

作业四：

通过 HTTP 访问某个网页，使用 Wireshark 对整个过程中的数据段进行捕获，分析 TCP 连接建立、数据传输、连接关闭的全过程，至少对其中 5 个典型的 TCP 数据段进行详细分析，给出界面截图，并同时提交捕获文件。

截图示例如下：

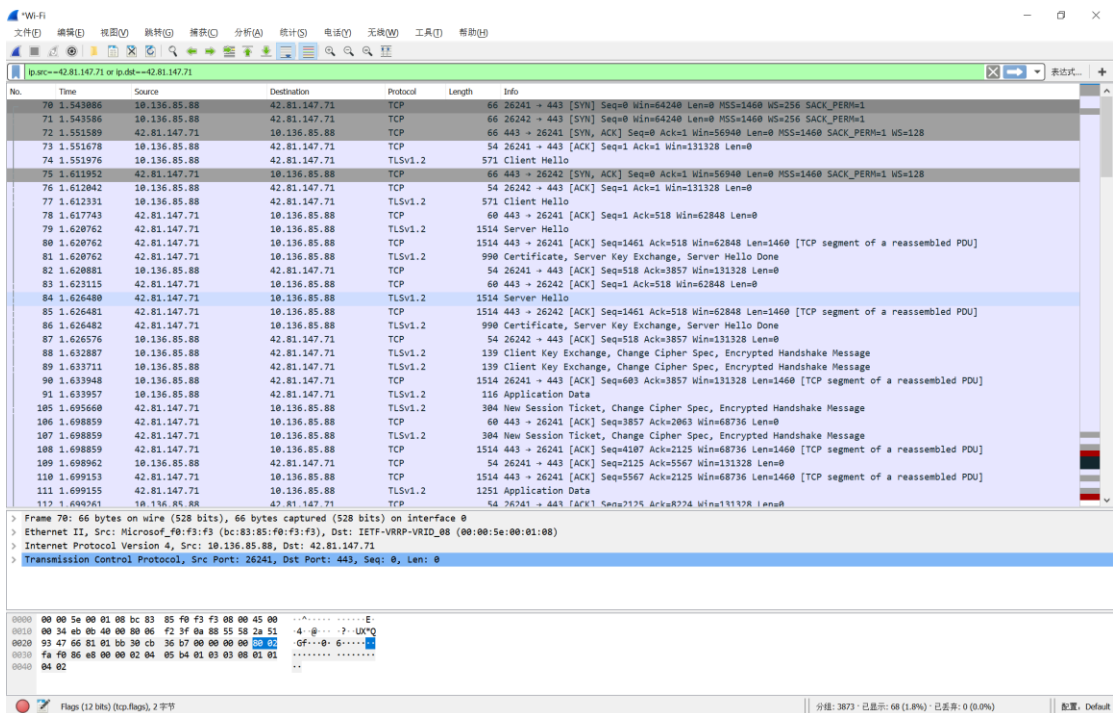
The screenshot displays the Wireshark interface with a capture of network traffic on the 'WLAN' interface. The packet list pane shows several TCP segments, including retransmissions and a successful connection establishment. The packet details pane for packet 117 (Frame 117) is expanded, showing the Ethernet II header, Internet Protocol Version 4 header, and the Transmission Control Protocol header. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
96	13.223667	10.134.152.178	192.168.3.11	TCP	66	[TCP Retransmission] 58327 → 7680 [SYN]
97	13.365832	2001:250:401:6545:f981...	2607:f8b0:4005:80a::20...	TCP	86	58329 → 443 [SYN] Seq=0 Win=64800 Len=0
101	14.159179	2001:250:401:6545:f981...	2607:f8b0:4005:80a::20...	TCP	86	[TCP Retransmission] 58328 → 443 [SYN]
102	14.371992	2001:250:401:6545:f981...	2607:f8b0:4005:80a::20...	TCP	86	[TCP Retransmission] 58329 → 443 [SYN]
103	14.861601	2001:250:401:6545:f981...	2607:f8b0:4005:80a::20...	TCP	86	58330 → 443 [SYN] Seq=0 Win=64800 Len=0
113	15.520306	10.134.152.178	39.156.66.179	TCP	66	58331 → 443 [SYN] Seq=0 Win=64240 Len=0
115	15.522063	10.134.152.178	222.30.45.190	TCP	66	58332 → 80 [SYN] Seq=0 Win=64240 Len=0
116	15.522408	10.134.152.178	222.30.45.190	TCP	66	58333 → 80 [SYN] Seq=0 Win=64240 Len=0
117	15.524994	222.30.45.190	10.134.152.178	TCP	66	80 → 58332 [SYN, ACK] Seq=0 Ack=1 Win=43
118	15.524994	222.30.45.190	10.134.152.178	TCP	66	80 → 58333 [SYN, ACK] Seq=0 Ack=1 Win=43
119	15.525101	10.134.152.178	222.30.45.190	TCP	54	58332 → 80 [ACK] Seq=1 Ack=1 Win=131328
120	15.525149	10.134.152.178	222.30.45.190	TCP	54	58333 → 80 [ACK] Seq=1 Ack=1 Win=131328
121	15.525482	10.134.152.178	222.30.45.190	HTTP	1170	GET / HTTP/1.1
122	15.529250	222.30.45.190	10.134.152.178	TCP	60	80 → 58333 [ACK] Seq=1 Ack=1117 Win=4608
123	15.533563	39.156.66.179	10.134.152.178	TCP	66	443 → 58331 [SYN, ACK] Seq=0 Ack=1 Win=64800
124	15.533593	10.134.152.178	39.156.66.179	TCP	54	58331 → 443 [ACK] Seq=1 Ack=1 Win=132096

Frame 117: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: IETF-VRRP-VRID_0e (00:00:5e:00:01:0e), Dst: IntelCor_20:fa:21 (a4:c4:94:20:fa:21)
> Internet Protocol Version 4, Src: 222.30.45.190, Dst: 10.134.152.178
> Transmission Control Protocol, Src Port: 80, Dst Port: 58332, Seq: 0, Ack: 1, Len: 0

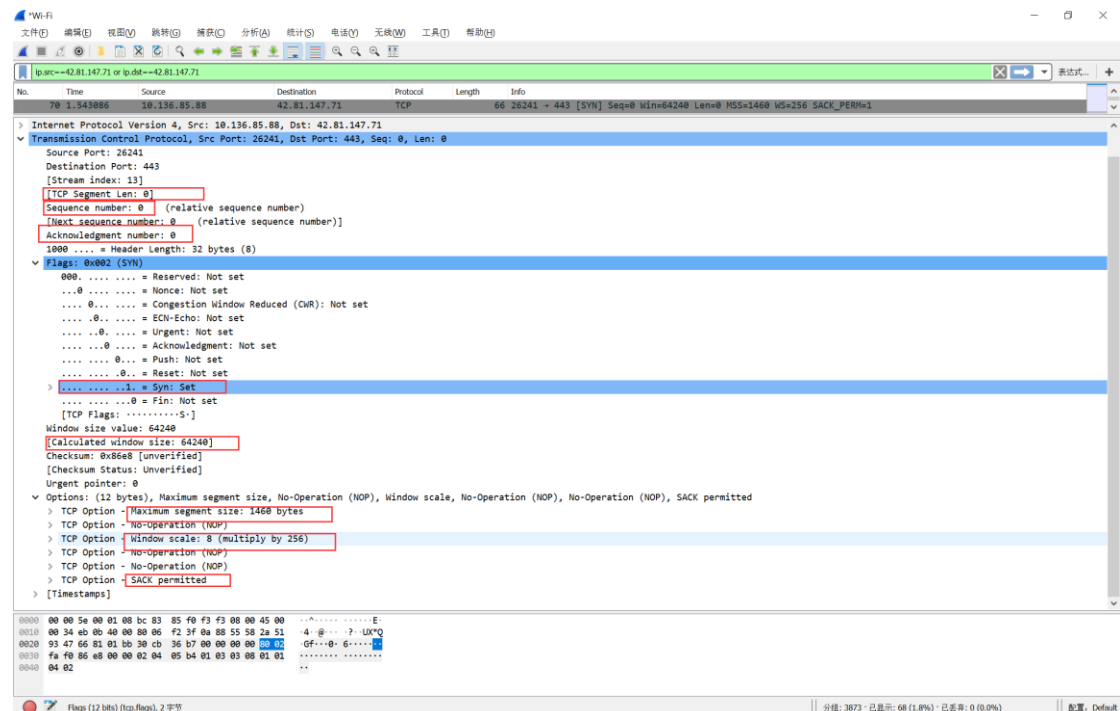
0000 a4 c4 94 20 fa 21 00 00 5e 00 01 0e 08 00 45 00 ... !... ^.....E-
0010 00 34 00 00 40 00 3e 06 8d af de 1e 2d be 0a 86 ... 4...@->.....
0020 98 b2 00 50 e3 dc 98 b8 14 b8 96 39 6b 38 80 12 ... P.....9k8..
0030 aa aa 88 0c 00 00 02 04 ff d7 01 01 04 02 01 03
wireshark_4593C3C7-48F9-4D6E-828...F1C_20191113082446_a20936.pcapng | Packets: 305 · Displayed: 167 (54.8%) · Dropped: 0 (0.0%) | Profile: Default

作业：因为目前网站大多使用 HTTPS 协议，因此此次作业捕获的包是来自 <https://www.huawei.com/> 的。关于 TCP 的分析基本不受影响。



一：三次握手建立连接：

第一次握手：



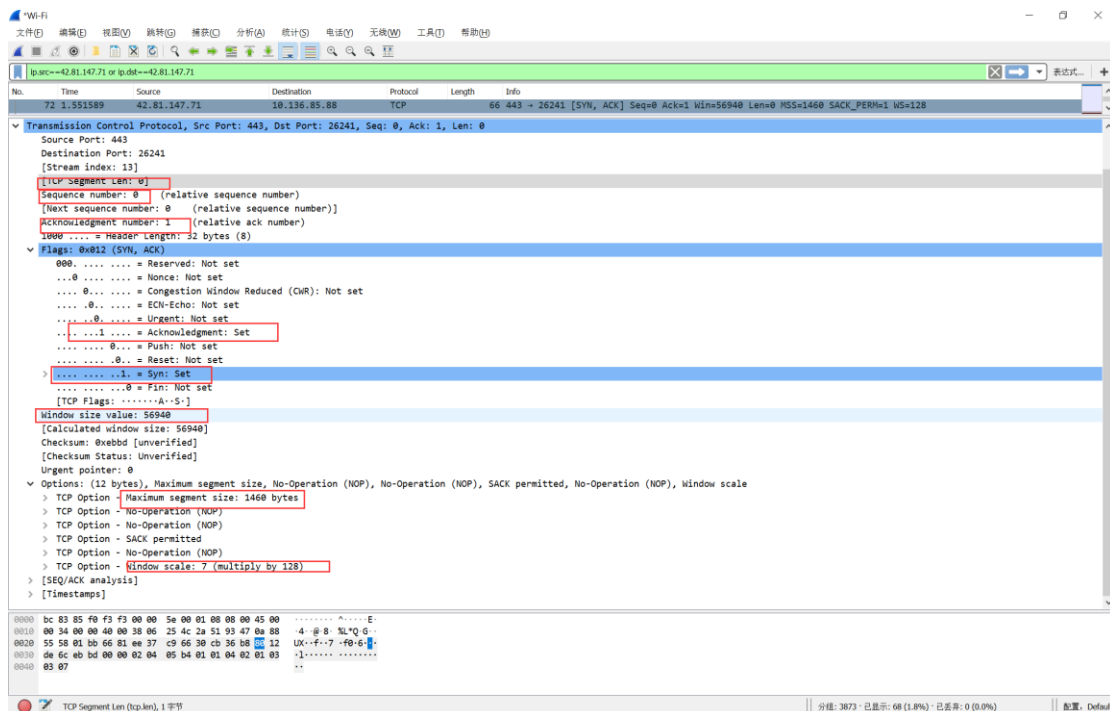
第 70 条：客户端发送 SYN 到服务器，请求建立连接，并进入 SYN_SENT 状态

- 源 IP：10.136.85.88
- 目标 IP：42.81.147.71
- SYN：标志位，表示请求建立连接
- Seq = 0：初始建立连接值为 0，数据包的相对序列号从 0 开始，表示当前还没有发送数据
- Ack=0：初始建立连接值为 0，已经收到包的数量，表示当前没有接收到数据
- WIN=64240 发送窗口，来自 Window size: 64240,
- Len = 0
- MSS = 1460 来自 Maximum segment size: 1460 byte，最长报

文段, TCP 包所能携带的最大数据量, 不包含 TCP 头和 Option。
一般为 MTU 值减去 IPv4 头部(至少 20 字节)和 TCP 头部(至少 20 字节)得到

- WS = 256 来自 windows scale : 8(multiply by 256): 窗口扩张, 放在 TCP 头之外的 Option, 向对方声明一个 shift count, 作为 2 的指数, 再乘以 TCP 定义的接收窗口, 得到真正的 TCP 窗口
- SACK_PERM = 1 来自 SACK permitted

第二次握手:

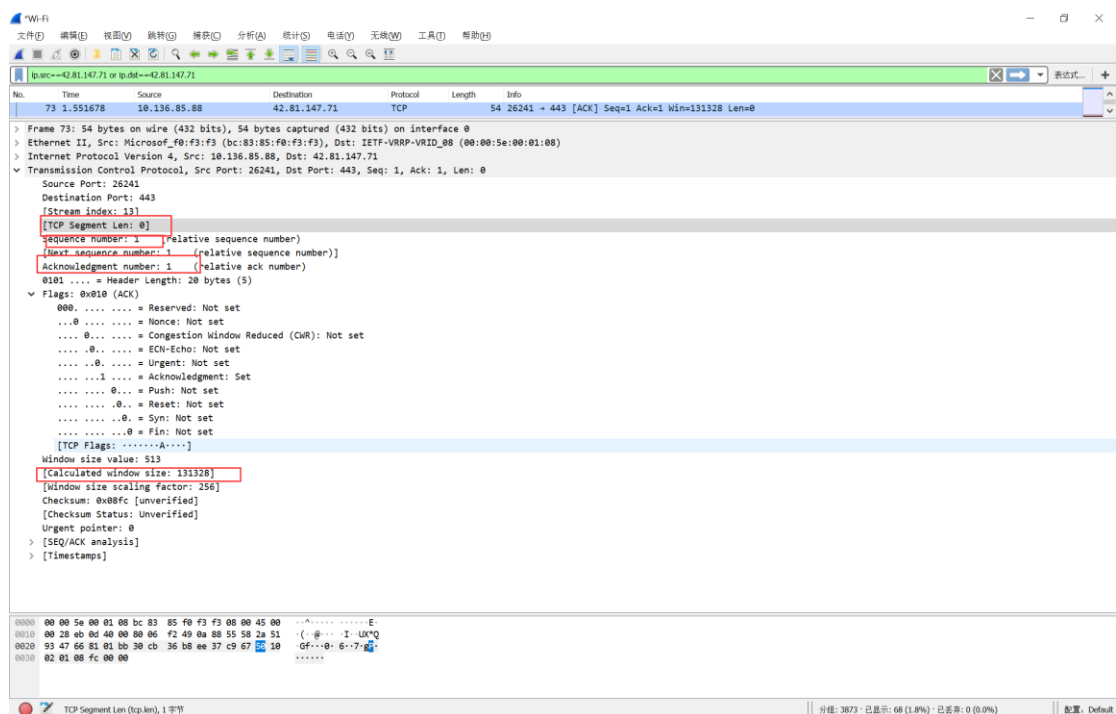


第 72 条: 服务器收到请求后, 回送 SYN+ACK 信令到客户端, 此时服务器进入 SYN_RECV 状态

- 源 IP: 42.81.147.71

- 目标 IP: 192.168.43.136
- SYN + ACK: 标志位, 同意建立连接, 并回送 SYN+ACK
- Seq = 0 : 初始建立值为 0, 表示当前还没有发送数据
- Ack = 1 : 表示当前端成功接收的数据位数, 尽管客户端没有发送任何有效数据, 确认号还是被加 1, 这是因为接收的包中包含 SYN 或 FIN 标志位(并不会对有效数据的计数产生影响, 因为含有 SYN 或 FIN 标志位的包并不携带有效数据)
- Win=56940 接收窗口
- Len=0
- 其它标志和第一包相同。

第三次握手:



客户端收到 SYN+ACK 包, 向服务器发送确认 ACK 包, 客户端进入

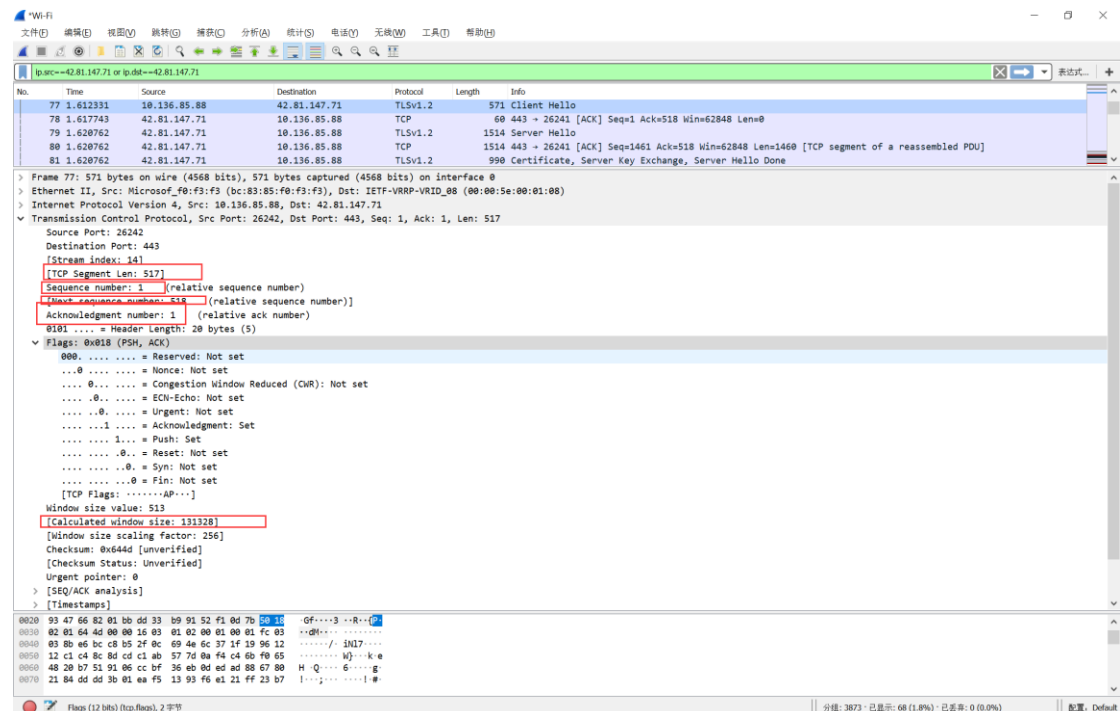
ESTABLISHED 状态，服务器收到请求后也进入 ESTABLISHED 状态，完成三次握手，此时 TCP 连接成功，客户端与服务器开始传送数据。

- 源 IP: 10.136.85.88
- 目标 IP: 42.81.147.71
- ACK : 标志位，表示已经收到记录
- Seq = 1 : 表示当前已经发送 1 个数据
- Ack = 1 : 表示当前端成功接收的数据位数，尽管客户端没有发送任何有效数据，确认号还是被加 1，这是因为接收的包中包含 SYN 或 FIN 标志位(并不会对有效数据的计数产生影响，因为含有 SYN 或 FIN 标志位的包并不携带有效数据)
- Win=131328 窗口扩张了。
- Len=0

由于是 HTTPS 协议，所以在之后还会有交换密钥的过程。

二：数据传输

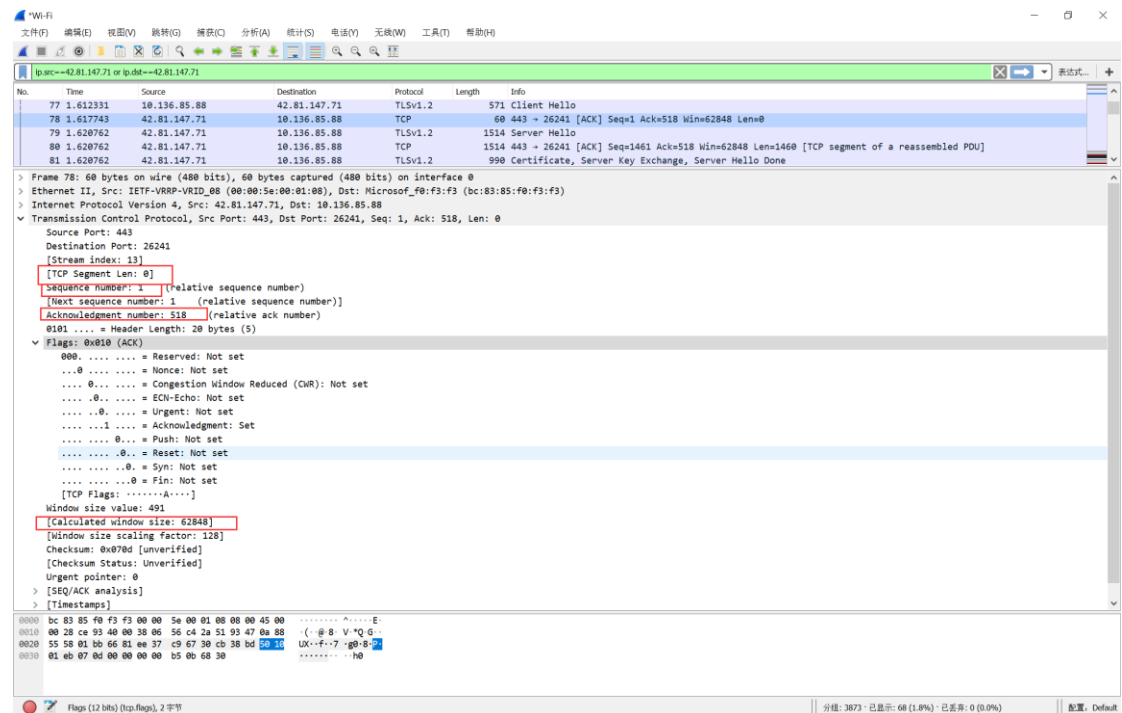
客户端->服务器



第 77 条：包含客户端向服务器发送的 Client Hello 信息，以便开始一个新的加密会话连接。对于内容不做分析。

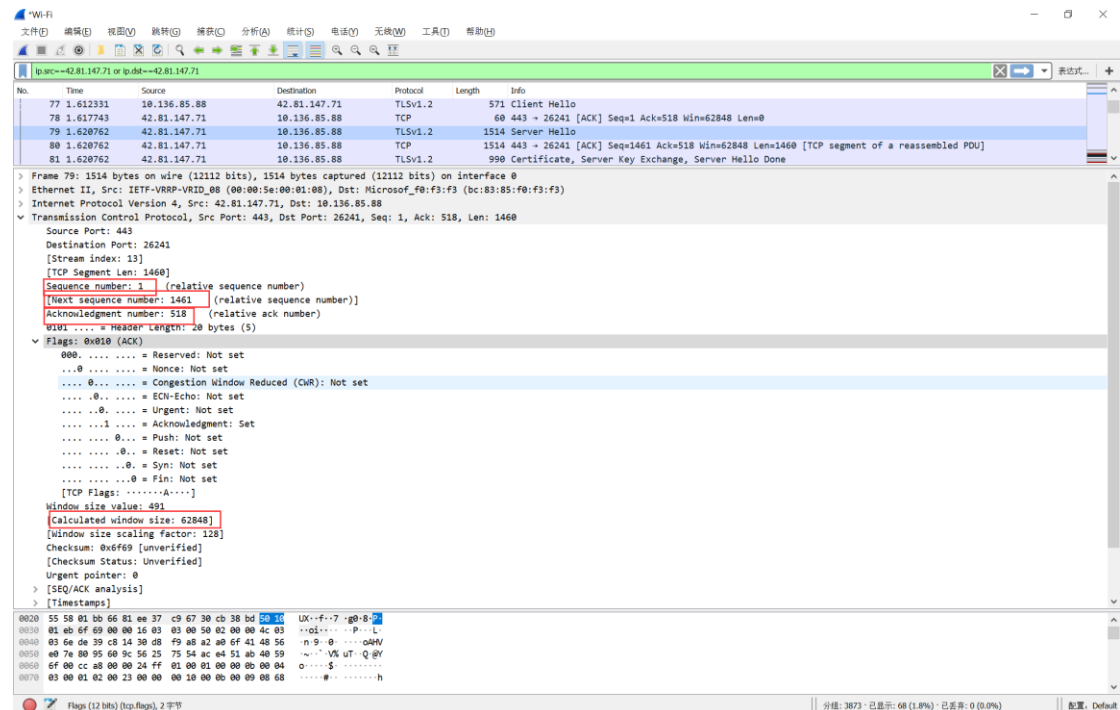
- 源 IP: 10.136.85.88
- 目标 IP: 42.81.147.71
- Seq=1: 表示当前已经发送了 1 个数据。
- Next Sequence number: 在完成这次的发送后,当发送新的数据时, Seq 从 518 开始编码
- Ack = 1: 表示当前端成功接收的 1 个数据位数
- Win=131328
- Len=517 说明里面包含有数据。
- TCP payload: Hypertext Transfer Protocol 部分的数据量,本次传送的数据量为 517 byte

服务器->客户端



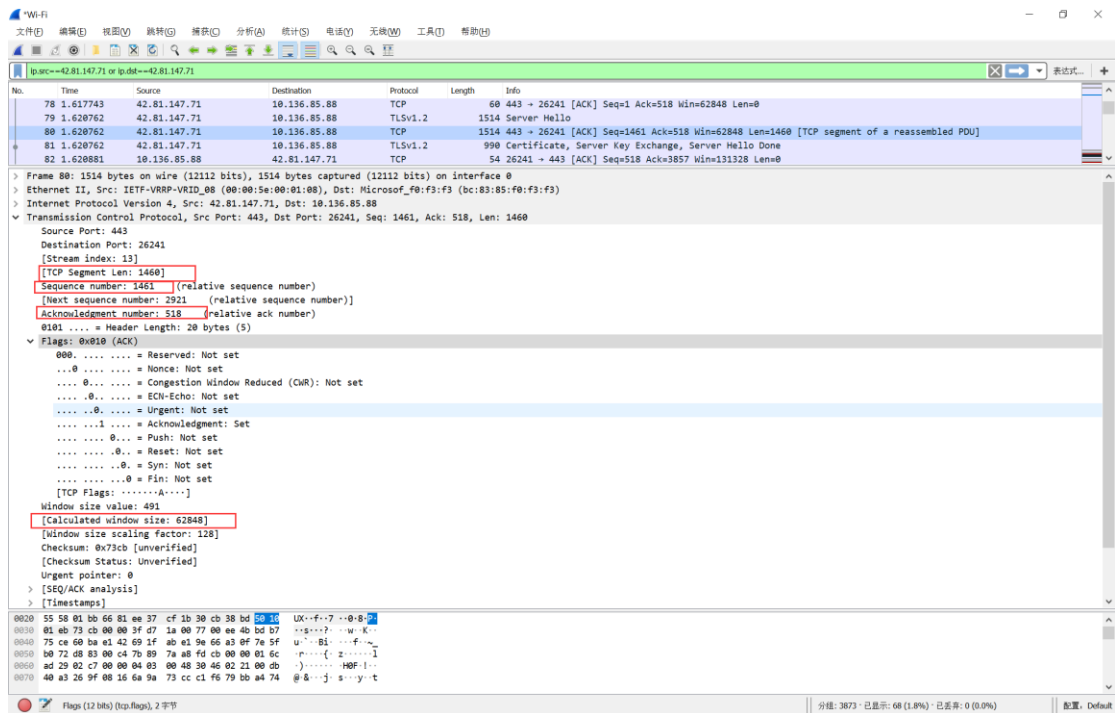
第 78 条：服务端接受发送过来的 517 个字节数据，在接收请求过程中，发送一次 ACK 包，

- 源 IP：42.81.147.71
- 目标 IP：10.136.85.88
- Seq=1, Ack=518, 说明共收到 518 个字节的数据，期待下一次传过来的数据的 seq 为 518。
- Win=62848,
- Len=0, 说明里面没有数据。



第 79 条：服务器第二次向客户端发送数据，内容是 server hello，也就是告诉客户端，请求已经收到。

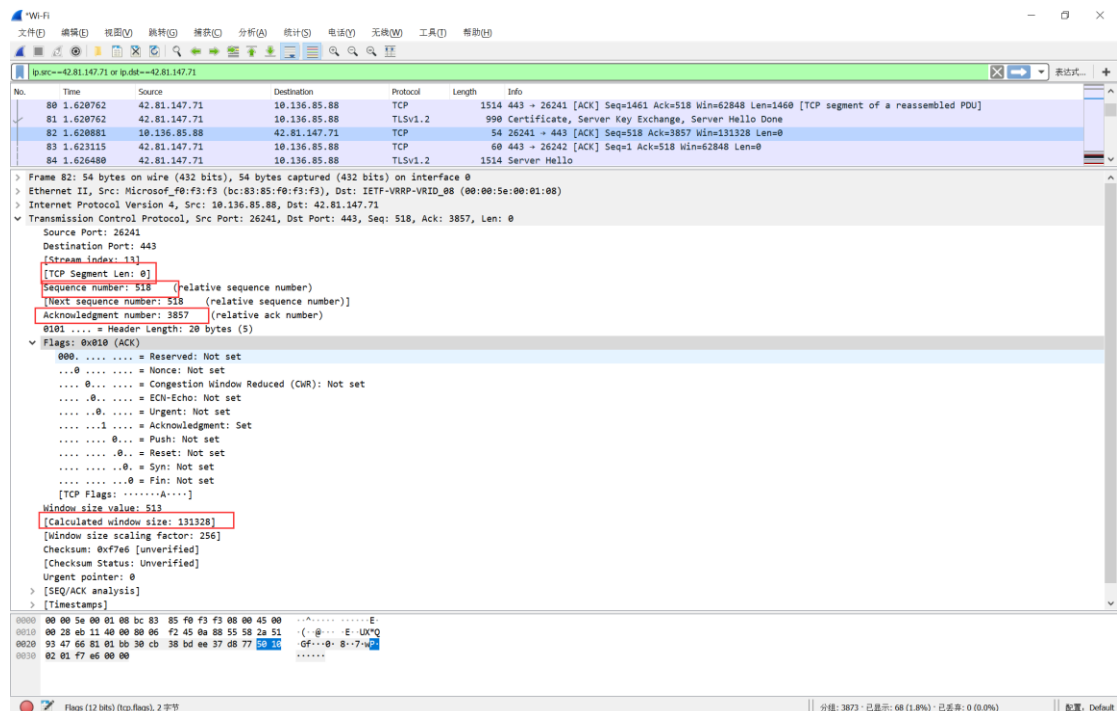
- 源 IP：42.81.147.71
- 目标 IP：192.168.43.136
- Seq=1, Ack=518, 说明共收到 518 个字节的数据，期待下一次传过来的数据的 seq 为 518。
- Win=62848,
- Len=1460,



第 59 条，还有一个 TCP 包给客户端。

- 源 IP：42.81.147.71
- 目标 IP：192.168.43.136
- Seq=1461，Ack=518，说明共收到 518 个字节的数据，期待下一次传过来的数据的 seq 为 518。
- Win=62848，
- Len=1460，

客户端->服务器



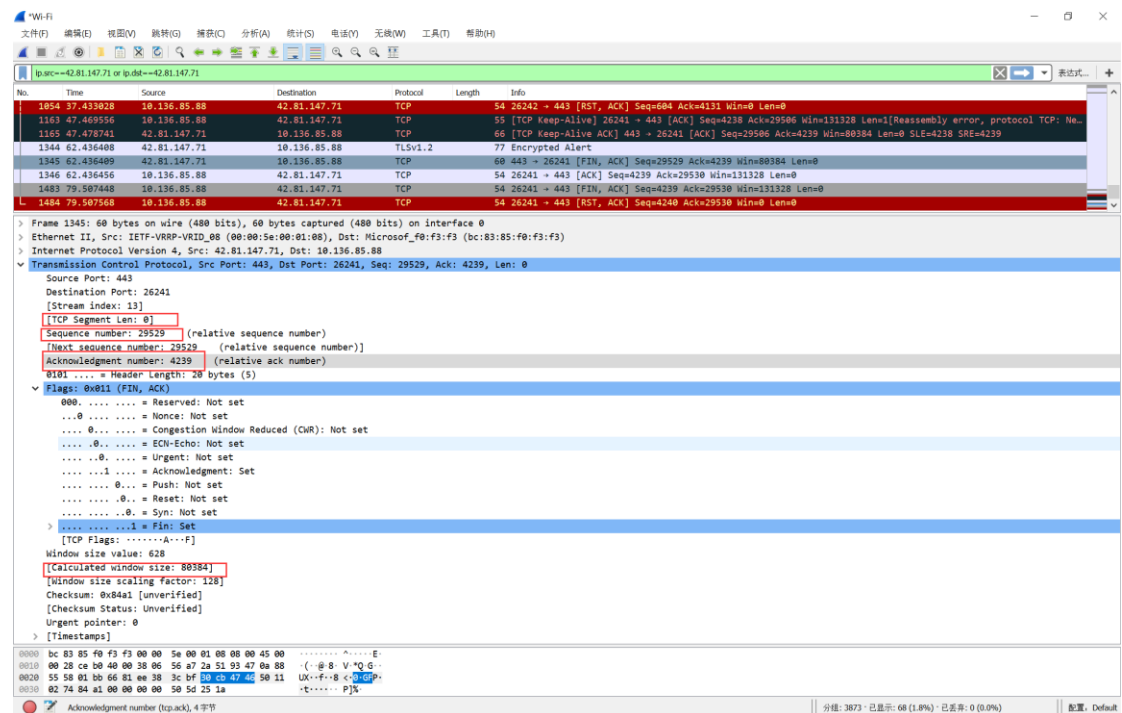
第 82 条告诉服务器，服务器的 hello 客户端收到了，此后应该交换密钥。

- 源 IP: 10.136.85.88
- 目标 IP: 42.81.147.71
- Seq=518: 也就是服务器所期待的字节号。
- Ack = 3857: 表示之前服务器发送来的数据已经收到。
- Win=131328
- Len=0 说明里面没有包含数据。

然后如此往复，客户端不断与服务器进行数据交流。在 wire shark 里有 TLS 的数据信息：如应用数据，交换密钥等等。

三：四次挥手关闭连接：

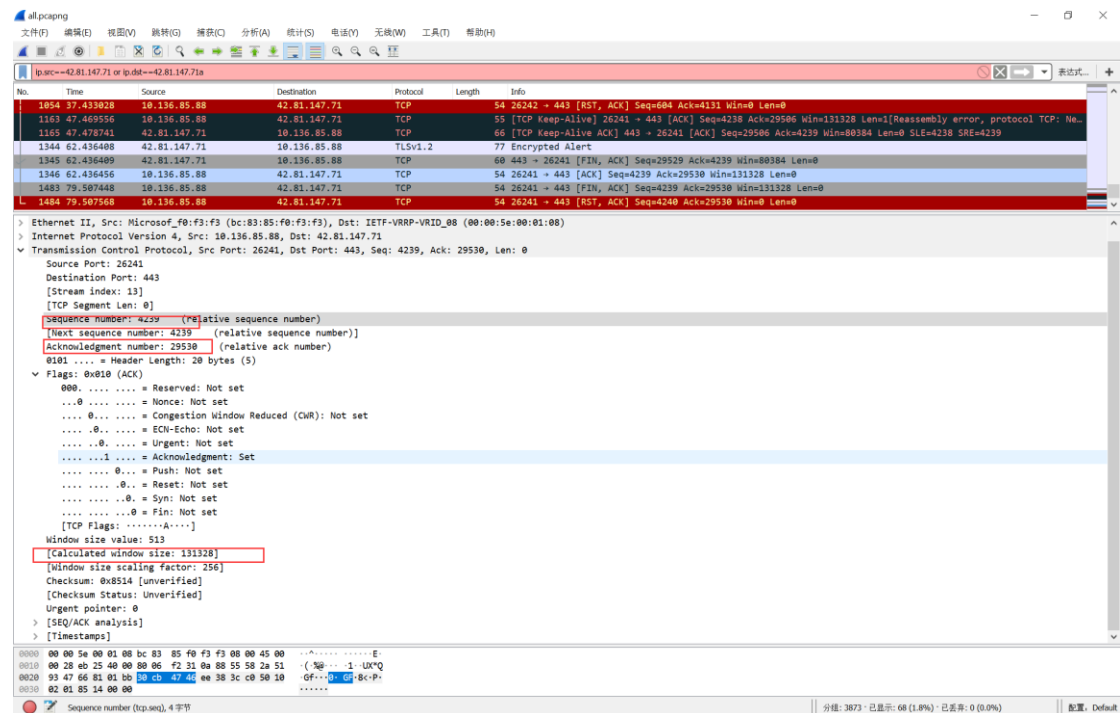
第一次挥手



第 1345 条：服务器向客户端（不应该是客户端向服务器？）发送 FIN 关闭连接的请求。

- 源 IP：42.81.147.71
- 目标 IP：192.168.43.136
- Seq=29529, Ack=4239. 表示收到了之前的 4293 个字节的数据。
- Win=80384,
- Len=0,
- [FIN, ACK]这两位置 1，表示没有数据请求，并且确认之前的数据。

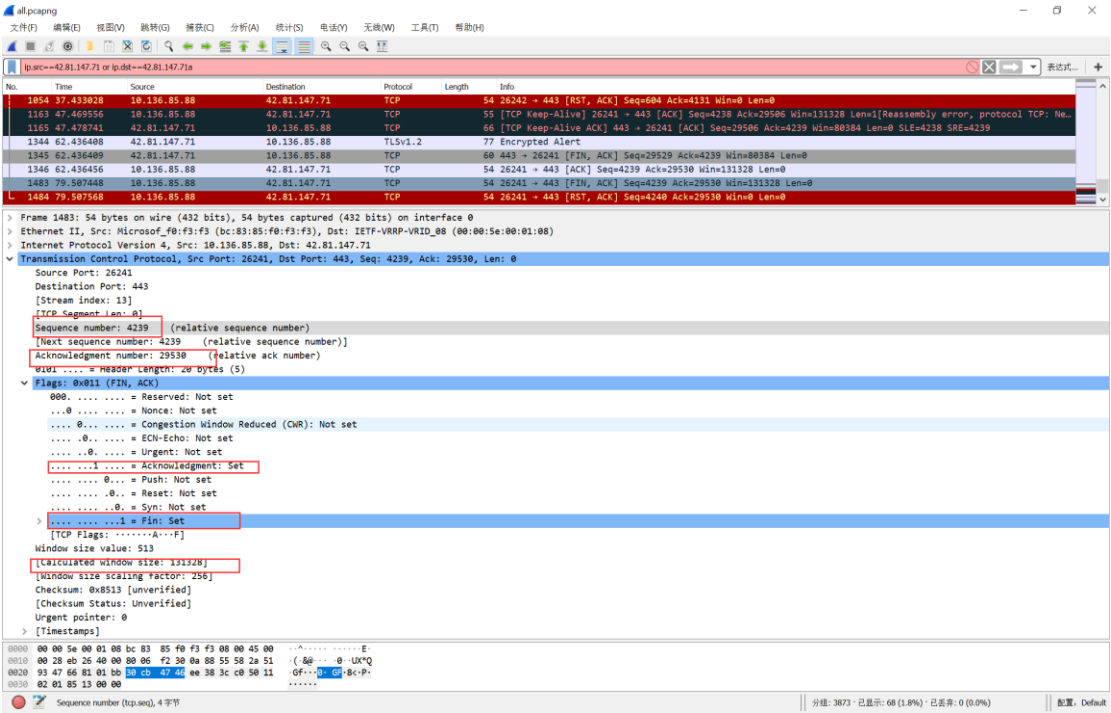
第二次挥手



客户端向服务器发送确认信号，知道服务器没有信息发送了。

- 源 IP: 10.136.85.88
- 目标 IP: 42.81.147.71
- Seq=4239: 也就是服务器所期待的字节号。
- Ack = 29530: 表示之前服务器发送来的数据已经收到。
- Win=131328
- Len=0 说明里面没有包含数据。

第三次挥手



客户端也向服务器发送请求关闭连接的请求，在发送之后告诉服务器，知道服务器没有数据了，我也没有数据传输了。你关闭了吧，我也关闭了。所以就关闭了。之前还有相同的操作，就等待服务器和客户端，看看对方还有没有信息传送，在延时后，没有信息交流就正式关闭了。

第四次挥手

在数据包里没有捕获到第四次挥手连接就关闭了。