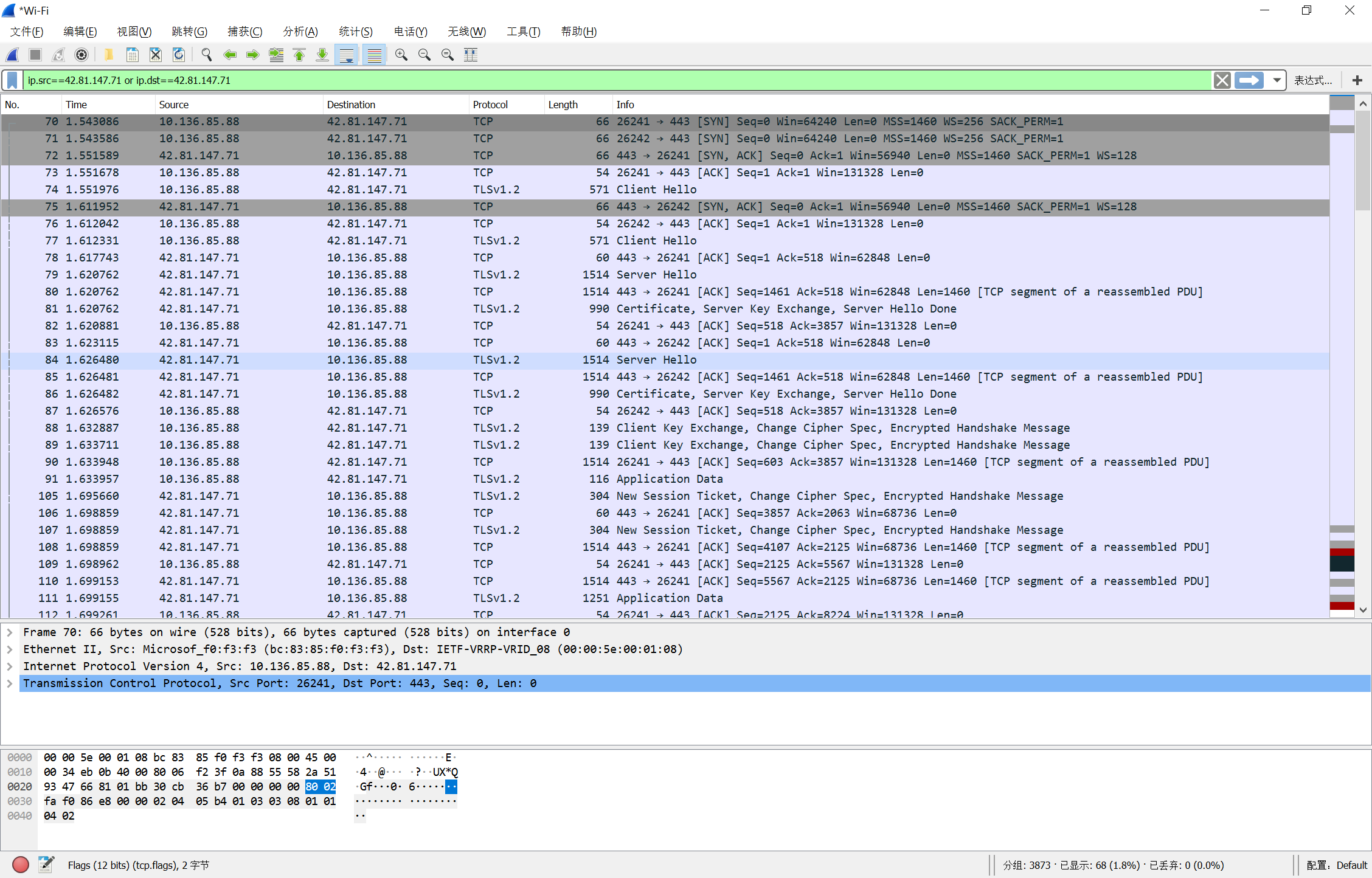
作业四：

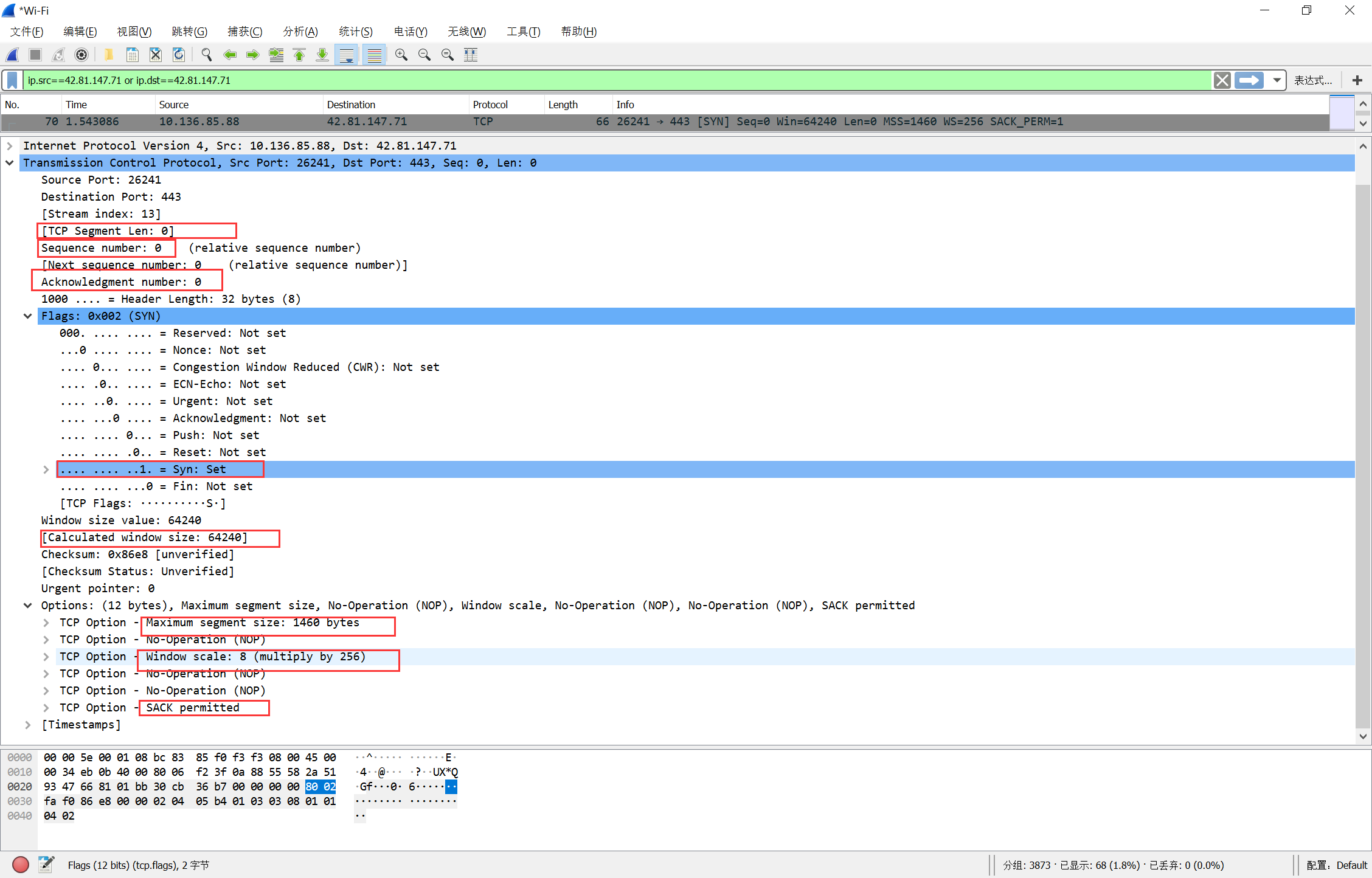
# 通过HTTP访问某个网页，使用Wireshark对整个过程中的数据段进行捕获，分析TCP连接建立、数据传输、连接关闭的全过程，至少对其中5个典型的TCP数据段进行详细分析，给出界面截图，并同时提交捕获文件。

# 截图示例如下：

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：

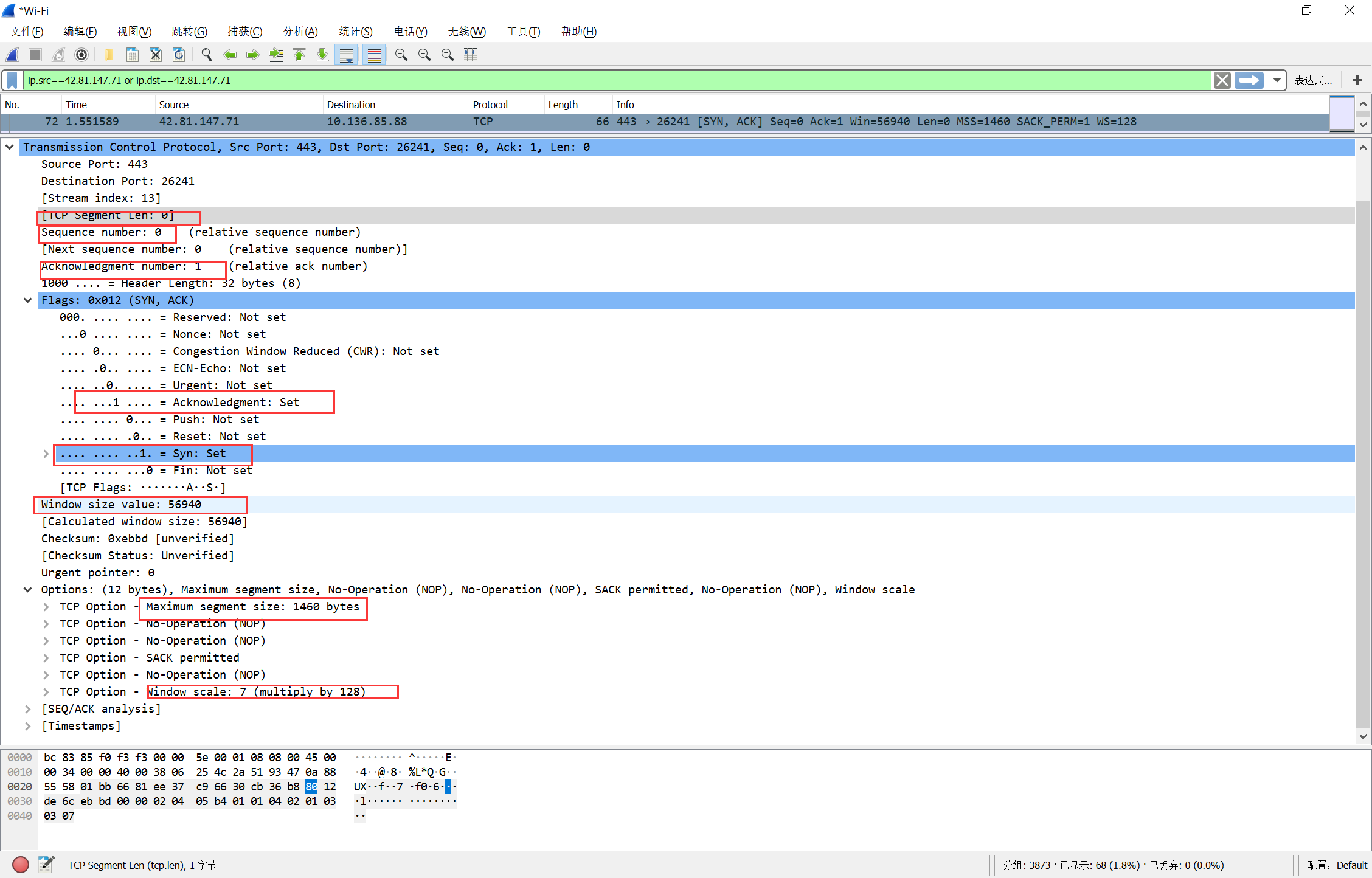
## 第一次握手：



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

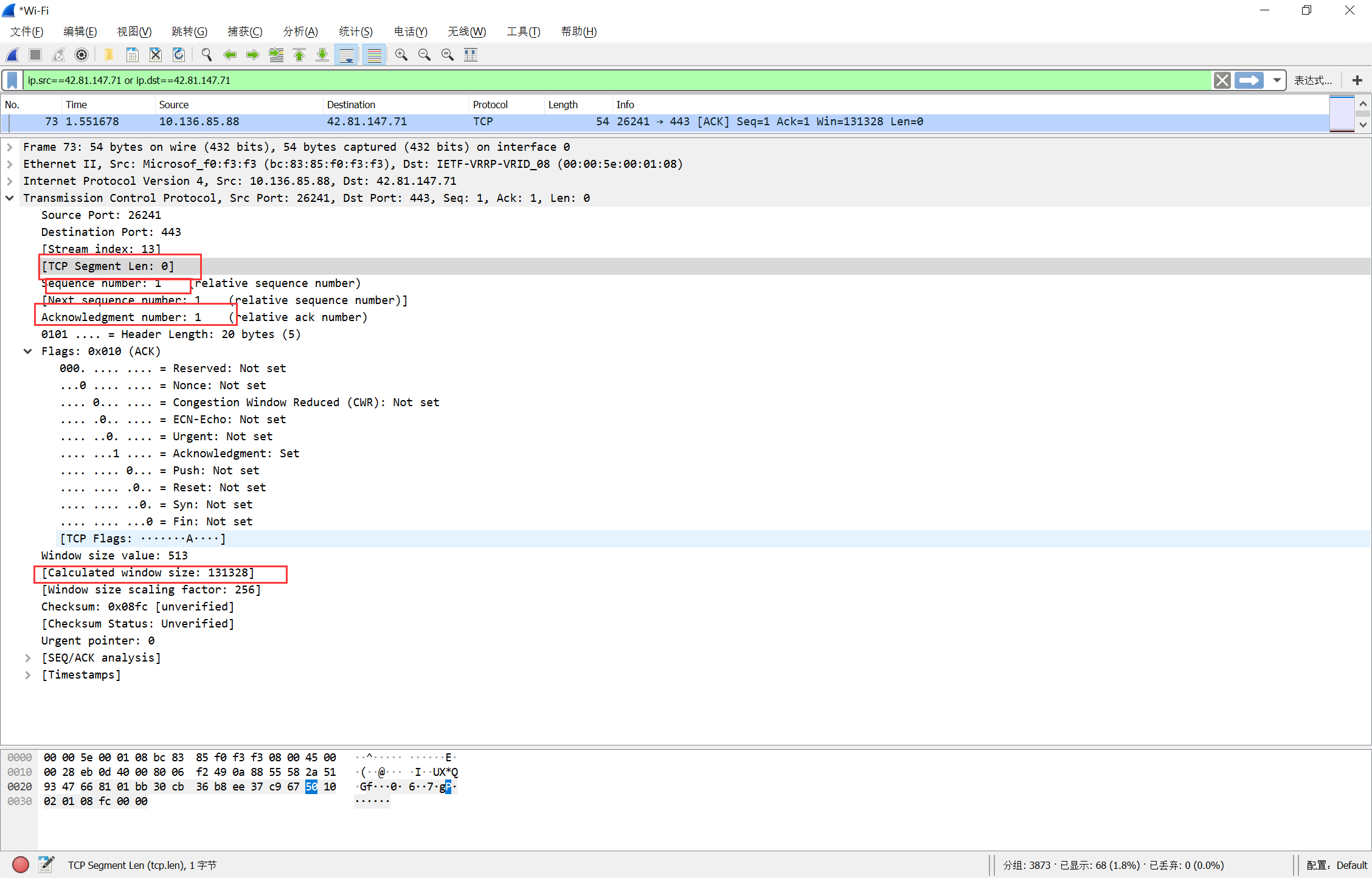
## 第二次握手：



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：



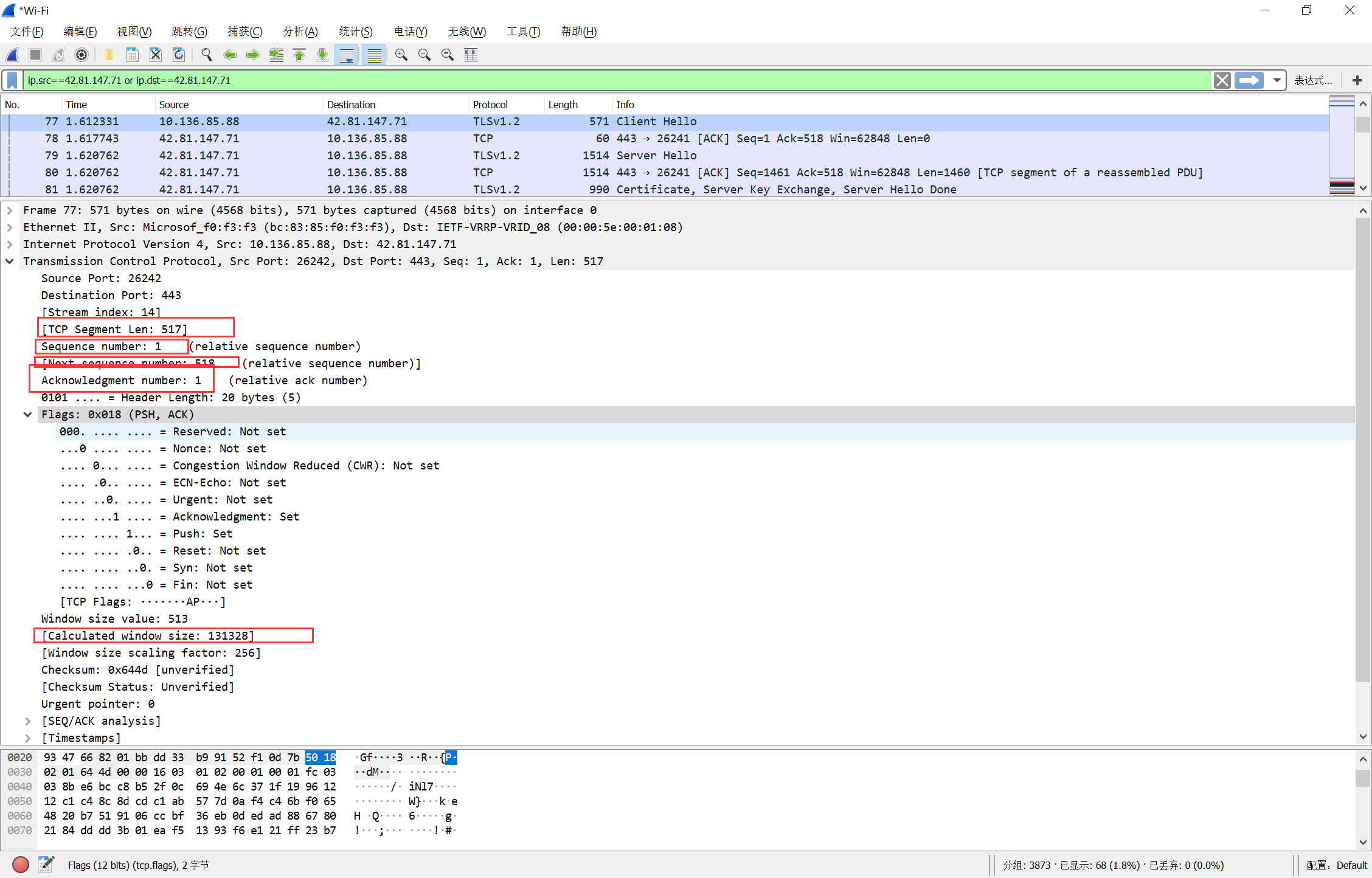
客户端收到SYN+ACK包，向服务器发送确认ACK包，客户端进入ESTABLISHED状态，服务器收到请求后也进入ESTABLISHED状态，完成三次握手，此时TCP连接成功，客户端与服务器开始传送数据。

* 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

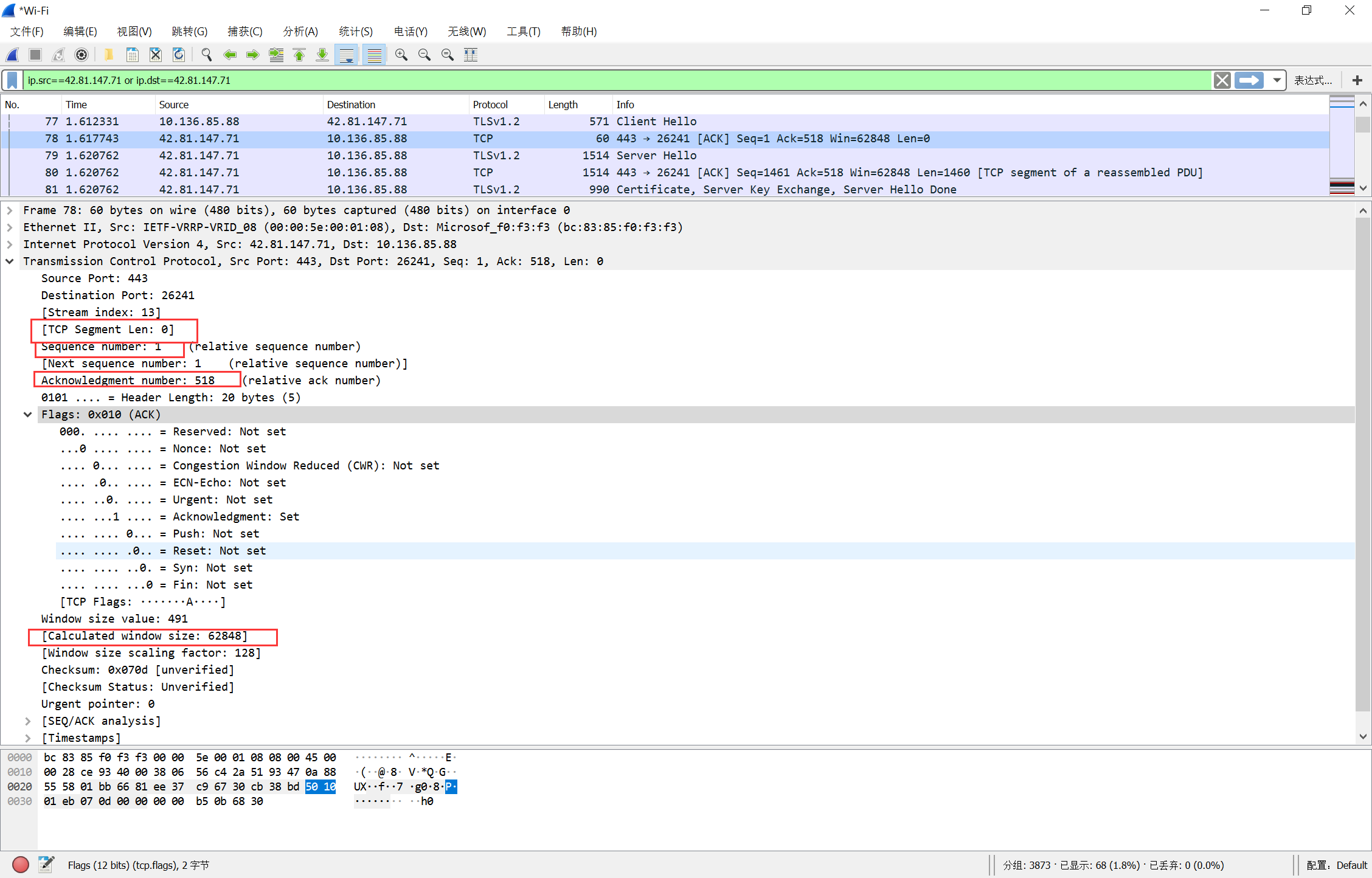
## 客户端->服务器



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。

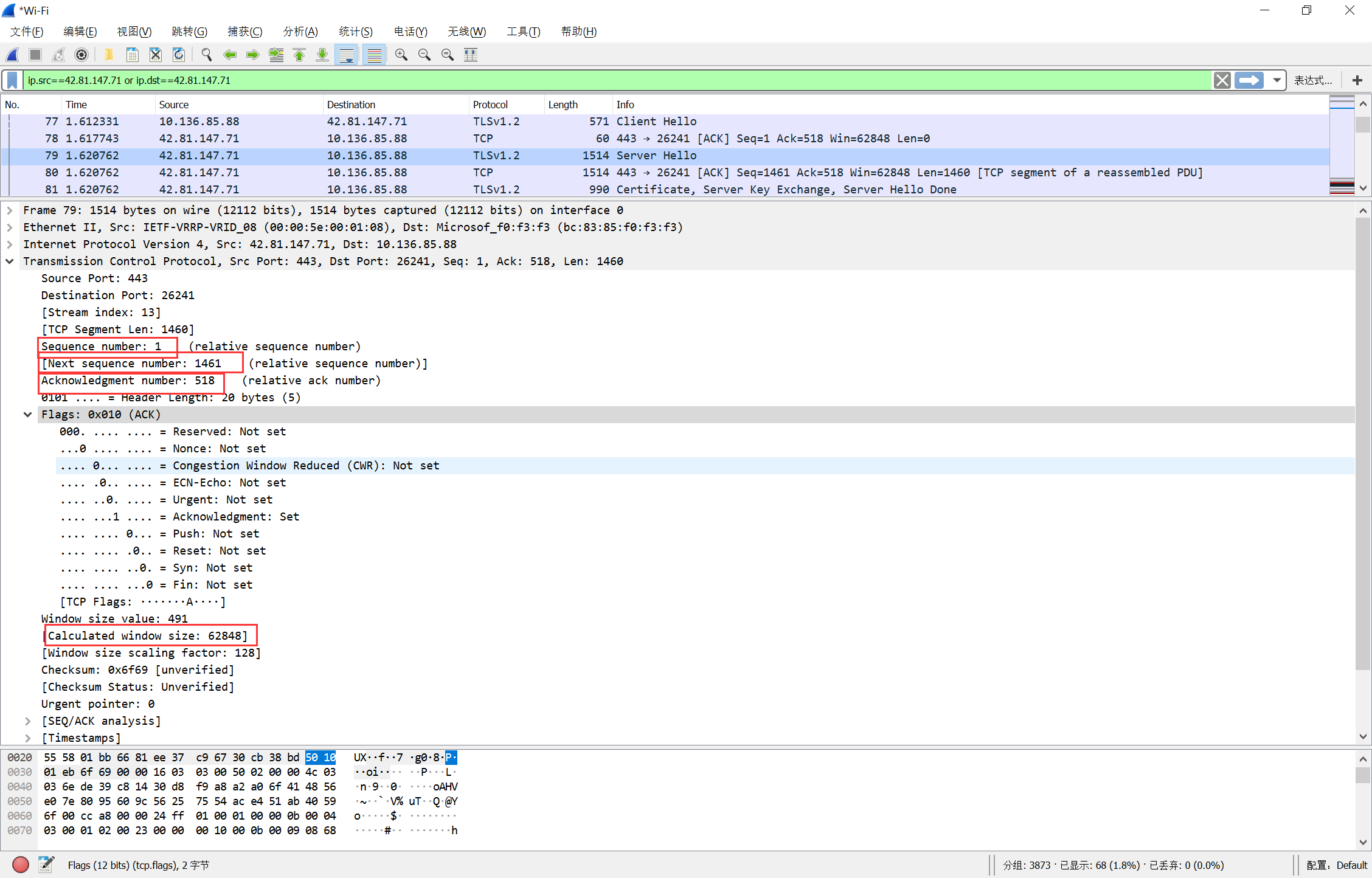
* 源IP：10.136.85.88
* 目标IP：42.81.147.71
* Seq=1：表示当前已经发送了1个数据。
* Next Sequence number：在完成这次的发送后，当发送新的数据时，Seq从518开始编码
* Ack = 1 : 表示当前端成功接收的1个数据位数
* Win=131328
* Len=517 说明里面包含有数据。
* TCP playload：Hypertext Transfer Protocol 部分的数据量，本次传送的数据量为517 byte

## 服务器->客户端



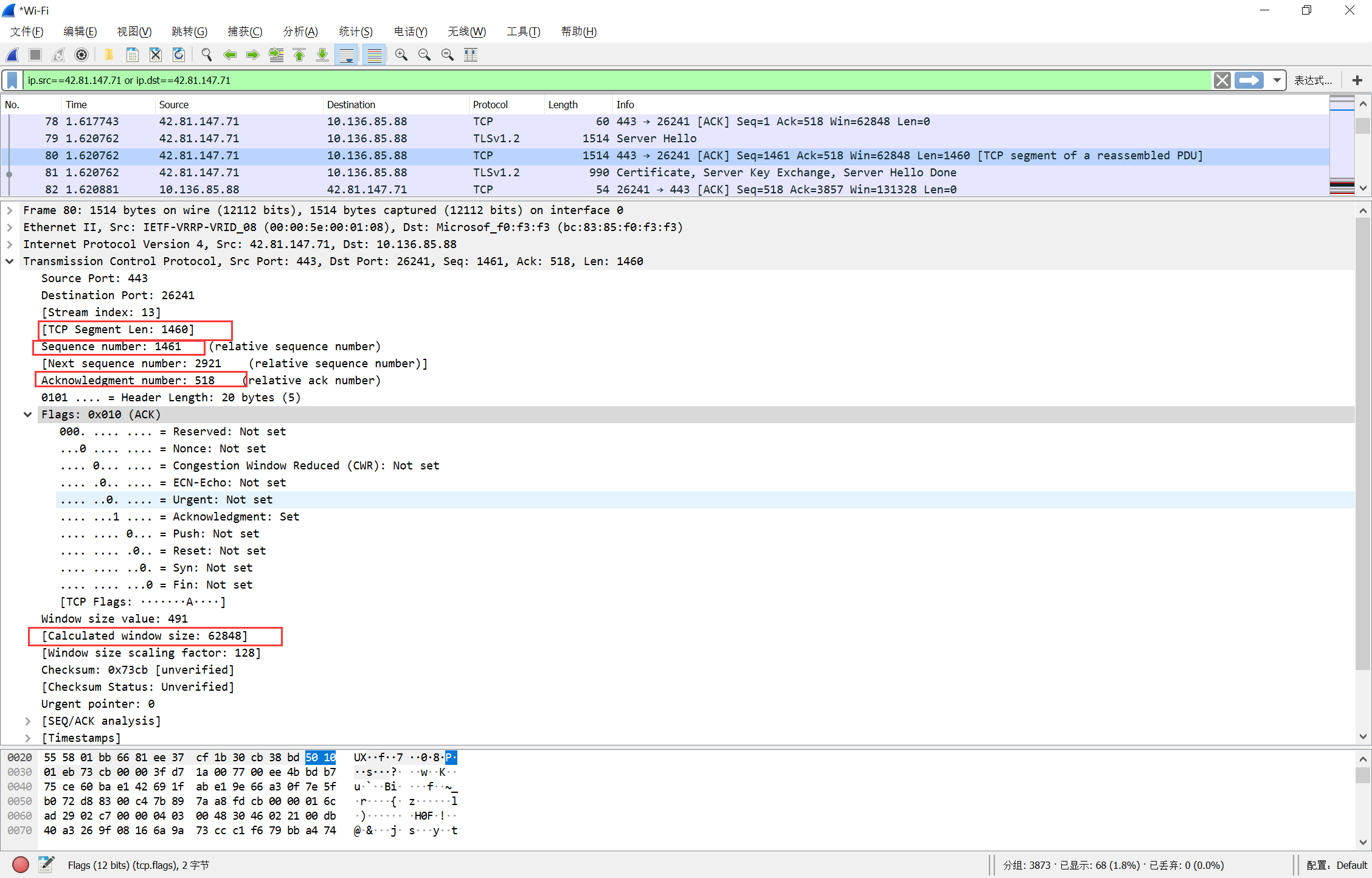
第78条：服务端接受发送过来的517个字节数据，在接收请求过程中，发送一次ACK包，

* 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。

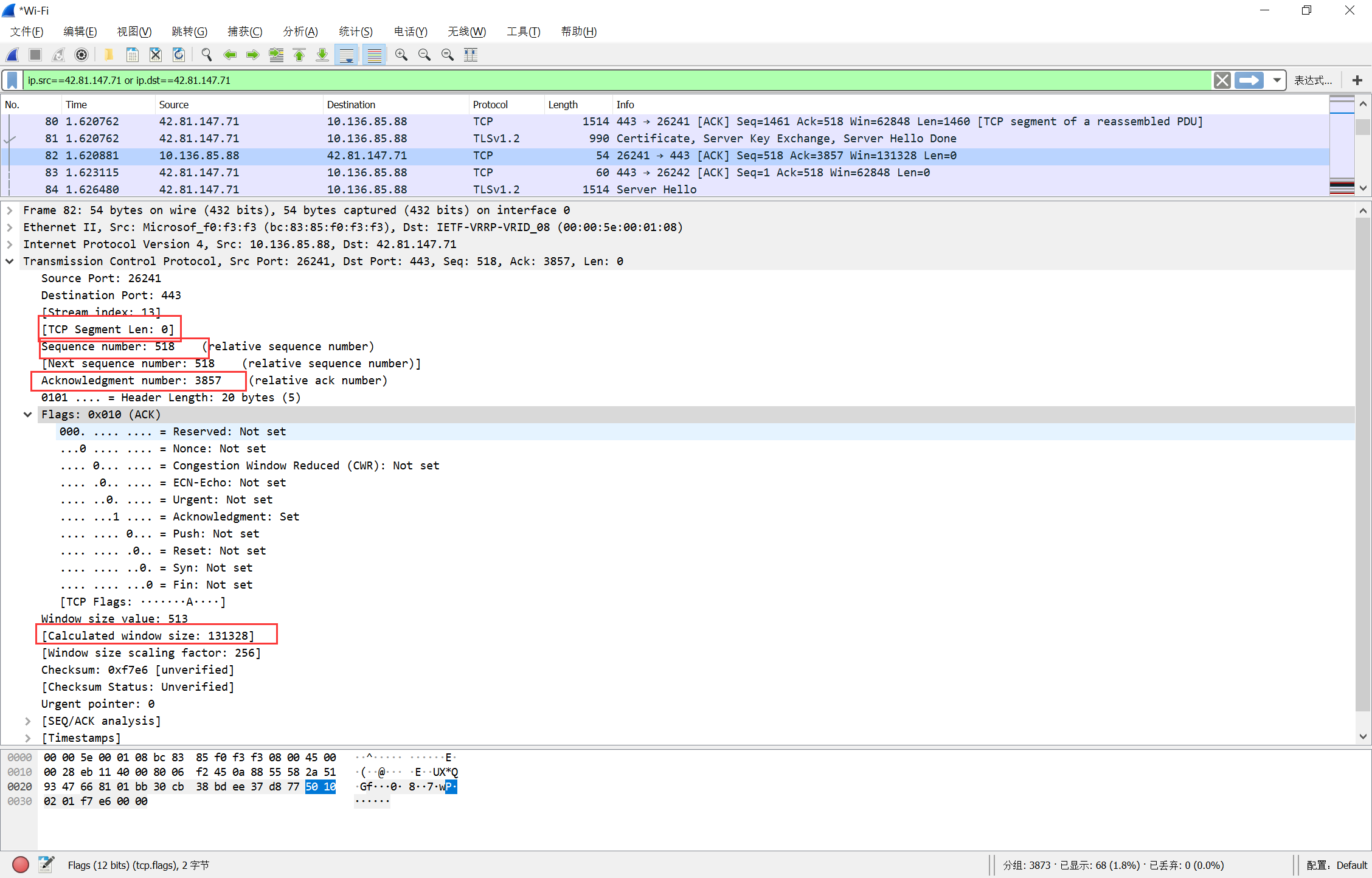
* 源IP：42.81.147.71
* 目标IP：192.168.43.136
* Seq=1，Ack=518，说明共收到518个字节的数据，期待下一次传过来的数据的seq为518。
* Win=62848，
* Len=1460，



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



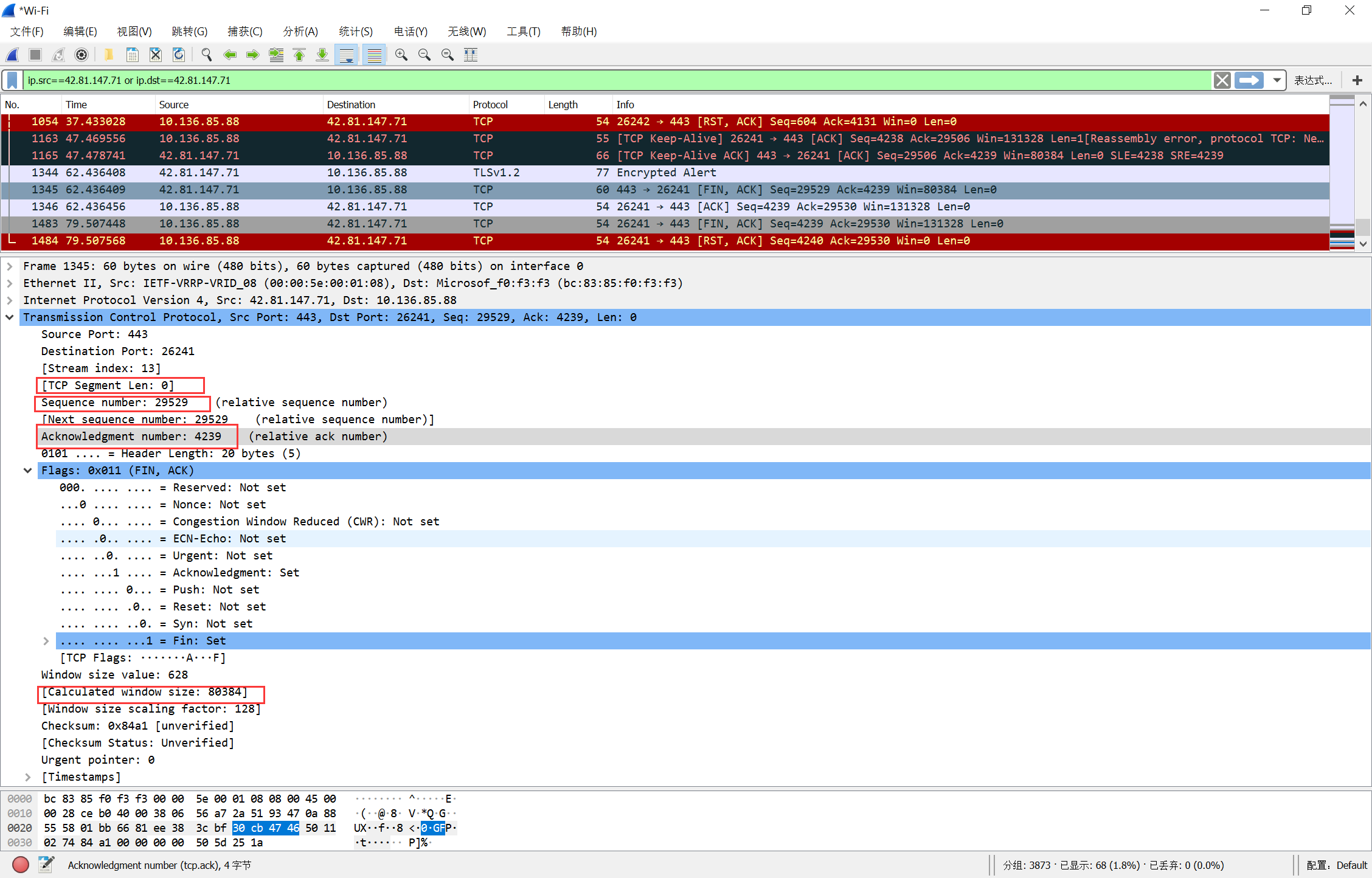
第82条告诉服务器，服务器的hello客户端收到了，此后应该交换密钥。

* 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：

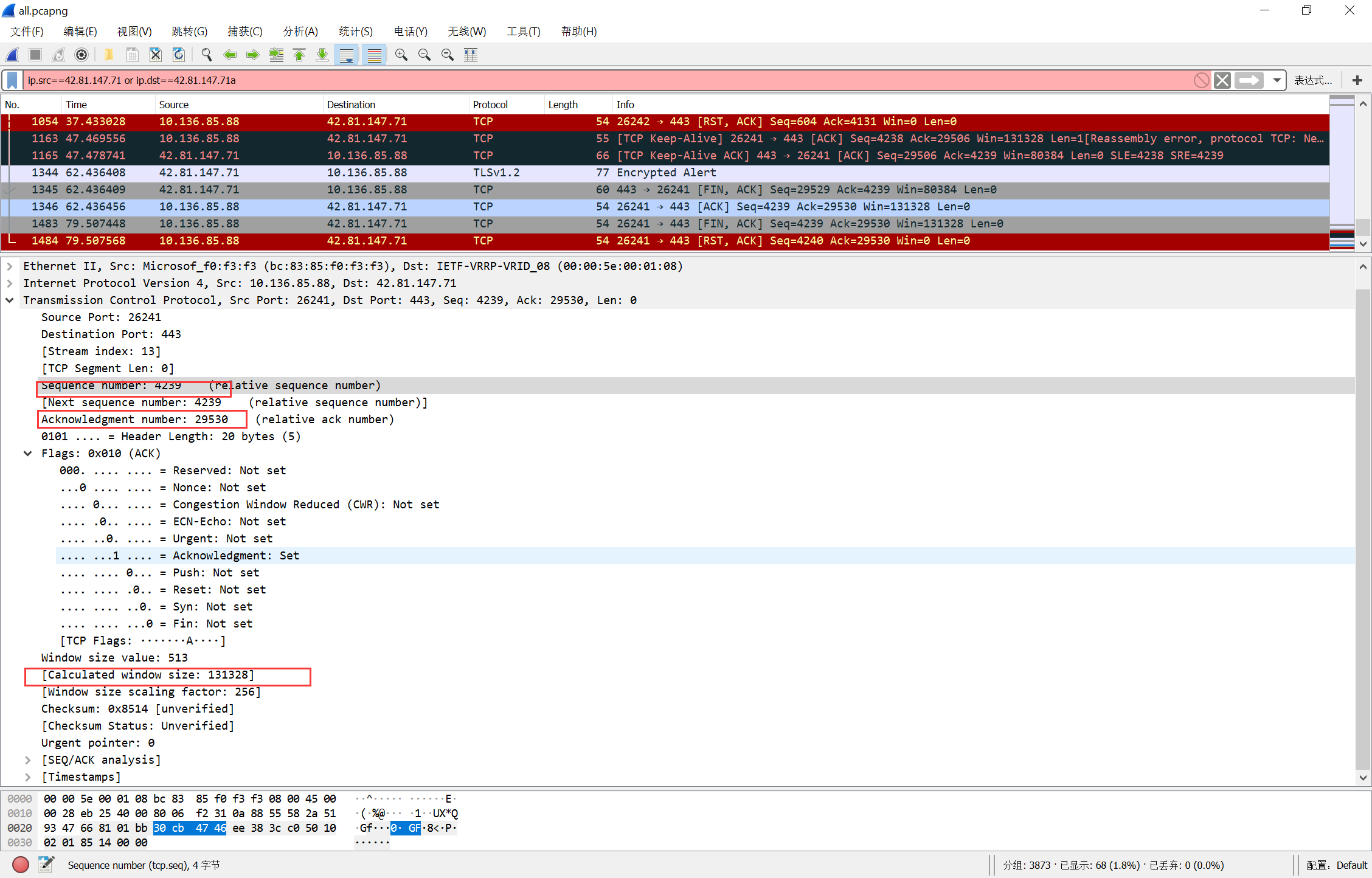
## 第一次挥手



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。

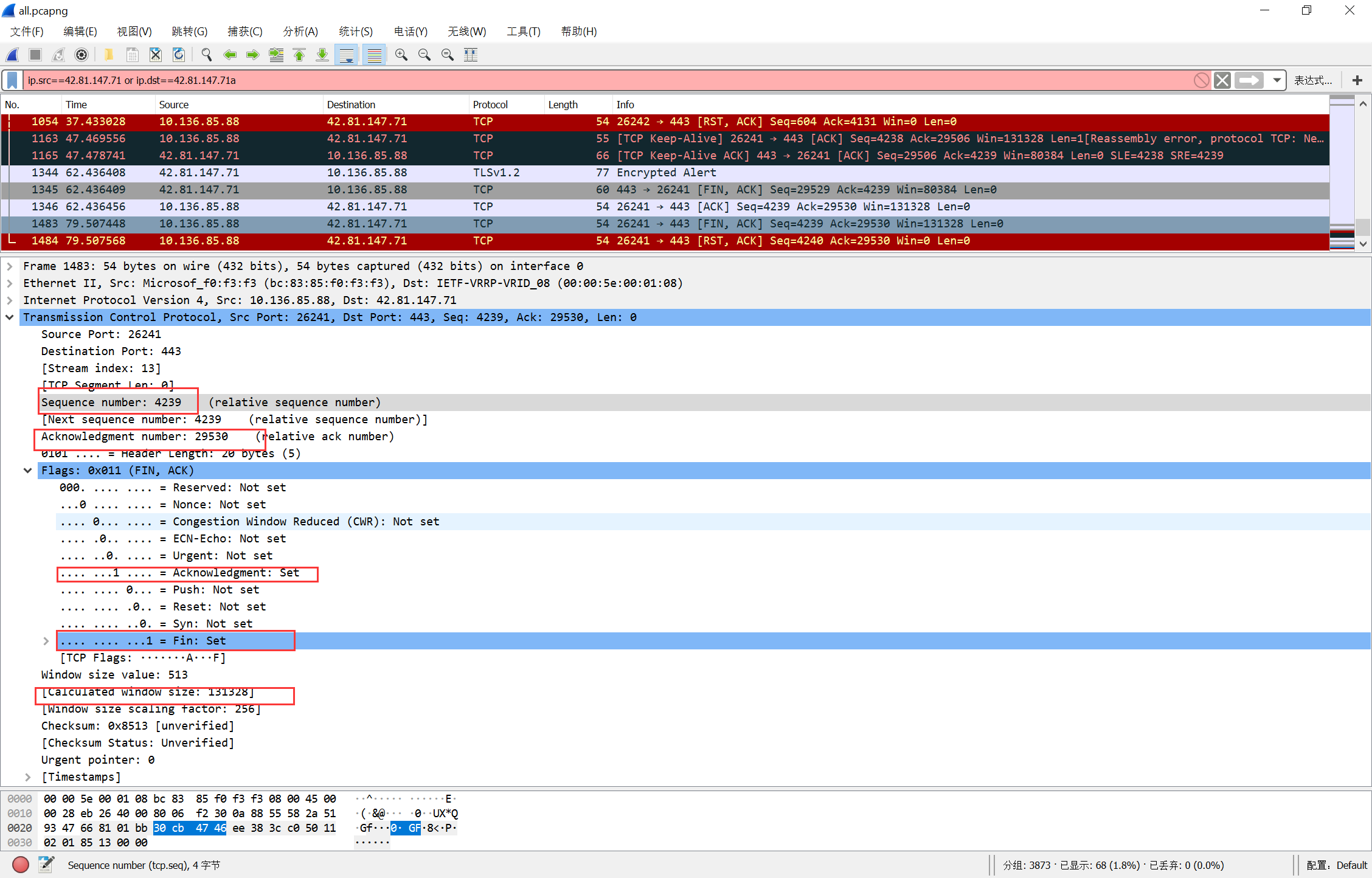
## 第二次挥手



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。