**实验二 TCP协议分析**

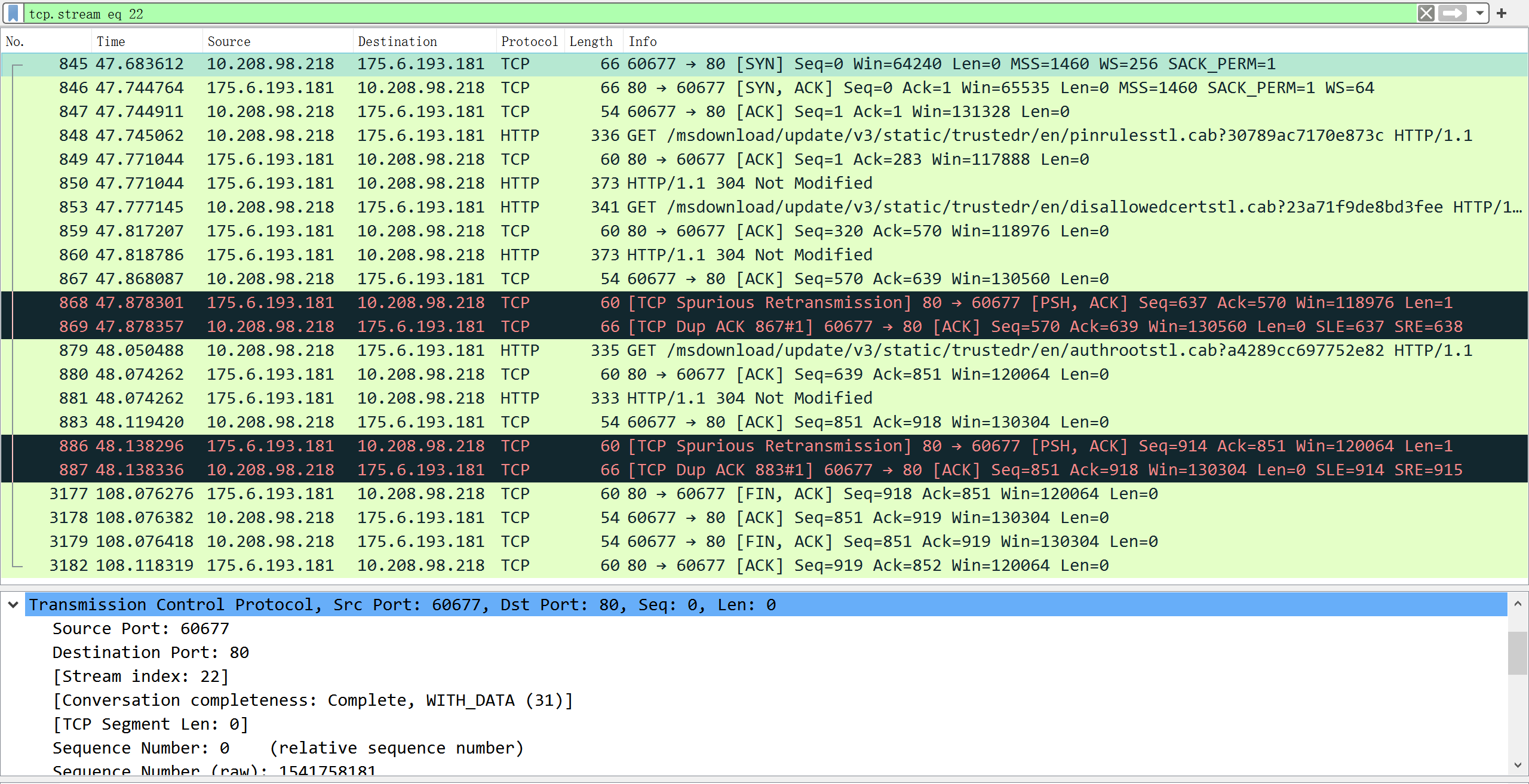
09019204 曹邹颖

1. **实验内容**

观察 TCP 三次握手与四次挥手报文，注意报文收发过程中，双方 TCP 状态的变化。以本次捕获的报文为依据，分别画出本次 TCP 连接三次握手与四次挥手的时序图，结合 TCP 状态机，在双方各阶段标出对应的TCP 状态。选择其中一个TCP 报文，配合Wireshark 截图，分析该报文 TCP 首部各字段的定义、值及其含义。

1. **实验步骤与分析**
2. **分析TCP三次握手与四次挥手报文**
3. **选择一个TCP会话**

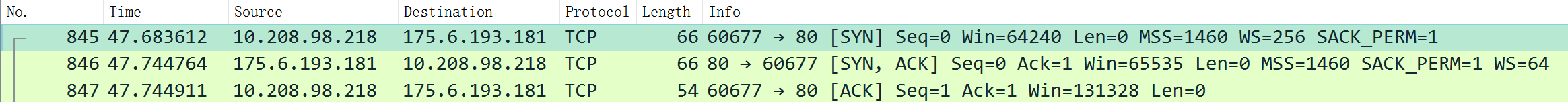
打开Wireshark选用WLAN进行捕获，此时打开浏览器访问一个网站，在捕获的流量数据里，鼠标点击任何一个TCP数据包，右键菜单中“追踪流”功能，再选择TCP，Wireshark就会显示这个TCP会话所有的数据包，并且列表在一个新的窗口中显示。



1. **分析TCP三次握手**
2. **首先找到TCP连接建立过程的三次握手**

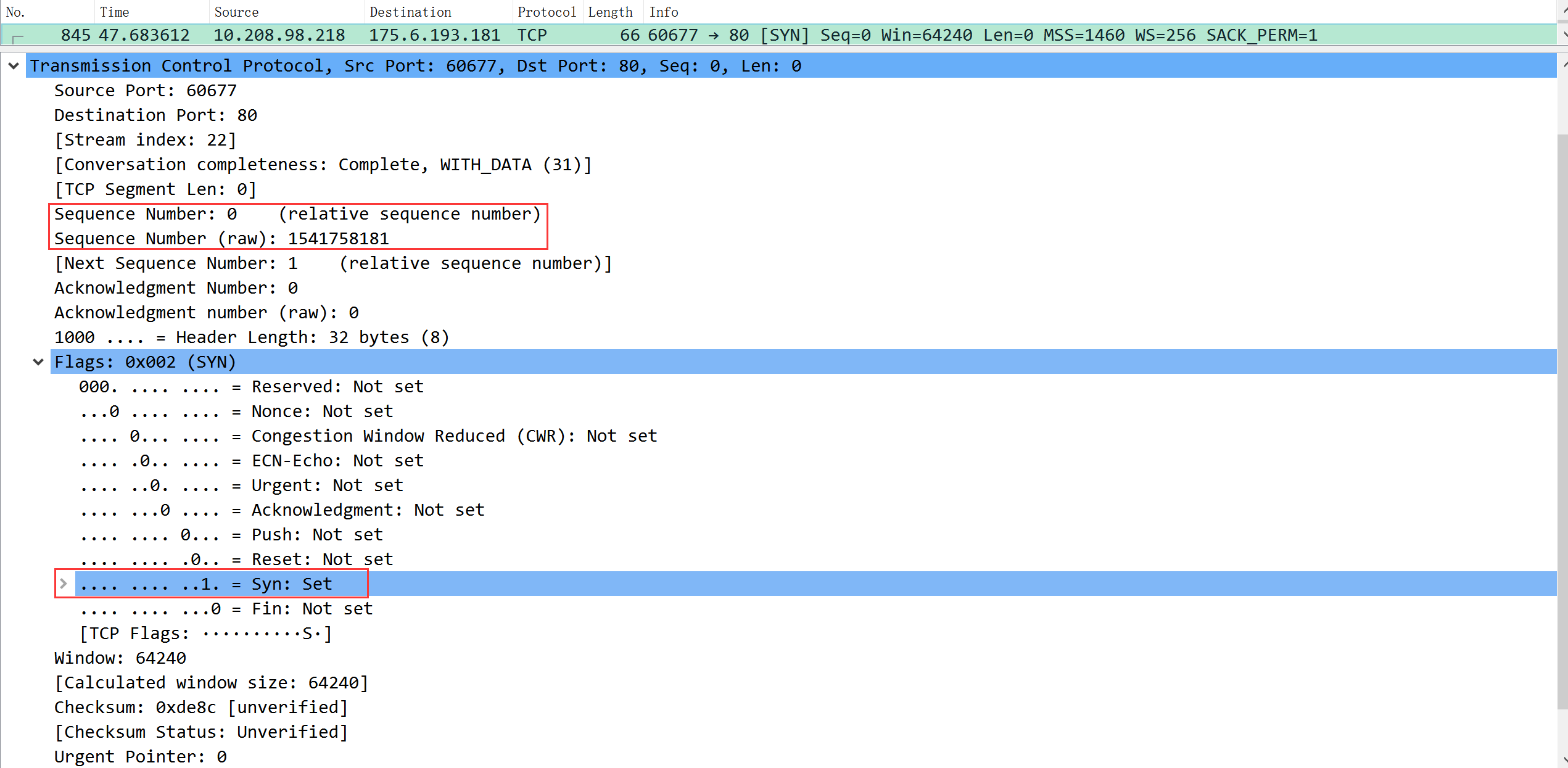
通过追踪任何一个TCP数据流，这个数据流开始的三个数据包都是其连接建立过程的三次握手。或者，也可以使用Flags标志位进行检索，例如三次握手的第二个数据包非常特殊，tcp.flags.syn == 1 && tcp.flags.ack == 1，可以利用这个特点发现一个三次握手过程。

因此找到TCP连接建立过程的三次握手如下：



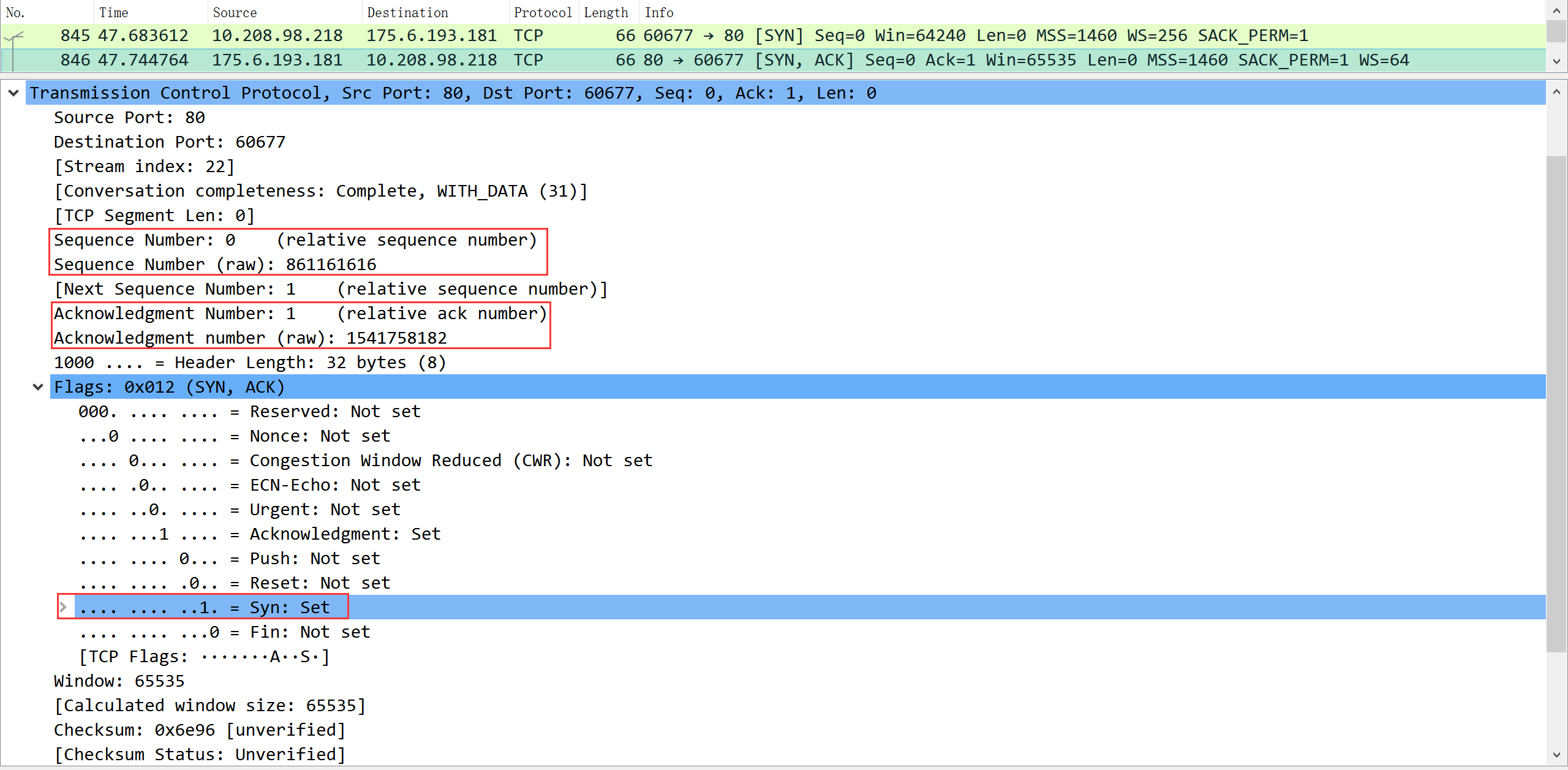
1. **第一次握手**

客户端向服务器发送连接请求报文，标志位SYN置1，随机选择的初始序号为1541758181，相对序号为0，为方便后续分析，这里序号选取相对序号seq=client\_isn=0。



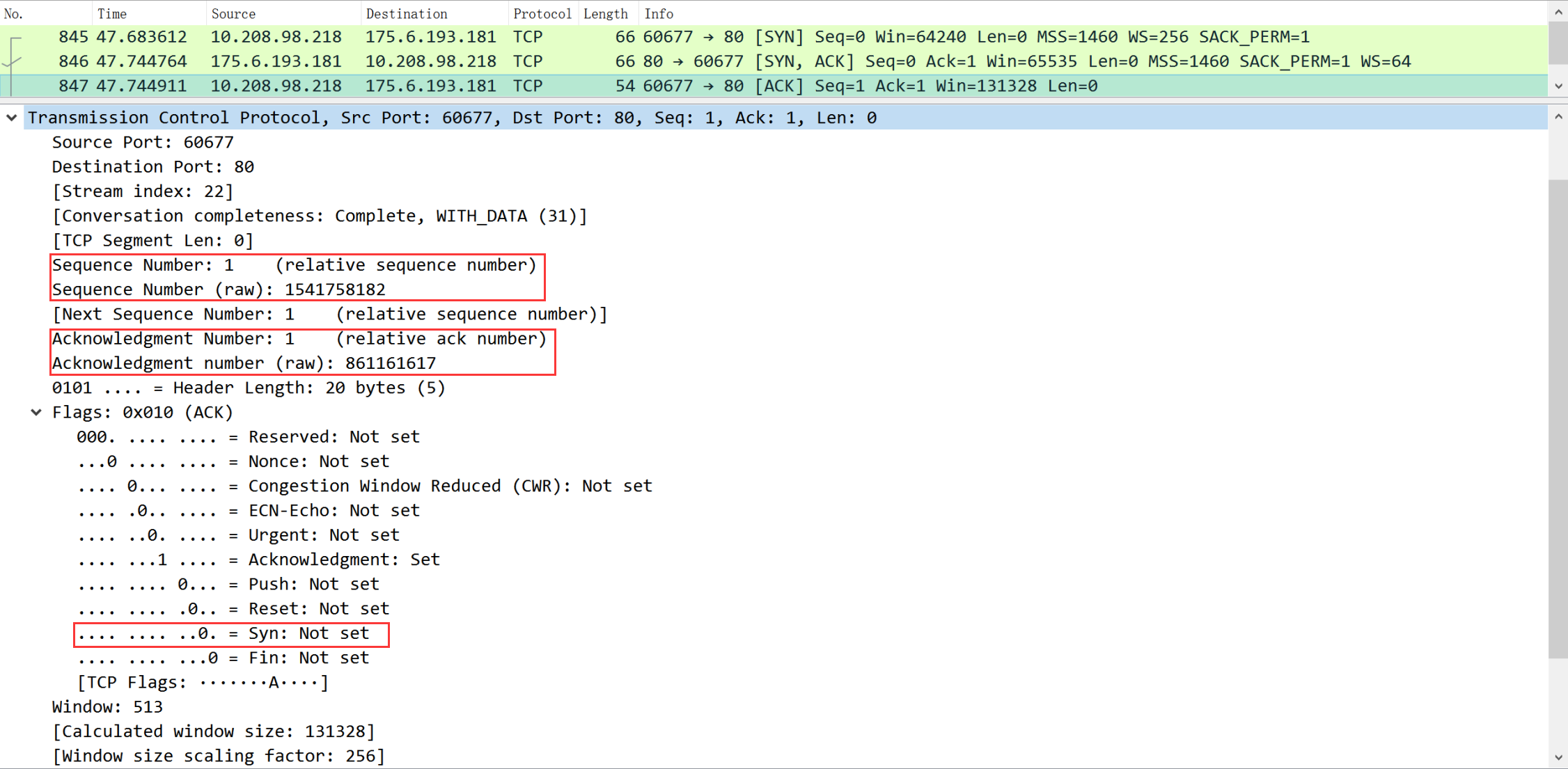
1. **第二次握手**

服务器收到客户端TCP SYN报文段，为该TCP连接分配TCP缓存和变量并向该客户TCP发送允许连接的报文段，该报文SYN被置为1，ACK（相对）被置为客户序列号+1=client\_isn+1=1（看原始ACK也是客户原始序列号+1=1541758182），自己初始序号选择为861161616，相对序号为0，同样为方便后续分析，选取seq=server\_isn=0；



1. **第三次握手**

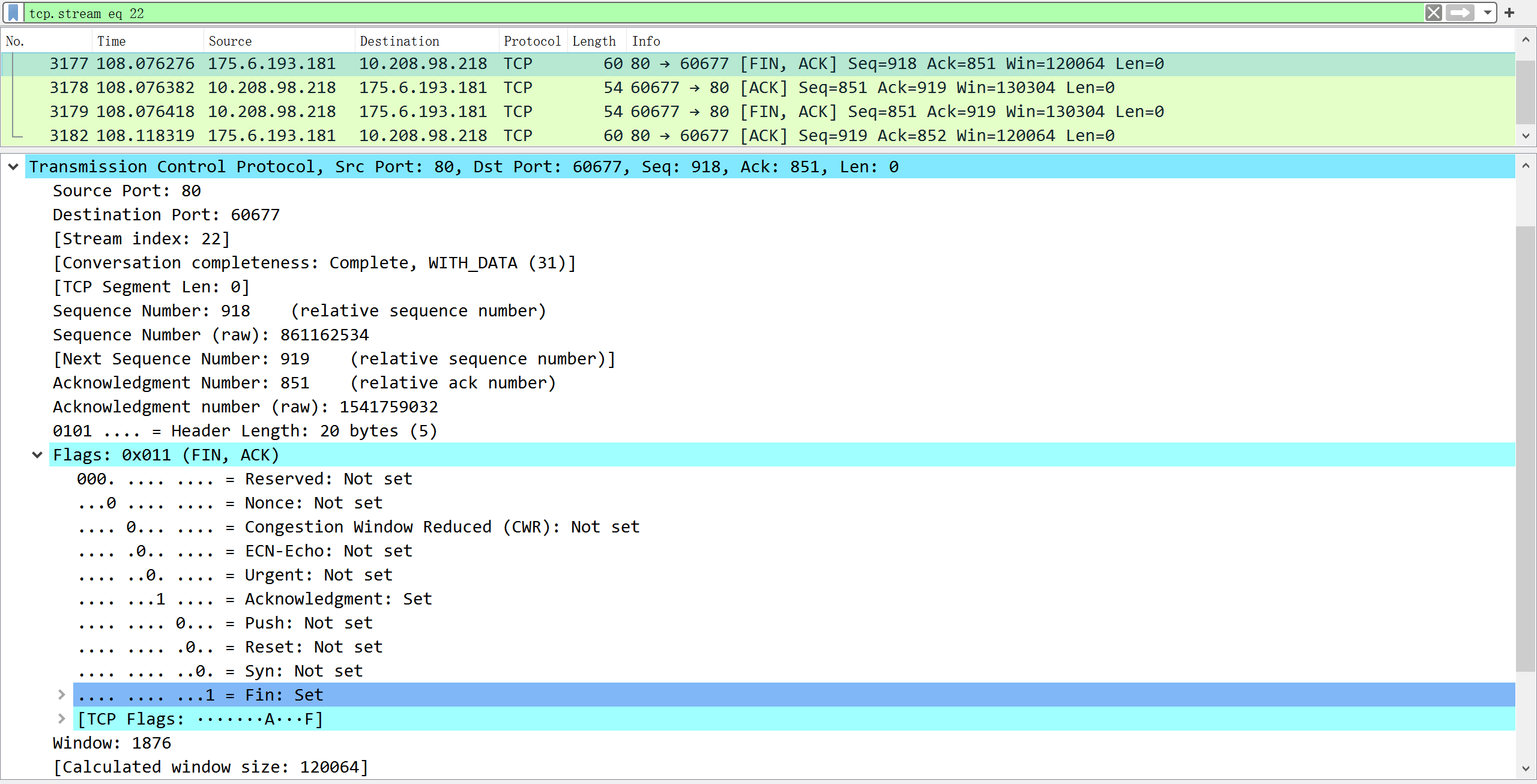
客户端收到服务器发来的SYNACK报文段后检查ACK是否正确，即第一次握手发送的序号加1 (client\_isn+1=1)，以及标志位ACK是否为1。若正确，客户端则向服务器发送另一个报文段，这最后一个报文段对服务器的允许连接的报文段进行确认，通过ACK置server\_isn+1=1，SYN置0来完成。第三次握手发送的报文段可在负载中接待客户到服务器的数据。一旦完成上述三个步骤，客户和服务器就可以传送数据了，在以后每一个报文段中SYN标志位均置0。



1. **分析TCP四次挥手**
2. **首先找到TCP断开连接的四次挥手**

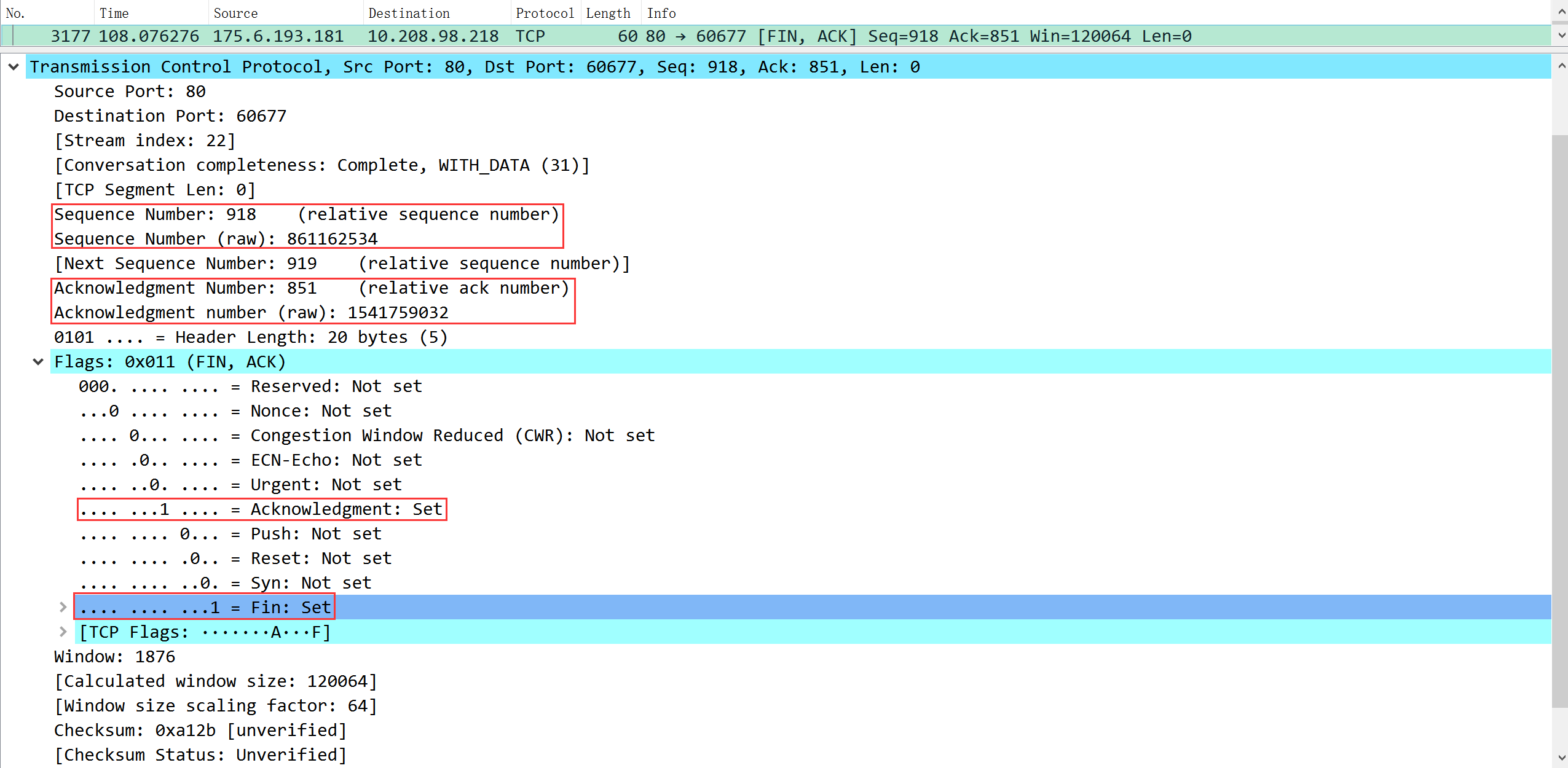
在上述追踪的TCP数据流最后的四个数据包恰是TCP连接的正常关闭过程的四次挥手。或者，也可以使用 Flags 标志位进行检索，例如发起TCP 连接终止的数据包非常特殊，tcp.flags.fin == 1，可以利用这个特点发现一次终止过程。

因此找到TCP断开连接的四次挥手如下：



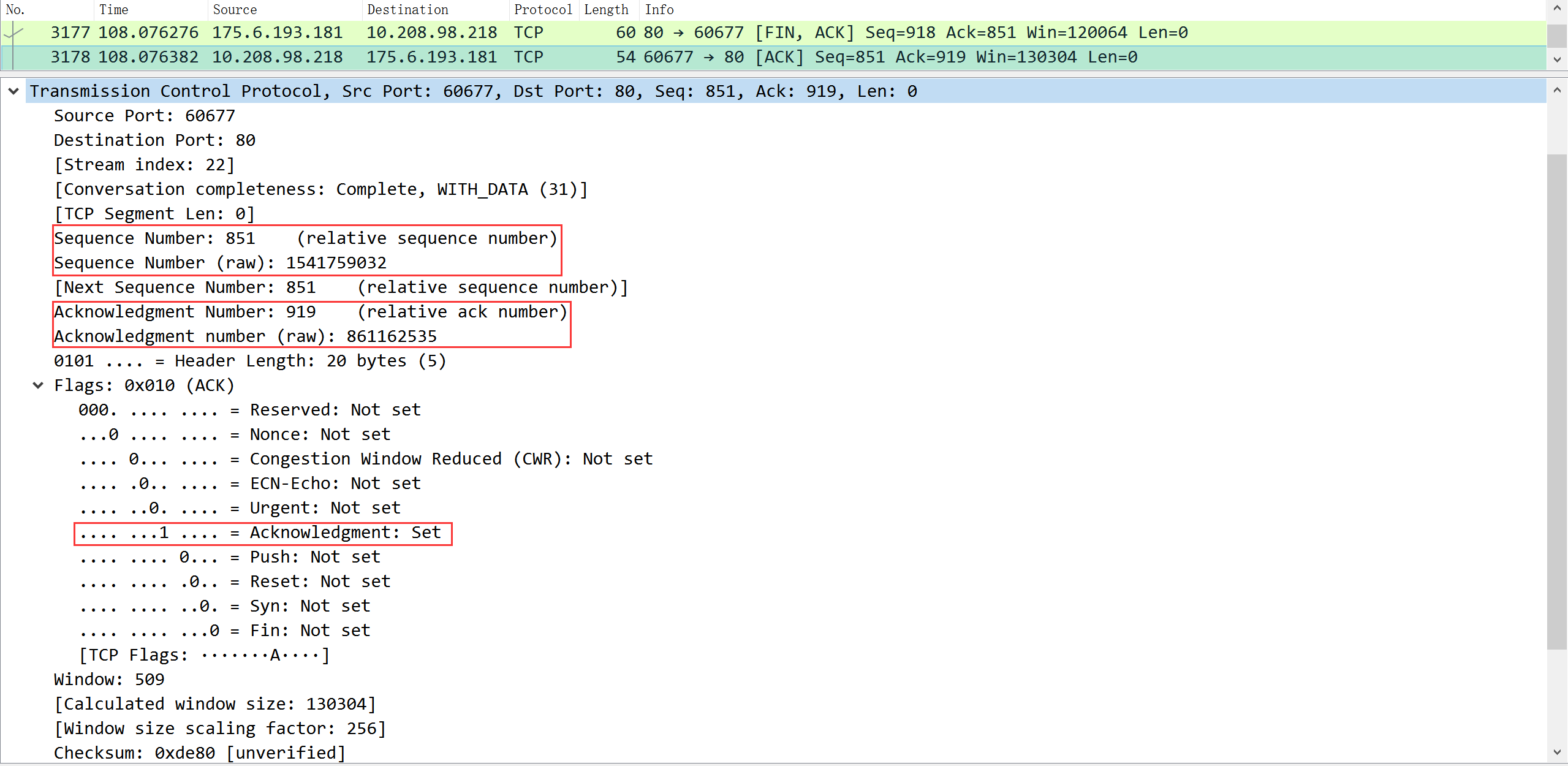
1. **第一次挥手**

当客户打算关闭连接时，向服务器发送一个用于关闭连接的TCP报文段，将标志位FIN和ACK置1，序号为1541758181，相对序号为918，确认序号（相对）为851。



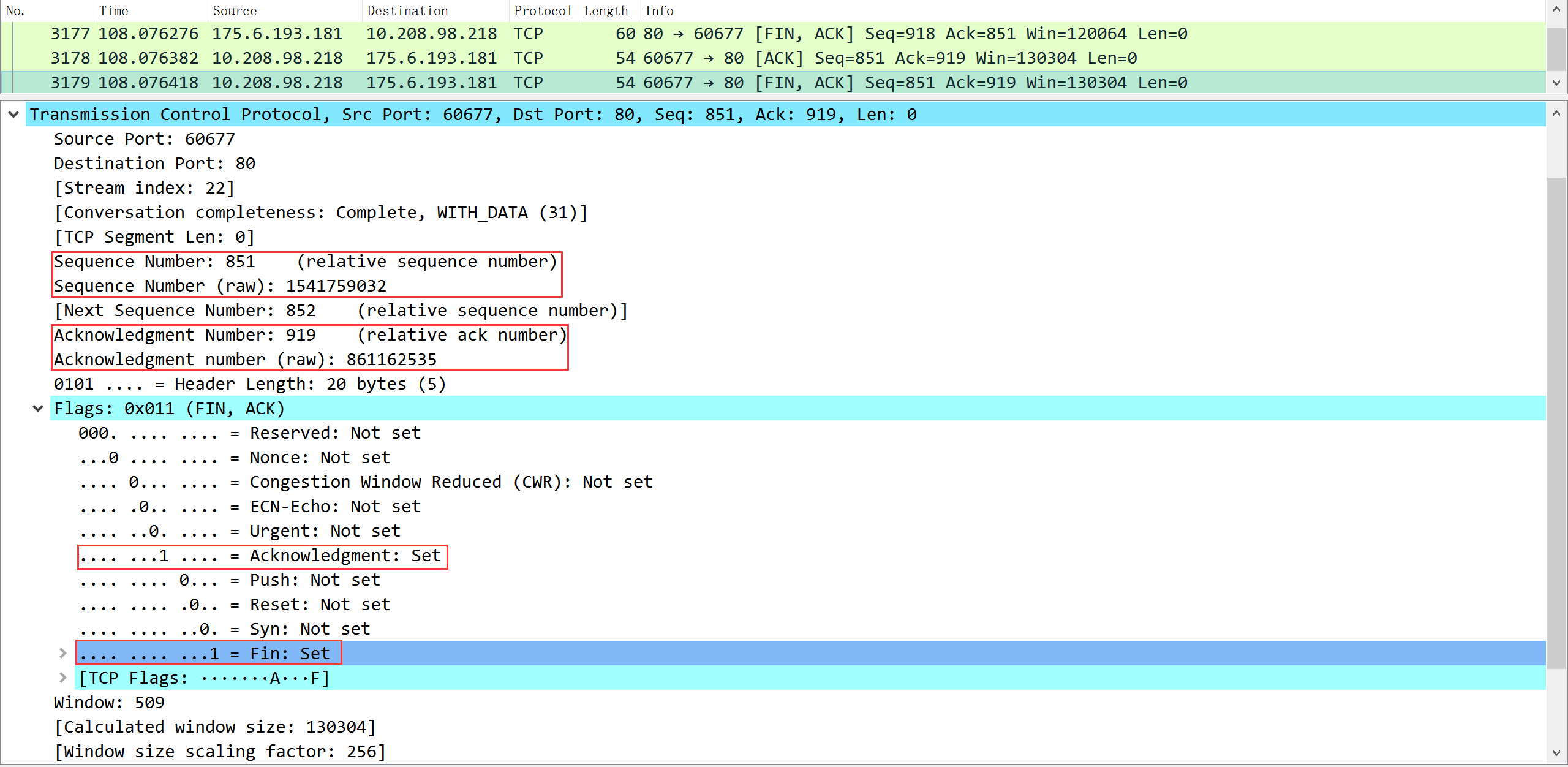
1. **第二次挥手**

当服务器收到该报文段后，就向发送方回送一个确认报文段，标志位ACK=1，序号（相对）为收到的确认序号=851，确认序号（相对）为收到的序号+1=918+1=919。



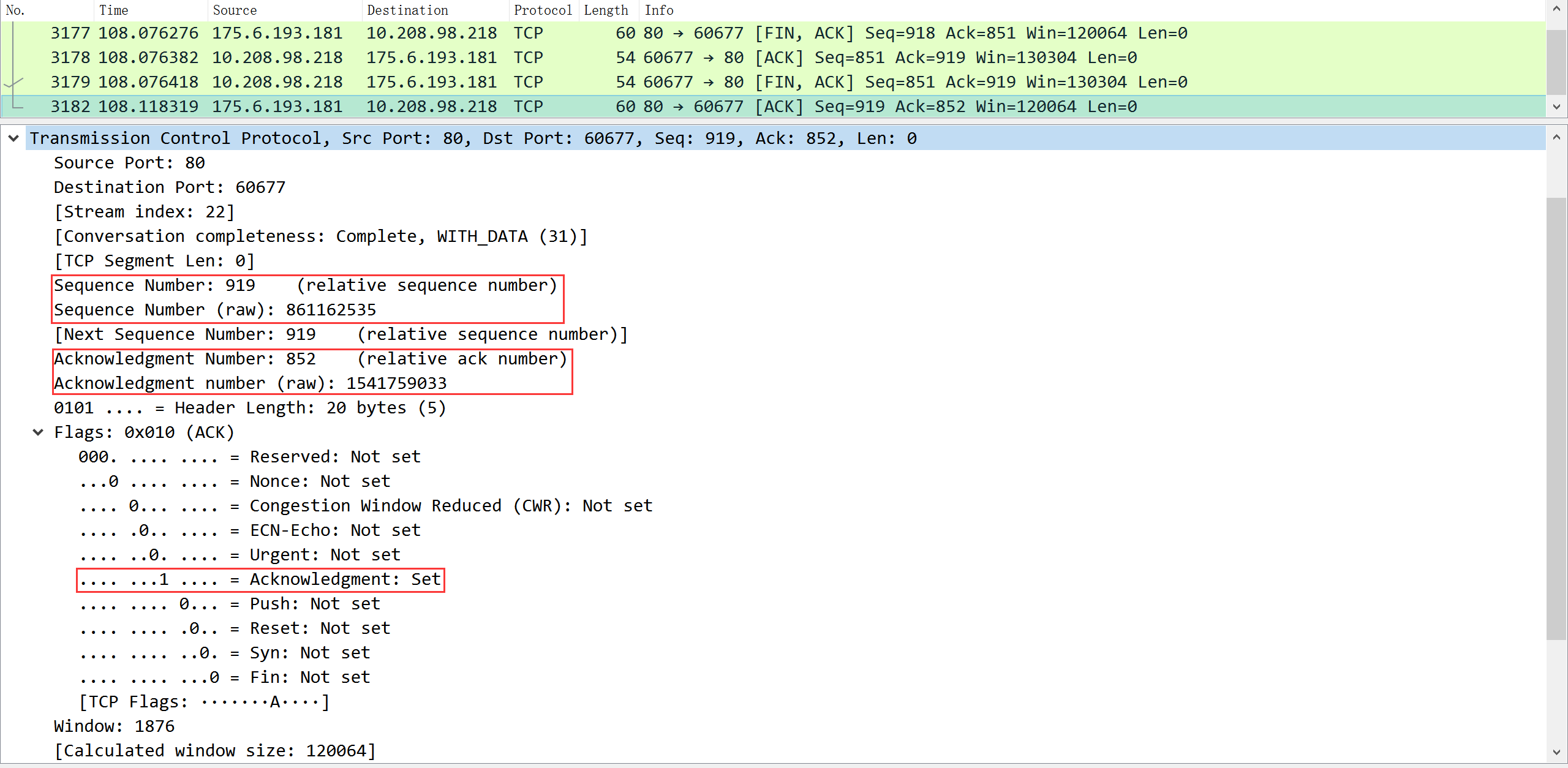
1. **第三次挥手**

服务器关闭与客户端的连接，发送它自己的终止报文段，其标志位FIN和ACK置1，序号（相对）为851，确认序号（相对）为919。

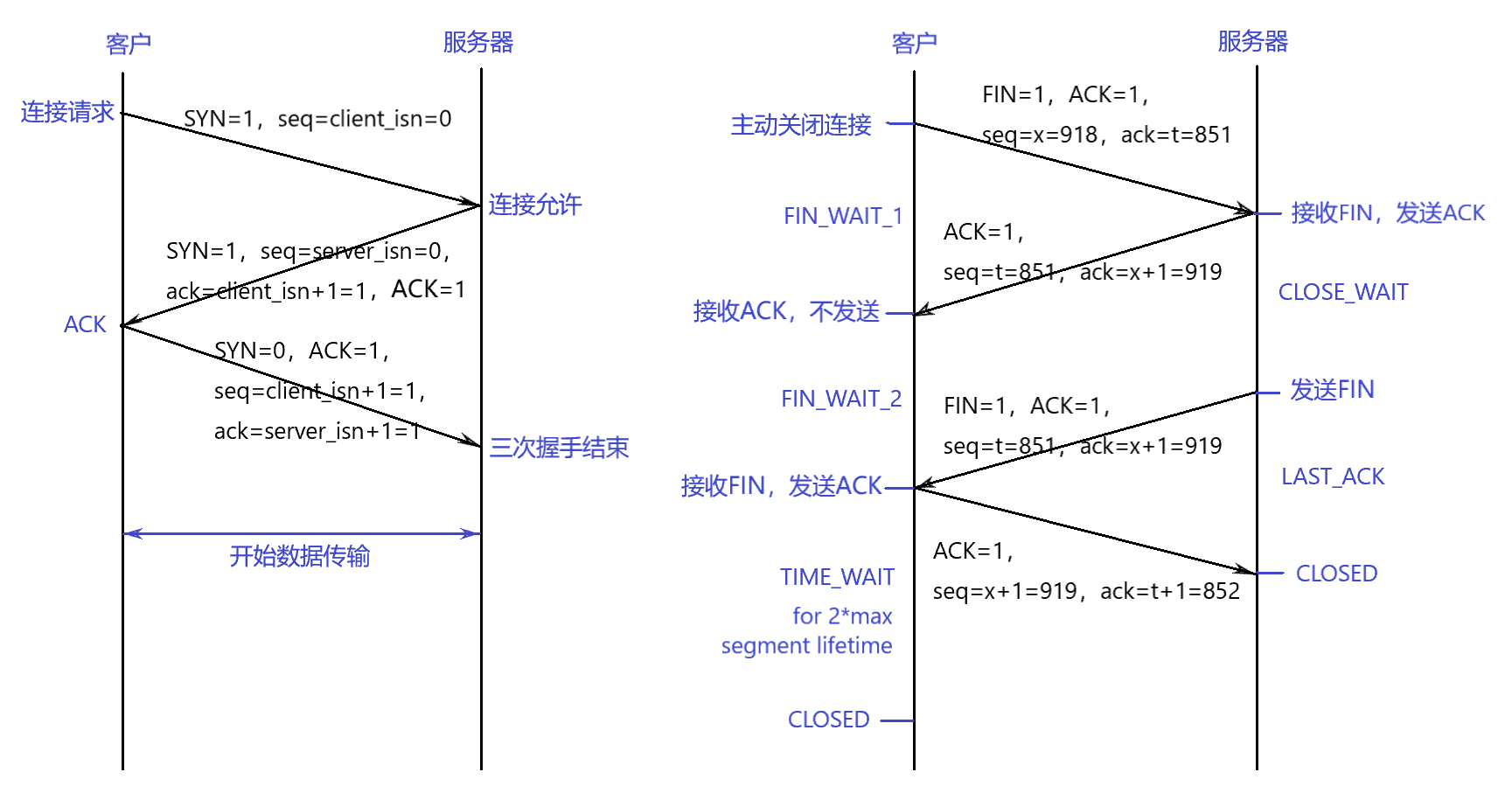


1. **第四次挥手**

客户端收到服务器发送的终止报文段之后，进行确认，发回一个标志位ACK置1的报文段，序号（相对）为收到的确认序号=919，确认序号（相对）为收到的序号+1=851+1=852。

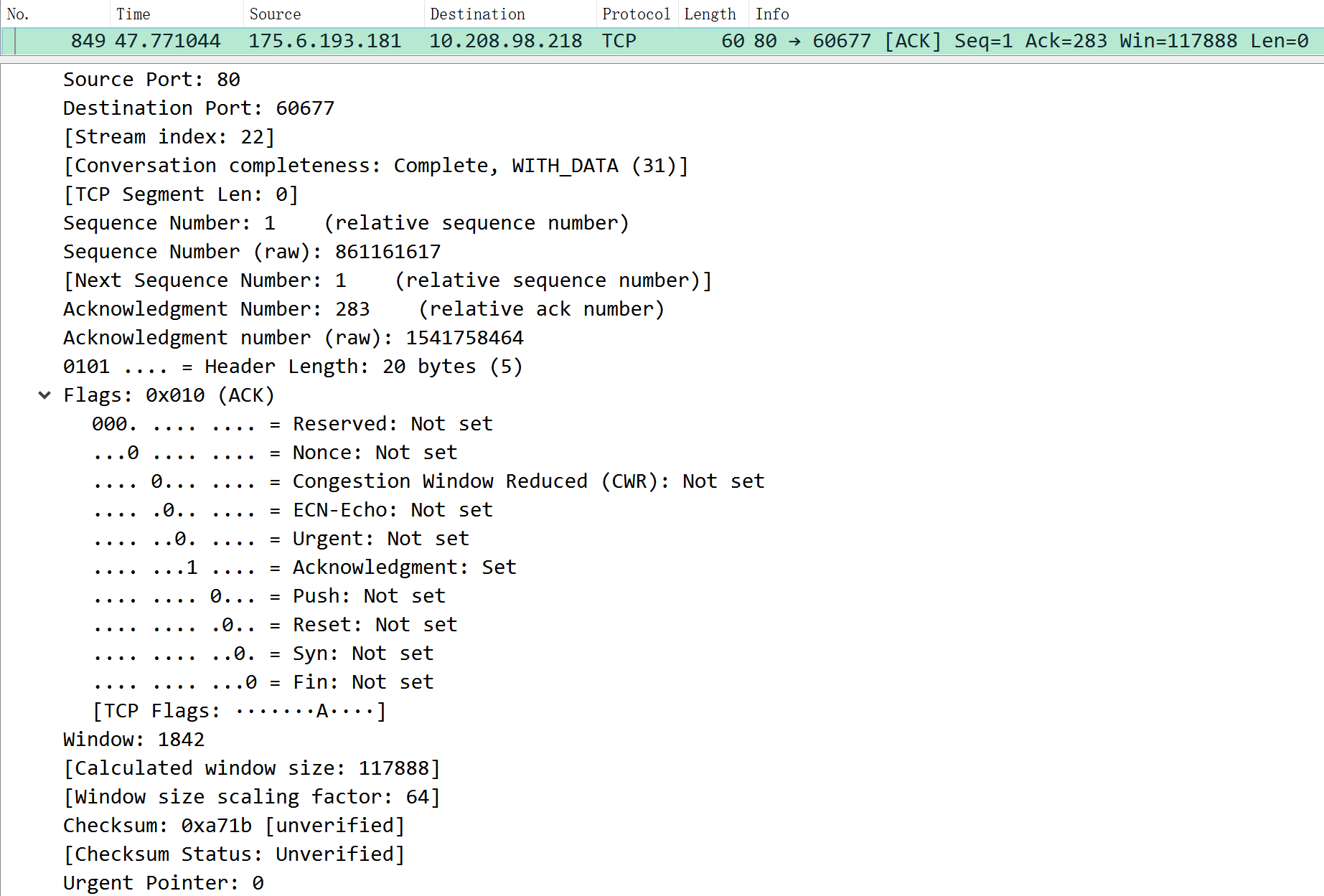


1. **本次TCP连接三次握手与四次挥手的时序图**

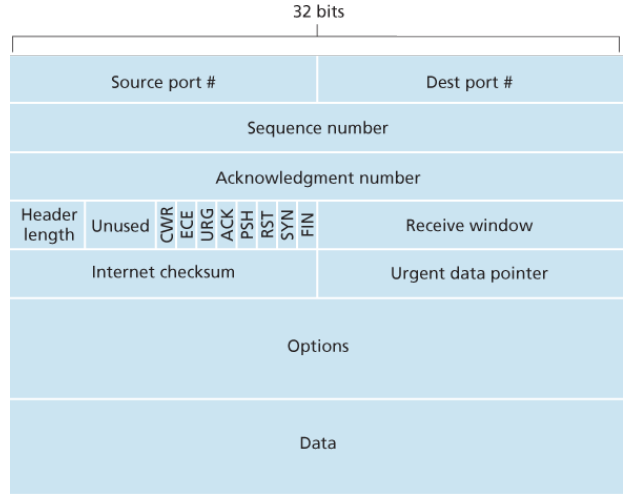


1. **分析一个TCP 报文**

选择追踪的该TCP数据流中任意一个TCP 报文如下，分析该报文TCP首部各字段的定义、值及其含义。



对照TCP报文的格式，逐句分析：



Source Port: 80 16位源端口号

Destination Port: 60677 16位目的端口号

[Stream index: 22]

[Conversation completeness: Complete, WITH\_DATA (31)]

[TCP Segment Len: 0]

Sequence Number: 1 (relative sequence number) 32位序列号

Sequence Number (raw): 861161617

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

Acknowledgment Number: 283 (relative ack number) 32位确认序列号

Acknowledgment number (raw): 1541758464

0101 .... = Header Length: 20 bytes (5) 4位头部长度（0101）+ Unused

Flags: 0x010 (ACK) Flags字段

000. .... .... = Reserved: Not set 保留

...0 .... .... = Nonce: Not set

.... 0... .... = Congestion Window Reduced (CWR): Not set

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

.... ..0. .... = Urgent: Not set当URG=1时，表示报文段中有紧急数据，应尽快传送

.... ...1 .... = Acknowledgment: Set ACK=1代表一个确认的TCP包，否则不是确认包

.... .... 0... = Push: Not set PSH=1代表接收端尽快的交付给应用进程

.... .... .0.. = Reset: Not set RST=1代表TCP连接中出现严重差错，必须释放连接， 再重新建立连接

.... .... ..0. = Syn: Not set在建立连接是用来同步序号。SYN=1，ACK=0表示连接请 求报文段；SYN=1，ACK=1表示同意建立连接。

.... .... ...0 = Fin: Not set FIN=1表示此报文段的发送端的数据已经发送完毕，并要求 释放传输连接。

[TCP Flags: ·······A····]

Window: 1842 用来控制对方发送的数据量，通知发放已确定的发送窗口上限。

[Calculated window size: 117888]

[Window size scaling factor: 64]

Checksum: 0xa71b [unverified] 校验和，检验的范围包括首部和数据这两部分

[Checksum Status: Unverified]

Urgent Pointer: 0 紧急指针在URG=1时有效，它指出本报文段中的紧急数据的字节数。

[Timestamps]

[SEQ/ACK analysis]

1. **实验体会**

本次实验通过使用Wireshark观察TCP 协议报文，分析通信时序，理解了TCP 连接管理的过程，掌握了TCP工作原理与实现，明确了TCP报文首部各字段的定义、值及其含义。

在观察TCP三次握手与四次挥手报文的过程中，我观察到在TCP四次挥手中第二、三两次挥手的seq序号是等于第一次挥手的ACK序号，原因是第二、三次挥手是服务器发送的报文，其seq序号应为服务器接收FIN，发送ACK前的上一条报文段序号＋其segment len=客户端对其确认的ACK序号即客户端期待从服务器收到的下一个报文的序号，通过此对TCP报文段序号和确认号的特征加深了认识。