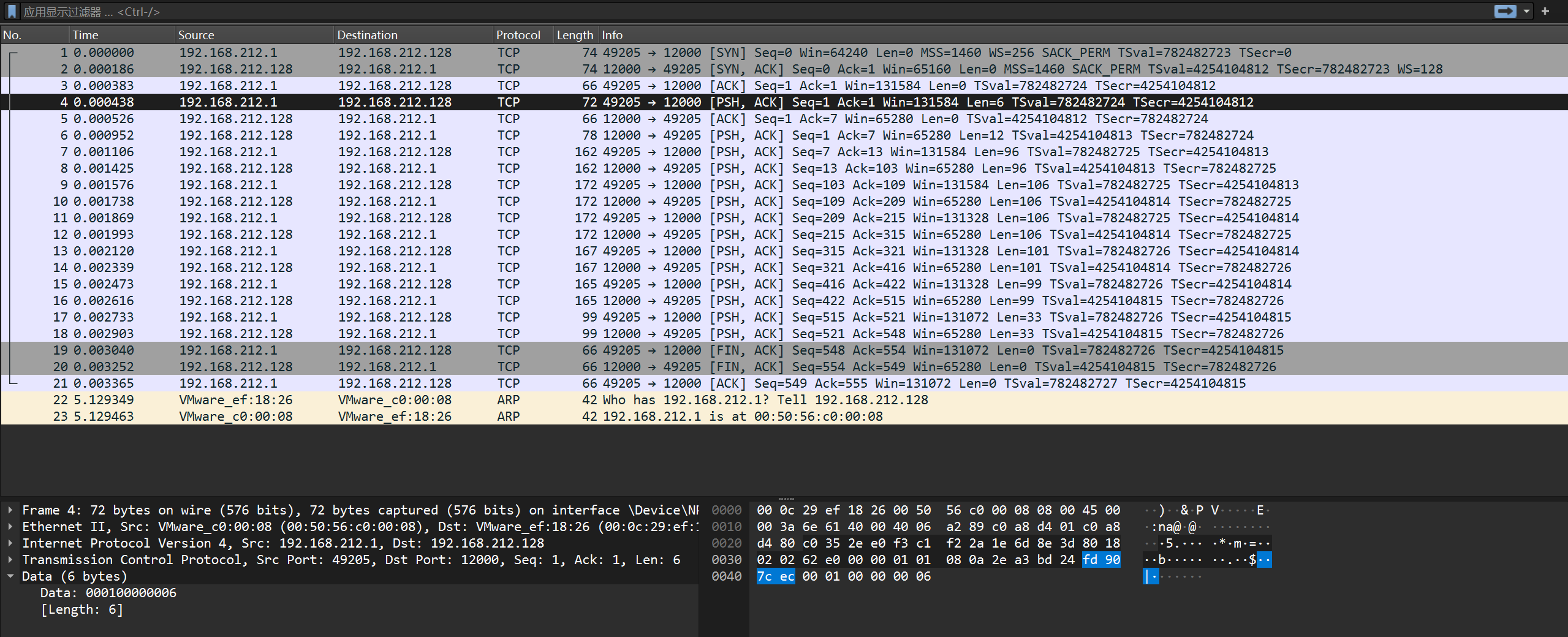
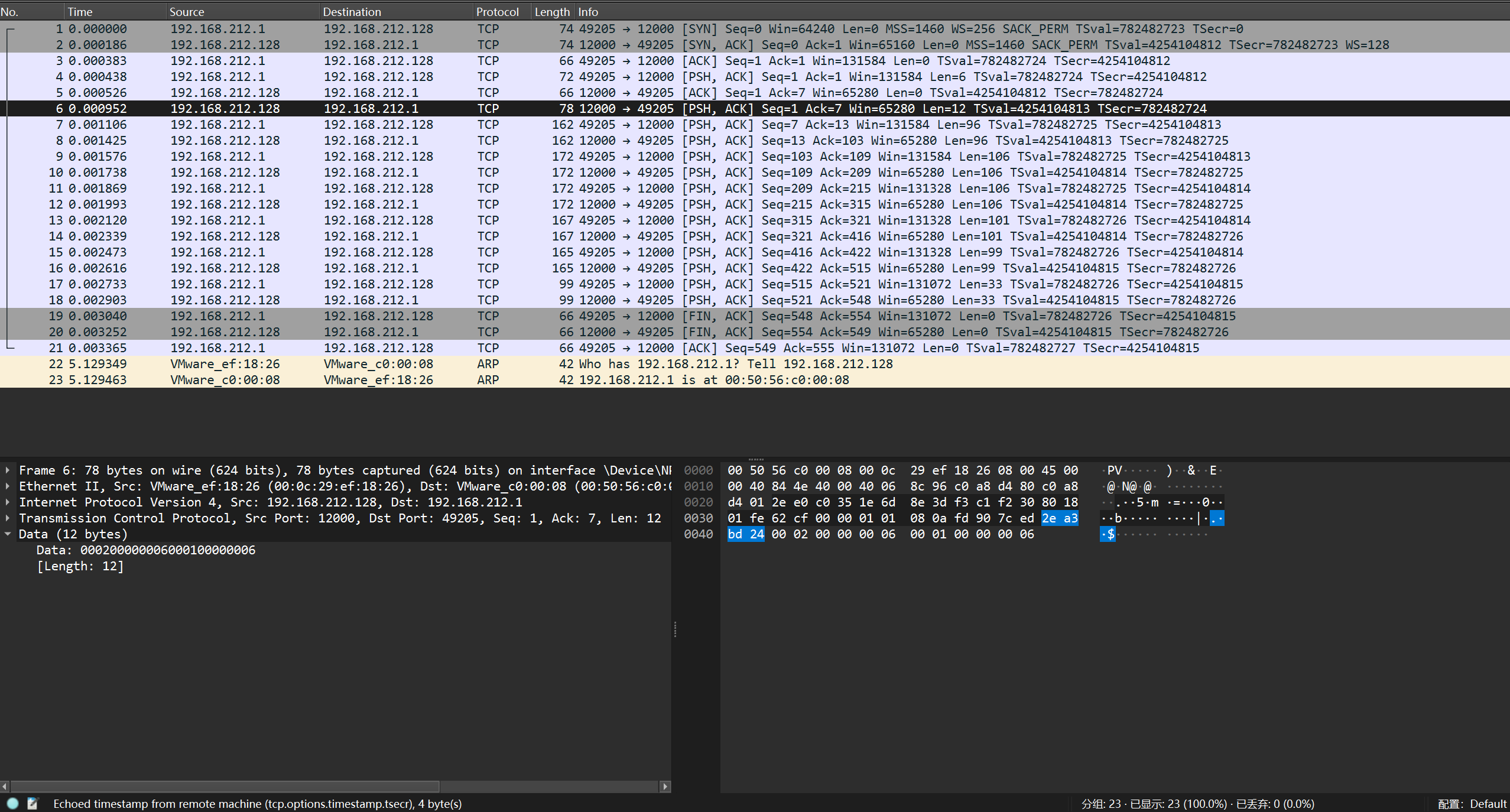
**a.**

data前四位为type

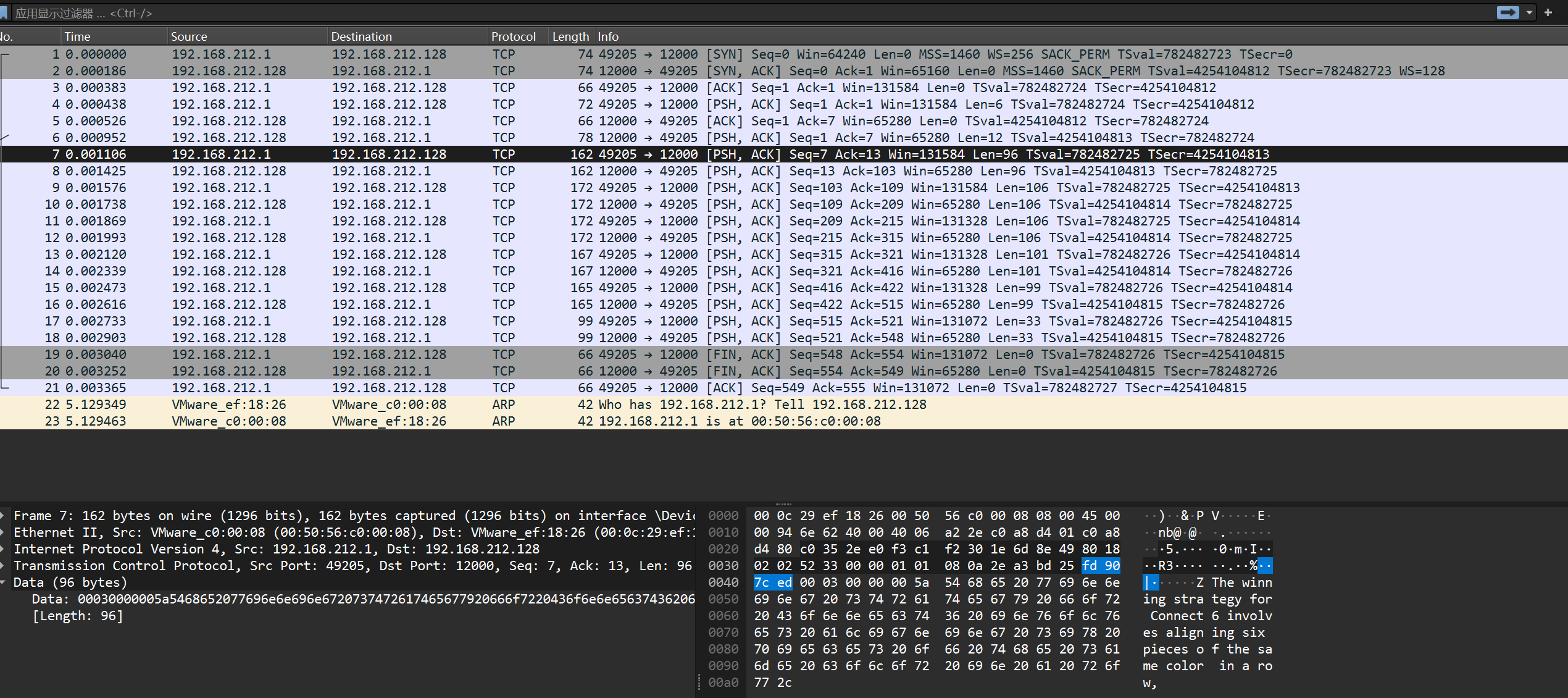
Type1:



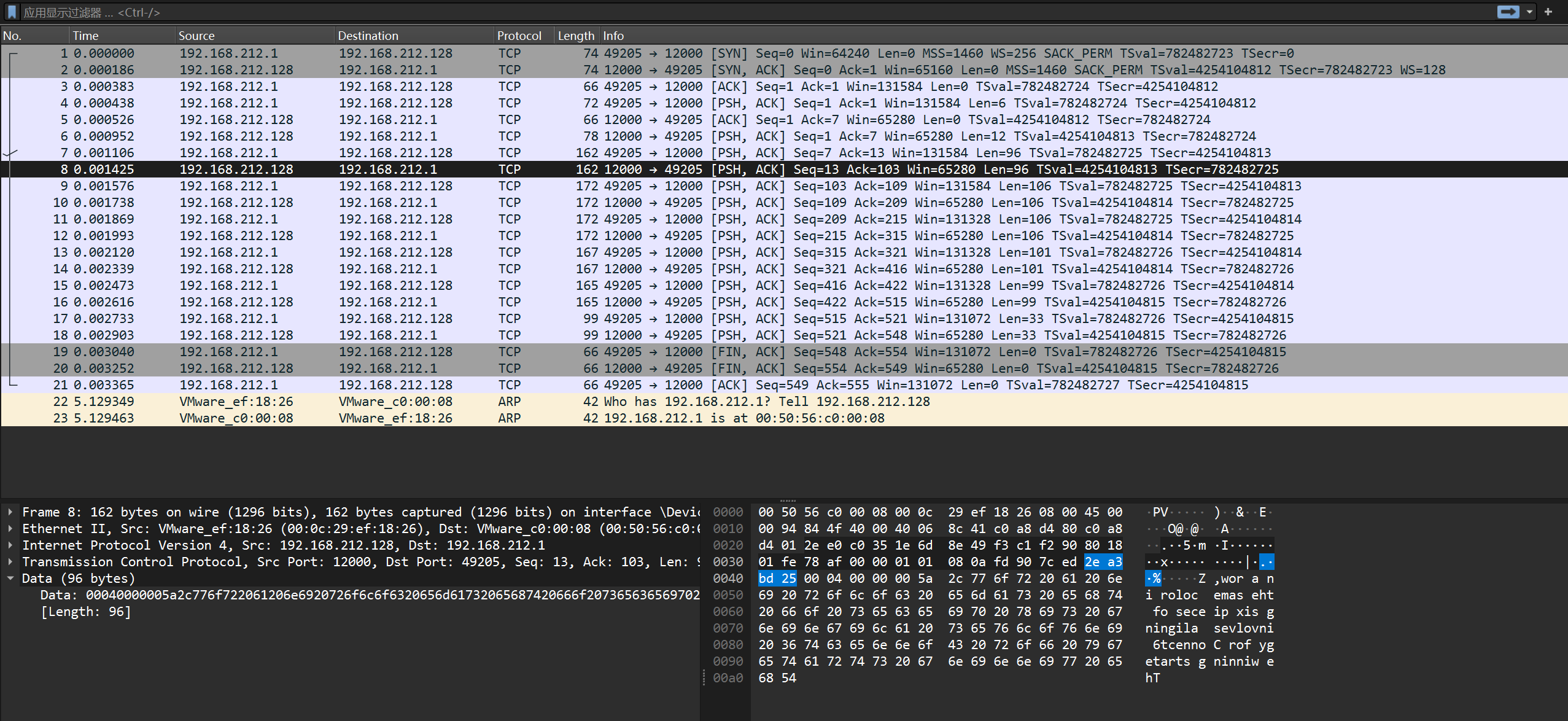
Type2:



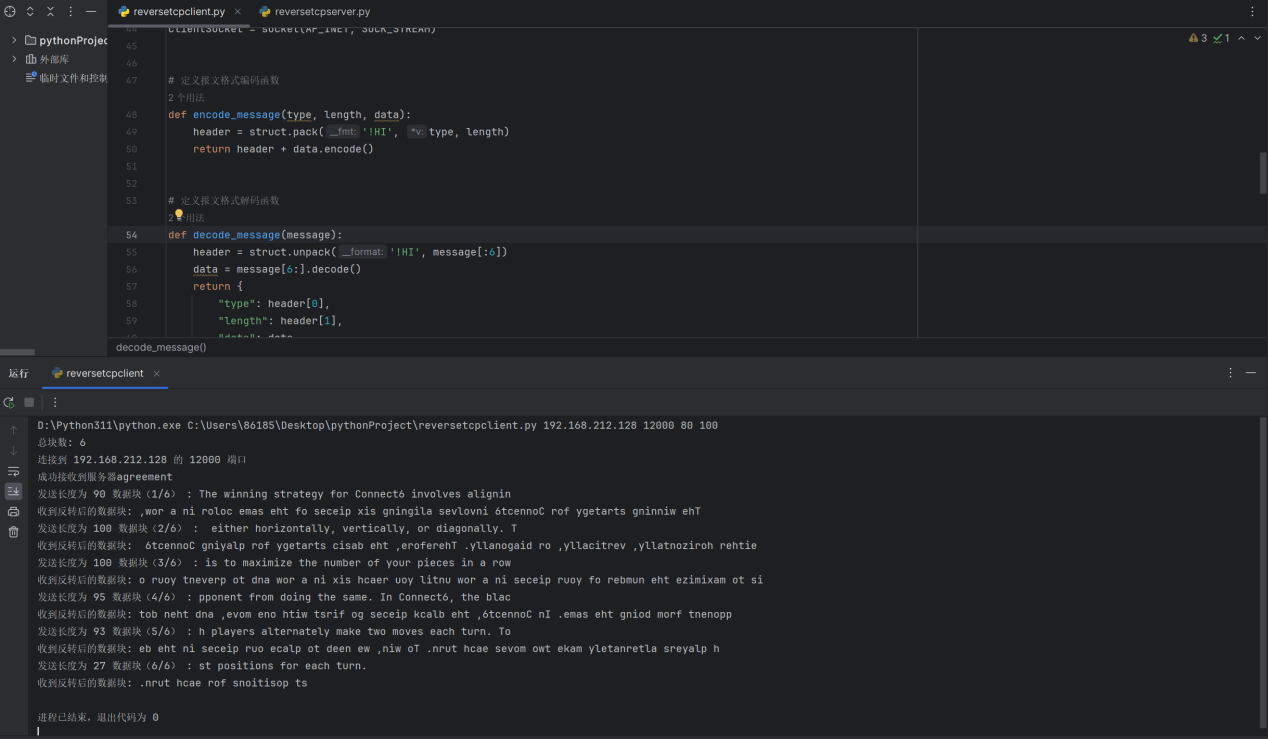
Type3:



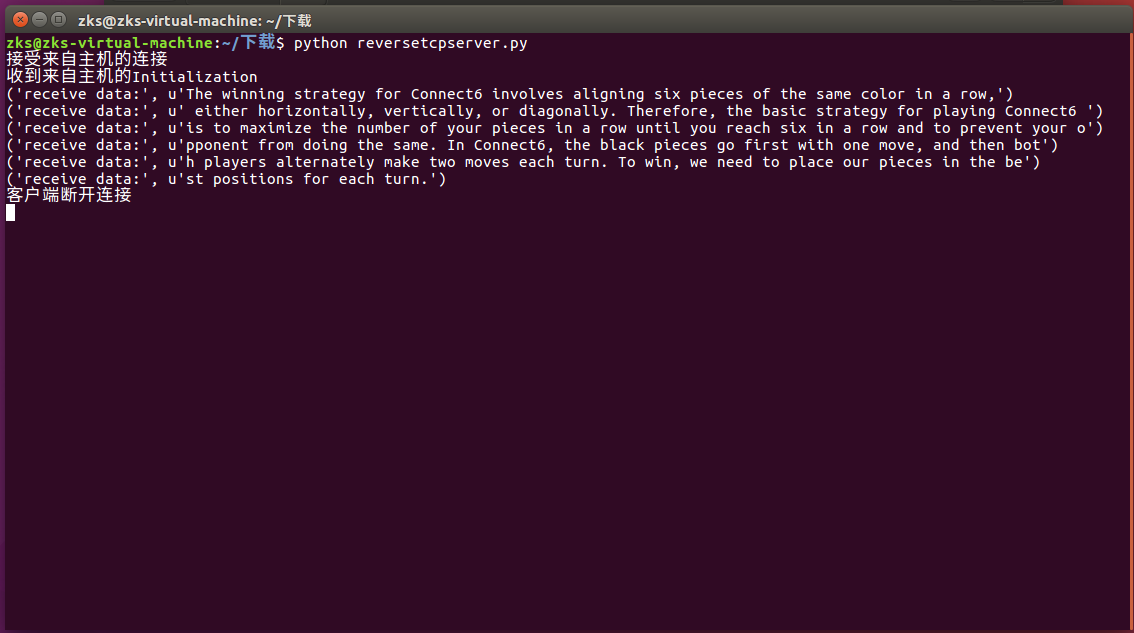
Type4:



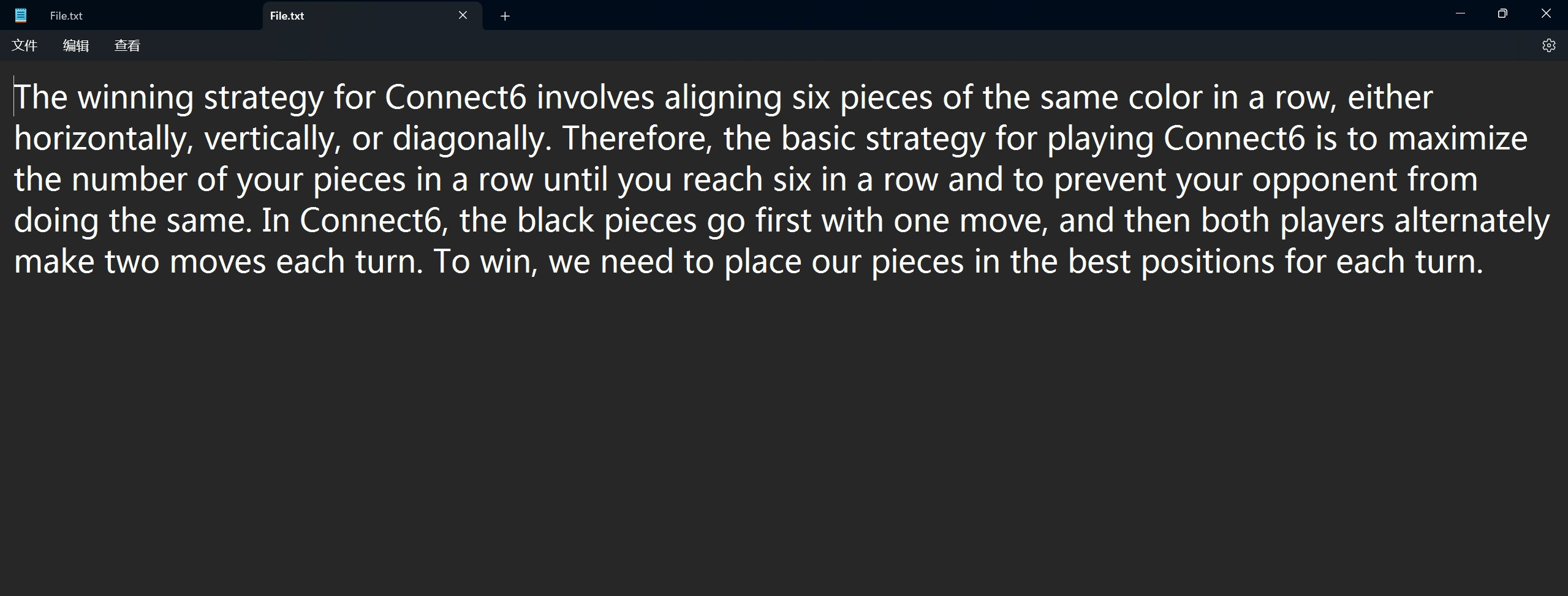
Client端:



Server端：

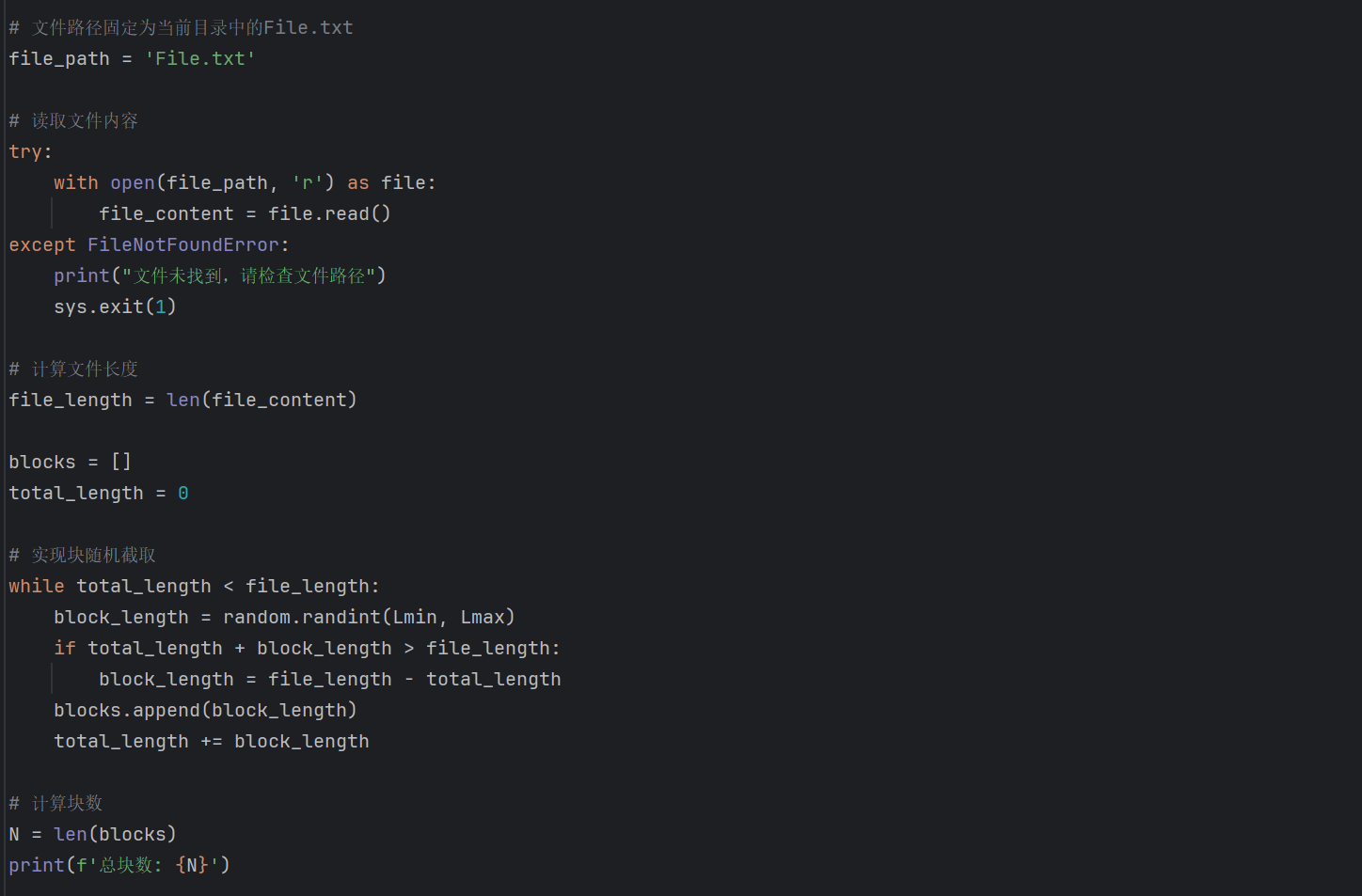


File英文文件：

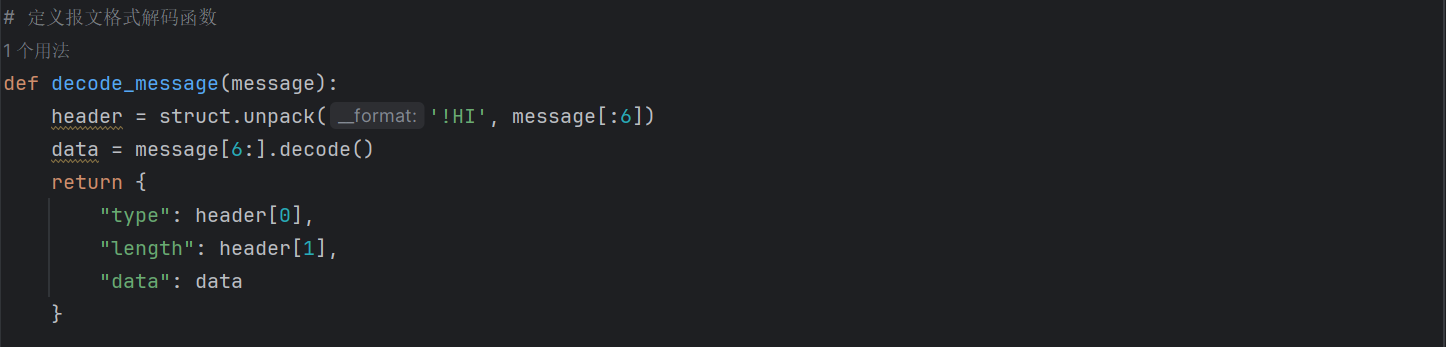


**b.**

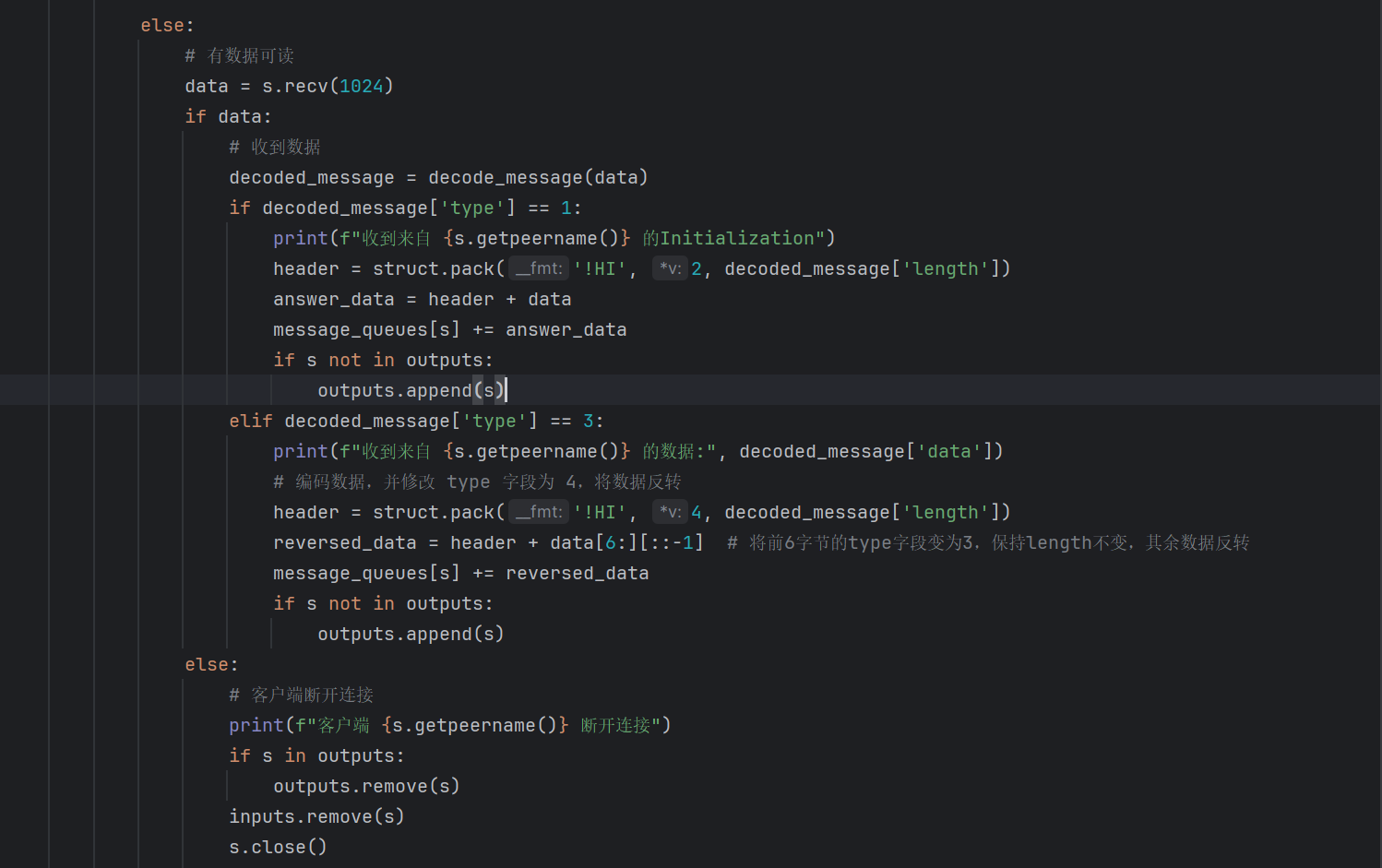
难点在于如何在发送数据报之前实现随机截取文件并计算块数。使用文件相关操作完成文件内容的读取，使用random库中randint(Lmin,Lmax)函数实现随机选择一个Lmin，Lmax范围内的数，而Lmin和Lmax已经实现在命令行中指定。利用列表blocks记录随机得到的每一块长度供后续传输数据时使用，同时blocks的长度也代表了块数。注意若剩余文件数据不够截取则将其剩余内容作为最后一块数据。



另一个难点是如何实现数据type字段更改，length字段不变，其余数据反转。与Task2 UDP套接字编程同样使用struct库编码解码数据报，前两个Bytes为type字段，之后四个Bytes为length字段，剩余内容为实际数据。



服务器端收到type为1的Initialization的数据报便返回没有数据内容，type为2的agreement报文。若收到type为3的reverseRequest报文则修改type为2，length不变，利用data[6:][::-1]实现数据报前六个Bytes之后内容的反转作为新的实际数据发送。



最后的难点在于如何实现非阻塞式TCP套接字编程，使服务端可以同时响应多个客户端。利用select库监视多个套接字的可读性、可写性和异常条件的事件实现多个客户端的响应控制。

select.select()函数接收并维护三个套接字列表：

readable：要检查是否有数据到达的套接字（对于监听套接字来说，是新连接是否到来）。

writable：准备好发送数据的套接字。

exceptional：要检查是否有错误的套接字。

程序中需要创建并使用以下列表：

inputs：此列表包含服务器要读取数据的所有套接字。最初，它只包含服务器的监听套接字 serverSocket。当接受一个新的客户端连接时，客户端套接字会被添加到此列表中。

outputs：此列表包含有数据排队等待发送的套接字。如果有套接字需要向客户端发送数据，它会被添加到此列表中。一旦数据发送完成，该套接字会从列表中移除。

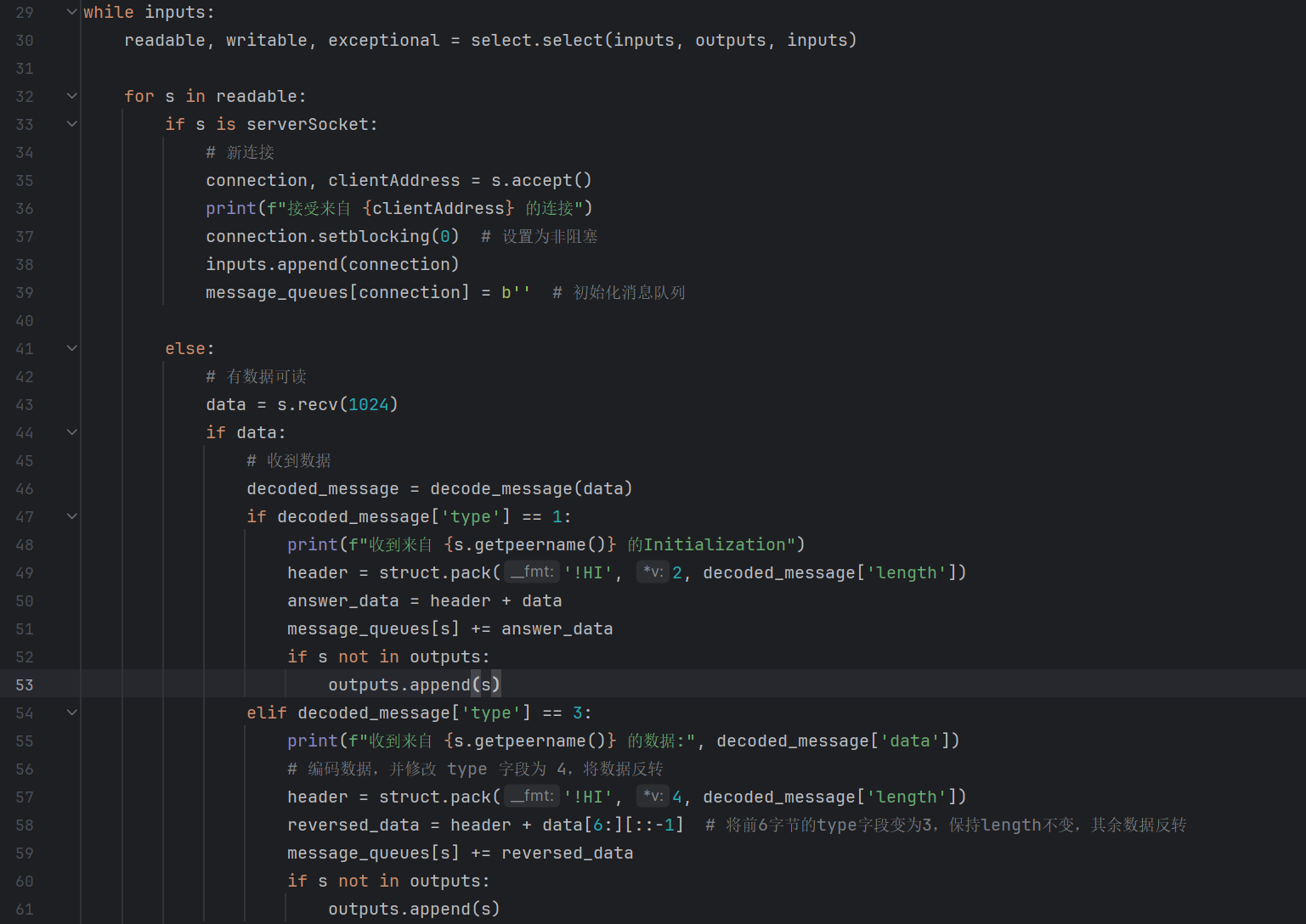
exceptional：此列表包含应监视是否有异常条件或错误的套接字。

还需要创建并维护消息队列：

message\_queues：此字典将每个客户端套接字映射到其对应的消息队列。消息队列是一个累积待发送给客户端的数据的字节字符串。当数据准备好发送时，会存储在此队列中，并将套接字添加到 outputs 列表中。一旦数据发送完毕，队列会被清空。

具体实现流程如下：一个新客户端连接，serverSocket 是可读的，则服务器接受连接，并将客户端套接字添加到 inputs。若客户端发送数据并且客户端套接字是可读的，服务器读取数据，处理它，并在需要响应时将响应排队到 message\_queues，并将客户端套接字添加到 outputs。服务器发送数据时，若客户端套接字是可写的则服务器发送排队的数据，并清空消息队列。由此实现了单线程模式下高效地管理多个客户端连接。





**c.**

主要使用到了TCP套接字编程，文件操作，select库相关知识。利用套接字编程相关知识建立连接，实现数据传输，对报文的编码解码以及对字段的具体控制。利用文件操作相关知识实现文件读取，截取分块以及内容存储与反转。利用select库相关知识实现单线程模式下管理多个客户端连接。经过完成该任务我对TCP数据传输过程以及非阻塞式连接有了更深刻的理解。

**d.**

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