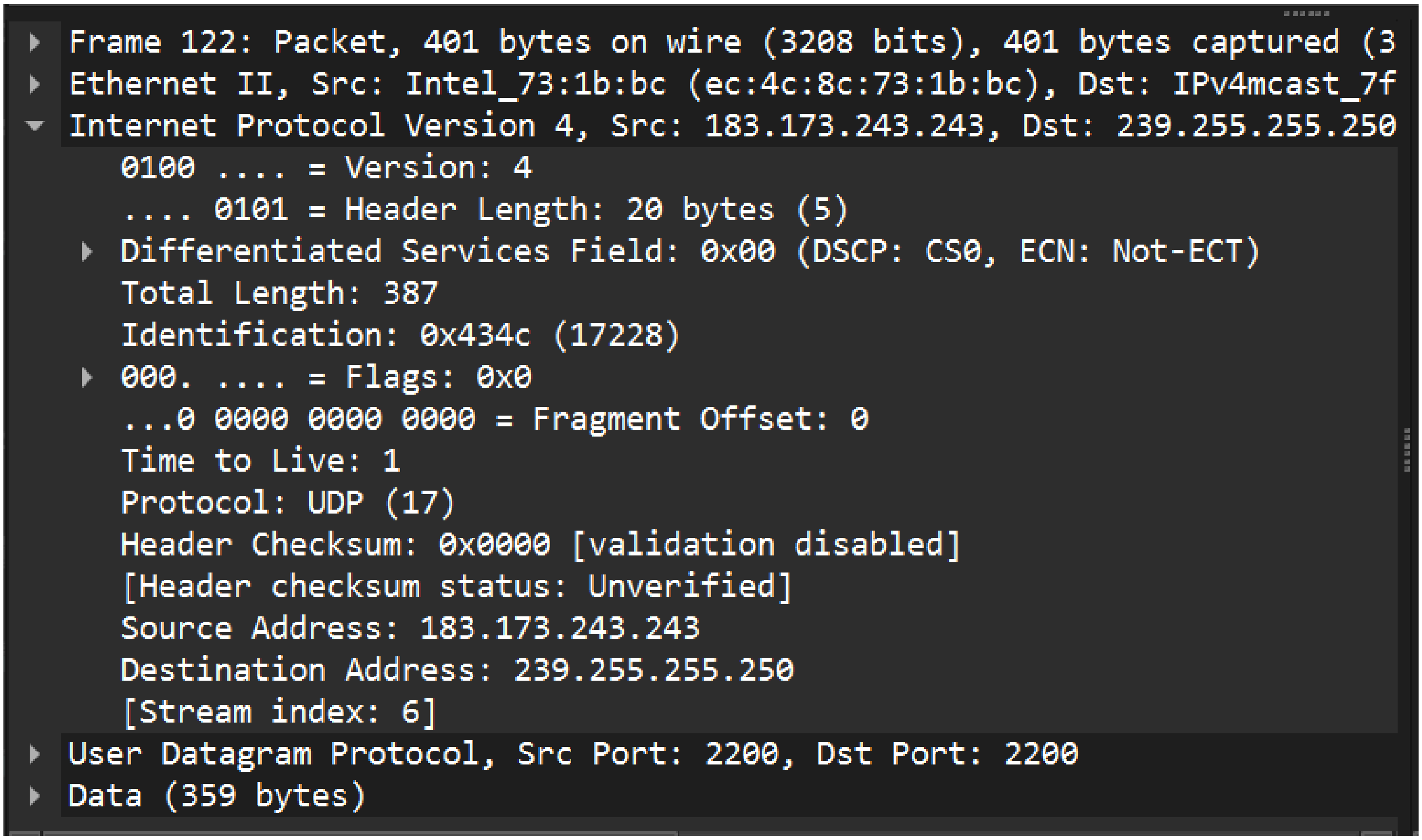
Wireshark 抓包实验报告

实验 1：观察 UDP 消息

(1) UDP 数据包在 IP 层的类型编号

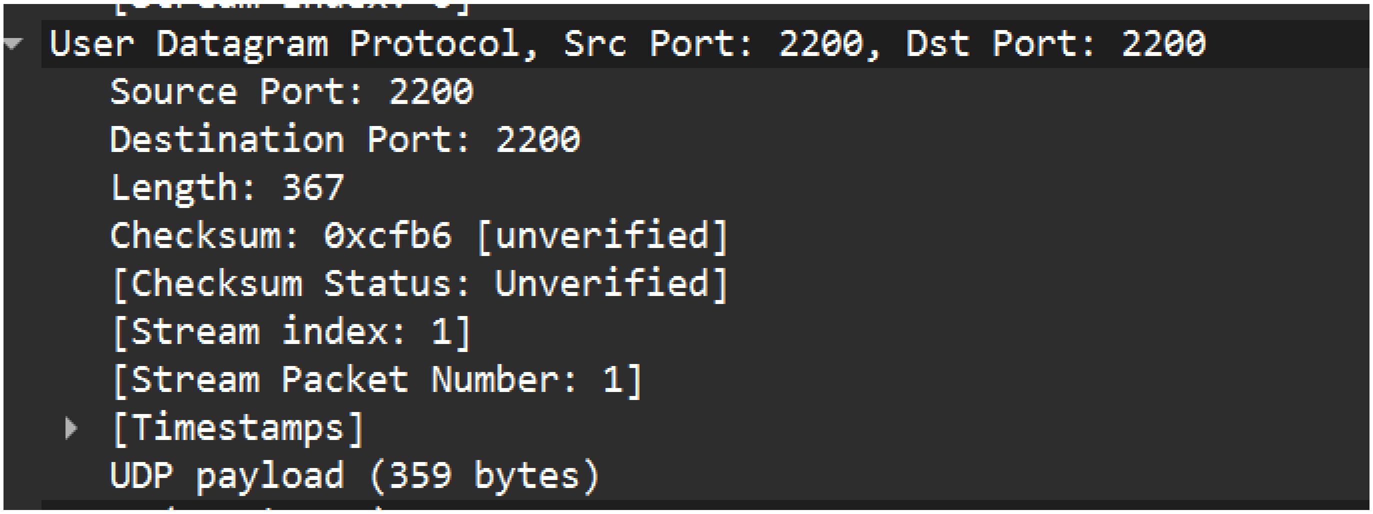
在 IP 数据报的首部中，“Protocol”字段标识了上层协议类型。对于 UDP 协议，其协议编号为 17（十六进制为 `0x11`）。



```
▸ Frame 122: Packet, 401 bytes on wire (3208 bits), 401 bytes captured (3
▸ Ethernet II, Src: Intel_73:1b:bc (ec:4c:8c:73:1b:bc), Dst: IPv4mcast_7f
▾ Internet Protocol Version 4, Src: 183.173.243.243, Dst: 239.255.255.250
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 387
    Identification: 0x434c (17228)
  ▸ 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 1
    Protocol: UDP (17)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 183.173.243.243
    Destination Address: 239.255.255.250
    [Stream index: 6]
▸ User Datagram Protocol, Src Port: 2200, Dst Port: 2200
▸ Data (359 bytes)
```

(2) UDP 数据包头字段依次是?

UDP 首部共有 8 个字节，字段依次为：源端口号 (Source Port)、目的端口号 (Destination Port)、长度 (Length) 以及校验和 (Checksum)。

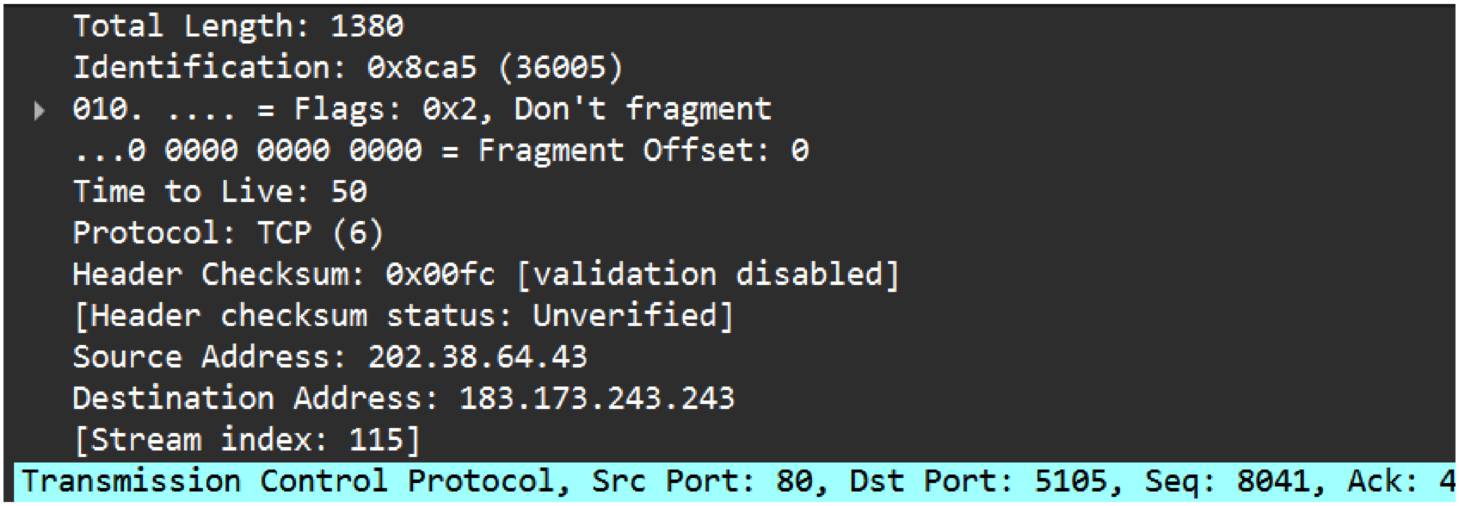


```
User Datagram Protocol, Src Port: 2200, Dst Port: 2200
    Source Port: 2200
    Destination Port: 2200
    Length: 367
    Checksum: 0xcfb6 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
    [Stream Packet Number: 1]
    [Timestamps]
    UDP payload (359 bytes)
```

实验 2：观察 TCP 消息

(1) TCP 数据包在 IP 层的类型编号

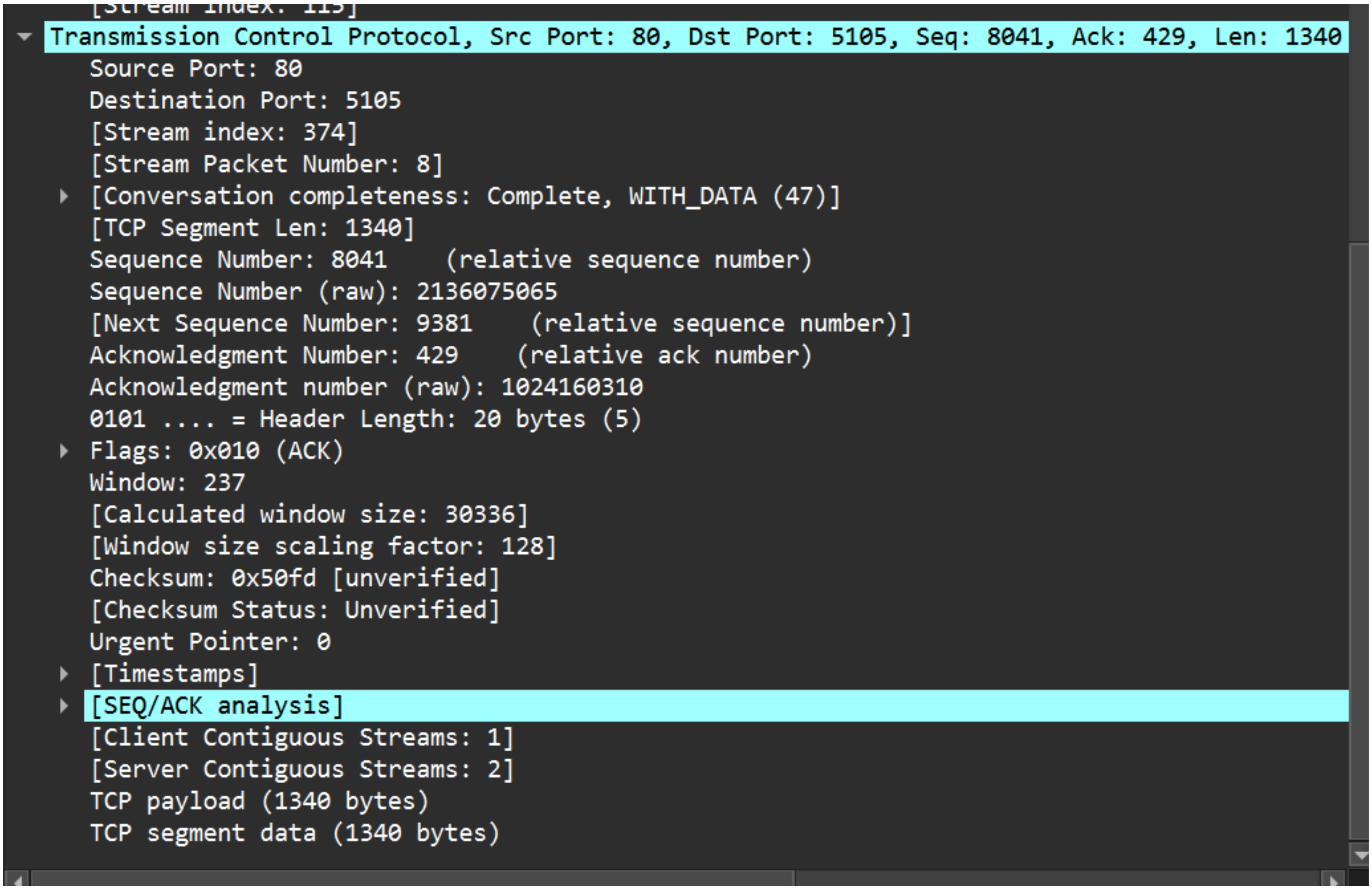
在 IP 数据报首部的“Protocol”字段中，TCP 协议的类型编号为 6。



```
    Total Length: 1380
    Identification: 0x8ca5 (36005)
    010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 50
    Protocol: TCP (6)
    Header Checksum: 0x00fc [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 202.38.64.43
    Destination Address: 183.173.243.243
    [Stream index: 115]
Transmission Control Protocol, Src Port: 80, Dst Port: 5105, Seq: 8041, Ack: 4
```

(2) TCP 数据包头字段依次是?

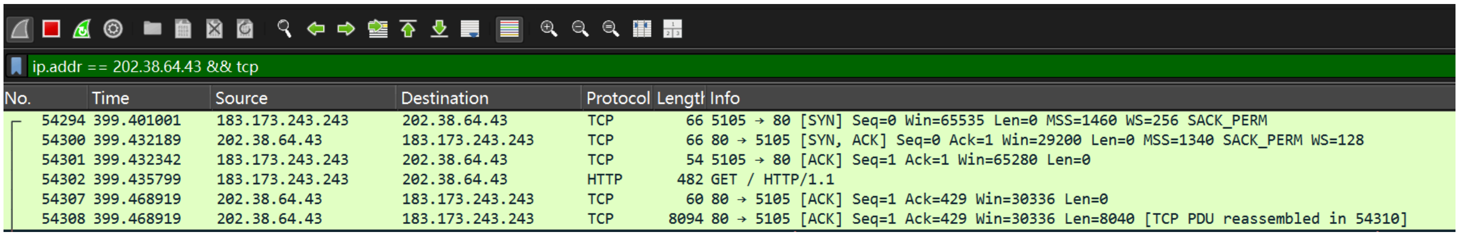
TCP 首部包含：源端口、目的端口、序列号、确认号、数据偏移、保留位、控制标志位（URG/ACK/PSH/RST/SYN/FIN）、窗口大小、校验和、紧急指针以及可选的选项字段。



```
    [Stream index: 115]
▾ Transmission Control Protocol, Src Port: 80, Dst Port: 5105, Seq: 8041, Ack: 429, Len: 1340
      Source Port: 80
      Destination Port: 5105
      [Stream index: 374]
      [Stream Packet Number: 8]
    ▸ [Conversation completeness: Complete, WITH_DATA (47)]
      [TCP Segment Len: 1340]
      Sequence Number: 8041    (relative sequence number)
      Sequence Number (raw): 2136075065
      [Next Sequence Number: 9381    (relative sequence number)]
      Acknowledgment Number: 429    (relative ack number)
      Acknowledgment number (raw): 1024160310
      0101 .... = Header Length: 20 bytes (5)
    ▸ Flags: 0x010 (ACK)
      Window: 237
      [Calculated window size: 30336]
      [Window size scaling factor: 128]
      Checksum: 0x50fd [unverified]
      [Checksum Status: Unverified]
      Urgent Pointer: 0
    ▸ [Timestamps]
    ▸ [SEQ/ACK analysis]
      [Client Contiguous Streams: 1]
      [Server Contiguous Streams: 2]
      TCP payload (1340 bytes)
      TCP segment data (1340 bytes)
```

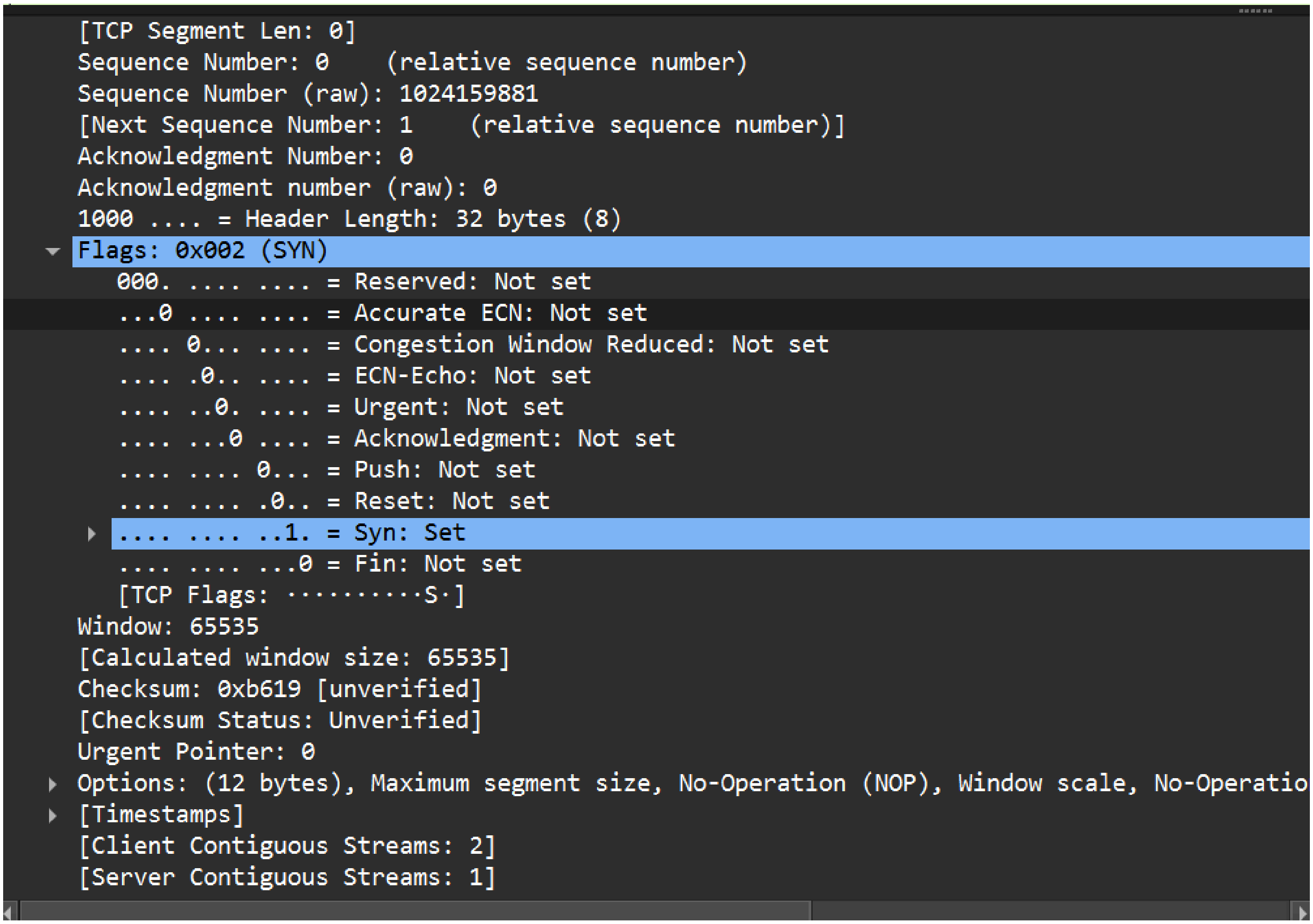
(3) 三次握手过程及选项协商

特点：第一次 (SYN)：标记位为 SYN，$Seq=0$，$Ack=0$。第二次 (SYN+ACK)：标记位为 SYN, ACK，$Seq=0$，$Ack=1$ (对原 Seq 加 1)。第三次 (ACK)：标记位为 ACK，$Seq=1$，$Ack=1$。选项协商例子：图中可见 MSS=1460，表示发送方在本连接中能接收的最大报文段长度为 1460 字节。

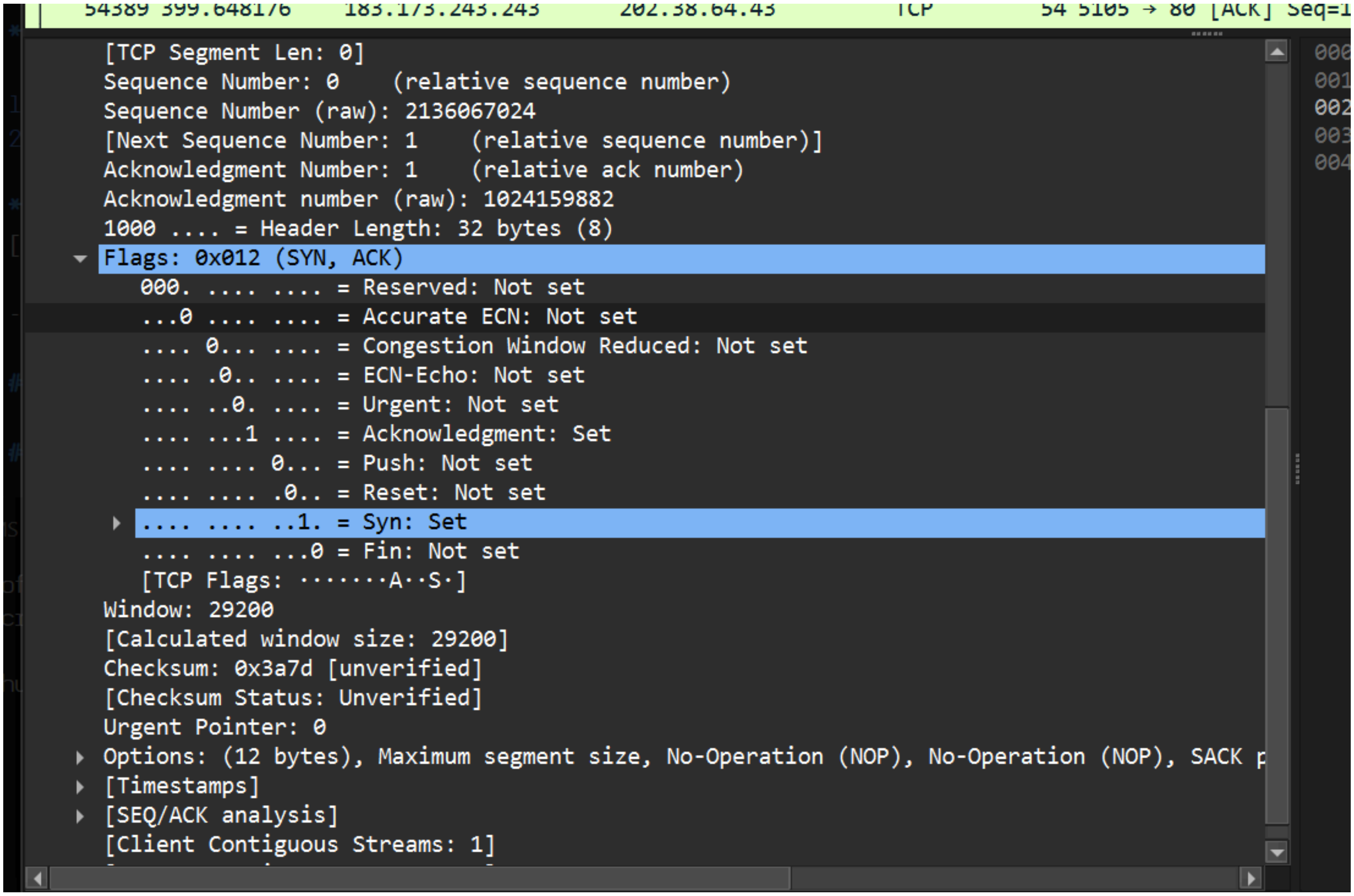


ip.addr == 202.38.64.43 && tcp

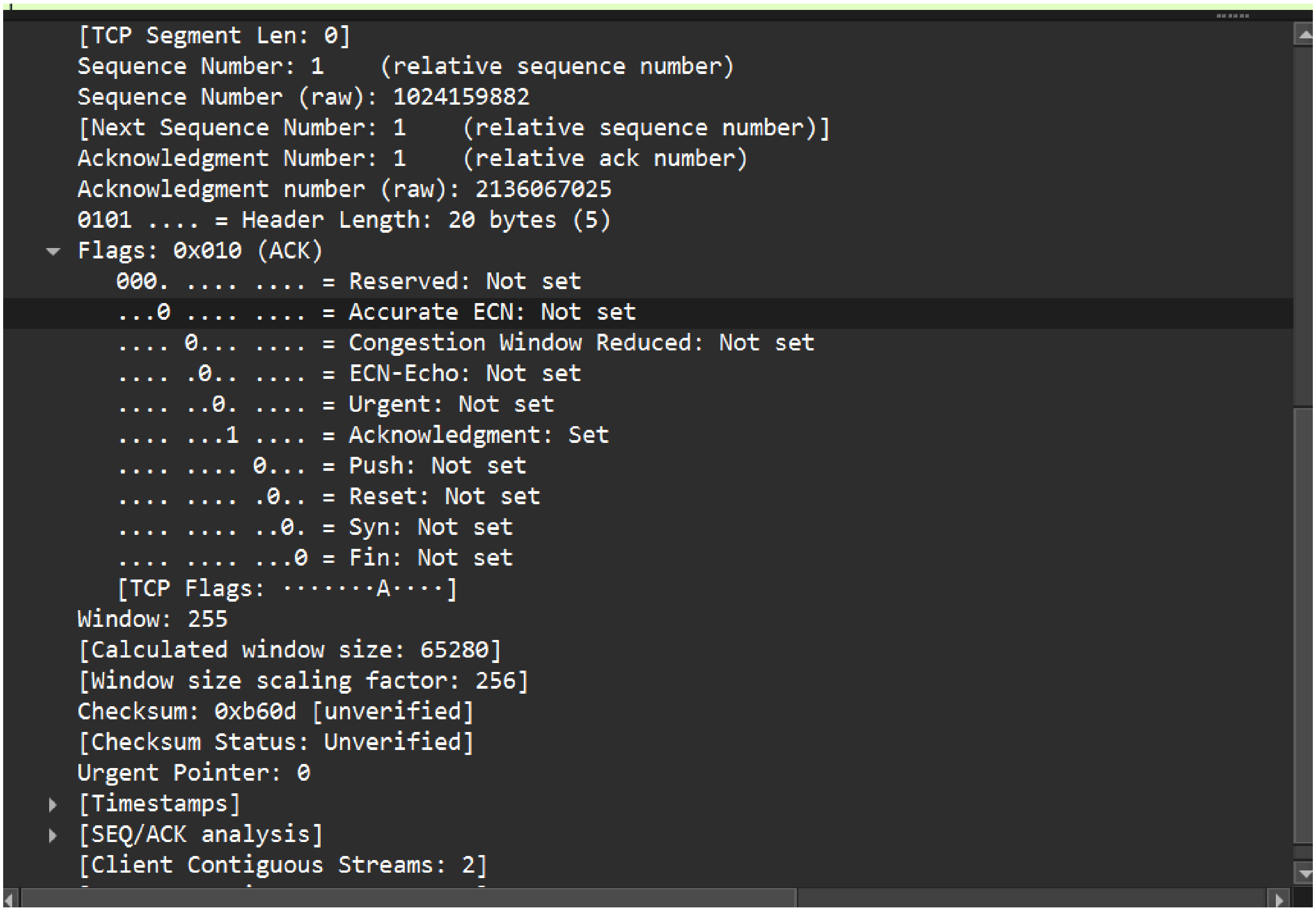
No.	Time	Source	Destination	Protocol	Lengtl	Info
54294	399.401001	183.173.243.243	202.38.64.43	TCP	66	5105 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
54300	399.432189	202.38.64.43	183.173.243.243	TCP	66	80 → 5105 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1340 SACK_PERM WS=128
54301	399.432342	183.173.243.243	202.38.64.43	TCP	54	5105 → 80 [ACK] Seq=1 Ack=1 Win=65280 Len=0
54302	399.435799	183.173.243.243	202.38.64.43	HTTP	482	GET / HTTP/1.1
54307	399.468919	202.38.64.43	183.173.243.243	TCP	60	80 → 5105 [ACK] Seq=1 Ack=429 Win=30336 Len=0
54308	399.468919	202.38.64.43	183.173.243.243	TCP	8094	80 → 5105 [ACK] Seq=1 Ack=429 Win=30336 Len=8040 [TCP PDU reassembled in 54310]



```
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1024159881
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
    Flags: 0x002 (SYN)
       000. .... .... = 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
       .... .... ..1. = Syn: Set
       .... .... ...0 = Fin: Not set
       [TCP Flags: ··········S·]
    Window: 65535
    [Calculated window size: 65535]
    Checksum: 0xb619 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
    Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operatio
    [Timestamps]
    [Client Contiguous Streams: 2]
    [Server Contiguous Streams: 1]
```



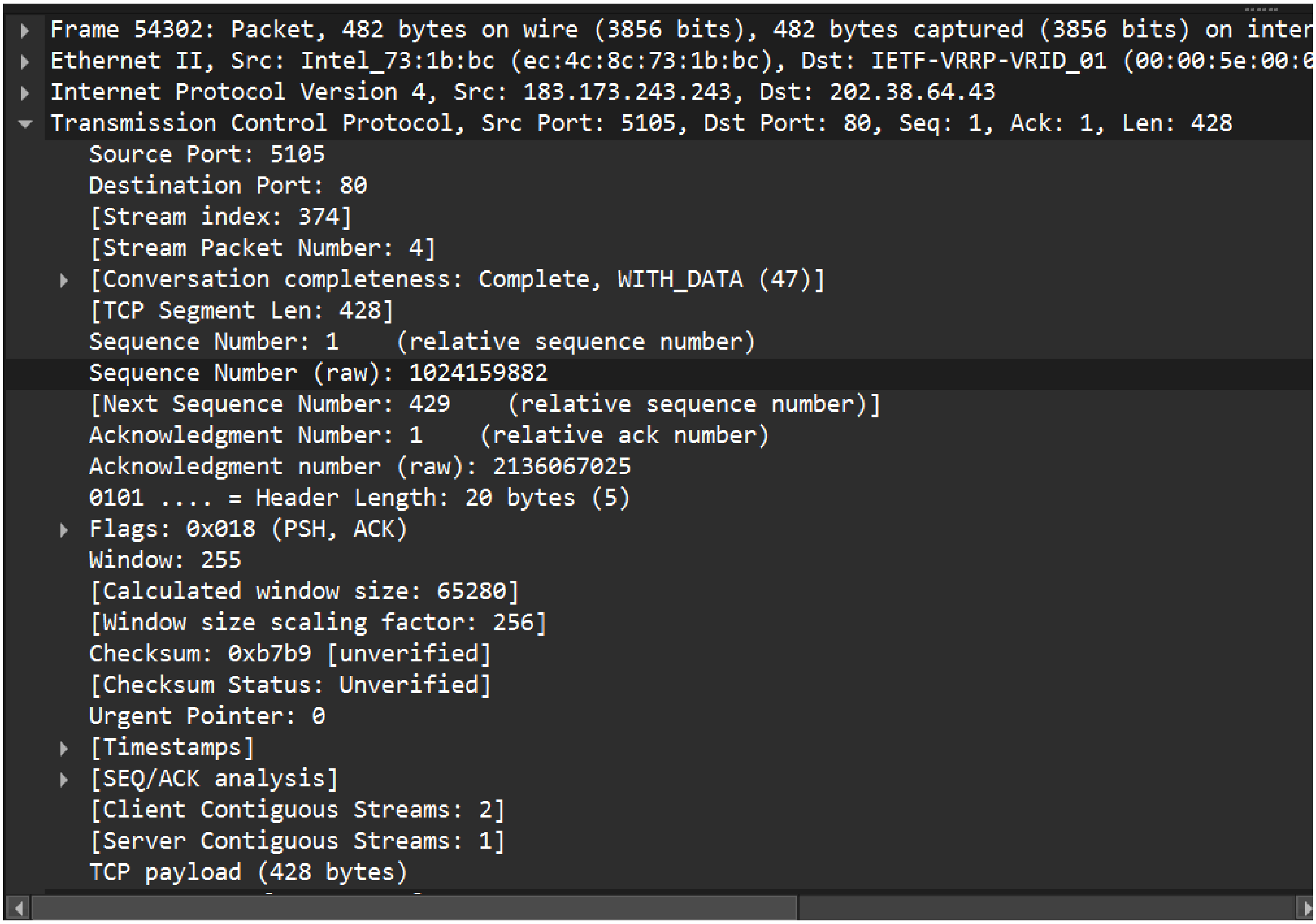
```
54389 399.648176    183.173.243.243       202.38.64.43          TCP       54 5105 → 80 [ACK] Seq=1

    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 2136067024
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1024159882
    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
       .... .... ...0 = Fin: Not set
       [TCP Flags: ·······A··S·]
    Window: 29200
    [Calculated window size: 29200]
    Checksum: 0x3a7d [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK p
  ▸ [Timestamps]
  ▸ [SEQ/ACK analysis]
    [Client Contiguous Streams: 1]

000
001
002
003
004
```



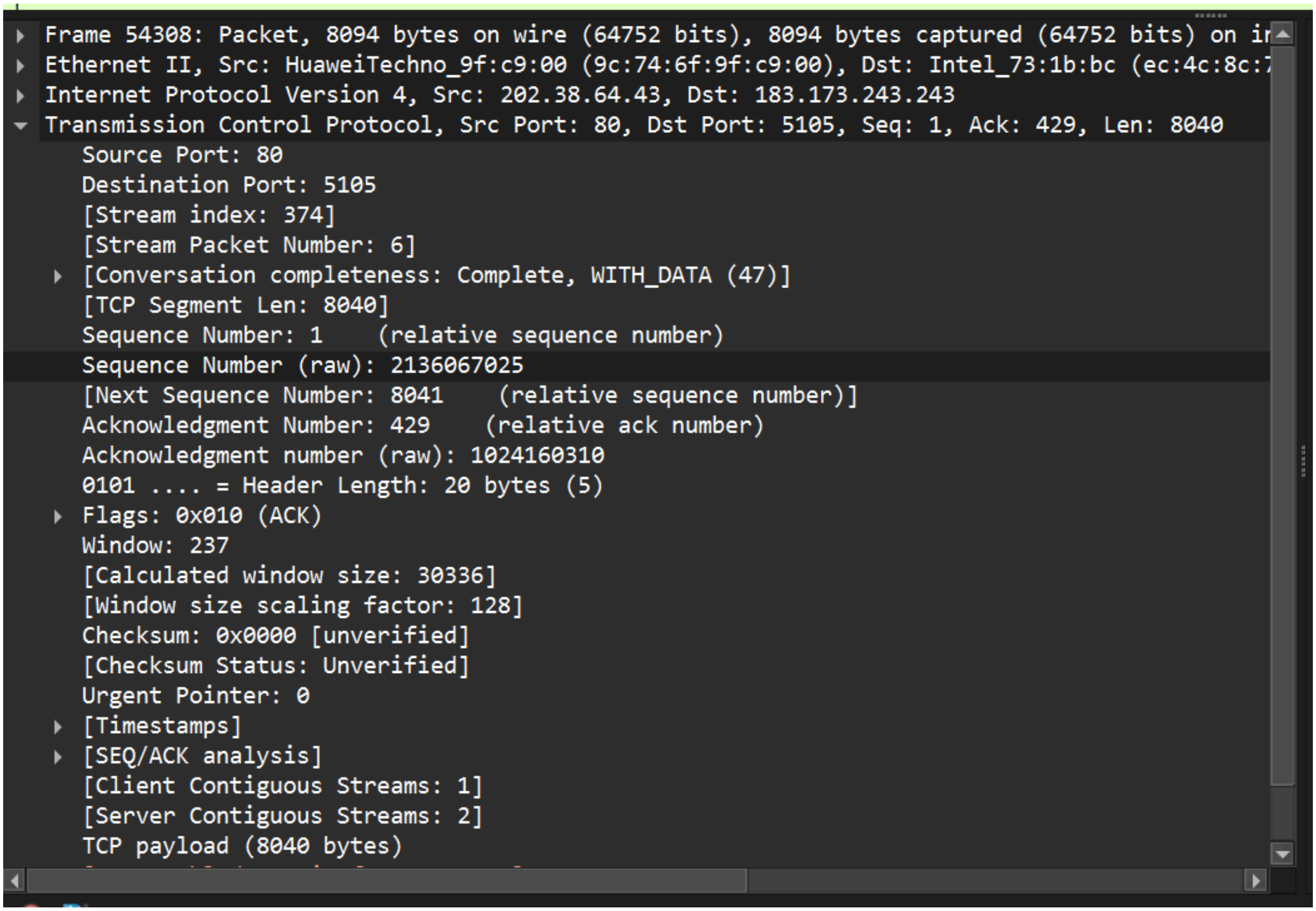
```
     [TCP Segment Len: 0]
     Sequence Number: 1    (relative sequence number)
     Sequence Number (raw): 1024159882
     [Next Sequence Number: 1    (relative sequence number)]
     Acknowledgment Number: 1    (relative ack number)
     Acknowledgment number (raw): 2136067025
     0101 .... = 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: 255
     [Calculated window size: 65280]
     [Window size scaling factor: 256]
     Checksum: 0xb60d [unverified]
     [Checksum Status: Unverified]
     Urgent Pointer: 0
   ▸ [Timestamps]
   ▸ [SEQ/ACK analysis]
     [Client Contiguous Streams: 2]
```

(4) 序列号与确认序列号的关系

序列号增长与包长关系：下一个数据包的 $Seq =$ 当前 $Seq +$ 当前包的 TCP Payload 长度。确认号表示接收方已经成功收到该序号之前的所有数据，并期待接收该序号开始的数据。



```
▸ Frame 54302: Packet, 482 bytes on wire (3856 bits), 482 bytes captured (3856 bits) on inter
▸ Ethernet II, Src: Intel_73:1b:bc (ec:4c:8c:73:1b:bc), Dst: IETF-VRRP-VRID_01 (00:00:5e:00:0
▸ Internet Protocol Version 4, Src: 183.173.243.243, Dst: 202.38.64.43
▾ Transmission Control Protocol, Src Port: 5105, Dst Port: 80, Seq: 1, Ack: 1, Len: 428
     Source Port: 5105
     Destination Port: 80
     [Stream index: 374]
     [Stream Packet Number: 4]
   ▸ [Conversation completeness: Complete, WITH_DATA (47)]
     [TCP Segment Len: 428]
     Sequence Number: 1    (relative sequence number)
     Sequence Number (raw): 1024159882
     [Next Sequence Number: 429    (relative sequence number)]
     Acknowledgment Number: 1    (relative ack number)
     Acknowledgment number (raw): 2136067025
     0101 .... = Header Length: 20 bytes (5)
   ▸ Flags: 0x018 (PSH, ACK)
     Window: 255
     [Calculated window size: 65280]
     [Window size scaling factor: 256]
     Checksum: 0xb7b9 [unverified]
     [Checksum Status: Unverified]
     Urgent Pointer: 0
   ▸ [Timestamps]
   ▸ [SEQ/ACK analysis]
     [Client Contiguous Streams: 2]
     [Server Contiguous Streams: 1]
     TCP payload (428 bytes)
```



```
▸ Frame 54308: Packet, 8094 bytes on wire (64752 bits), 8094 bytes captured (64752 bits) on ir
▸ Ethernet II, Src: HuaweiTechno_9f:c9:00 (9c:74:6f:9f:c9:00), Dst: Intel_73:1b:bc (ec:4c:8c:7
▸ Internet Protocol Version 4, Src: 202.38.64.43, Dst: 183.173.243.243
▾ Transmission Control Protocol, Src Port: 80, Dst Port: 5105, Seq: 1, Ack: 429, Len: 8040
     Source Port: 80
     Destination Port: 5105
     [Stream index: 374]
     [Stream Packet Number: 6]
   ▸ [Conversation completeness: Complete, WITH_DATA (47)]
     [TCP Segment Len: 8040]
     Sequence Number: 1    (relative sequence number)
     Sequence Number (raw): 2136067025
     [Next Sequence Number: 8041    (relative sequence number)]
     Acknowledgment Number: 429    (relative ack number)
     Acknowledgment number (raw): 1024160310
     0101 .... = Header Length: 20 bytes (5)
   ▸ Flags: 0x010 (ACK)
     Window: 237
     [Calculated window size: 30336]
     [Window size scaling factor: 128]
     Checksum: 0x0000 [unverified]
     [Checksum Status: Unverified]
     Urgent Pointer: 0
   ▸ [Timestamps]
   ▸ [SEQ/ACK analysis]
     [Client Contiguous Streams: 1]
     [Server Contiguous Streams: 2]
     TCP payload (8040 bytes)
```

实验 3：简述题

(1) TCP 选项还支持什么特殊功能?

除了 MTU/MSS 协商外，TCP 选项还支持：

窗口缩放 (Window Scale)：扩展 16 位窗口限制，支持更大带宽延迟积。
时间戳 (Timestamps)：用于计算 RTT 及防止序列号回绕 (PAWS)。
选择性确认 (SACK)：允许只重传丢失的分段，提高重传效率。

(2) 反射 DoS 攻击为什么大多使用基于 UDP 的公共服务?

1. 无须握手：UDP 无连接，伪造源 IP 发送请求后，服务器直接回复受害者，而 TCP 需要三次握手验证 IP 有效性。
2. 放大倍数高：DNS、NTP 等协议请求小、回复大，能产生巨大的流量放大效应，使受害者带宽快速耗尽。