

华东师范大学数据科学与工程学院期末项目报告

课程名称：计算机网络与编程

年级：2022 级

上机实践成绩：

指导教师：张召

姓名：李芳

学号：10214602404

上机实践名称：week 13_TCP 协议分析

上机实践日期：2024.05.24

上机实践编号：

组号：

上机实践时间：

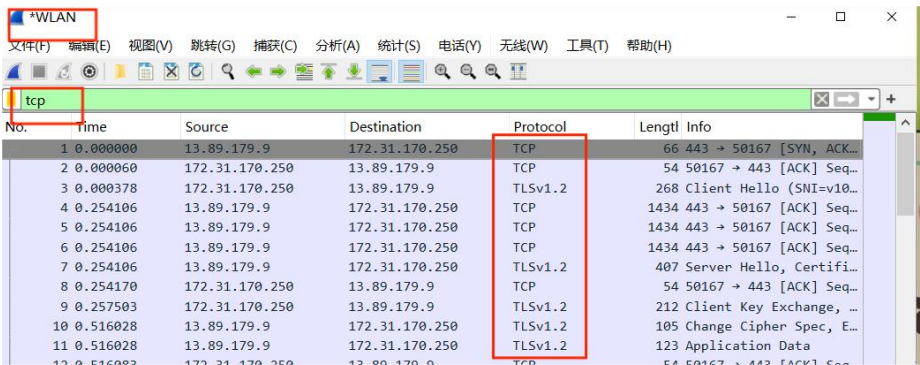
一、题目要求及实现情况

task1: 利用Wireshark抓取一个TCP数据包，查看其具体数据结构和实际的数据（要求根据报文结构正确标识每个部分），请将实验结果附在实验报告中。

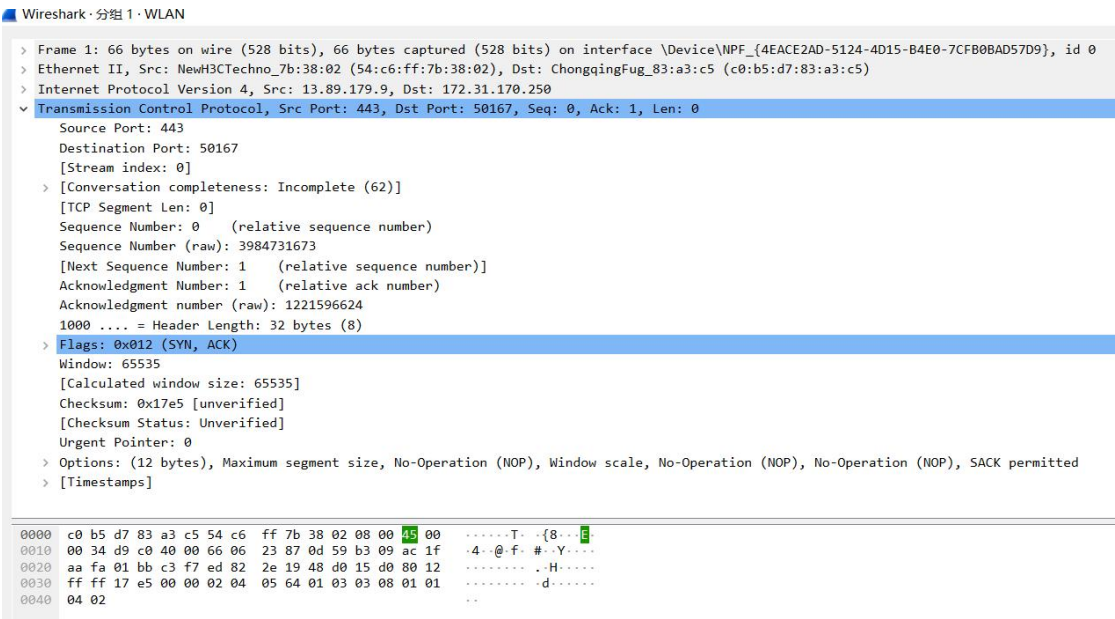
无操作捕获 TCP 时截图&分析：

➤ 操作流程：

- 先打开 wireshark，选择 wlan 选项进行数据包捕获：



- 选中其中一个 TCP 数据包进行数据包分析：



➤ 传输层协议数据包分析：

■ 捕获 TCP 数据包报文段具体信息：

```

Transmission Control Protocol, Src Port: 443, Dst Port: 50167, Seq: 0, Ack: 1, Len: 0
  Source Port: 443
  Destination Port: 50167
  [Stream index: 0]
  [Conversation completeness: Incomplete (62)]
    ..1. .... = RST: Present
    ...1 .... = FIN: Present
    .... 1... = Data: Present
    .... .1.. = ACK: Present
    .... ..1. = SYN-ACK: Present
    .... ...0 = SYN: Absent
    [Completeness Flags: RFDAS·]
  [TCP Segment Len: 0]
  Sequence Number: 0      (relative sequence number)
  Sequence Number (raw): 3984731673
  [Next Sequence Number: 1      (relative sequence number)]
  Acknowledgment Number: 1      (relative ack number)
  Acknowledgment number (raw): 1221596624
  1000 .... = Header Length: 32 bytes (8)
  Flags: 0x012 (SYN, ACK)
    000. .... = Reserved: Not set
    ...0 .... = Accurate ECN: Not set
    .... 0... = Congestion Window Reduced: 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
    [Expert Info (Chat/Sequence): Connection establish acknowledge (SYN+ACK): server port 443]
    [Connection establish acknowledge (SYN+ACK): server port 443]
    [Severity level: Chat]
    [Group: Sequence]
    .... .... ...0 = Fin: Not set
    [TCP Flags: .....A·S·]
  Window: 65535
  [Calculated window size: 65535]
  Checksum: 0x17e5 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-
  Operation (NOP), SACK permitted
    TCP Option - Maximum segment size: 1380 bytes
      Kind: Maximum Segment Size (2)
      Length: 4
      MSS Value: 1380
    TCP Option - No-Operation (NOP)
      Kind: No-Operation (1)
  
```

TCP Option - Window scale: 8 (multiply by 256)

Kind: Window Scale (3)

Length: 3

Shift count: 8

[Multiplier: 256]

TCP Option - No-Operation (NOP)

Kind: No-Operation (1)

TCP Option - No-Operation (NOP)

Kind: No-Operation (1)

TCP Option - SACK permitted

Kind: SACK Permitted (4)

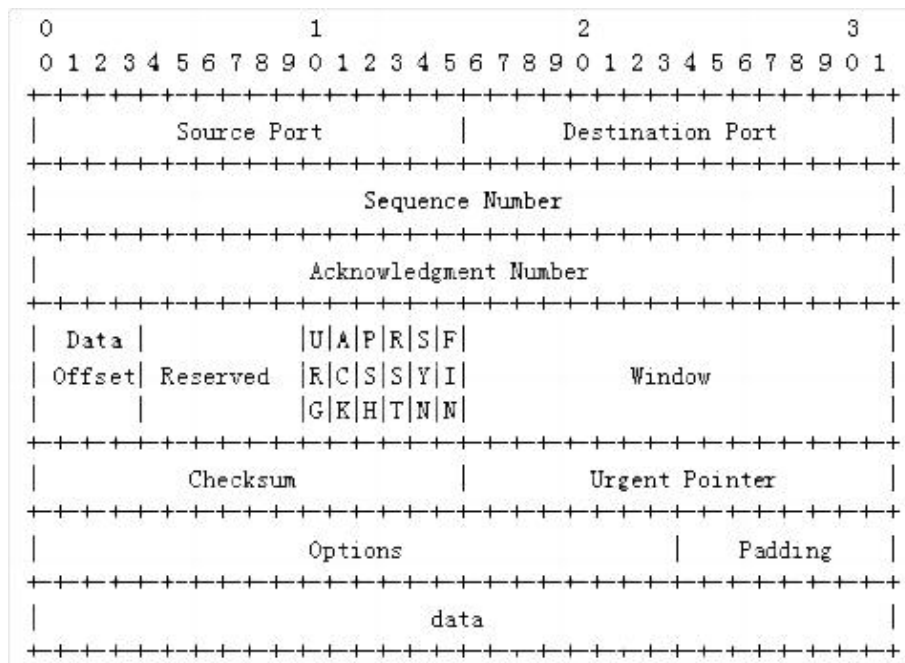
Length: 2

[Timestamps]

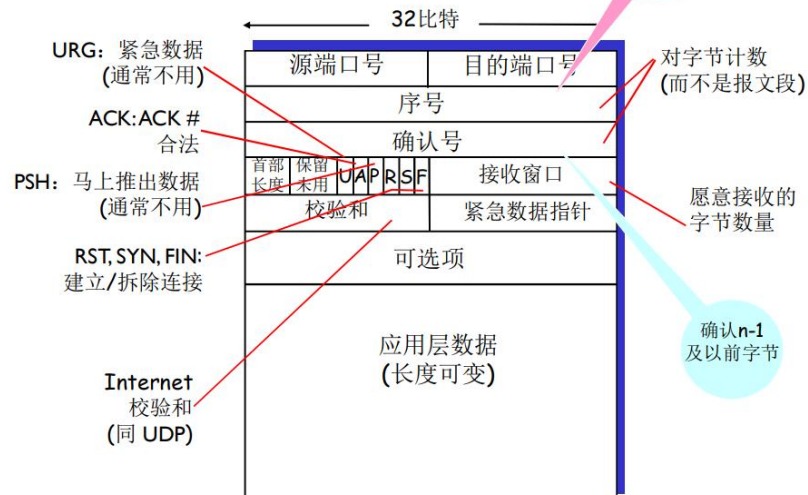
[Time since first frame in this TCP stream: 0.000000000 seconds]

[Time since previous frame in this TCP stream: 0.000000000 seconds]

■ TCP 报文段结构：



TCP报文段结构



■ 根据默认 TCP 报文结构分析捕获数据包：

- ◆ 源端口 (Source Port) : 443
- ◆ 目标端口 (Destination Port) : 50167
- ◆ 序列号 (Sequence Number) :
 - 序列号 (相对) : 0
 - 序列号 (原始) : 3984731673
 - 下一序列号: 1 (相对)
- ◆ 确认号 (Acknowledgment Number) :
 - 确认号 (相对) : 1
 - 确认号 (原始) : 1221596624
- ◆ 数据偏移 (Header Length) :
 - 首部长度: 32 字节
- ◆ 标志位 (Flags) : 0x012 (SYN, ACK)
 - Acknowledgment (ACK) : 已设置
 - Synchronize (SYN) : 已设置
 - Reset (RST) : 未设置
 - Finish (FIN) : 未设置
 -
- ◆ 接收窗口 (Window) : 65535
- ◆ 校验和 (Checksum) : 0x17e5 (未验证)
- ◆ 紧急指针 (Urgent Pointer) : 0
- ◆ 选项 (Options) :
 - 最大段大小 (MSS) : 1380 字节
 - No-Operation (NOP)

- 窗口缩放 (Window scale) : 8 (乘以 256)
- SACK 许可 (SACK permitted)
- ◆ 时间戳 (Timestamps) :
 - 从该 TCP 流的第一个帧以来的时间: 0.000000000 秒
 - 从该 TCP 流的前一个帧以来的时间: 0.000000000 秒
- 总结: 选中的数据包是一个 TCP 连接建立过程中的 SYN-ACK 包, 发送方端口是 443 (HTTPS), 目标端口是 50167 (客户端端口)。序列号和确认号表明这是连接的第一个数据段。标志位的设置 (SYN 和 ACK) 确认了这是连接建立的响应包。选项部分包含了最大段大小、窗口缩放和 SACK 许可等信息, 确保连接的可靠性和性能。

JAVA 实现 TCP 捕获时截图&分析:

➤ 操作流程:

- 先打开 wireshark, 选择 Adepter 选项进行数据包捕获
- 然后运行上上周 TCP 传输 1e8 文件的代码进行 TCP 数据包捕获

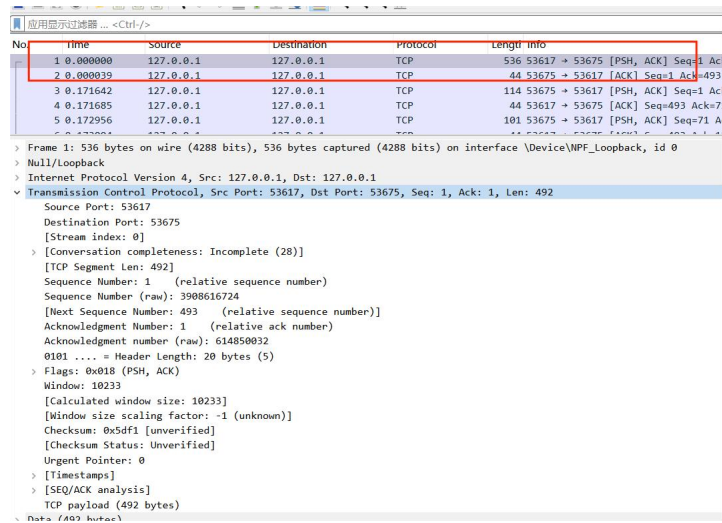
The image displays two screenshots related to a TCP file transfer experiment.

The top screenshot shows an IDE with two Java files: `TCPFileSender.java` and `TCPFileReceiver.java`. The `TCPFileSender.java` code is as follows:

```
1 package HW9.bntask;  
2  
3 import java.io.*;  
4 import java.net.Socket;  
5 import java.security.NoSuchAlgorithmException;  
6 import java.util.Random;  
7  
8 public class TCPFileSender {  
9     public static void main(String[] args) throws IOException, NoSuchAlgorithmException {  
10         String fileName = "checksum1e8_tcp.txt";  
11         FileInputStream fis = new FileInputStream(fileName);  
12         FileWriter fileWriter = new FileWriter(fileName);  
13         Socket socket = new Socket("127.0.0.1", 9091);  
14         BufferedInputStream bis = new BufferedInputStream(fis);  
15         OutputStream os = socket.getOutputStream();  
16         try {  
17             Random random = new Random( seed: 2023);  
18             for (int i = 0; i < 1e8; i++) {
```

The bottom screenshot shows the Wireshark network protocol analyzer. The packet list pane shows a series of TCP packets. The selected packet (No. 1) is a SYN-ACK packet from 127.0.0.1 to 127.0.0.1, port 443 to 50167. The packet details pane shows the TCP header and options, including the window scale factor (8) and SACK permitted flag.

➤ 选中其中一个进行同样的 TCP 报文格式分析：



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	536	53617 → 53675 [PSH, ACK] Seq=1 Ac
2	0.000039	127.0.0.1	127.0.0.1	TCP	44	53675 → 53617 [ACK] Seq=1 Ack=493
3	0.171642	127.0.0.1	127.0.0.1	TCP	114	53675 → 53617 [PSH, ACK] Seq=1 Ac
4	0.171685	127.0.0.1	127.0.0.1	TCP	44	53617 → 53675 [ACK] Seq=493 Ack=7
5	0.172956	127.0.0.1	127.0.0.1	TCP	101	53675 → 53617 [PSH, ACK] Seq=71 A

Frame 1: 536 bytes on wire (4288 bits), 536 bytes captured (4288 bits) on interface \Device\NPF_{...} id 0

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 53617, Dst Port: 53675, Seq: 1, Ack: 1, Len: 492

Source Port: 53617

Destination Port: 53675

[Stream index: 0]

[Conversation completeness: Incomplete (28)]

[TCP Segment Len: 492]

Sequence Number: 1 (relative sequence number)

Sequence Number (raw): 3908616724

[Next Sequence Number: 493 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 614850032

0101 = Header Length: 20 bytes (5)

Flags: 0x018 (PSH, ACK)

Window: 10233

[Calculated window size: 10233]

[Window size scaling factor: -1 (unknown)]

Checksum: 0x5df1 [unverified]

[Checksum Status: Unverified]

Urgent Pointer: 0

[Timestamps]

[SEQ/ACK analysis]

TCP payload (492 bytes)

Data (492 bytes)

■ 报文：

Transmission Control Protocol, Src Port: 53617, Dst Port: 53675, Seq: 1, Ack: 1, Len: 492

Source Port: 53617

Destination Port: 53675

[Stream index: 0]

[Conversation completeness: Incomplete (28)]

..0. = RST: Absent

...1 = FIN: Present

.... 1... = Data: Present

.... .1.. = ACK: Present

.... ..0. = SYN-ACK: Absent

.... ...0 = SYN: Absent

[Completeness Flags: ·FDA··]

[TCP Segment Len: 492]

Sequence Number: 1 (relative sequence number)

Sequence Number (raw): 3908616724

[Next Sequence Number: 493 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 614850032

0101 = Header Length: 20 bytes (5)

Flags: 0x018 (PSH, ACK)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

.... 0... = Congestion Window Reduced: Not set

.... .0.. = ECN-Echo: Not set

.... ..0. = Urgent: Not set

.... ...1 = Acknowledgment: Set

.... 1... = Push: Set

....0.. = Reset: Not set

....0. = Syn: Not set

....0 = Fin: Not set

[TCP Flags:AP···]

Window: 10233

[Calculated window size: 10233]

[Window size scaling factor: -1 (unknown)]

Checksum: 0x5df1 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
[Timestamps]
[Time since first frame in this TCP stream: 0.000000000 seconds]
[Time since previous frame in this TCP stream: 0.000000000 seconds]
[SEQ/ACK analysis]
[Bytes in flight: 492]
[Bytes sent since last PSH flag: 492]
TCP payload (492 bytes)

➤ 分析:

- 源端口 (Source Port) : 53617
- 目标端口 (Destination Port) : 53675
- 序列号 (Sequence Number) :
 - ◆ 序列号 (相对) : 1
 - ◆ 序列号 (原始) : 3908616724
 - ◆ 下一序列号: 493 (相对)
- 确认号 (Acknowledgment Number) :
 - ◆ 确认号 (相对) : 1
 - ◆ 确认号 (原始) : 614850032
- 数据偏移 (Header Length) :
 - ◆ 报头长度: 20 字节
- 标志位 (Flags) :
 - ◆ 0x018 (PSH, ACK)
 - ◆ Acknowledgment (ACK) : 已设置
 - ◆ Push (PSH) : 已设置
 - ◆ Reset (RST) : 未设置
 - ◆ Synchronize (SYN) : 未设置
 - ◆ Finish (FIN) : 未设置
- 窗口大小 (Window Size) : 10233
- 校验和 (Checksum) : 0x5df1 (未验证)
- 紧急指针 (Urgent Pointer) : 0
- 时间戳 (Timestamps) :
 - ◆ 从该 TCP 流的第一个帧以来的时间: 0.000000000 秒
 - ◆ 从该 TCP 流的前一个帧以来的时间: 0.000000000 秒
- TCP Payload: 数据负载长度: 492 字节
- 总结: 选中的是一个普通的 TCP 数据包, 源端口为 53617, 目标端口为 53675。序列号和确认号表明这是连接中的第二个数据段, 数据负载长度为 492 字节。标志位的设置 (PSH 和 ACK) 确认了这

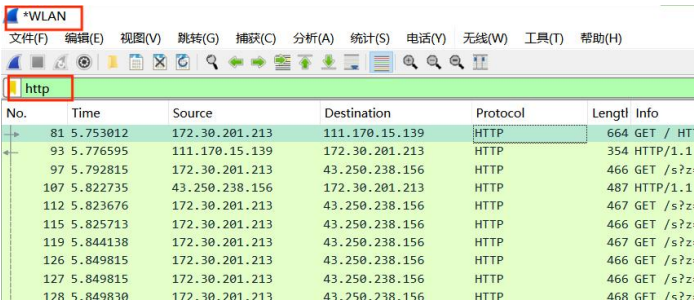
是一个包含数据的推送报文段。窗口大小和校验和等其他字段确保了连接的可靠性和完整性。

task2: 根据TCP三次握手的交互图和抓到的TCP报文详细分析三次握手过程，请将实验结果附在实验报告中。

三次握手过程分析:

➤ 操作流程:

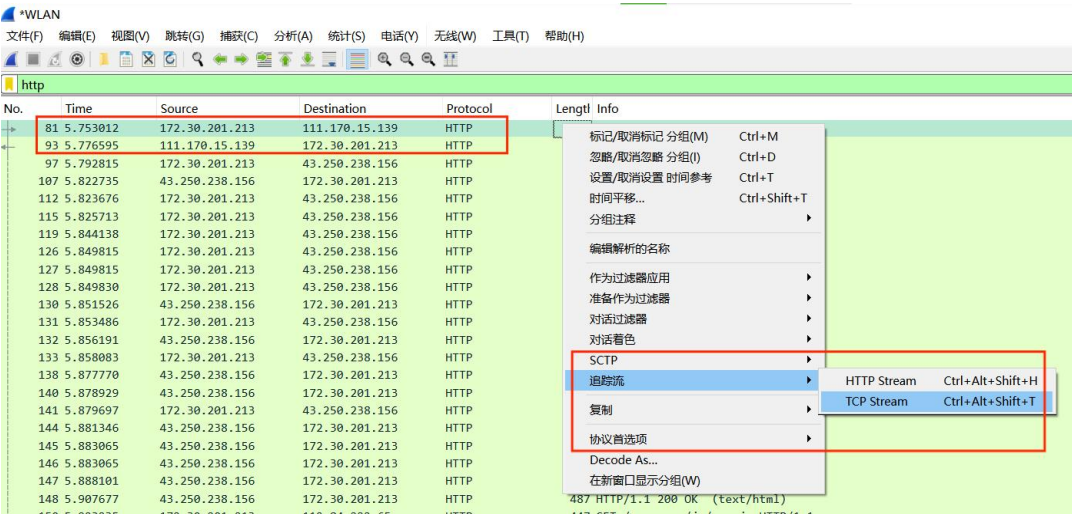
- 打开 http 协议的网站: 人民网 www.people.com.cn，捕获 http 数据包:



The screenshot shows a Wireshark packet capture on the *WLAN interface. The filter is set to 'http'. The packet list shows several GET requests. The first packet (No. 81) is highlighted with a red box.

No.	Time	Source	Destination	Protocol	Length	Info
81	5.753012	172.30.201.213	111.170.15.139	HTTP	664	GET / HTTP/1.1
93	5.776595	111.170.15.139	172.30.201.213	HTTP	354	HTTP/1.1 304 Not Modified
97	5.792815	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1
107	5.822735	43.250.238.156	172.30.201.213	HTTP	487	HTTP/1.1 200 OK (text/html)
112	5.823676	172.30.201.213	43.250.238.156	HTTP	467	GET /s?z HTTP/1.1
115	5.825713	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1
119	5.844138	172.30.201.213	43.250.238.156	HTTP	467	GET /s?z HTTP/1.1
126	5.849815	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1
127	5.849815	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1
128	5.849830	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1

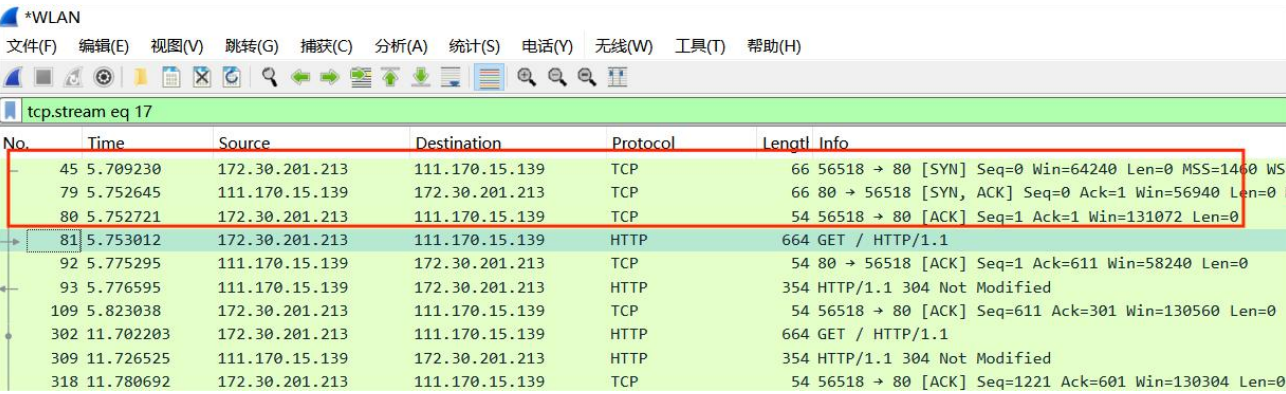
- 对第一个 GET 方法的 http 数据包进行 TCP 追踪流:



The screenshot shows the same Wireshark packet capture. The first packet (No. 81) is selected, and the 'Follow' (追踪流) menu is open. The 'TCP Stream' option is highlighted with a red box.

No.	Time	Source	Destination	Protocol	Length	Info
81	5.753012	172.30.201.213	111.170.15.139	HTTP	664	GET / HTTP/1.1
93	5.776595	111.170.15.139	172.30.201.213	HTTP	354	HTTP/1.1 304 Not Modified
97	5.792815	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1
107	5.822735	43.250.238.156	172.30.201.213	HTTP	487	HTTP/1.1 200 OK (text/html)
112	5.823676	172.30.201.213	43.250.238.156	HTTP	467	GET /s?z HTTP/1.1
115	5.825713	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1
119	5.844138	172.30.201.213	43.250.238.156	HTTP	467	GET /s?z HTTP/1.1
126	5.849815	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1
127	5.849815	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1
128	5.849830	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1
130	5.851526	43.250.238.156	172.30.201.213	HTTP	466	GET /s?z HTTP/1.1
131	5.853486	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1
132	5.856191	43.250.238.156	172.30.201.213	HTTP	466	GET /s?z HTTP/1.1
133	5.858083	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1
138	5.877770	43.250.238.156	172.30.201.213	HTTP	466	GET /s?z HTTP/1.1
140	5.878929	43.250.238.156	172.30.201.213	HTTP	466	GET /s?z HTTP/1.1
141	5.879697	172.30.201.213	43.250.238.156	HTTP	466	GET /s?z HTTP/1.1
144	5.881346	43.250.238.156	172.30.201.213	HTTP	466	GET /s?z HTTP/1.1
145	5.883065	43.250.238.156	172.30.201.213	HTTP	466	GET /s?z HTTP/1.1
146	5.883065	43.250.238.156	172.30.201.213	HTTP	466	GET /s?z HTTP/1.1
147	5.888101	43.250.238.156	172.30.201.213	HTTP	466	GET /s?z HTTP/1.1
148	5.907677	43.250.238.156	172.30.201.213	HTTP	466	GET /s?z HTTP/1.1

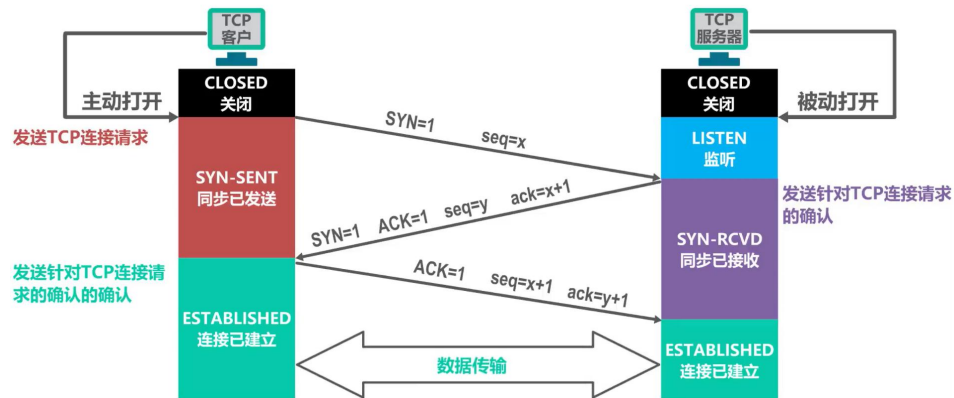
- 得到 TCP 三次握手的过程数据包:



The screenshot shows the same Wireshark packet capture. The filter is set to 'tcp.stream eq 17'. The packet list shows the TCP three-way handshake. The first three packets (No. 45, 79, 80) are highlighted with a red box.

No.	Time	Source	Destination	Protocol	Length	Info
45	5.709230	172.30.201.213	111.170.15.139	TCP	66	56518 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=0
79	5.752645	111.170.15.139	172.30.201.213	TCP	66	80 → 56518 [SYN, ACK] Seq=0 Ack=1 Win=56940 Len=0 MSS=1460 WS=0
80	5.752721	172.30.201.213	111.170.15.139	TCP	54	56518 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
81	5.753012	172.30.201.213	111.170.15.139	HTTP	664	GET / HTTP/1.1
92	5.775295	111.170.15.139	172.30.201.213	TCP	54	80 → 56518 [ACK] Seq=1 Ack=611 Win=58240 Len=0
93	5.776595	111.170.15.139	172.30.201.213	HTTP	354	HTTP/1.1 304 Not Modified
109	5.823038	172.30.201.213	111.170.15.139	TCP	54	56518 → 80 [ACK] Seq=611 Ack=301 Win=130560 Len=0
302	11.702203	172.30.201.213	111.170.15.139	HTTP	664	GET / HTTP/1.1
309	11.726525	111.170.15.139	172.30.201.213	HTTP	354	HTTP/1.1 304 Not Modified
318	11.780692	172.30.201.213	111.170.15.139	TCP	54	56518 → 80 [ACK] Seq=1221 Ack=601 Win=130304 Len=0

➤ 三次握手过程分析：



① 首先客户端向服务器发送一个 SYN 包，并等待服务器确认，其中：

标志位为 SYN，表示请求建立连接；

序号为 $\text{Seq} = x$ (x 一般取随机数)；

随后客户端进入 SYN-SENT 阶段。

② 服务器接收到客户端发来的 SYN 包后，对该包进行确认后结束 LISTEN 阶段，并返回一段 TCP 报文，其中：

标志位为 SYN 和 ACK，表示确认客户端的报文 Seq 序号有效，服务器能正常接收客户端发送的数据，并同意创建新连接；

序号为 $\text{Seq} = y$ ；

确认号为 $\text{Ack} = x + 1$ ，表示收到客户端的序号 Seq 并将其值加 1 作为自己确认号 Ack 的值，随后服务器端进入 SYN-RCV 阶段。

③ 客户端接收到发送的 SYN + ACK 包后，明确了从客户端到服务器的数据传输是正常的，从而结束 SYN-SENT 阶段。并返回最后一段报文。其中：

标志位为 ACK，表示确认收到服务器端同意连接的信号；

序号为 $\text{Seq} = x + 1$ ，表示收到服务器端的确认号 Ack，并将其值作为自己的序号值；

确认号为 $\text{Ack} = y + 1$ ，表示收到服务器端序号 seq，并将其值加 1 作为自己的确认号 Ack 的值。随后客户端进入 ESTABLISHED。

- ✓ 当服务器端收到来自客户端确认收到服务器数据的报文后，得知从服务器到客户端的数据传输是正常的，从而结束 SYN-RECV 阶段，进入 ESTABLISHED 阶段，从而完成三次握手。

➤ 报文分析：

- 第一次：Seq=0 是客户端 TCP 序号；设置 SYN 标志位，请求建立连接

No.	Time	Source	Destination	Protocol	Length	Info
45	5.709230	172.30.201.213	111.170.15.139	TCP	66	56518 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
79	5.752645	111.170.15.139	172.30.201.213	TCP	66	80 → 56518 [SYN, ACK] Seq=0 Ack=1 Win=56940 Len=0 MSS=1380 SACK_PERM WS=128
80	5.752721	172.30.201.213	111.170.15.139	TCP	54	56518 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
81	5.753012	172.30.201.213	111.170.15.139	HTTP	664	GET / HTTP/1.1

> Frame 45: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{4EACE2AD-5124-4D15-B4E0-7CFB0BAD57D9}, id 0	0000
> Ethernet II, Src: ChongqingFug_83:a3:c5 (c0:b5:d7:83:a3:c5), Dst: NewH3CTechno_7b:38:02 (54:c6:ff:7b:38:02)	0010
> Internet Protocol Version 4, Src: 172.30.201.213, Dst: 111.170.15.139	0020
> Transmission Control Protocol, Src Port: 56518, Dst Port: 80, Seq: 0, Len: 0	0030
Source Port: 56518	0040
Destination Port: 80	
[Stream index: 17]	
> [Conversation completeness: Complete, WITH_DATA (31)]	
[TCP Segment Len: 0]	
Sequence Number: 0 (relative sequence number)	
Sequence Number (raw): 3718308960	
[Next Sequence Number: 1 (relative sequence number)]	
Acknowledgment Number: 0	
Acknowledgment number (raw): 0	
1000 = Header Length: 32 bytes (8)	
> Flags: 0x002 (SYN)	
0000 = Reserved: Not set	
...0 = Accurate ECN: Not set	
....0... = Congestion Window Reduced: Not set	
....0... = ECN-Echo: Not set	
....0... = Urgent: Not set	
....0... = Acknowledgment: Not set	
....0... = Push: Not set	
....0... = Reset: Not set	
>0...1... = Syn: Set	
....0...0... = Fin: Not set	
[TCP Flags:S.]	
Window: 64240	
[Calculated window size: 64240]	
Checksum: 0xd9dd [unverified]	
[Checksum Status: Unverified]	
Urgent Pointer: 0	
> Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted	
> [Timestamps]	

- 第二次：Seq = 0 是服务端 TCP 序号；Ack=0+1，设置标志位 SYN 和 ACK，表示确认客户端的报文 Seq 序号有效，服务器能正常接收客户端发送的数据，并同意创建新连接

No.	Time	Source	Destination	Protocol	Length	Info
45	5.709230	172.30.201.213	111.170.15.139	TCP	66	56518 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
79	5.752645	111.170.15.139	172.30.201.213	TCP	66	80 → 56518 [SYN, ACK] Seq=0 Ack=1 Win=56940 Len=0 MSS=1380 SACK_PERM WS=128
80	5.752721	172.30.201.213	111.170.15.139	TCP	54	56518 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
81	5.753012	172.30.201.213	111.170.15.139	HTTP	664	GET / HTTP/1.1

> Frame 79: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{4EACE2AD-5124-4D15-B4E0-7CFB0BAD57D9}, id 0	0000
> Ethernet II, Src: NewH3CTechno_7b:38:02 (54:c6:ff:7b:38:02), Dst: ChongqingFug_83:a3:c5 (c0:b5:d7:83:a3:c5)	0010
> Internet Protocol Version 4, Src: 111.170.15.139, Dst: 172.30.201.213	0020
> Transmission Control Protocol, Src Port: 80, Dst Port: 56518, Seq: 0, Ack: 1, Len: 0	0030
Source Port: 80	0040
Destination Port: 56518	
[Stream index: 17]	
> [Conversation completeness: Complete, WITH_DATA (31)]	
[TCP Segment Len: 0]	
Sequence Number: 0 (relative sequence number)	
Sequence Number (raw): 2543537381	
[Next Sequence Number: 1 (relative sequence number)]	
Acknowledgment Number: 1 (relative ack number)	
Acknowledgment number (raw): 3718308961	
1000 = Header Length: 32 bytes (8)	
> Flags: 0x012 (SYN, ACK)	
0000 = Reserved: Not set	
...0 = Accurate ECN: Not set	
....0... = Congestion Window Reduced: Not set	
....0... = ECN-Echo: Not set	
....0... = Urgent: Not set	
>0...1... = Acknowledgment: Set	
....0...0... = Push: Not set	
....0...0... = Reset: Not set	
>0...1... = Syn: Set	
....0...0... = Fin: Not set	
[TCP Flags:A..S.]	
Window: 56940	
[Calculated window size: 56940]	
Checksum: 0x1821 [unverified]	
[Checksum Status: Unverified]	
Urgent Pointer: 0	
> Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale	
> [Timestamps]	

- 第三次：Seq=1 是客户端 TCP 序号，表示收到服务器端的确认号 ACK=1，并将其值作为自己的

序号值；设置标志位 ACK；Ack=0+1，表示收到服务器端序号 Seq=0，并将其值加 1 作为自己的确认号 Ack 的值

No.	Time	Source	Destination	Protocol	Length	Info
45	5.709230	172.30.201.213	111.170.15.139	TCP	66	56518 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
79	5.752645	111.170.15.139	172.30.201.213	TCP	66	80 → 56518 [SYN, ACK] Seq=0 Ack=1 Win=56940 Len=0 MSS=1380 SACK_PERM WS=128
80	5.752721	172.30.201.213	111.170.15.139	TCP	54	56518 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
81	5.753012	172.30.201.213	111.170.15.139	HTTP	664	GET / HTTP/1.1

> Frame 80: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{4EACE2AD-5124-4D15-B4E0-7CFB0BAD57D9}, id 0

> Ethernet II, Src: Chongqingfug_83:a3:c5 (c0:b5:d7:83:a3:c5), Dst: NewH3CTechno_7b:38:02 (54:c6:ff:7b:38:02)

> Internet Protocol Version 4, Src: 172.30.201.213, Dst: 111.170.15.139

> Transmission Control Protocol, Src Port: 56518, Dst Port: 80, Seq: 1, Ack: 1, Len: 0

Source Port: 56518

Destination Port: 80

[Stream index: 17]

> [Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 0]

Sequence Number: 1 (relative sequence number)

Sequence Number (raw): 3718308961

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 2343597982

0101 = Header Length: 20 bytes (5)

> Flags: 0x010 (ACK)

0000 = Reserved: Not set

...0 = Accurate ECN: Not set

....0... = Congestion Window Reduced: Not set

....0... = ECN-Echo: Not set

....0... = Urgent: Not set

....1... = Acknowledgment: Set

....0... = Push: Not set

....0... = Reset: Not set

....0... = Syn: Not set

....0... = Fin: Not set

[TCP Flags:A....]

Window: 512

[Calculated window size: 131072]

[Window size scaling factor: 256]

Checksum: 0x3510 [unverified]

[Checksum Status: Unverified]

Urgent Pointer: 0

> [Timestamps]

至此，三次握手结束，正式建立 TCP 连接，随后开始 HTTP 协议的数据传输。

task3: 根据TCP四次挥手的交互图和抓到的TCP报文详细分析四次挥手过程，请将实验结果附在实验报告中。

四次挥手过程分析:

➤ 操作流程:

■ 选择 task2 中捕获 http 数据包中获取 gif 图片的 http 数据包进行 TCP 追踪流

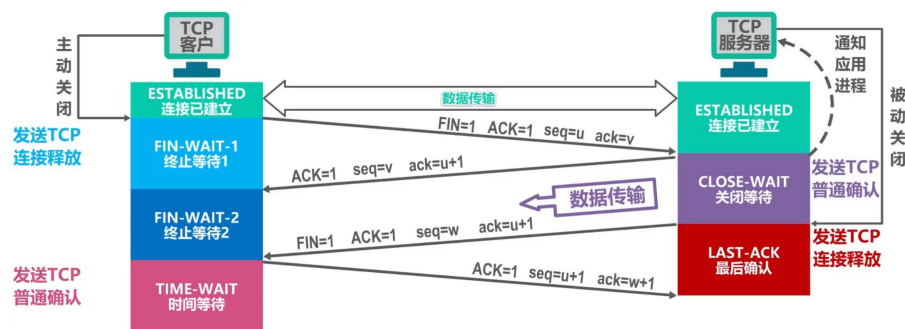
No.	Time	Source	Destination	Protocol	Length	Info
394	12.418676	119.39.205.30	172.30.201.213	HTTP	259	HTTP/1.1 200 (text/plain)
396	12.419166	221.122.98.131	172.30.201.213	HTTP		
483	14.715016	172.30.201.213	36.110.220.110	HTTP		
501	14.821543	172.30.201.213	111.170.15.139	HTTP		
517	14.826363	172.30.201.213	43.250.238.156	HTTP		
518	14.826479	172.30.201.213	43.250.238.156	HTTP		
519	14.826580	172.30.201.213	43.250.238.156	HTTP		
520	14.826769	172.30.201.213	43.250.238.156	HTTP		
521	14.826887	172.30.201.213	43.250.238.156	HTTP		
522	14.827052	172.30.201.213	36.110.220.110	HTTP		
555	14.847524	111.170.15.139	172.30.201.213	HTTP		
561	14.848572	111.170.15.139	172.30.201.213	HTTP		
570	14.849295	172.30.201.213	111.170.15.139	HTTP		
571	14.849523	172.30.201.213	111.170.15.139	HTTP		
572	14.850945	172.30.201.213	111.170.15.139	HTTP		
573	14.854848	43.250.238.156	172.30.201.213	HTTP		
574	14.854848	43.250.238.156	172.30.201.213	HTTP		
575	14.854848	43.250.238.156	172.30.201.213	HTTP		
583	14.855920	172.30.201.213	111.170.15.139	HTTP		
584	14.856112	172.30.201.213	111.170.15.139	HTTP		
585	14.856125	172.30.201.213	111.170.15.139	HTTP		
586	14.857555	43.250.238.156	172.30.201.213	HTTP		
588	14.857675	43.250.238.156	172.30.201.213	HTTP		
594	14.858509	172.30.201.213	111.170.15.139	HTTP		
598	14.861403	36.110.220.110	172.30.201.213	HTTP		

■ 得到以下完整 HTTP 借助 TCP 获取数据的数据流，对最后四条挥手进行详细分析：

No.	Time	Source	Destination	Protocol	Length	Info
383	12.358664	172.30.201.213	221.122.98.131	TCP	66	56551 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK
386	12.387588	221.122.98.131	172.30.201.213	TCP	66	80 → 56551 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1380
387	12.387678	172.30.201.213	221.122.98.131	TCP	54	56551 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
388	12.387920	172.30.201.213	221.122.98.131	HTTP	745	GET /1.gif?z=15&a=18faaacd82d&b=%u4EBA%u6C11%u7F51_%u7F51%
395	12.418676	221.122.98.131	172.30.201.213	TCP	54	80 → 56551 [ACK] Seq=1 Ack=692 Win=30592 Len=0
396	12.419166	221.122.98.131	172.30.201.213	HTTP	380	HTTP/1.1 200 OK (GIF89a)
397	12.419166	221.122.98.131	172.30.201.213	TCP	54	80 → 56551 [FIN, ACK] Seq=327 Ack=692 Win=30592 Len=0
398	12.419321	172.30.201.213	221.122.98.131	TCP	54	56551 → 80 [ACK] Seq=692 Ack=328 Win=130560 Len=0
399	12.419581	172.30.201.213	221.122.98.131	TCP	54	56551 → 80 [FIN, ACK] Seq=692 Ack=328 Win=130560 Len=0
400	12.451623	221.122.98.131	172.30.201.213	TCP	54	80 → 56551 [ACK] Seq=328 Ack=693 Win=30592 Len=0

> Frame 396: 380 bytes on wire (3040 bits), 380 bytes captured (3040 bits) on interface \Device\NPF_{4EACE2AD-5124-4D15-B4E0-7CFB0BAD57D9}, id 0
 > Ethernet II, Src: NewH3CTechno_7b:38:02 (54:c6:ff:7b:38:02), Dst: ChongqingFug_83:a3:c5 (c0:b5:d7:83:a3:c5)
 > Internet Protocol Version 4, Src: 221.122.98.131, Dst: 172.30.201.213

➤ 四次挥手过程分析：



① 首先客户端向服务器发送一段 TCP 报文表明其想要释放 TCP 连接，其中：

标记位为 FIN，表示请求释放连接；

序号为 Seq = u；

随后客户端进入 FIN-WAIT-1 阶段，即半关闭阶段，并且停止向服务端发送通信数据。

② 服务器接收到客户端请求断开连接的 FIN 报文后，结束 ESTABLISHED 阶段，进入 CLOSE-WAIT 阶段并返回一段 TCP 报文，其中：

标记位为 ACK，表示接收到客户端释放连接请求；

序号为 Seq = v；

确认号为 Ack = u + 1，表示是在收到客户端报文的基础上，将其序号值加 1 作为本段报文确认号 Ack 的值；

随后服务器开始准备释放服务器端到客户端方向上的连接。

客户端收到服务器发送过来的 TCP 报文后，确认服务器已经收到了客户端连接释放的请求，随后客户端结束 FIN-WAIT-1 阶段，进入 FIN-WAIT-2 阶段。

③ 服务器端在发出 ACK 确认报文后，服务器端会将遗留的待传数据传送给客户端，待传输完成后即经过 CLOSE-WAIT 阶段，便做好了释放服务器端到客户端的连接准备，再次向客户端发出一段 TCP 报文，其中：

标记位为 FIN 和 ACK，表示已经准备好释放连接了；

序号为 $\text{Seq} = w$ ；

确认号 $\text{Ack} = u + 1$ ，表示是在收到客户端报文的基础上，将其序号 Seq 的值加 1 作为本段报文确认号 Ack 的值。

随后服务器端结束 CLOSE-WAIT 阶段，进入 LAST-ACK 阶段。并且停止向客户端发送数据。

④ 客户端收到从服务器发来的 TCP 报文，确认了服务器已经做好释放连接的准备，于是结束 FIN-WAIT-2 阶段，进入 TIME-WAIT 阶段，并向服务器发送一段报文，其中：

标记位为 ACK，表示接收到服务器准备好释放连接的信号；

序号为 $\text{Seq} = u + 1$ ，表示是在已收到服务器报文的基础上，将其确认号 Ack 值作为本段序号的值；

确认号为 $\text{Ack} = w + 1$ ，表示是在收到了服务器报文的基础上，将其序号 Seq 的值作为本段报文确认号的值。

随后客户端开始在 TIME-WAIT 阶段等待 2 MSL。服务器端收到从客户端发出的 TCP 报文之后结束 LAST-ACK 阶段，进入 CLOSED 阶段。

✓ 由此正式确认关闭服务器端到客户端方向上的连接。客户端等待完 2 MSL 之后，结束 TIME-WAIT 阶段，进入 CLOSED 阶段，由此完成「四次挥手」。

➤ 报文分析：

■ 第一次：Seq = 327 是客户端的 TCP 序号；Ack=692；设置标记位 ACK 和 FIN，表示请求释放连接

No.	Time	Source	Destination	Protocol	Length	Info
396	12.419166	221.122.98.131	172.30.201.213	HTTP	380	HTTP/1.1 200 OK (GIF89a)
397	12.419166	221.122.98.131	172.30.201.213	TCP	54	80 → 56551 [FIN, ACK] Seq=327 Ack=692 Win=30592 Len=0
398	12.419321	172.30.201.213	221.122.98.131	TCP	54	56551 → 80 [ACK] Seq=692 Ack=328 Win=130560 Len=0
399	12.419581	172.30.201.213	221.122.98.131	TCP	54	56551 → 80 [FIN, ACK] Seq=692 Ack=328 Win=130560 Len=0
400	12.451623	221.122.98.131	172.30.201.213	TCP	54	80 → 56551 [ACK] Seq=328 Ack=693 Win=30592 Len=0

> Ethernet II, Src: NewH3CTechno_7b:38:02 (54:c6:ff:7b:38:02), Dst: ChongqingFug_83:a3:c5 (c0:b5:d7:83:a3:c5)

> Internet Protocol Version 4, Src: 221.122.98.131, Dst: 172.30.201.213

▼ Transmission Control Protocol, Src Port: 80, Dst Port: 56551, Seq: 327, Ack: 692, Len: 0

Source Port: 80

Destination Port: 56551

[Stream index: 37]

> [Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 0]

Sequence Number: 327 (relative sequence number)

Sequence Number (raw): 375966269

[Next Sequence Number: 328 (relative sequence number)]

Acknowledgment Number: 692 (relative ack number)

Acknowledgment number (raw): 91429191

0101 = Header Length: 20 bytes (5)

▼ Flags: 0x011 (FIN, ACK)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

.... 0... = Congestion Window Reduced: Not set

.... .0.. = ECN-Echo: Not set

.... ..0. = Urgent: Not set

.... ...1 = Acknowledgment: Set

....0... = Push: Not set

....0.. = Reset: Not set

....0.. = Syn: Not set

>1 = Fin: Set

> [TCP Flags:A...F]

Window: 239

[Calculated window size: 30592]

[Window size scaling factor: 128]

- 第二次: Seq = 692 是服务端 TCP 序号; Ack=327+1=328, 设置标志位 ACK, 表示接收到客户端释放连接请求

No.	Time	Source	Destination	Protocol	Length	Info
396	12.419166	221.122.98.131	172.30.201.213	HTTP	380	HTTP/1.1 200 OK (GIF89a)
397	12.419166	221.122.98.131	172.30.201.213	TCP	54	80 → 56551 [FIN, ACK] Seq=327 Ack=692 Win=30592 Len=0
398	12.419321	172.30.201.213	221.122.98.131	TCP	54	56551 → 80 [ACK] Seq=692 Ack=328 Win=130560 Len=0
399	12.419581	172.30.201.213	221.122.98.131	TCP	54	56551 → 80 [FIN, ACK] Seq=692 Ack=328 Win=130560 Len=0
400	12.451623	221.122.98.131	172.30.201.213	TCP	54	80 → 56551 [ACK] Seq=328 Ack=693 Win=30592 Len=0

> Ethernet II, Src: ChongqingFug_83:a3:c5 (c0:b5:d7:83:a3:c5), Dst: NewH3CTechno_7b:38:02 (54:c6:ff:7b:38:02)

> Internet Protocol Version 4, Src: 172.30.201.213, Dst: 221.122.98.131

▼ Transmission Control Protocol, Src Port: 56551, Dst Port: 80, Seq: 692, Ack: 328, Len: 0

Source Port: 56551

Destination Port: 80

[Stream index: 37]

> [Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 0]

Sequence Number: 692 (relative sequence number)

Sequence Number (raw): 91429191

[Next Sequence Number: 692 (relative sequence number)]

Acknowledgment Number: 328 (relative ack number)

Acknowledgment number (raw): 375966270

0101 = Header Length: 20 bytes (5)

▼ Flags: 0x010 (ACK)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

.... 0... = Congestion Window Reduced: Not set

.... .0.. = ECN-Echo: Not set

.... ..0. = Urgent: Not set

.... ...1 = Acknowledgment: Set

....0... = Push: Not set

....0.. = Reset: Not set

....0.. = Syn: Not set

....0.. = Fin: Not set

[TCP Flags:A....]

Window: 510

[Calculated window size: 130560]

- 第三次: Seq = 692 是客户端 TCP 序号; Ack=328, 设置标志位 ACK 和 FIN, 表示已经准备好释放连接

No.	Time	Source	Destination	Protocol	Length	Info
396	12.419166	221.122.98.131	172.30.201.213	HTTP	380	HTTP/1.1 200 OK (GIF89a)
397	12.419166	221.122.98.131	172.30.201.213	TCP	54	80 → 56551 [FIN, ACK] Seq=327 Ack=692 Win=30592 Len=0
398	12.419321	172.30.201.213	221.122.98.131	TCP	54	56551 → 80 [ACK] Seq=692 Ack=328 Win=130560 Len=0
399	12.419581	172.30.201.213	221.122.98.131	TCP	54	56551 → 80 [FIN, ACK] Seq=692 Ack=328 Win=130560 Len=0
400	12.451623	221.122.98.131	172.30.201.213	TCP	54	80 → 56551 [ACK] Seq=328 Ack=693 Win=30592 Len=0

> Frame 399: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{4EACE2AD-5124-4D15-B4E0-7CFB0BAD57D9}, id 0
> Ethernet II, Src: ChongqingFug_83:a3:c5 (c0:b5:d7:83:a3:c5), Dst: NewH3CTechno_7b:38:02 (54:c6:ff:7b:38:02)
> Internet Protocol Version 4, Src: 172.30.201.213, Dst: 221.122.98.131
v Transmission Control Protocol, Src Port: 56551, Dst Port: 80, Seq: 692, Ack: 328, Len: 0
Source Port: 56551
Destination Port: 80
[Stream index: 37]
> [Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 692 (relative sequence number)
Sequence Number (raw): 91429191
[Next Sequence Number: 693 (relative sequence number)]
Acknowledgment Number: 328 (relative ack number)
Acknowledgment number (raw): 375966270
0101 = Header Length: 20 bytes (5)
v Flags: 0x011 (FIN, ACK)
000. = Reserved: Not set
...0 = Accurate ECN: Not set
... 0... = Congestion Window Reduced: Not set
... .0.. = ECN-Echo: Not set
... ..0. = Urgent: Not set
... ...1 = Acknowledgment: Set
...0... = Push: Not set
...0.. = Reset: Not set
...0. = Syn: Not set
...1 = Fin: Set
> [TCP Flags:A...F]
Window: 510

- 第四次: Seq = 328 是客户端 TCP 序号; Ack=692+1, 设置标志位 ACK, 表示确认释放连接

No.	Time	Source	Destination	Protocol	Length	Info
396	12.419166	221.122.98.131	172.30.201.213	HTTP	380	HTTP/1.1 200 OK (GIF89a)
397	12.419166	221.122.98.131	172.30.201.213	TCP	54	80 → 56551 [FIN, ACK] Seq=327 Ack=692 Win=30592 Len=0
398	12.419321	172.30.201.213	221.122.98.131	TCP	54	56551 → 80 [ACK] Seq=692 Ack=328 Win=130560 Len=0
399	12.419581	172.30.201.213	221.122.98.131	TCP	54	56551 → 80 [FIN, ACK] Seq=692 Ack=328 Win=130560 Len=0
400	12.451623	221.122.98.131	172.30.201.213	TCP	54	80 → 56551 [ACK] Seq=328 Ack=693 Win=30592 Len=0

> Frame 400: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{4EACE2AD-5124-4D15-B4E0-7CFB0BAD57D9}, id 0
> Ethernet II, Src: NewH3CTechno_7b:38:02 (54:c6:ff:7b:38:02), Dst: ChongqingFug_83:a3:c5 (c0:b5:d7:83:a3:c5)
> Internet Protocol Version 4, Src: 221.122.98.131, Dst: 172.30.201.213
v Transmission Control Protocol, Src Port: 80, Dst Port: 56551, Seq: 328, Ack: 693, Len: 0
Source Port: 80
Destination Port: 56551
[Stream index: 37]
> [Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 328 (relative sequence number)
Sequence Number (raw): 375966270
[Next Sequence Number: 328 (relative sequence number)]
Acknowledgment Number: 693 (relative ack number)
Acknowledgment number (raw): 91429192
0101 = Header Length: 20 bytes (5)
v Flags: 0x010 (ACK)
000. = Reserved: Not set
...0 = Accurate ECN: Not set
... 0... = Congestion Window Reduced: Not set
... .0.. = ECN-Echo: Not set
... ..0. = Urgent: Not set
... ...1 = Acknowledgment: Set
...0... = Push: Not set
...0.. = Reset: Not set
...0. = Syn: Not set
...0 = Fin: Not set
[TCP Flags:A...]
Window: 239

- 至此, 四次挥手结束, 服务器端到客户端方向上的连接关闭。

二、总结

本次实验上机，通过使用 Wireshark 工具抓取和分析 TCP 数据包，我深入了解了 TCP 协议的工作原理，特别是 TCP 建立连接的三次握手过程和断开连接的四次挥手过程。

Task1：我了解了 TCP 协议的工作原理：通过 Wireshark 抓包和分析，对 TCP 报文的结构和各字段的作用有了清晰的理解，包括源端口、目标端口、序列号、确认号、窗口大小、校验和、选项以及标志位等。

Task2：我学习 TCP 建立连接的三次握手过程：抓取并分析了 TCP 三次握手的报文，观察到 SYN、SYN-ACK、ACK 三个阶段的报文交互，理解了三次握手确保连接可靠建立的机制。

Task3：我学习 TCP 断开连接的四次挥手过程：通过对四次挥手过程的报文分析，了解了 FIN、ACK 标志位在断开连接中的作用，以及双方在关闭连接时的交互细节。

实验过程中遇到了问题：抓包数据量大，使用 Wireshark 进行 HTTP 捕获时，抓取到大量数据包，导致目标数据包难以定位。通过设置过滤条件，成功过滤追踪出所需的 TCP 报文。

总的来说，通过这次实验，我更加深入了解并掌握了 TCP 协议的基本工作原理和关键过程，并且复习了相关 HTTP 和 TCP 协议的理论知识，掌握了 wireshark 抓包追踪的操作，收获很大！