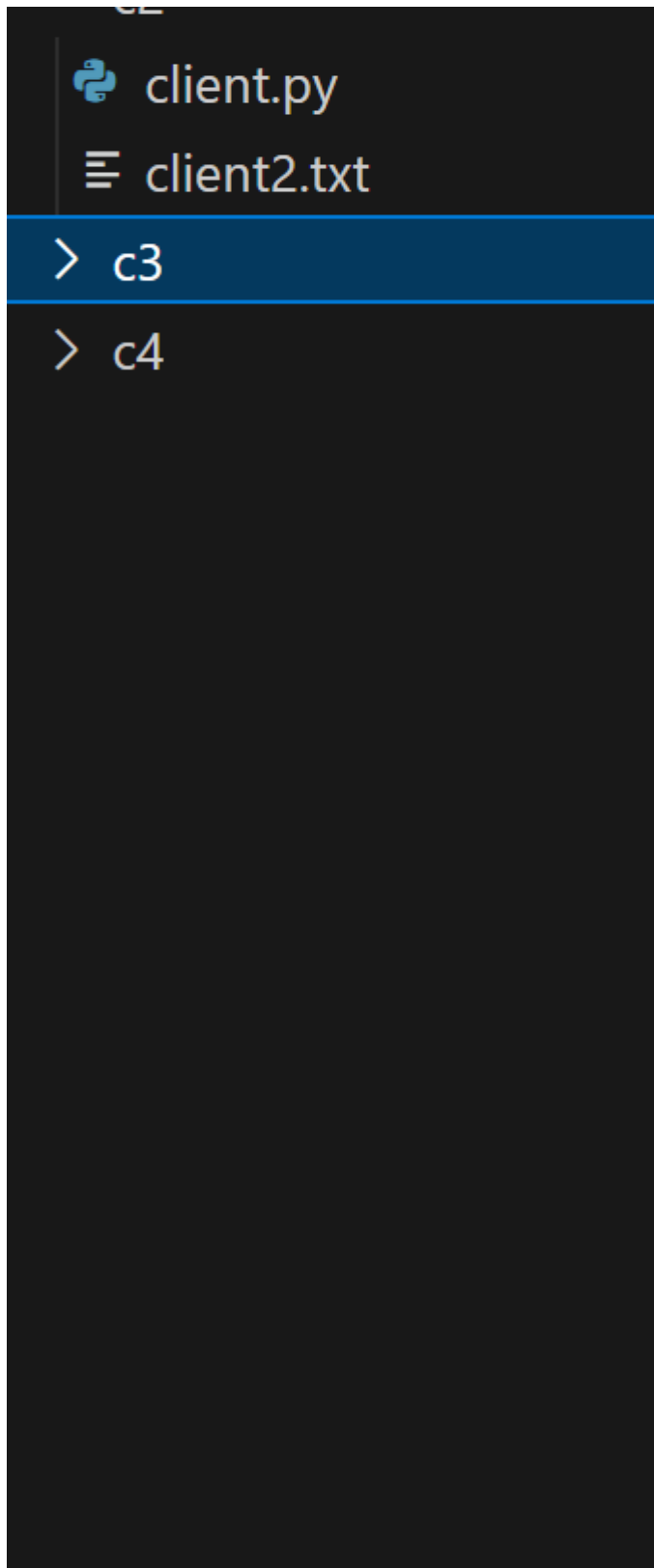
B 机器IP:192.168.43.152

B机器4个客户端文件夹内容:

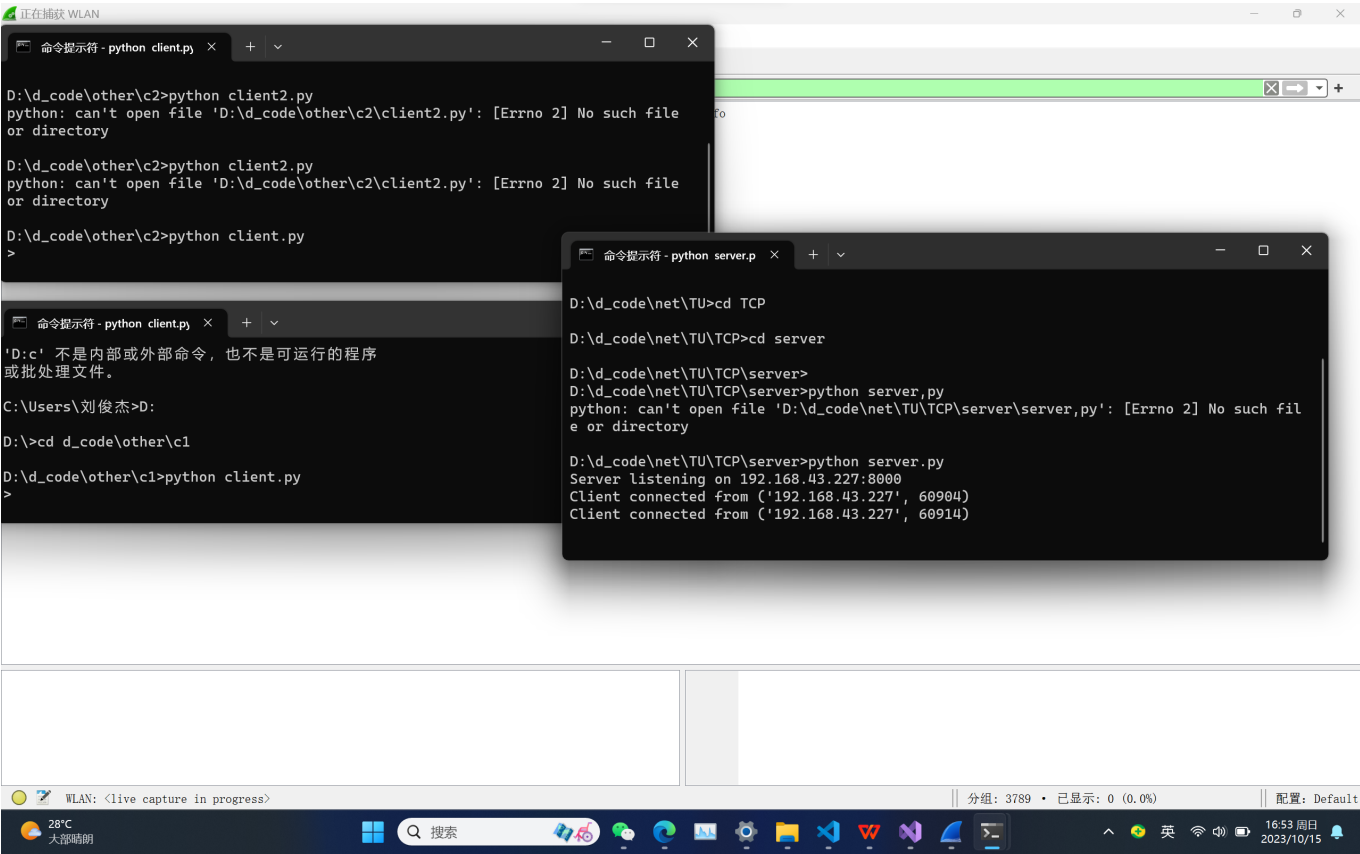


A机器4个客户端文件夹内容:



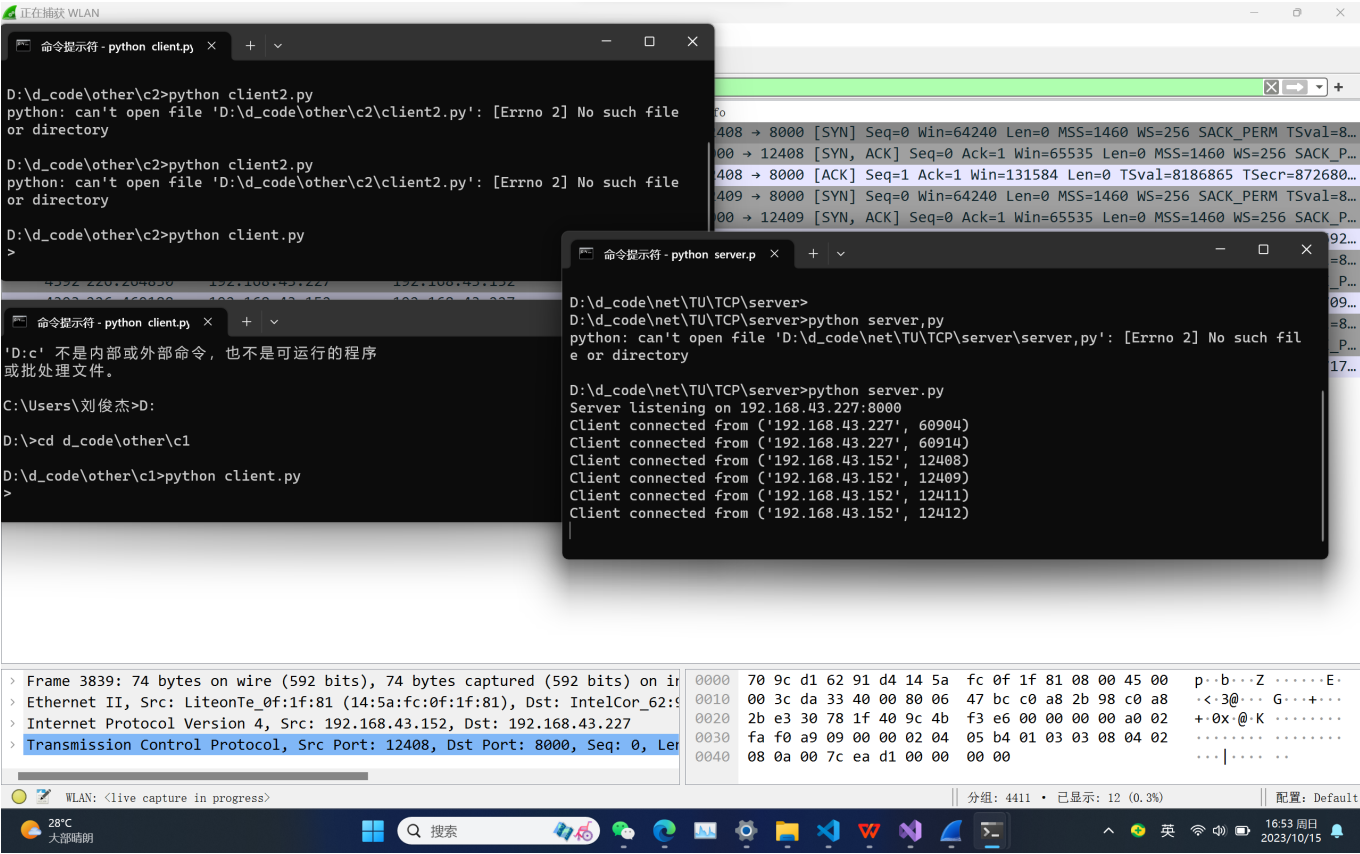


- 1) A机器运行服务器软件;
- 2) 在A机器也同时运行客户端软件2个客户端;



可以看到服务端显示了客户端和服务端的连接信息

3) 在客户端B机器的命令行同时运行4个客户端；



可以看到服务端显示了客户端和服务端的连接信息

4) 观看服务器端运行的日志。



A机器2个客户的运行日志：

命令提示符 - python client.py

Microsoft Windows [版本 10.0.22621.2428]  
(c) Microsoft Corporation. 保留所有权利。  
  
C:\Users\刘俊杰>D:  
  
D:\>cd d\_code\other\c1  
  
D:\d\_code\other\c1>python client.py  
> LIST  
  
-----  
RESPONSE FROM SERVER:  
client3.txt  
client4.txt  
client5.txt  
client6.txt  
ljj.txt  
server.log  
server.py  
-----  
> DOWNLOAD ljj.txt  
-----  
RESPONSE FROM SERVER:  
Download success  
-----  
> UPLOAD client1.txt  
-----  
RESPONSE FROM SERVER:  
Upload success  
-----  
>  
-----

命令提示符 - python client.py

(c) Microsoft Corporation. 保留所有权利。  
  
C:\Users\刘俊杰>D:  
  
D:\>cd d\_code\other\c2  
  
D:\d\_code\other\c2>python client.py  
> LIST  
  
-----  
RESPONSE FROM SERVER:  
client1.txt  
client3.txt  
client4.txt  
client5.txt  
client6.txt  
ljj.txt  
server.log  
server.py  
-----  
> DOWNLOAD ljj.txt  
-----  
RESPONSE FROM SERVER:  
Download success  
-----  
> UPLOAD client2.txt  
-----  
RESPONSE FROM SERVER:  
Upload success  
-----  
>  
-----

B机器4个客户的运行日志：








C:\Users\万元>E:  
E:\pythonpp\liu\client3>python client.py  
> LIST  
  
-----  
RESPONSE FROM SERVER:  
ljj.txt  
server.log  
server.py  
-----  
> UPLOAD client3.txt  
-----  
RESPONSE FROM SERVER:  
Upload success  
-----  
> DOWNLOAD ljj.txt  
-----  
RESPONSE FROM SERVER:  
Download success  
-----  
>  
-----

E:\>cd\pythonpp\liu\client4  
E:\pythonpp\liu\client4>python cli  
ent.py  
> LIST  
  
-----  
RESPONSE FROM SERVER:  
client3.txt  
client4.txt  
ljj.txt  
server.log  
server.py  
-----  
> UPLOAD client4.txt  
-----  
RESPONSE FROM SERVER:  
Upload success  
-----  
> DOWNLOAD ljj.txt  
-----  
RESPONSE FROM SERVER:  
Download success  
-----  
>  
-----

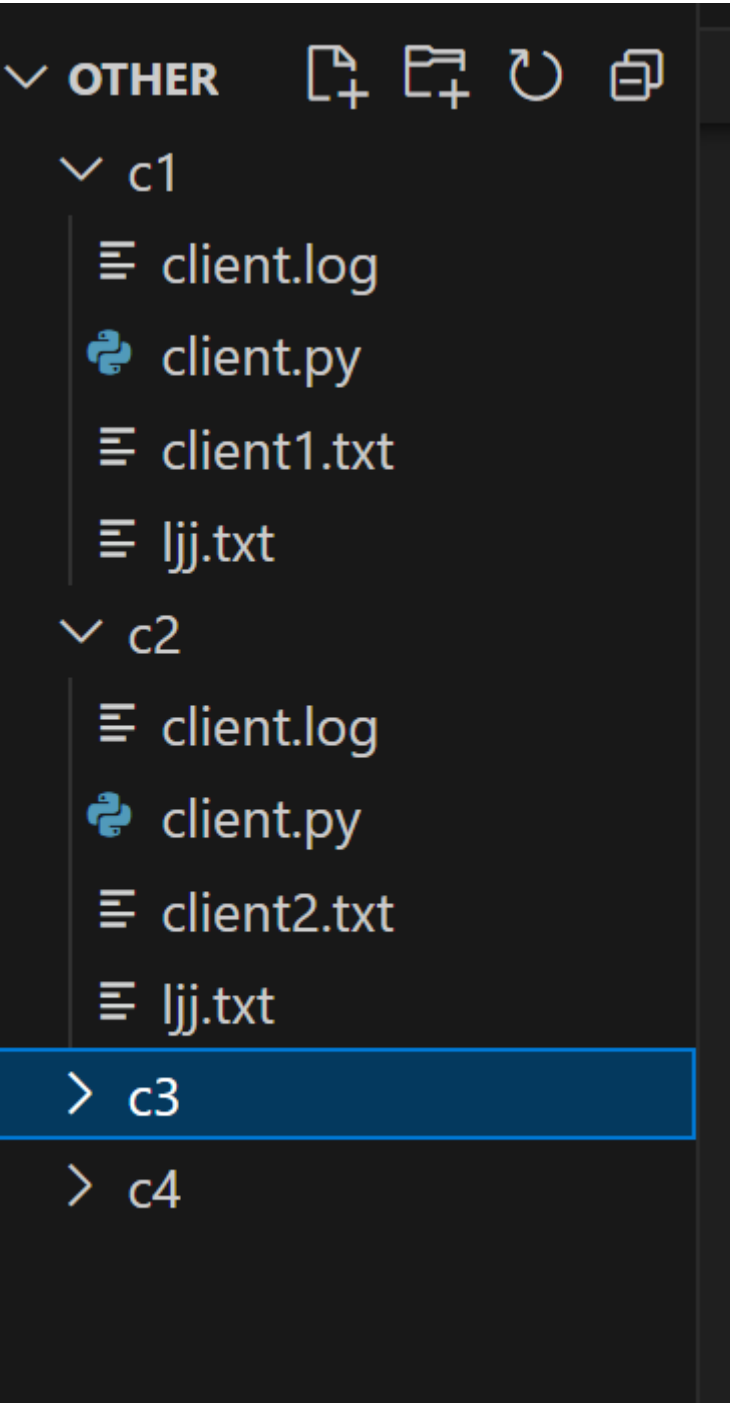
E:\pythonpp\liu\client5>python c  
lient.py  
> LIST  
  
-----  
RESPONSE FROM SERVER:  
client3.txt  
client4.txt  
ljj.txt  
server.log  
server.py  
-----  
> UPLOAD client5.txt  
-----  
RESPONSE FROM SERVER:  
Upload success  
-----  
> DOWNLOAD ljj.txt  
-----  
RESPONSE FROM SERVER:  
Download success  
-----  
>  
-----

E:\>cd\pythonpp\liu\client6  
E:\pythonpp\liu\client6>python client.py  
> LIST  
  
-----  
RESPONSE FROM SERVER:  
client3.txt  
client4.txt  
client5.txt  
ljj.txt  
server.log  
server.py  
-----  
> UPLOAD client6.txt  
-----  
RESPONSE FROM SERVER:  
Upload success  
-----  
> DOWNLOAD ljj.txt  
-----  
RESPONSE FROM SERVER:  
Download success  
-----  
>  
-----

A机器服务端文件夹下内容:

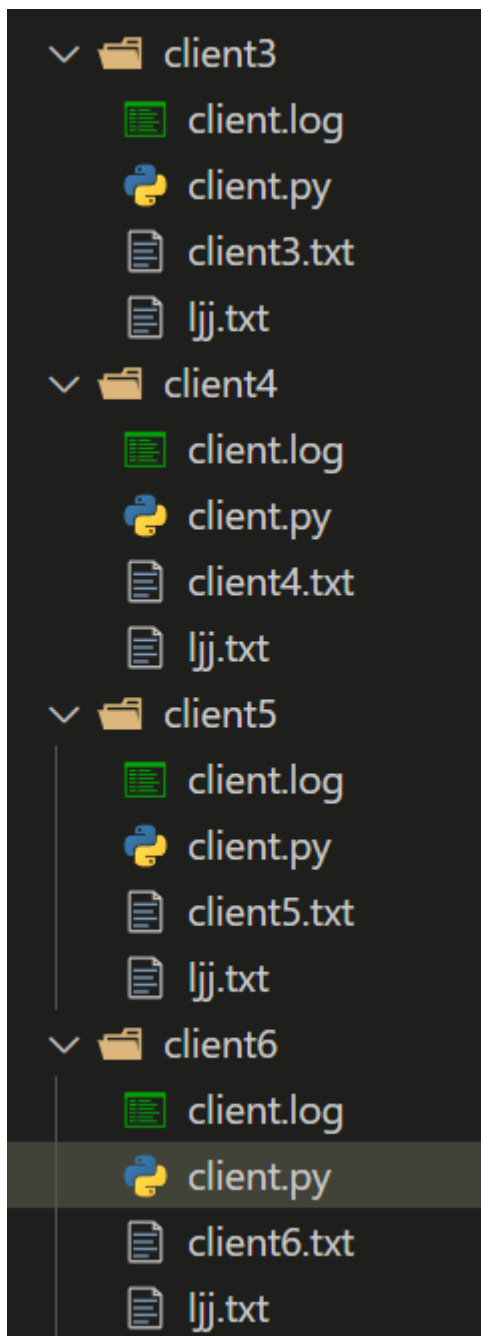
名称	修改日期	类型	大小
 server	2023/10/15 16:36	Python 源文件	3 KB
 server	2023/10/15 16:57	文本文档	1 KB
 ljj	2023/10/15 16:46	文本文档	0 KB
 client3	2023/10/15 16:54	文本文档	0 KB
 client4	2023/10/15 16:55	文本文档	0 KB
 client5	2023/10/15 16:56	文本文档	0 KB
 client6	2023/10/15 16:56	文本文档	0 KB

9 / 13



A机器2个客户文件夹下的内容:

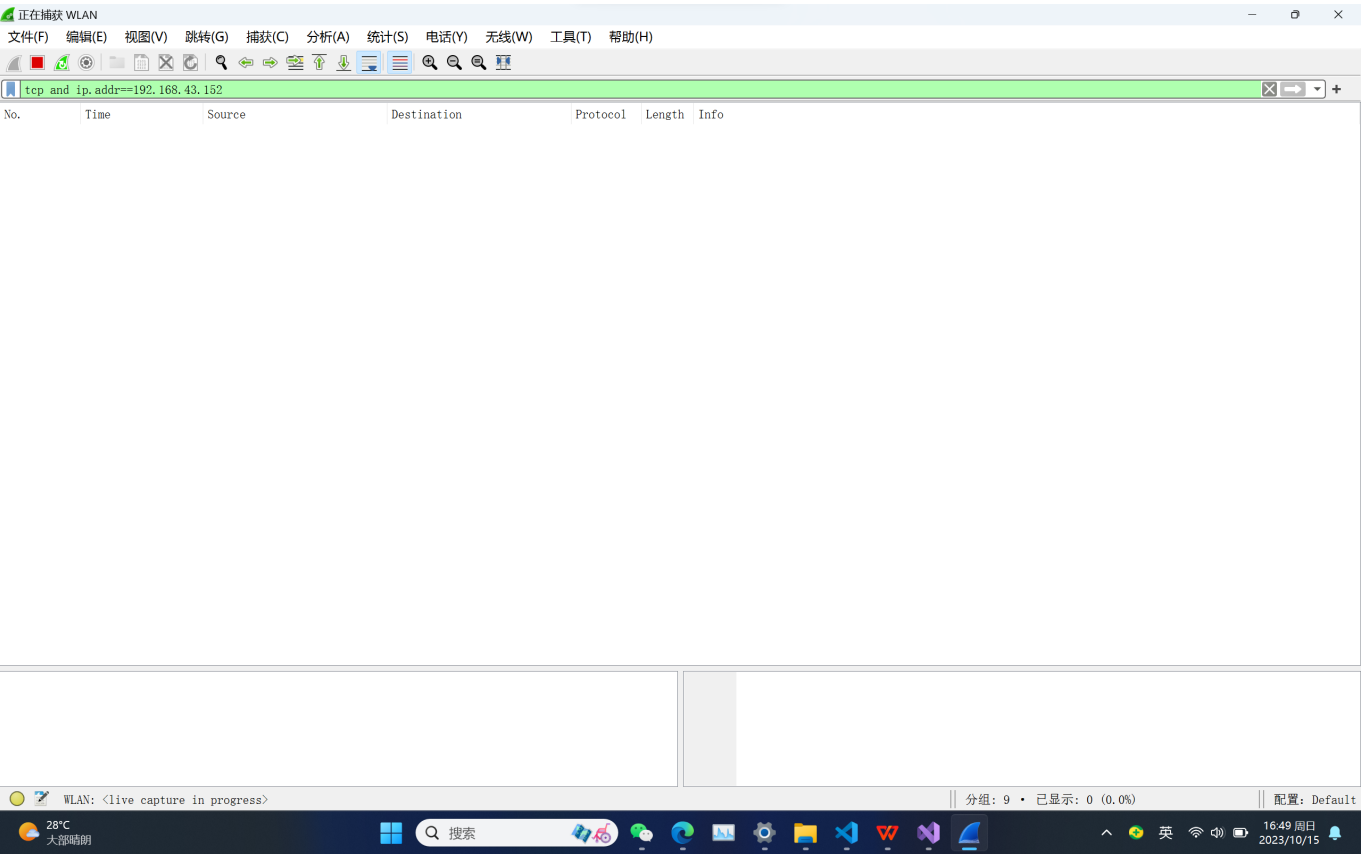
B机器4个客户文件



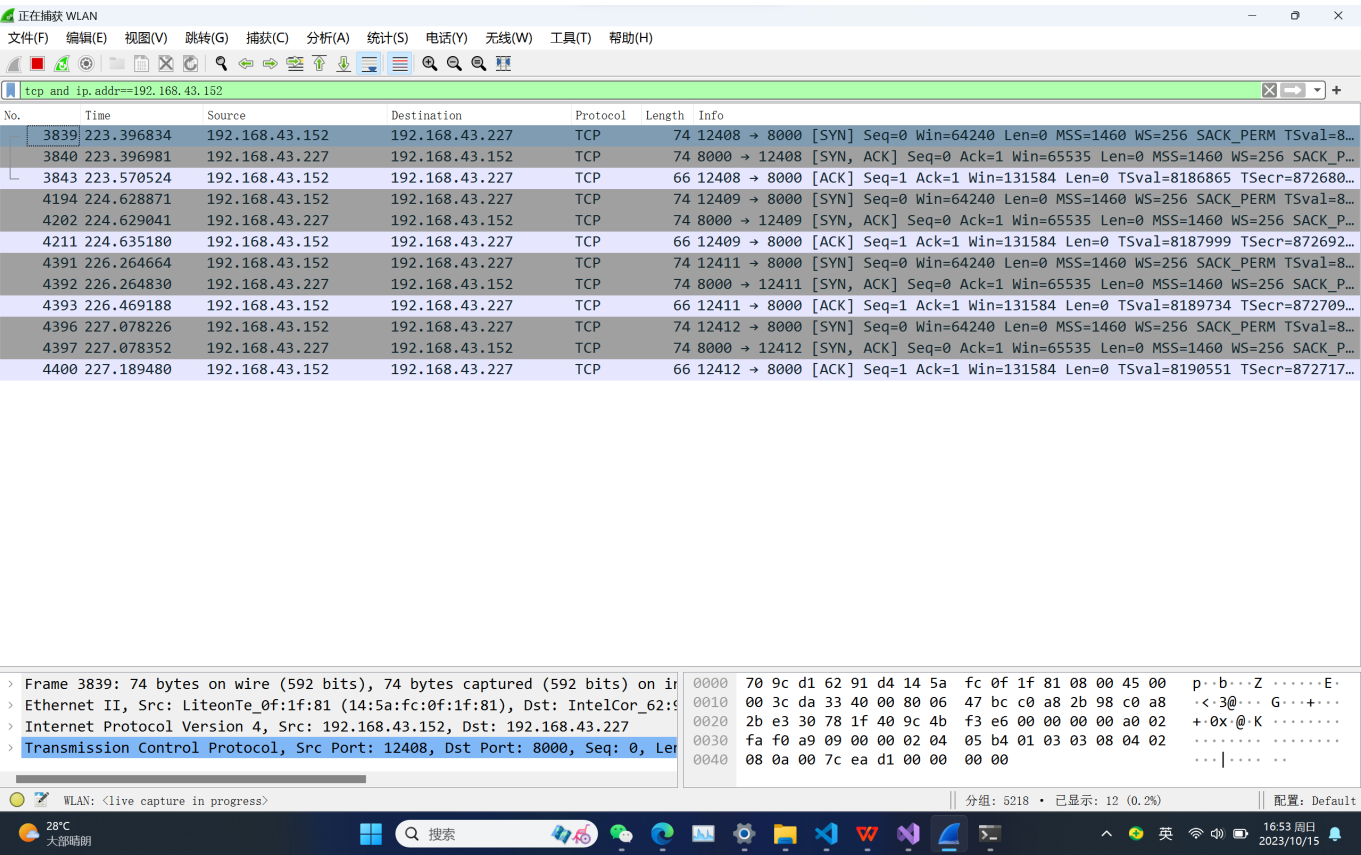
夹下的内容:

5) 请打开wireshark 观察、确认服务器端和客户端的传输层协议交互过程。

A 机器上客户端连接A机器服务器时没有抓到数据包：



B 机器上客户端连接A机器服务器时抓到的数据包：



每一次连接有三次握手的过程，四次连接总共抓到了12个数据包

服务机和客户端之间传输命令并执行:

正在捕获 WLAN

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

tcp and ip.addr==192.168.43.152

No.	Time	Source	Destination	Protocol	Length	Info
3839	223.396834	192.168.43.152	192.168.43.227	TCP	74	12408 → 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM TSval=8...
3840	223.396981	192.168.43.227	192.168.43.152	TCP	74	8000 → 12408 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_P...
3843	223.570524	192.168.43.152	192.168.43.227	TCP	66	12408 → 8000 [ACK] Seq=1 Ack=1 Win=131584 Len=0 TSval=8186865 TSecr=872680...
4194	224.628871	192.168.43.152	192.168.43.227	TCP	74	12409 → 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM TSval=8...
4202	224.629041	192.168.43.227	192.168.43.152	TCP	74	8000 → 12409 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_P...
4211	224.635180	192.168.43.152	192.168.43.227	TCP	66	12409 → 8000 [ACK] Seq=1 Ack=1 Win=131584 Len=0 TSval=8187999 TSecr=872692...
4391	226.264664	192.168.43.152	192.168.43.227	TCP	74	12411 → 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM TSval=8...
4392	226.264830	192.168.43.227	192.168.43.152	TCP	74	8000 → 12411 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_P...
4393	226.469188	192.168.43.152	192.168.43.227	TCP	66	12411 → 8000 [ACK] Seq=1 Ack=1 Win=131584 Len=0 TSval=8189734 TSecr=872709...
4396	227.078226	192.168.43.152	192.168.43.227	TCP	74	12412 → 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM TSval=8...
4397	227.078352	192.168.43.227	192.168.43.152	TCP	74	8000 → 12412 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_P...
4400	227.189480	192.168.43.152	192.168.43.227	TCP	66	12412 → 8000 [ACK] Seq=1 Ack=1 Win=131584 Len=0 TSval=8190551 TSecr=872717...
5992	261.552025	192.168.43.152	192.168.43.227	TCP	75	12408 → 8000 [PSH, ACK] Seq=1 Ack=1 Win=131584 Len=9 TSval=8224880 TSecr=8...
5993	261.552746	192.168.43.227	192.168.43.152	TCP	94	8000 → 12408 [PSH, ACK] Seq=1 Ack=10 Win=1049600 Len=28 TSval=87306192 TSe...
5997	261.696910	192.168.43.152	192.168.43.227	TCP	66	12408 → 8000 [ACK] Seq=10 Ack=29 Win=131584 Len=0 TSval=8225006 TSecr=8730...
6622	293.130799	192.168.43.152	192.168.43.227	TCP	84	12408 → 8000 [PSH, ACK] Seq=10 Ack=29 Win=131584 Len=18 TSval=8256391 TSec...
6623	293.180119	192.168.43.227	192.168.43.152	TCP	66	8000 → 12408 [ACK] Seq=29 Ack=28 Win=1049600 Len=0 TSval=87337819 TSecr=82...
6624	293.236570	192.168.43.152	192.168.43.227	TCP	69	12408 → 8000 [PSH, ACK] Seq=28 Ack=29 Win=131584 Len=3 TSval=8256601 TSecr...
6625	293.237047	192.168.43.227	192.168.43.152	TCP	80	8000 → 12408 [PSH, ACK] Seq=29 Ack=31 Win=1049600 Len=14 TSval=87337876 TS...
6626	293.338777	192.168.43.152	192.168.43.227	TCP	66	12408 → 8000 [ACK] Seq=31 Ack=43 Win=131584 Len=0 TSval=8256655 TSecr=8733...

> Frame 3839: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

> Ethernet II, Src: LiteonTe\_0f:1f:81 (14:5a:fc:0f:1f:81), Dst: IntelCor\_62:53:00 (82:53:00:00:00:00)

> Internet Protocol Version 4, Src: 192.168.43.152, Dst: 192.168.43.227

> Transmission Control Protocol, Src Port: 12408, Dst Port: 8000, Seq: 0, Len: 0

0000 70 9c d1 62 91 d4 14 5a fc 0f 1f 81 08 00 45 00 p·b···Z·····E·

0010 00 3c da 33 40 00 80 06 47 bc c0 a8 2b 98 c0 a8 ·<·3@··· G·····+

0020 2b e3 30 78 1f 40 9c 4b f3 e6 00 00 00 00 a0 02 +·0x·@·K·······

0030 fa f0 a9 09 00 00 02 04 05 b4 01 03 03 08 04 02 ······

0040 08 0a 00 7c ea d1 00 00 00 00 ···|·····

WLAN: <live capture in progress>

分组: 6848 · 已显示: 20 (0.3%)

配置: Default

28°C 大部晴朗

搜索

16:55 周日 2023/10/15

每次传输命令并执行两者之间有四条TCP数据包，其中带有push的数据包说明两者之间传递了数据信息，占用了缓冲区