华东师范大学数据科学与工程学院实验报告

课程名称：计算机网络与编程	**年级：**21 级	**上机实践成绩：**
指导教师：张召	**姓名：**杨茜雅	**学号：**10215501435
上机实践名称： Lab11		**上机实践日期：**2023.5.19
上机实践编号：11	**组号：**	**上机实践时间：**5.19

一、 实验目的

·了解 TCP 协议的工作原理

·学习TCP建立连接三次握手的过程

·学习 TCP 断开连接四次挥手的过程

二、 实验任务

·使用 Wireshark 快速了解 TCP 协议

三、 使用环境

Wireshark

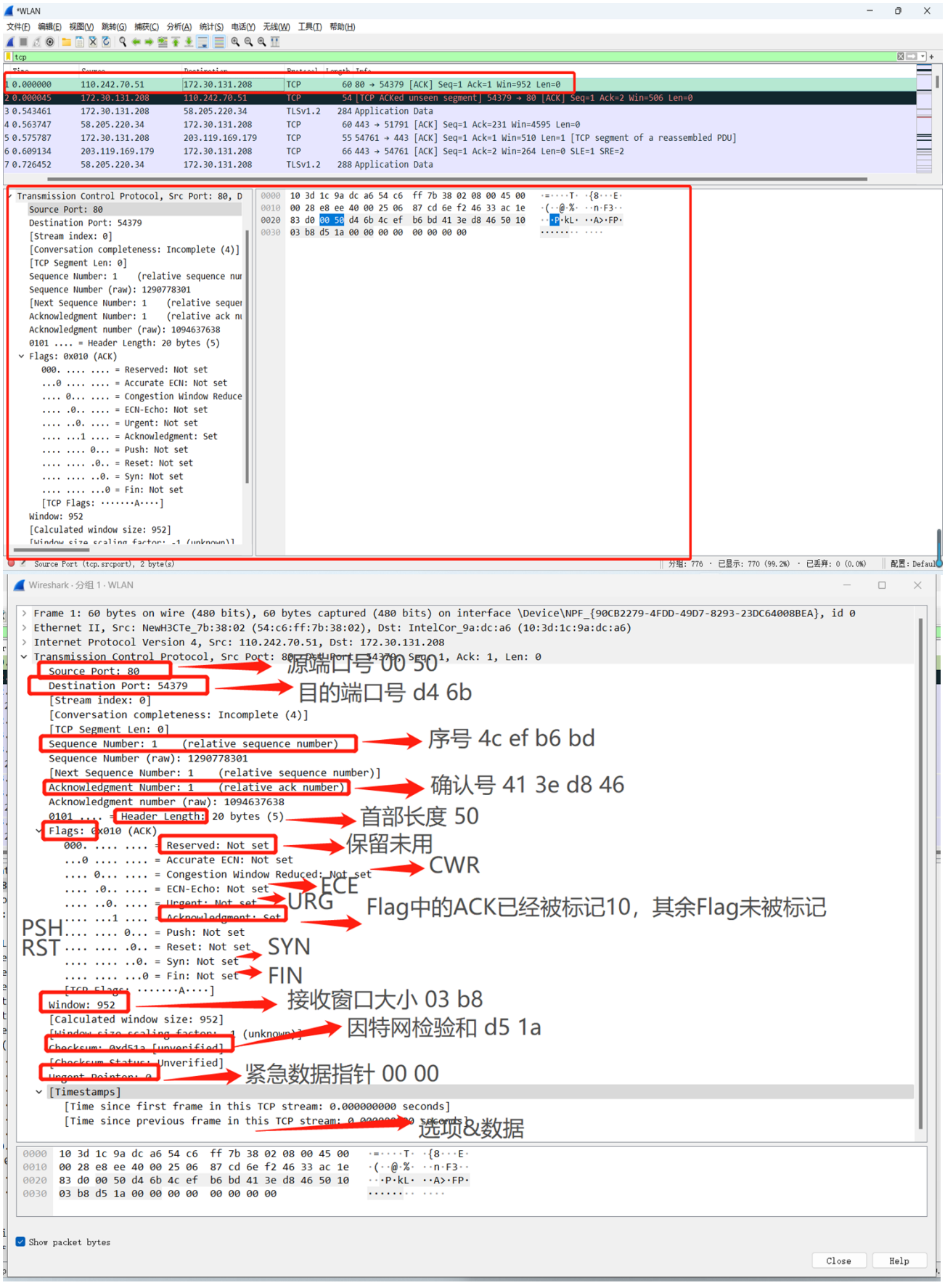
四、 实验过程

Task 1：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。

TCP是因特网运输层的面向连接的可靠的运输协议。TCP被称为是面向连接的（connection.oriented），这是因为在一个应用进程可以开始 向另一个应用进程发送数据之前，这两个进程必须先相互“握手”，即它们必须相互发送某些预备报文段，以建立确保数据传输的参数。作为TCP连接建立的一部分，连接的双方都将初始化与 TCP 连接相关的许多 TCP 状态变量。如图是 TCP 的报文段结构

图 3-29　TCP 报文段结构

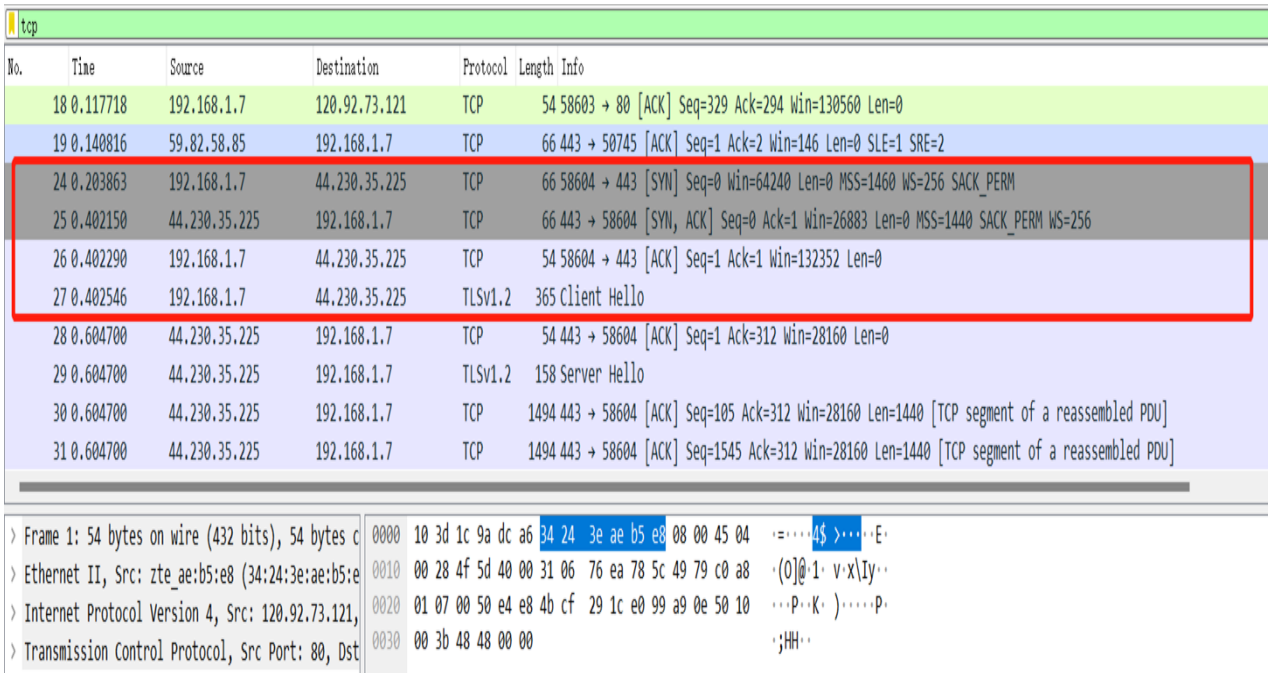
利用 Wireshark 抓包软件抓取一个 TCP 数据包并分析：



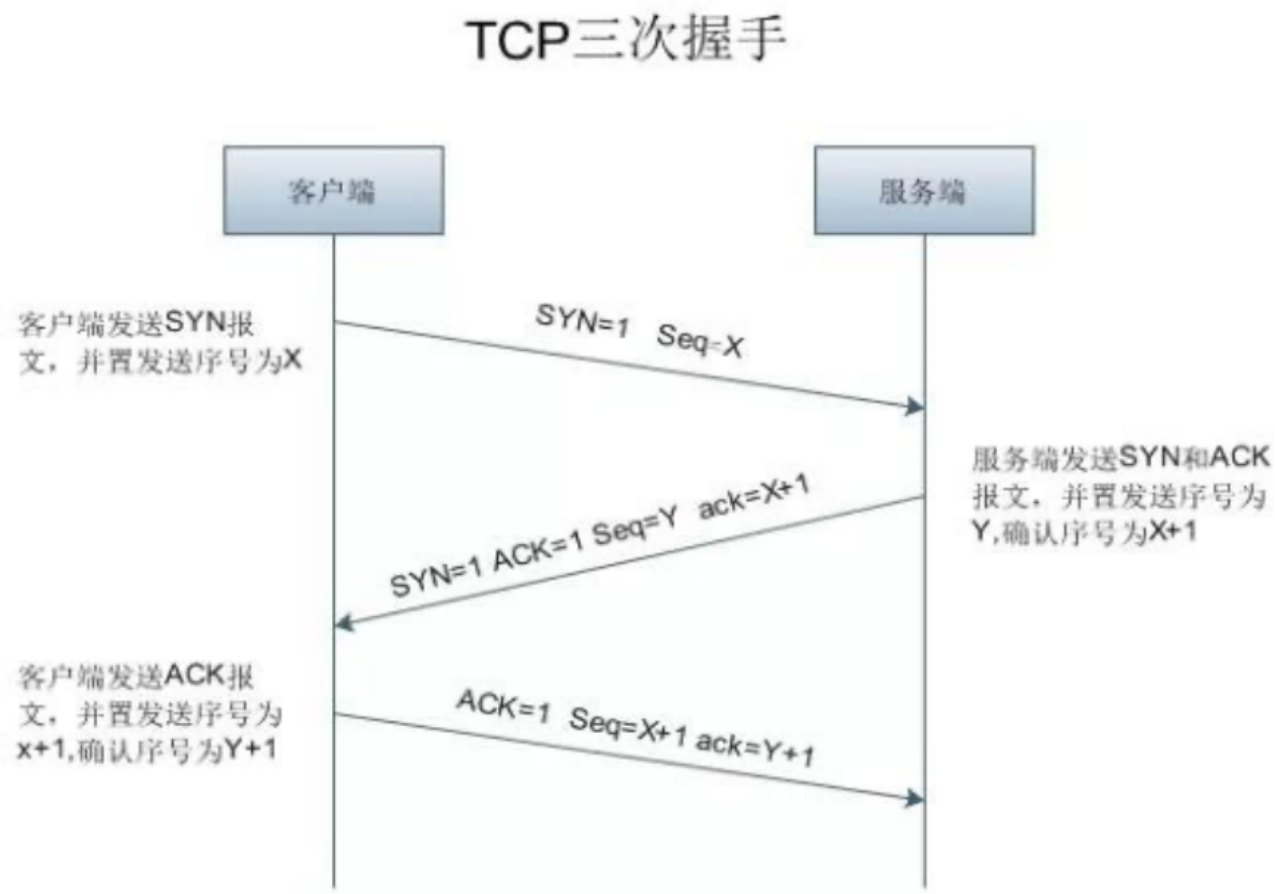
Task 2：根据TCP三次握手的交互图和抓到的TCP报文详细分析三次握手过程，请将实验结果附在实验报告中。

TCP建立连接时，会有三次握手过程，如下图所示，Wireshark截获到了三次握手的三个数据包。

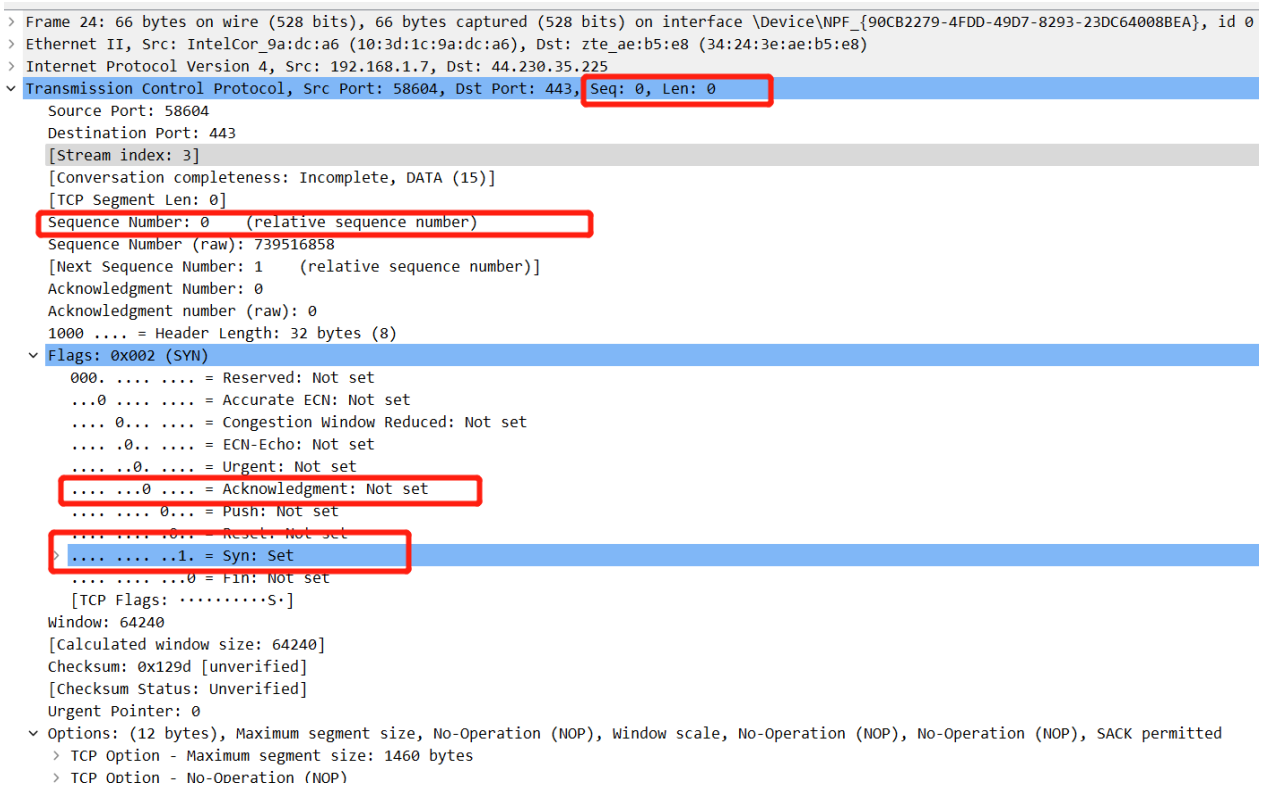
第四个包才 Client Hello 的，说明的确是使用 TCP 建立连接的。



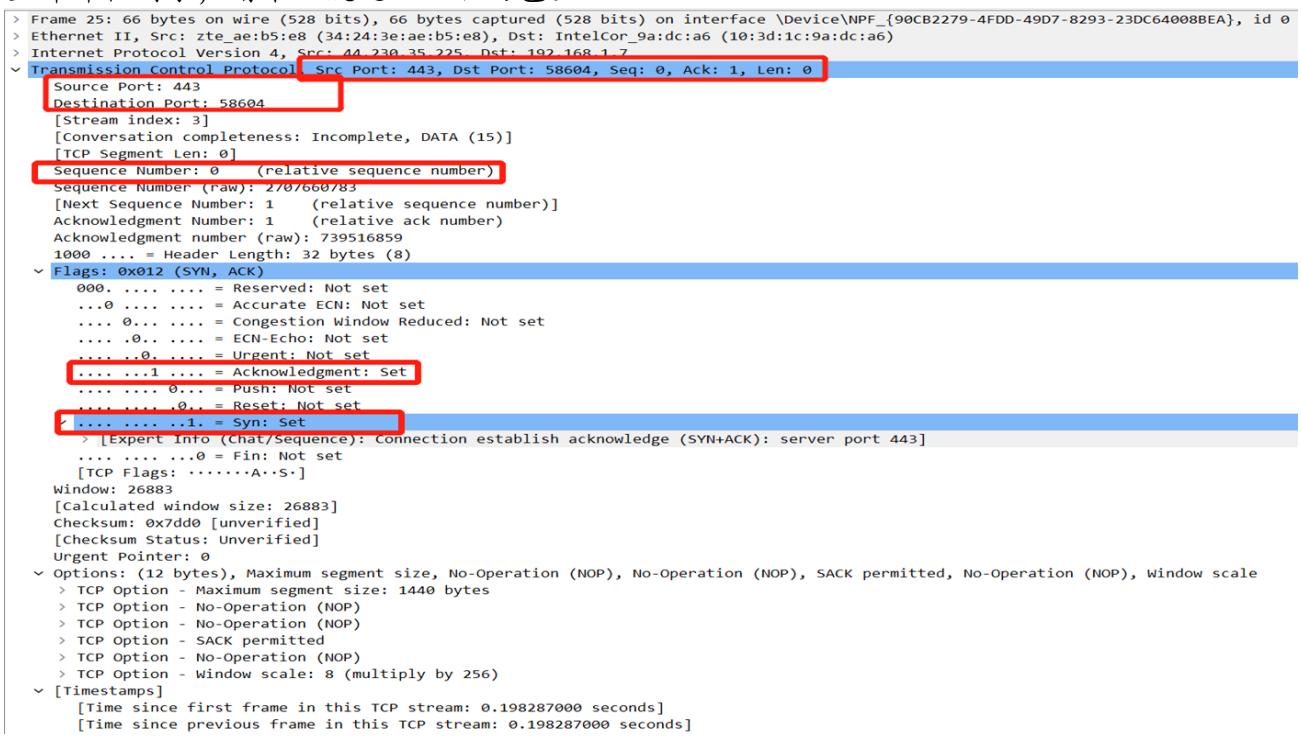
TCP 三次握手交互图：



第一次握手，客户端发送SYN报文，并置发送序号为0。在这个数据报中，Seq=0，说明一开始是从序号为0 的包发送的。我们可以看到这里SYN 这一位已经被设为1 了，因为还没收到来服务器的确认信息，因此这里ACK 设为0。



第二次握手，服务器发送SYN和ACK报文，并置发送序号为0，确认序号为1的报文。Seq=0，ACK = 1 ，因为TCP 是全双工通信的，因此发送来的第一个报文段也是从seq=0 开始的。但是这个报文段中包含了对我发的确认信息，因此这里ACK 被设置为1，这说明1 以前的包我全部都收到了，请本地发送1 以后的包。



第三次握手，客户端发送ACK报文，并置发送序号为1，确认序号为1。Seq=1，说明这是本地发送的第二个包了(第一个包Seq=0)；ACK = 1 说明已经收到了来自服务端的1 以前的所有包。

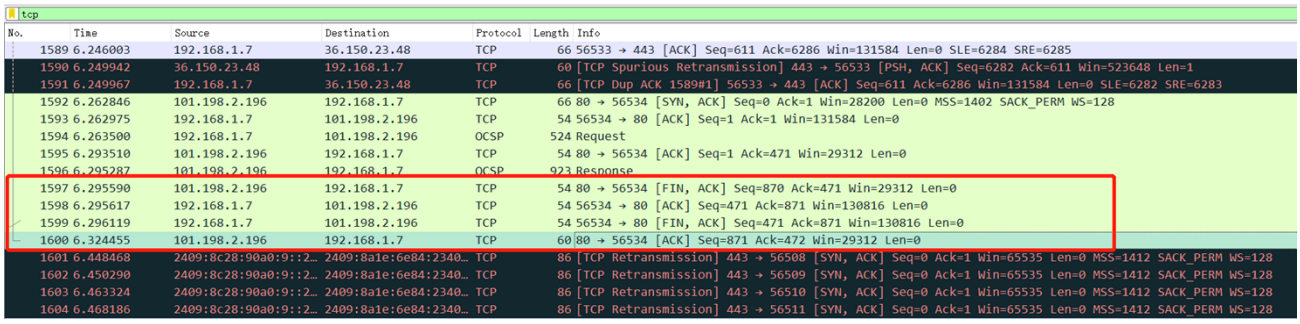
```
> Frame 26: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{90CB2279-4FDD-49D7-8293-23DC64008BEA}, id 0
> Ethernet II, Src: IntelCor_9a:dc:a6 (10:3d:1c:9a:dc:a6), Dst: zte_ae:b5:e8 (34:24:3e:ae:b5:e8)
> Internet Protocol Version 4, Src: 192.168.1.7, Dst: 44.230.35.225
v Transmission Control Protocol, Src Port: 58604, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
    Source Port: 58604
    Destination Port: 443
    [Stream index: 3]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 739516859
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 2707660784
    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····]
```

Client hello:

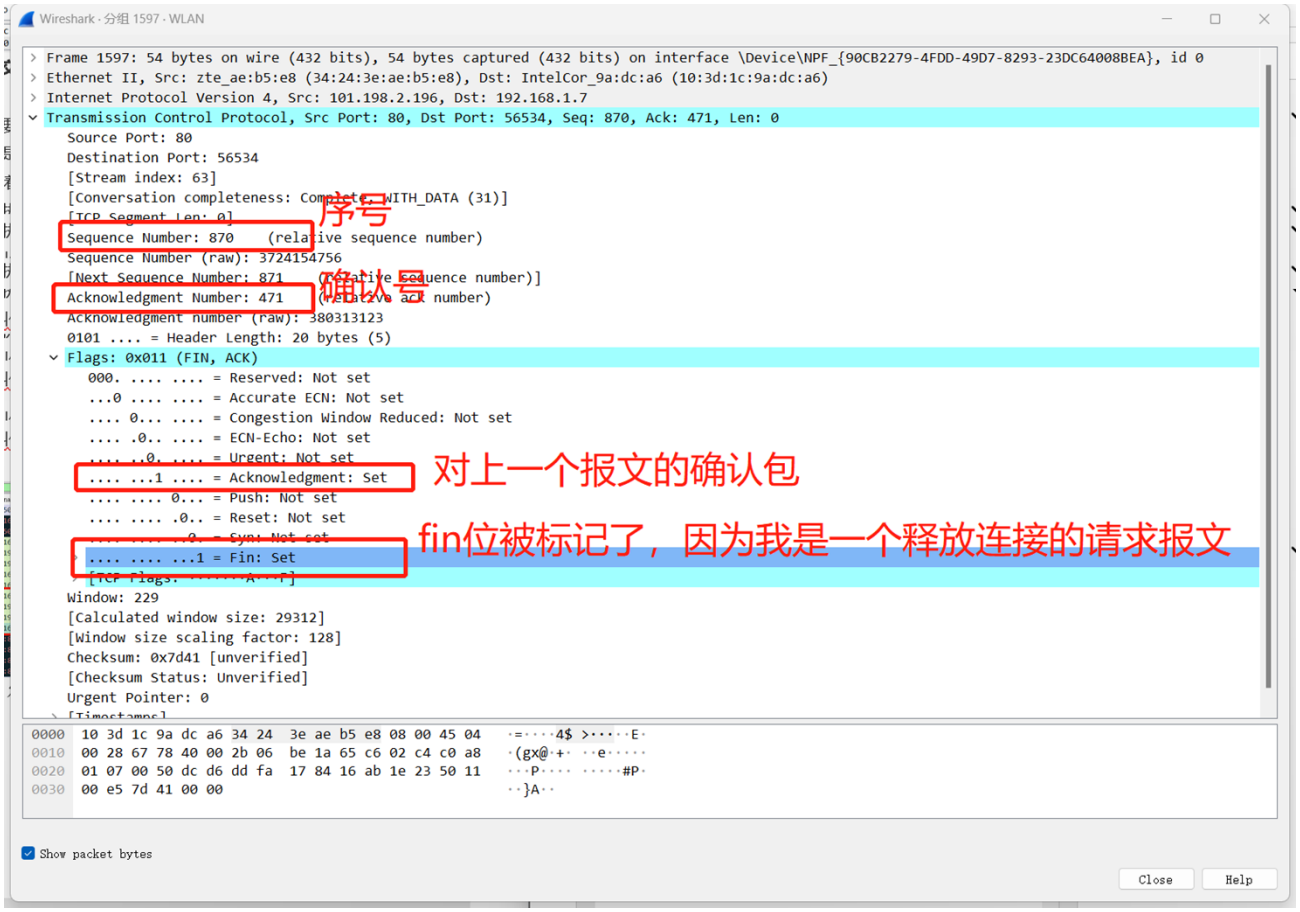
```
> Frame 27: 365 bytes on wire (2920 bits), 365 bytes captured (2920 bits) on interface \Device\NPF_{90CB2279-4FDD-49D7-8293-23DC64008BEA}, id 0
> Ethernet II, Src: IntelCor_9a:dc:a6 (10:3d:1c:9a:dc:a6), Dst: zte_ae:b5:e8 (34:24:3e:ae:b5:e8)
> Internet Protocol Version 4, Src: 192.168.1.7, Dst: 44.230.35.225
v Transmission Control Protocol, Src Port: 58604, Dst Port: 443, Seq: 1, Ack: 1, Len: 311
    Source Port: 58604
    Destination Port: 443
    [Stream index: 3]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 311]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 739516859
    [Next Sequence Number: 312    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 2707660784
    0101 .... = Header Length: 20 bytes (5)
  v 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: 517
    [Calculated window size: 132352]
    [Window size scaling factor: 256]
    Checksum: 0x13c8 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
    TCP payload (311 bytes)
v Transport Layer Security
  v TLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 306
    > Handshake Protocol: Client Hello

0000  34 24 3e ae b5 e8 10 3d  1c 9a dc a6 08 00 45 00   4$>····= ·····E·
0010  01 5f e6 07 40 00 80 06  00 00 c0 a8 01 07 2c e6   ·_··@··· ······,·
```

Task 3：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。

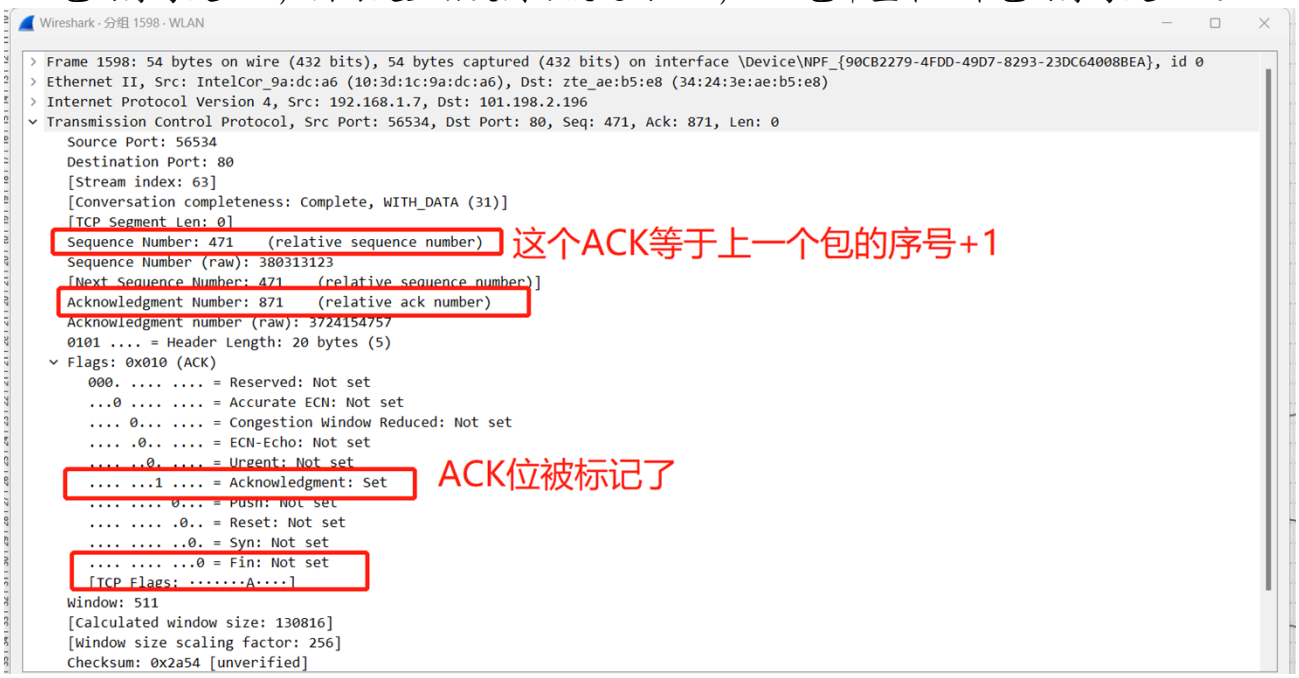
当通信双方完成数据传输，需要进行TCP连接的释放，由于TCP连接是全双工的，因此每个方向都必须单独进行关闭。这个原则是当一方完成它的数据发送任务后就能发送一个FIN来终止这个方向的连接。收到一个 FIN只意味着这一方向上没有数据流动，一个TCP连接在收到一个FIN后仍能发送数据。首先进行关闭的一方将执行主动关闭，而另一方执行被动关闭。因为正常关闭过程需要发送4个TCP帧，因此这个过程也叫作4次挥手。如下图所示，Wireshark截获到了四次挥手的四个数据包。



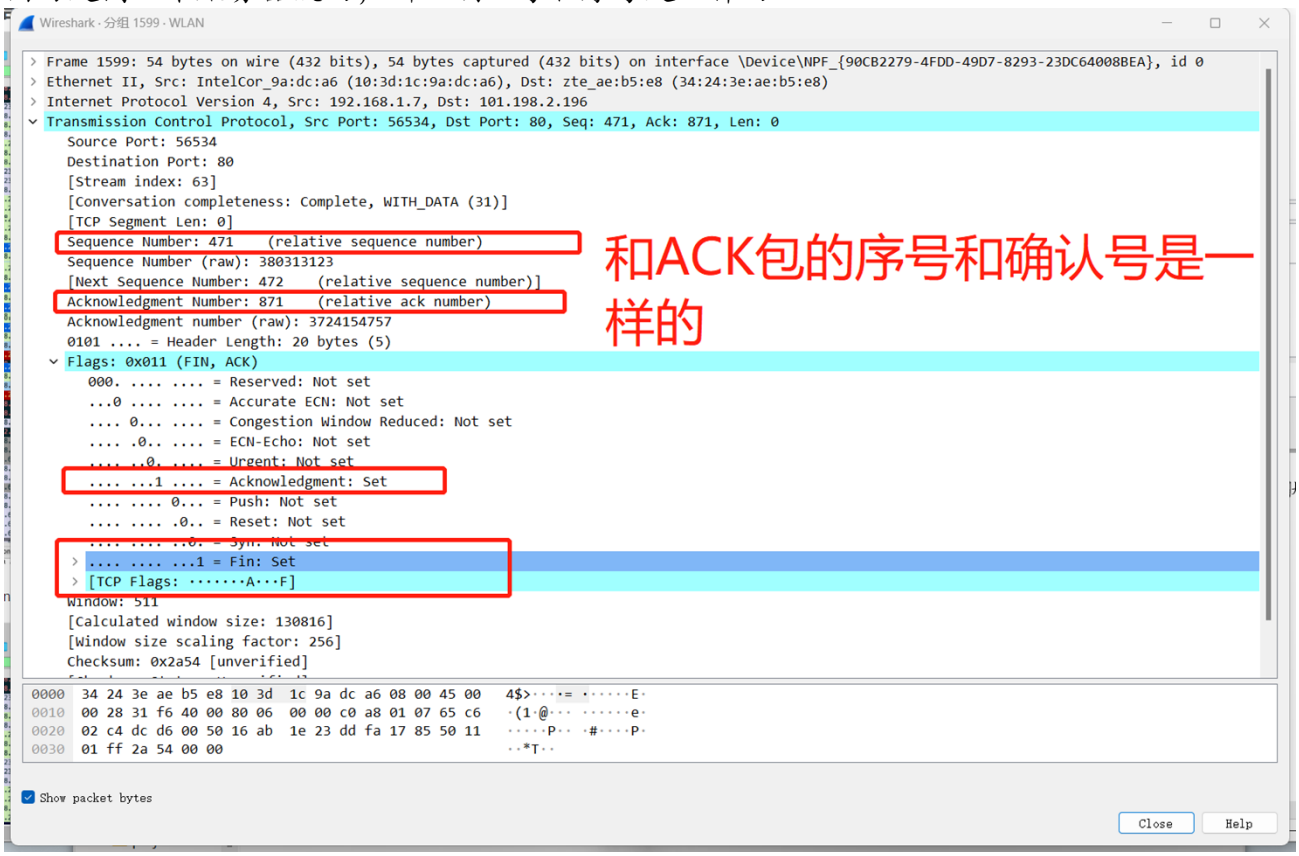
第一次挥手，主动关闭方发送 FIN=1，ACK=1，SEQ=870，ACK_SEQ=471，状态 FIN_WAIT_1。



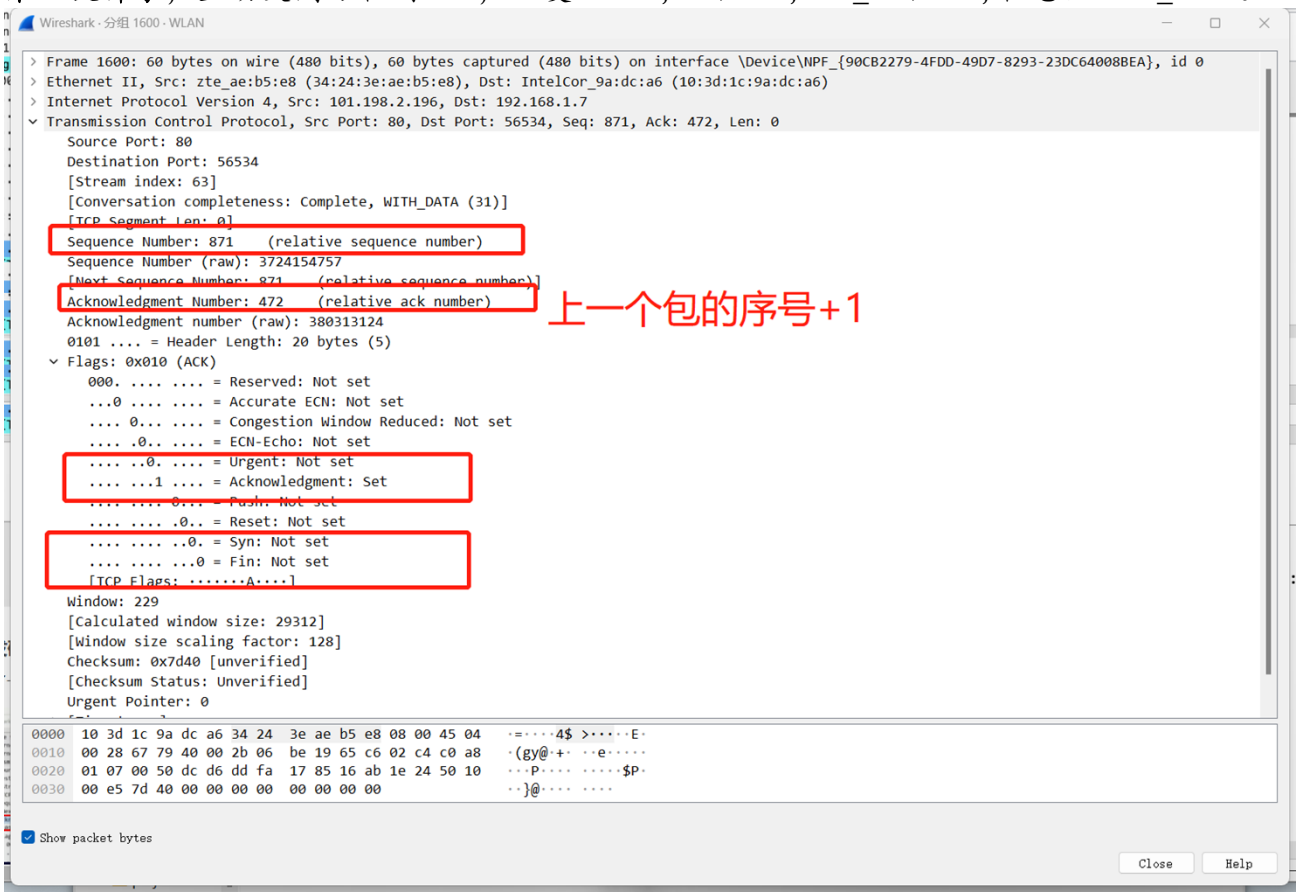
第二次挥手，被动关闭方收到 FIN，回复 ACK=1,SEQ=471,ACK_SEQ=871,状态 CLOSE_WAIT。
只有 ACK 位被标记了，其他位没有被标记，因为这就是一个确认消息。
ACK 包的序号是 471，因为是主动关闭方发送的 fin，ACK 包希望下一个包的序号是 471。



第三次挥手：close，被动关闭方发送 FIN=1（告诉主动关闭方我也要释放连接了），ACK=1. SEQ=471,ACK_SEQ=871,状态：LAST_ACK
因为是同一个服务器发的，所以确认号和序号是一样的



第四次挥手，主动关闭方收到 FIN，回复 ACK=1,SEQ=871,ACK_SEQ=472,状态：TIME_WAIT。



上一个包的序号+1

五、总结

通过本次实验，我使用Wireshark 了解到了TCP 协议，明确了TCP协议的工作原理，更直观地认识到了TCP 建立连接三次握手和断开连接四次握手的过程，并对具体的数据包进行分析，对TCP协议有了更深刻的理解。