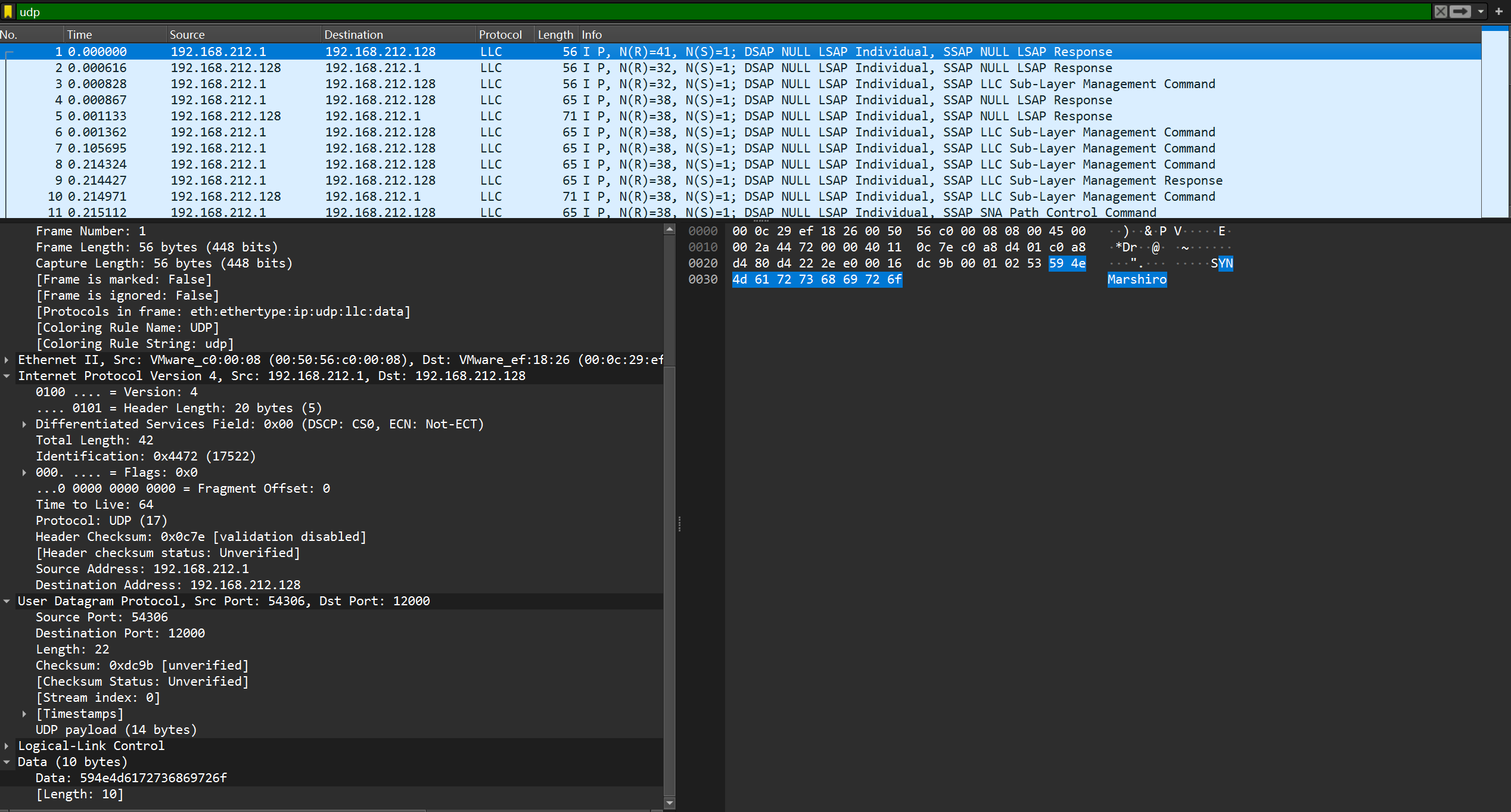
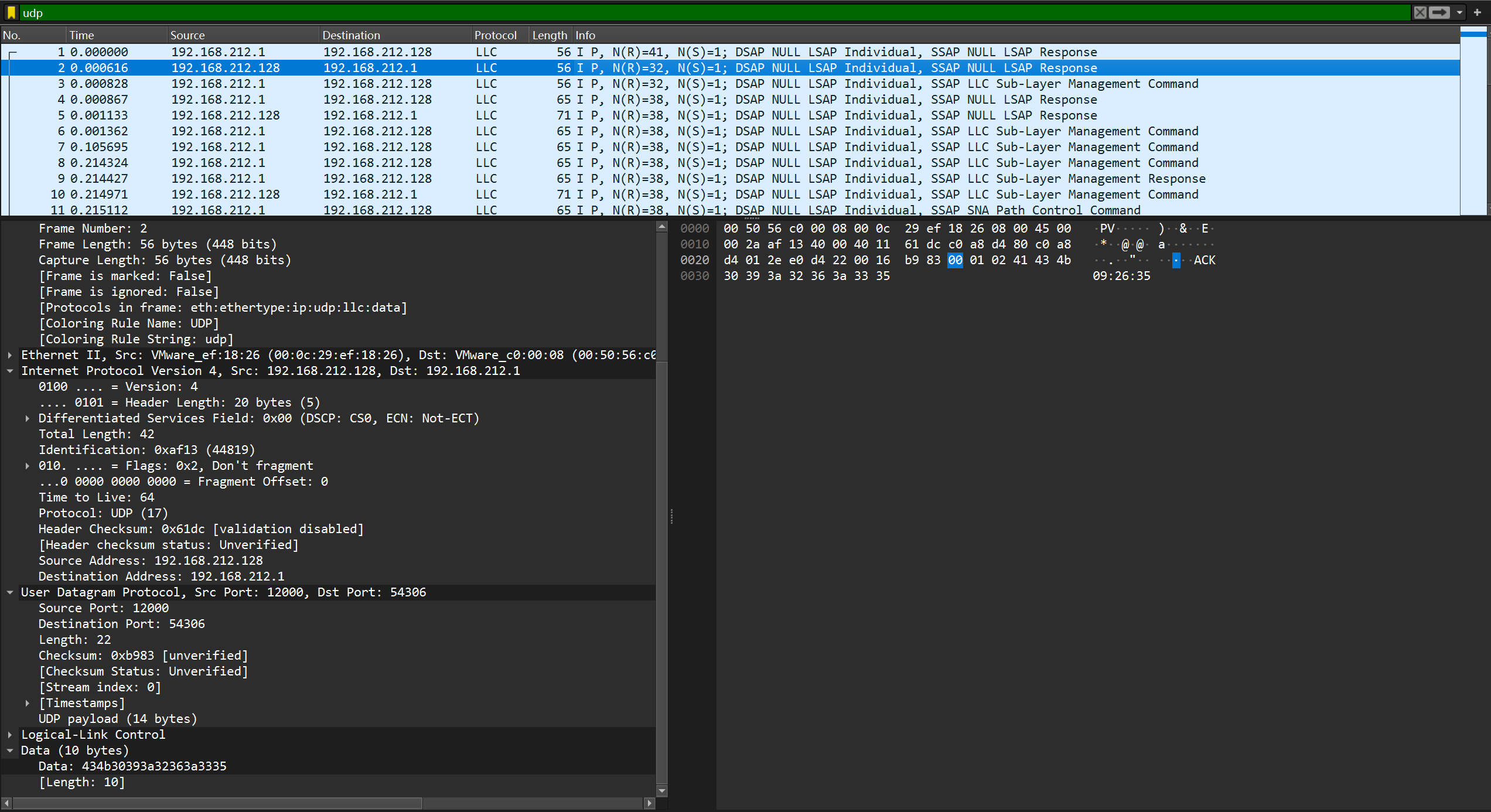
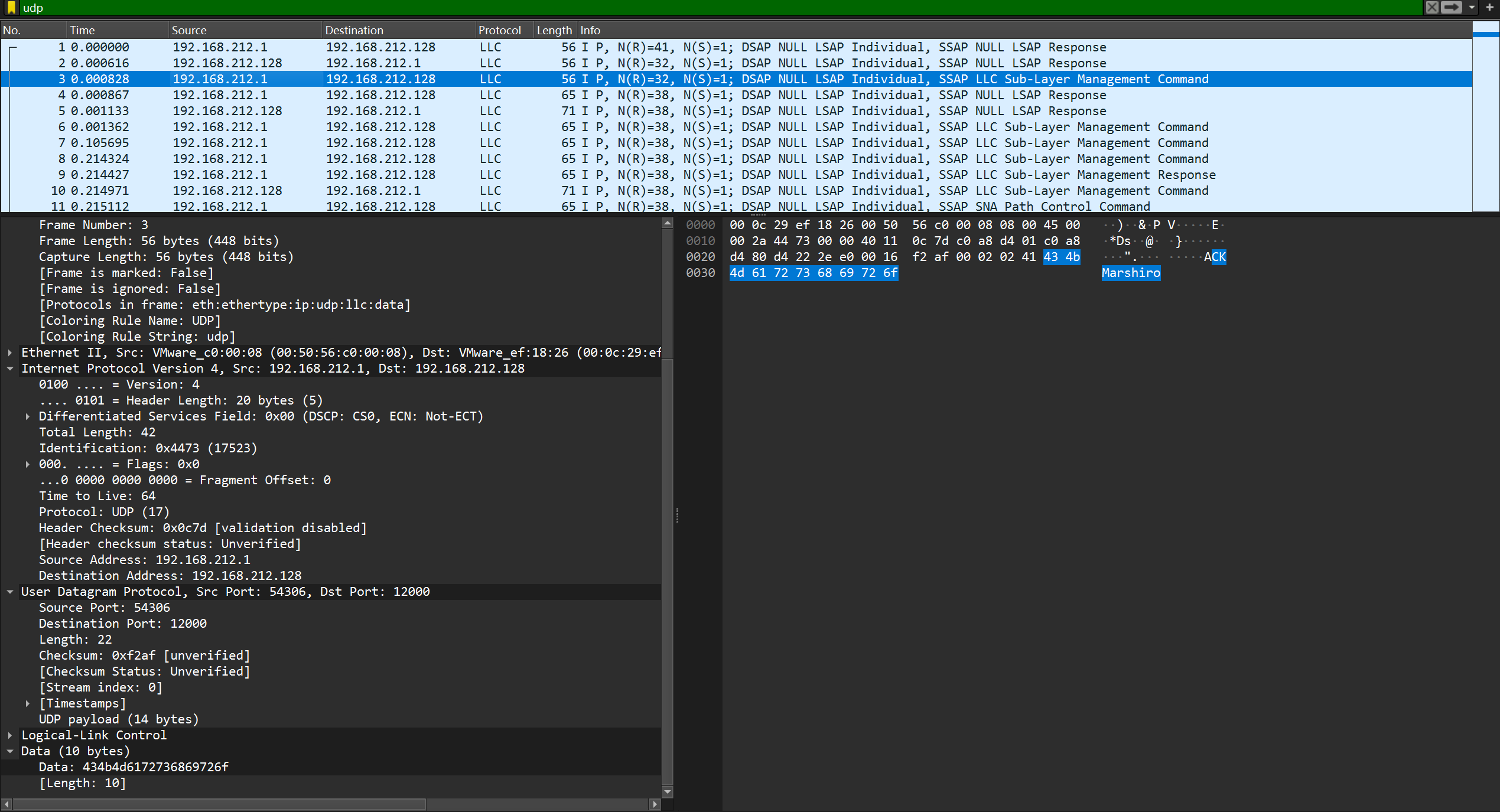
**a.**

**三次握手：**

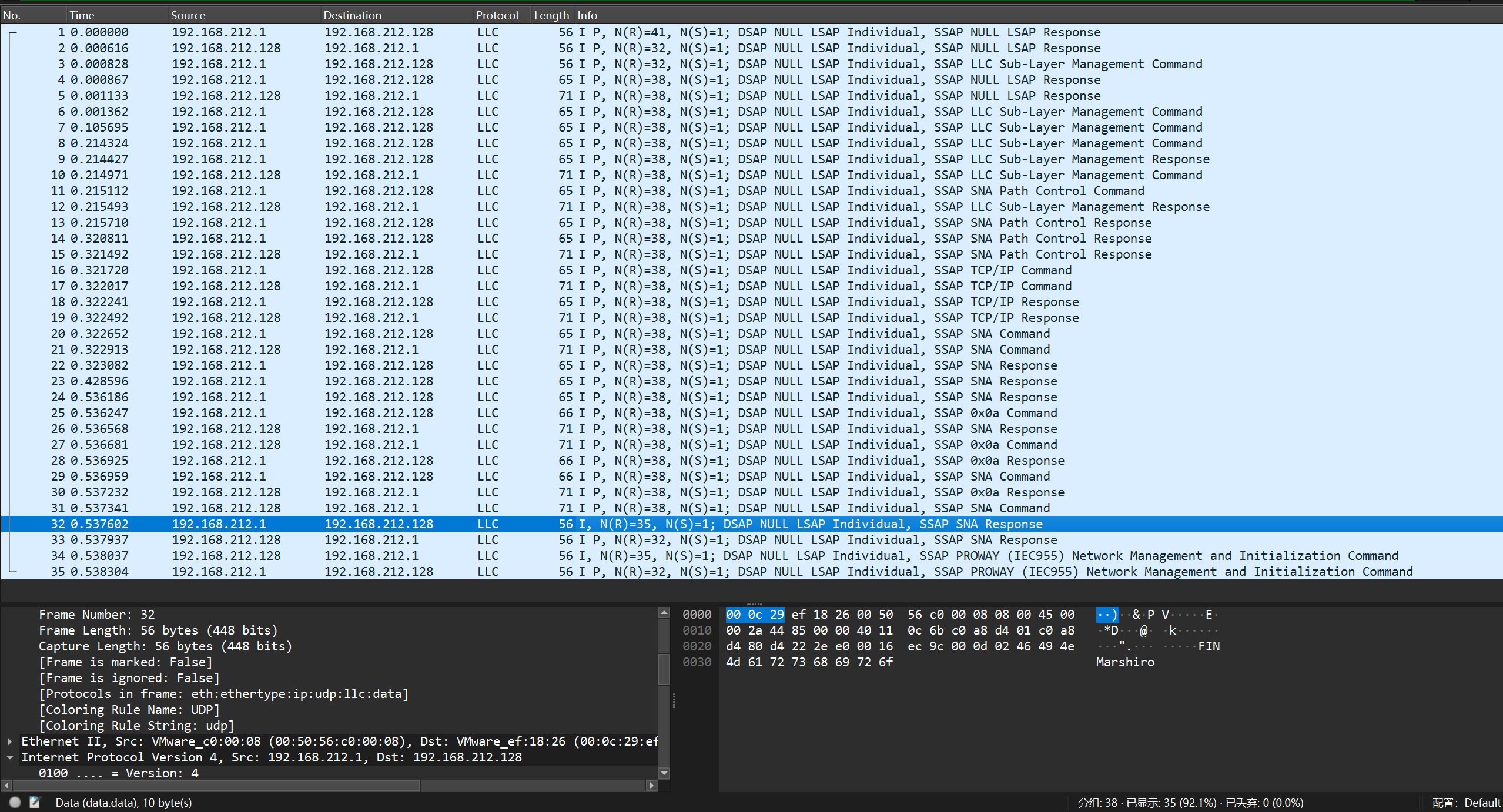




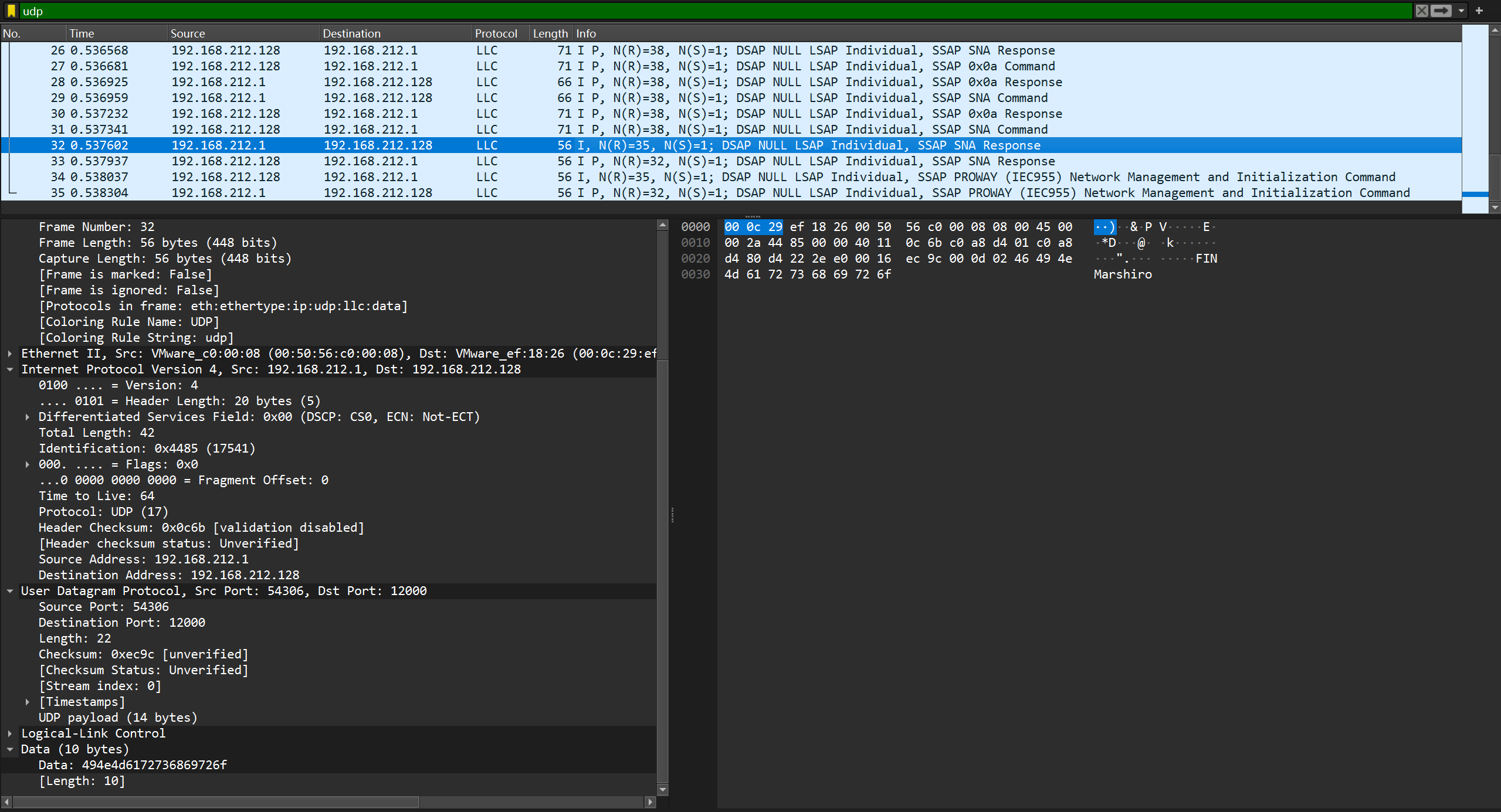


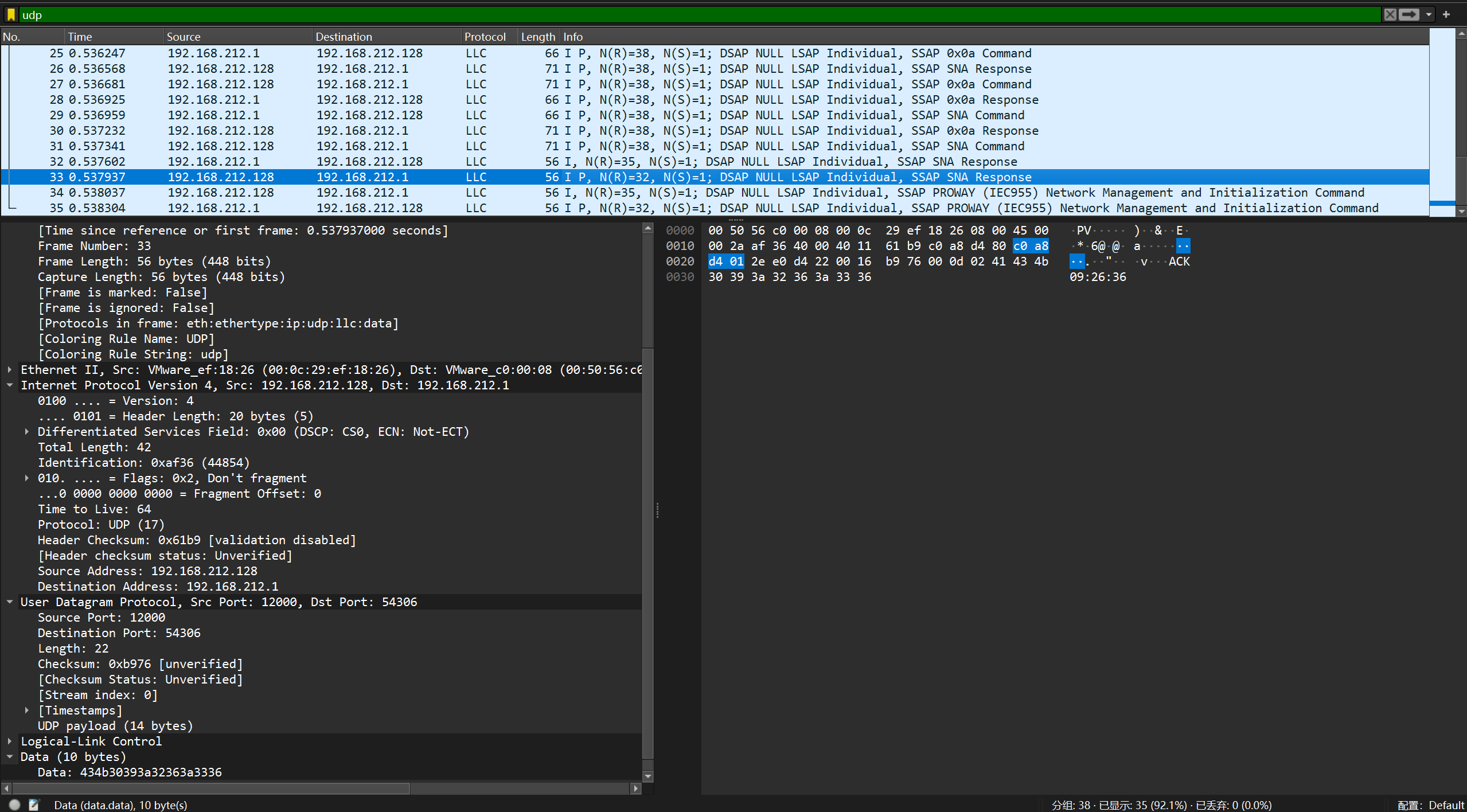
如图192.168.212.1作为client向server 192.168.212.128发送类型为SYN的连接请求，server回复ACK并附带时间，client回复ACK，此后连接建立并进入数据传输阶段，成功模拟实现了TCP三次握手过程

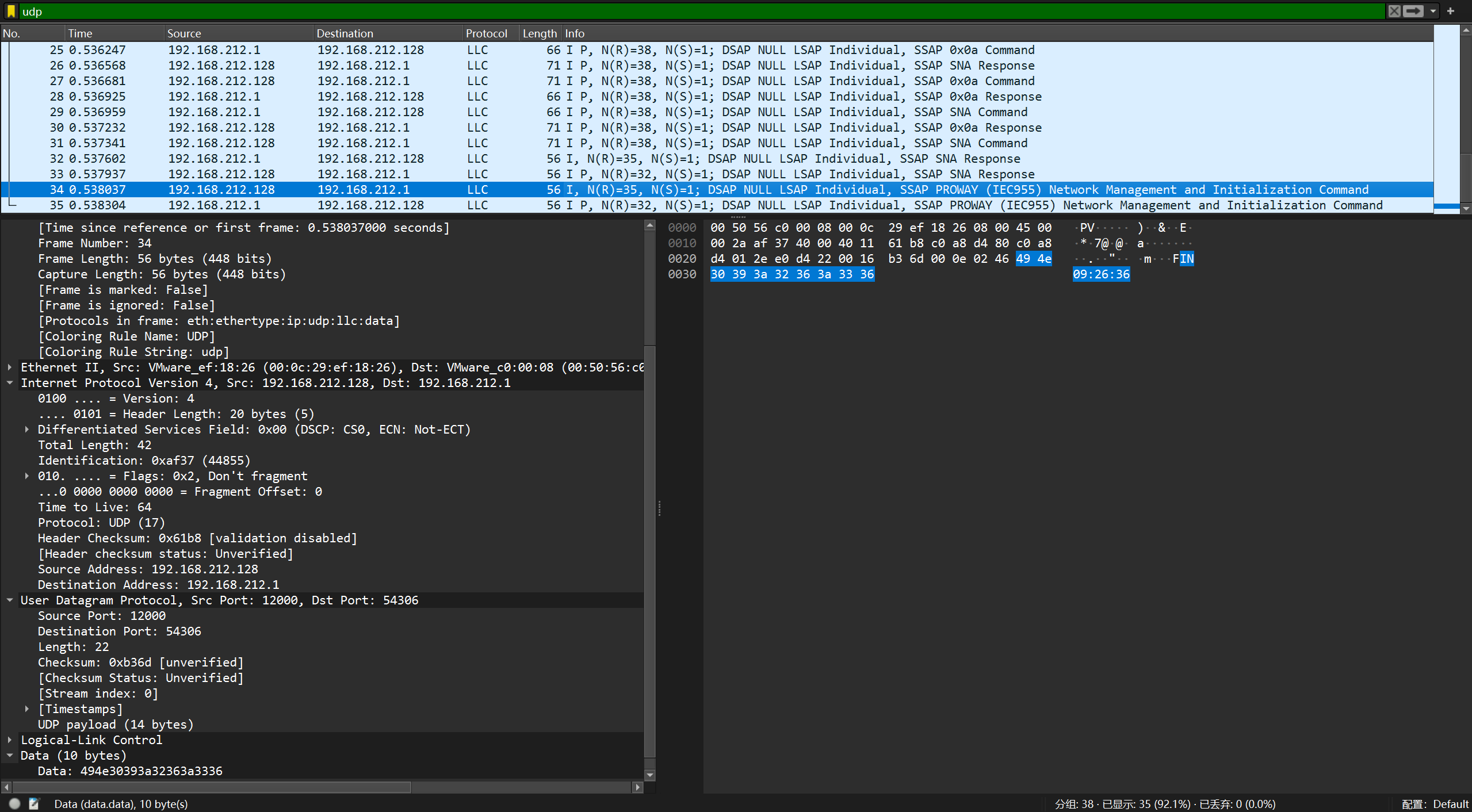
**全部包体**：

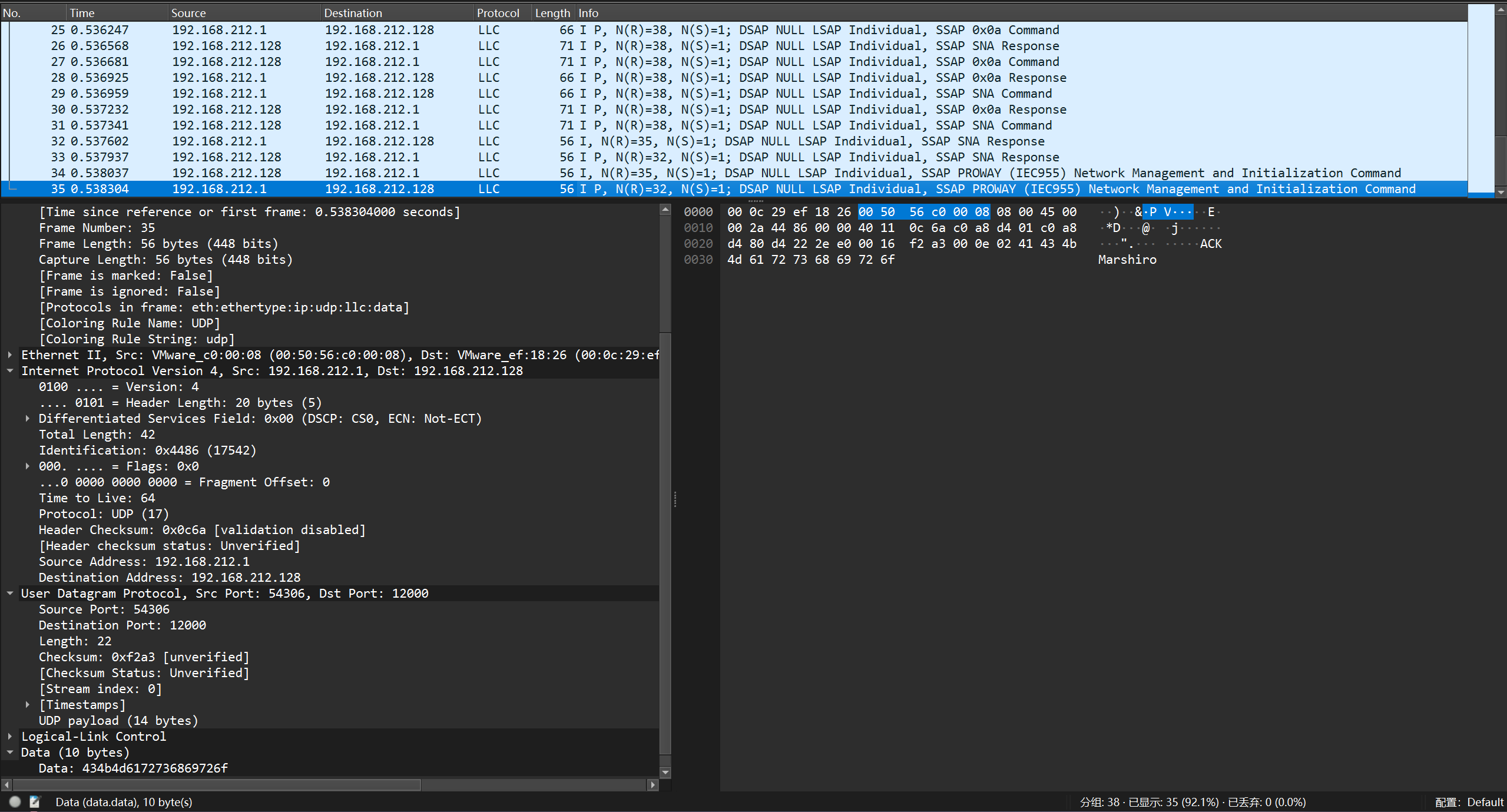


**四次挥手：**



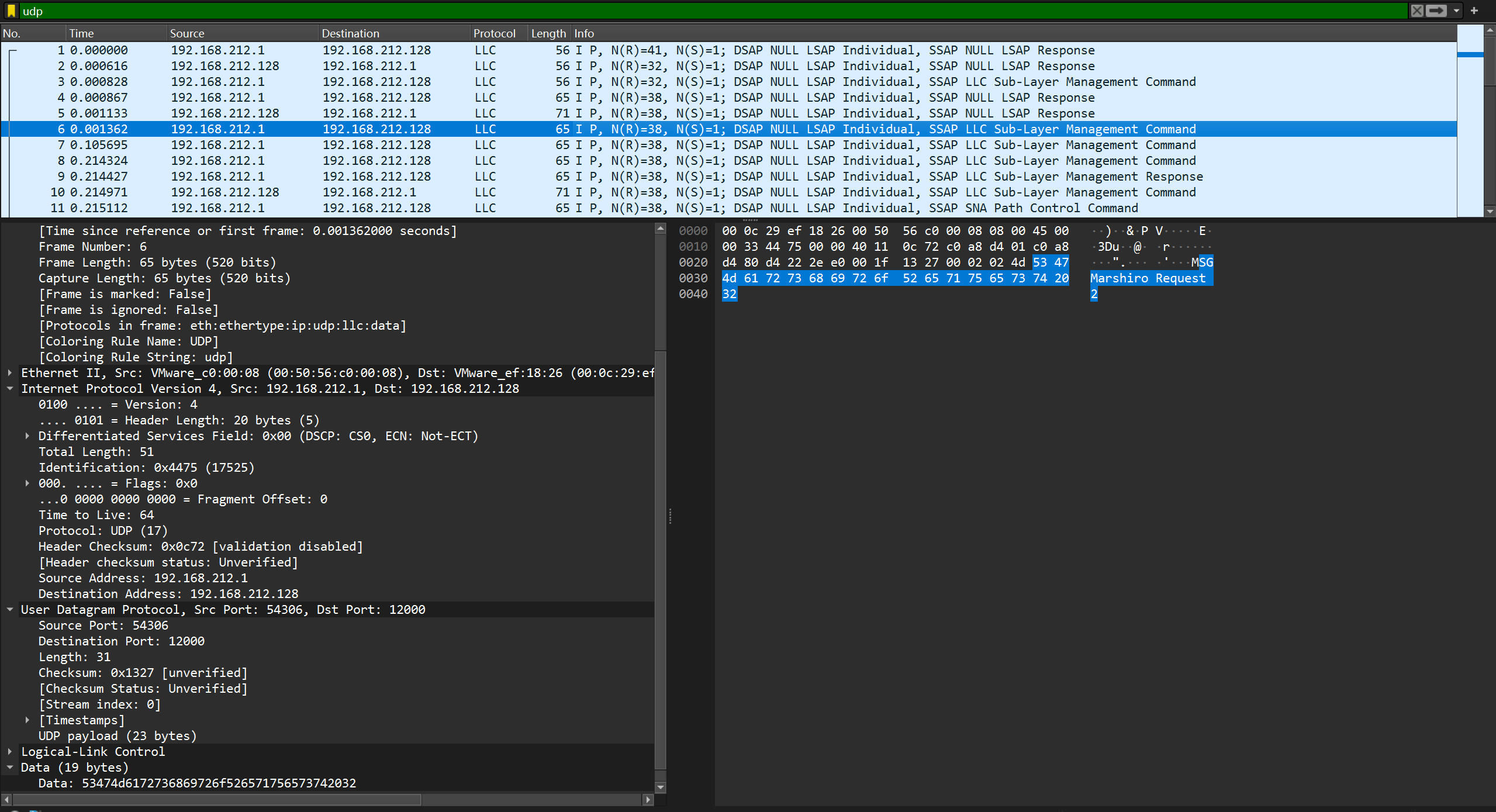


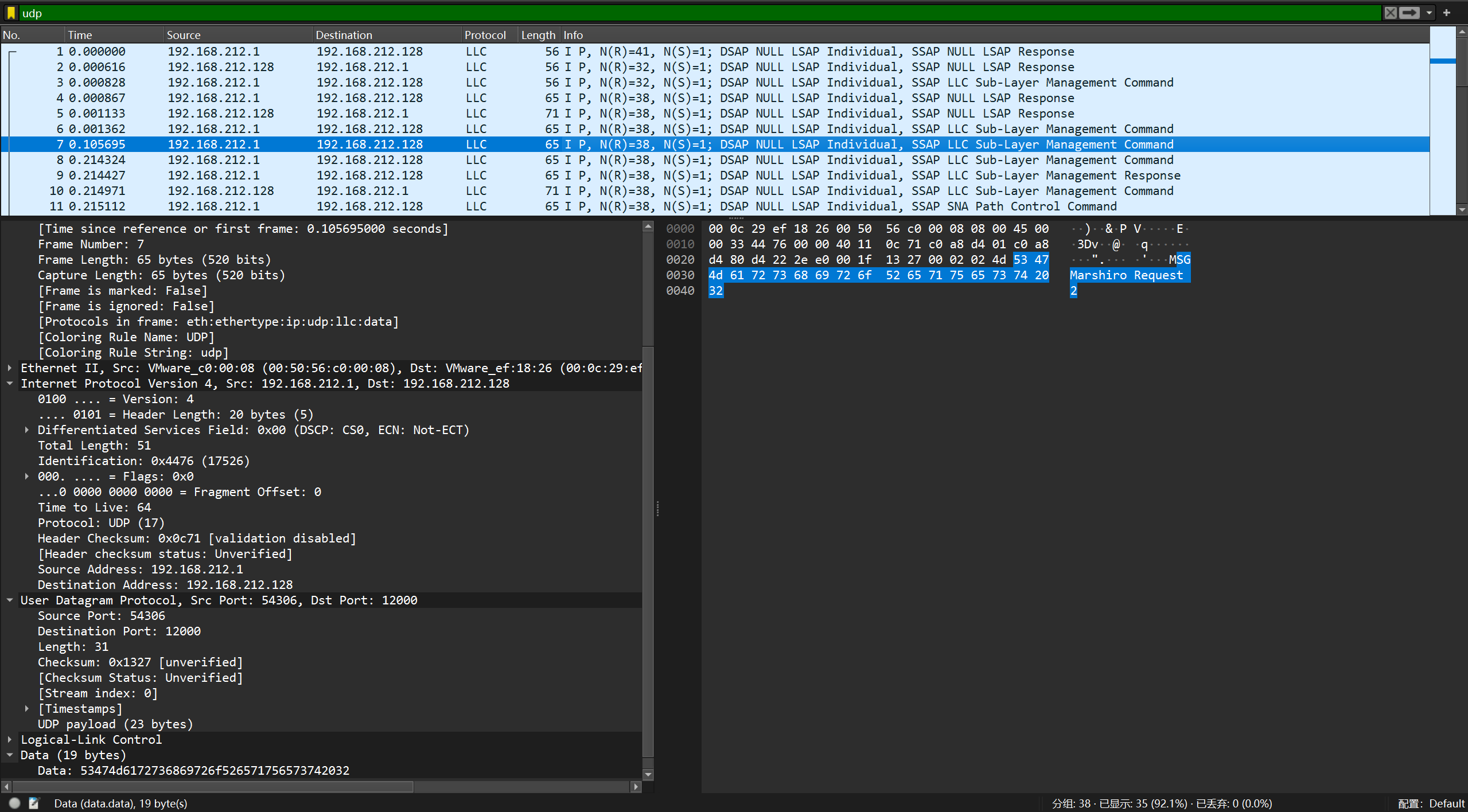


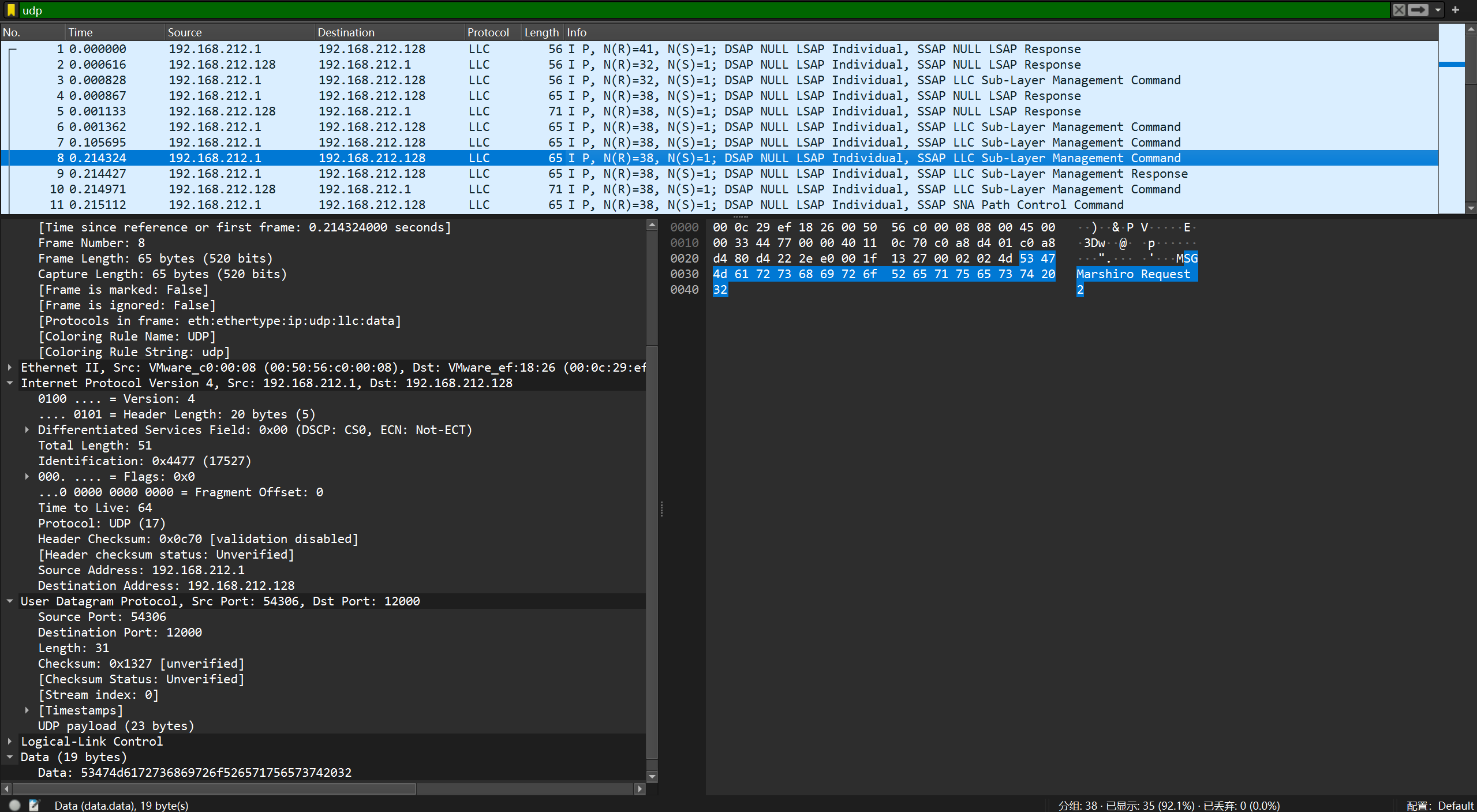


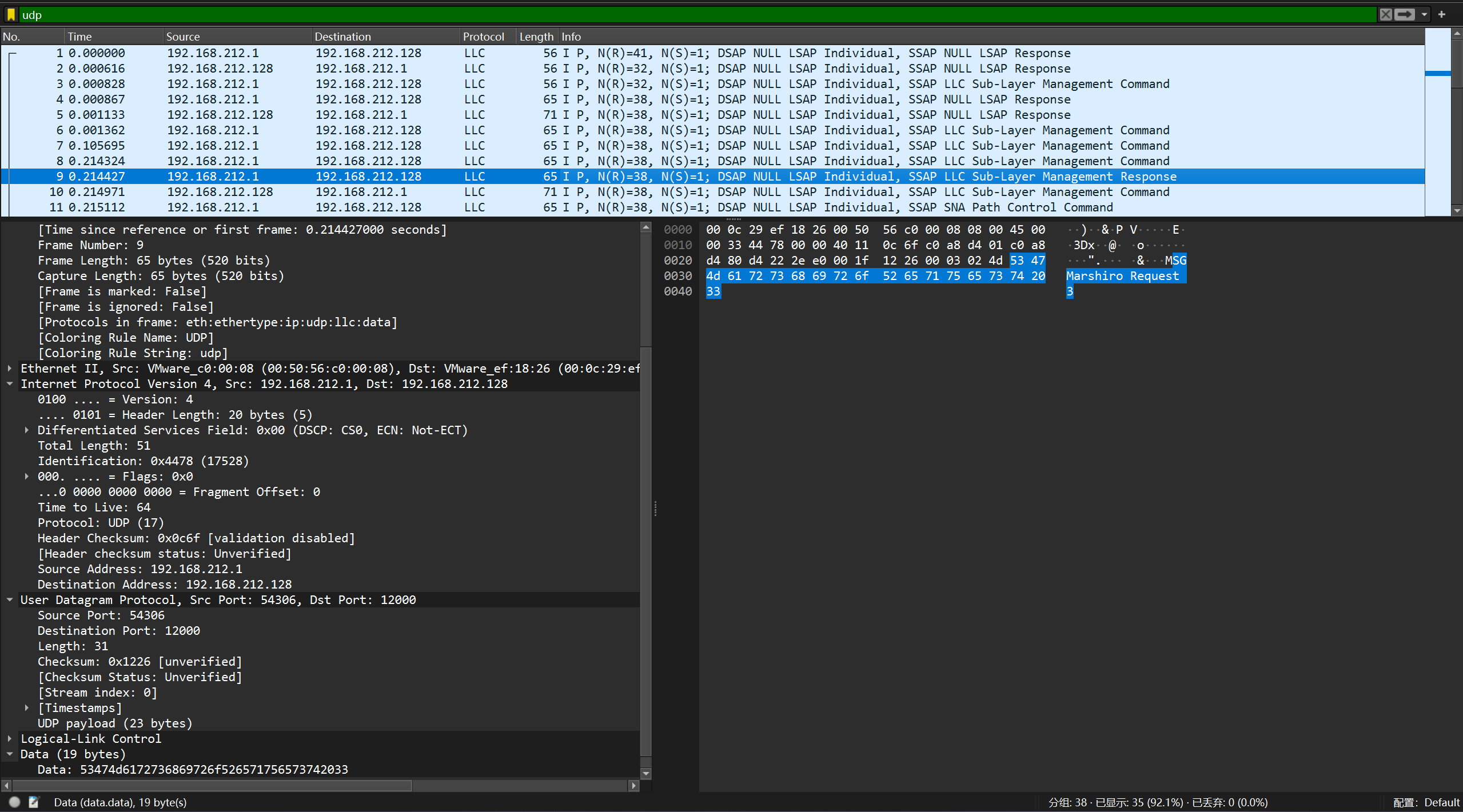
如图client向server发送FIN，server回复ACK然后再向client发送FIN，client接收到后也回复ACK，之后连接关闭，成功模拟实现了TCP四次挥手过程

**在发送第二条数据时两次重传：**









如图所示编号为2的MSG类型数据报发送了三次也没有收到回复，client转为发送编号为3的数据报

**b.**

实现关键在于如何设计报文格式，并在编码解码报文段时实现所设计的字段。我设计的client和server端报文格式一致为：2 Bytes seq\_no序号；1 Bytes ver 版本号；3Bytes字符串 msg\_type 报文类型；8 Bytes time 系统时间（client端为time\_filler无意义字符“Marshiro”进行填充）

解码编码利用struct库中pack和unpack函数实现，H，B，3s，8s分别对应2 Bytes，1 Bytes，3 Bytes字符串，8 Bytes字符串。将解码出的信息存入字典，方便后续查找使用。

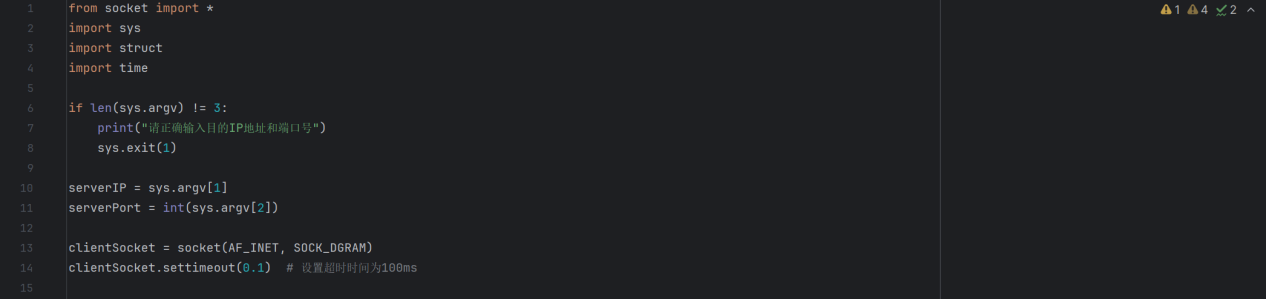


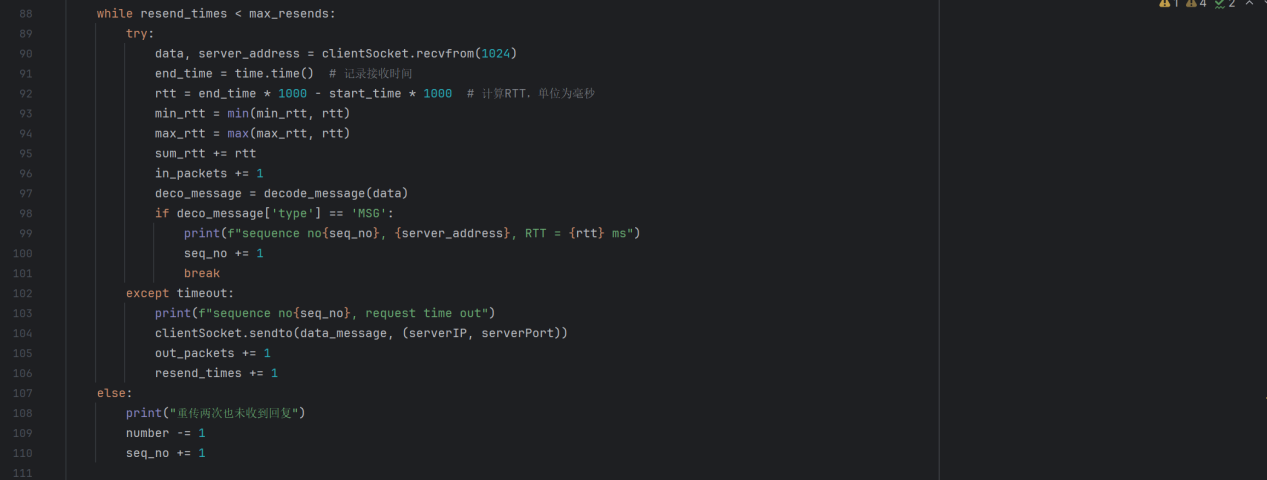


另一个关键点是服务器端如何在难以发生的UDP传输中模拟丢包，我使用了while循环接受客户端消息和continue结构，设置了一个随机数，若该随机数小于设置的丢包率便跳过该次循环即不对接受到的客户端数据报做处理也不回复，由此实现了对丢包情况的模拟。



最后一个关键点是客户端如何对超时情况做出响应，UDP套接字编程socket库提供了settimeout定时器，若发生超时情况则可以使用except捕捉timeout异常然进行重传，同时维护变量resend\_times记录重传次数，重传超过两次便丢弃该数据报转而发送下一个序号对应的数据报。





**c.**

主要使用到了TCP和UDP相关知识，通过完成该任务利用UDP模拟TCP过程我熟练掌握了UDP套接字编程的相关内容，同时对UDP数据传输过程以及TCP建立连接，数据传输，关闭连接过程有了更深的理解。

**d.**

https://github.com/Destiny-zks/BJFU-2024ComputerNetwork.git