Task2 UDP socket programming

1. 目的

(1) 掌握UDP socket程序结构。

(2) 掌握自定义应用层报文格式及报文交互过程，即自定义应用层协议。

1. 内容和分析

(1) client和server都是基于UDP的命令行程序。client运行在host os，server运行在guest os。client端启动时在命令行方式下指定<serverIP> <serverPort> <packet\_nums> <packet\_ver>，server指定<serverPort> <drop\_rate>。

(2) 模拟TCP连接建立过程，即建立完毕连接以后，才进入到数据传输阶段。模拟TCP连接释放过程，关闭client-server之间的交互。

为了完成这一过程，引入新的报文格式：b'\x00\x01' + 'SYN'；b'\x00\x02' + 'SYN-ACK'；b'\x00\x03' + 'FIN'；b'\x00\x04' + 'ACK'。前两个字节表示报文类型，后面用对应TCP中的报文名称填充。TCP连接建立即三次握手，TCP连接释放即四次挥手，使用套接字的send和recv即可实现。recv阻塞，需要超时做了异常处理。

(3) client发送packet\_nums个request数据包，由命令行指定packet\_nums。对每个reqeust数据包，server端采用随机不响应client请求的方式，来模拟丢包场景。丢包率自定，由命令行指定drop\_rate，需要注意的是这里的丢包率不等同于client 最后“汇总”的丢包率，而只是作为随机数判断时的依据。

UDP虽然不可靠，但指望发生UDP丢包事件也是概率极低的，所以“丢包”这种事，得自己在应用层程序udpserver中来构建。

(4) 给client设定一个超时时间timeout，从命令行获取。在创建并配置UDP socket时，使用 timeout 参数设置超时时间。在超时时间内：

a.若能接收到来自于server的响应报文，则计算本次交互的RTT(单位：ms)以及sever的系统时间(hh-mm-ss)。

b.若没有收到来自于server的回复，则认为发了丢包。若发生丢包，则重传；两次重传失败，则放弃本次重传。

(5) client端和server端的交互及本应用层的报文格式应该包括Seq no、ver字段，“其他内容”。

a. sequence\_number指的是发送的第几个报文，从1开始，按序递增。ver指的是版本号，由命令行确定，用于帮助区分不同客户端的数据包。

b. server->client报文中Seq no.与刚刚从client接收到的报文相对应。

c. “其他内容”：server-client中要存放系统时间（str类型）的长度（2 bytes），系统时间，其余字节用‘z’填充到198 bytes。长度用来帮助客户端读取时间str。

client->server剩余200 bytes用‘zwb#’\* 50作为填充。

(6) client端每发送一个报文，若收到了来自于server端的response,则print出“Seq、Ver、serverIP：Port、RTT”。若发生超时，即在timeout内没有收到来自于server的response,则print出“请求超时：Seq、尝试次数”。

server端每收到客户端的消息，都会打印出“client\_address、Seq、Ver（区分客户端的标志）、系统时间（%H:%M:%S）”。

(7) 当发送完packet\_nums个request报文后，client端打印出如下“汇总”信息：

接收到的udp packets数目；

丢包率(百分比：1-接收到的udppacket/发送的udppacket)；

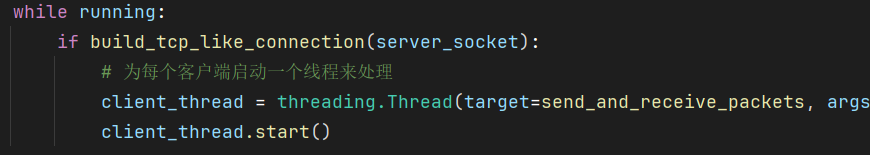
最大RTT、最小RTT、平均RTT、RTT的标准差，由numpy库快速计算；

server的整体响应时间(server最后一次response的系统时间与第一次response的系统时间之差)。

(8) 给server增加了一个exit线程，用于随时从命令行终止服务器。

(9) 即使在UDP中模拟了TCP的连接建立和释放过程，UDP协议本身还是无连接的。这意味着UDP服务器不需要像TCP服务器那样监听和接受连接请求。UDP服务器可以直接接收来自多个客户端的数据包。所有的客户端通信都通过同一个UDP套接字进行。但是，为了能够在处理多个客户端的并发请求时避免阻塞，最好使用多线程来处理每个客户端的请求。本Task 就是这么做的。

一开始想用如下所示方式：



但是很明显，当第二个客户端连接到来时，会导致连接冲突，因为 build\_tcp\_like\_connection 函数会阻塞，等待第一个客户端完成连接建立过程，如下图所示：



接下来尝试循环对每条来自客户端的消息，单独开一个线程来处理，如下所示：



服务器的套接字只有一个，一旦某个消息在处理过程中需要继续等待新消息，比如：一个客户端需要等待ACK消息，另一个客户端的数据包处理就会冲突。这也就意味着此时模拟TCP的建立和释放已不能完成，当然可以强行操作，也就是把SYN、SYN-ACK等消息当做一般的消息处理，不会存在客户端发出SYN后，必须等到SYN-ACK才开始进行数据交互。但这样的模拟已失去了意义。

故而尝试将具有“符合TCP机制”的模拟TCP连接和释放的版本命名为v1，对应的是udpclient\_v1、udpserver\_v1。而用于测试UDP的多端连接的版本命名为v2，对应的是udpclient\_v2、udpserver\_v2。

(10) 使用更常见的蛇形命名法（源自早期的开发环境）。使用pycharm的ctrl+alt+L格式化代码。使用try-except-finally捕获处理异常。

(11) encode将字符串转换为字节数据（bytes）；decode将接收到的字节数据（bytes）转换为字符串，这样可以方便程序对数据进行处理和显示。一般是对字符串使用，int 类型有专门的方法转换为 byte。

1. 测试和抓包

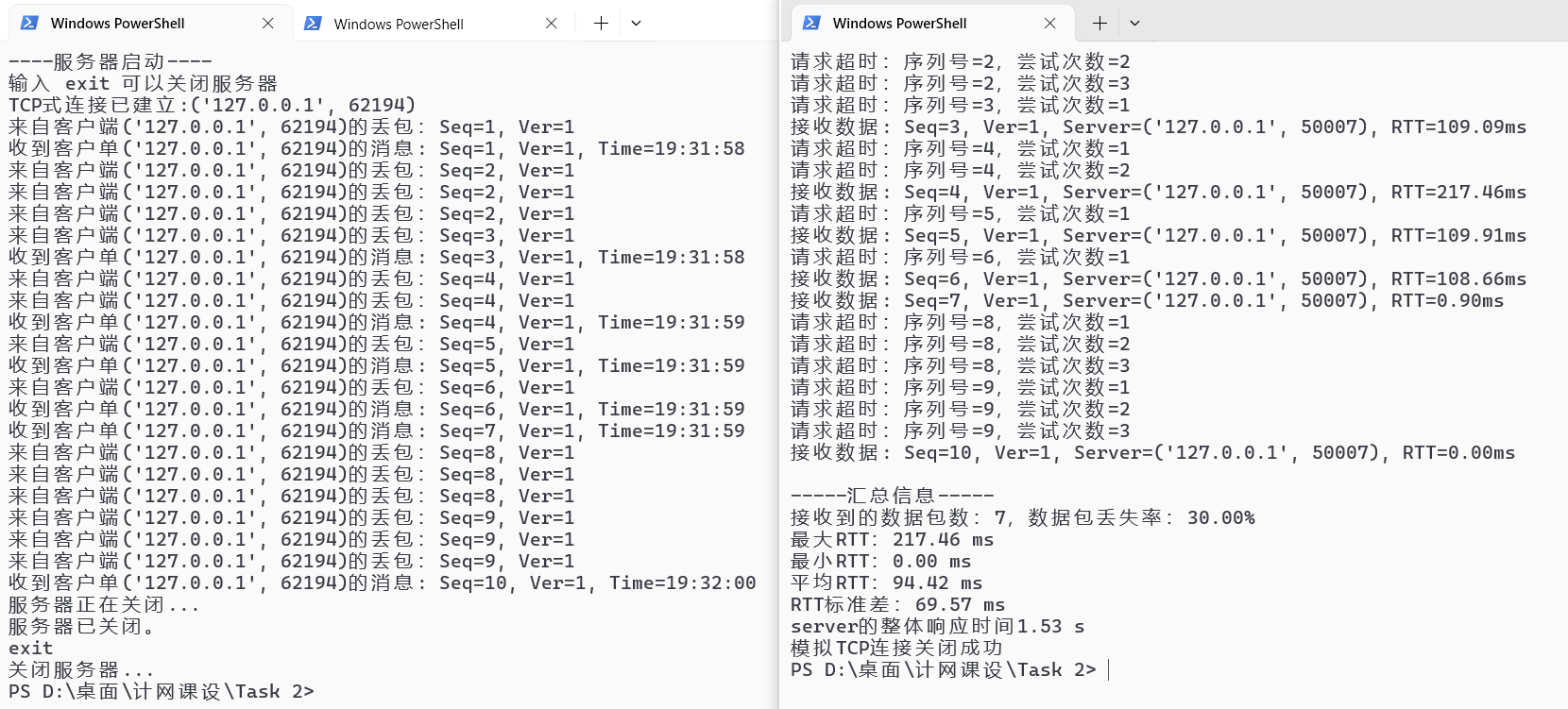
(1) 为了方便在本地计算机上的测试，即服务器和客户端在同一台计算机上运行。客户端连接到服务器的ip地址为127.0.0.1，即本地回环地址。如果在不同设备之间测试服务器和客户端代码，那么必须设置为服务器所在设备的实际 IP 地址。同一台机器上可以通过虚拟机来实现。

(2) 需要保证客户端和服务器使用的端口号一致，一般选择动态端口号。需要避免与已知的应用程序和服务器所使用的端口冲突。

(3) v1 版本下的测试与抓包。

终端运行结果如下所示，服务器输入python udpserver\_v1.py 50007 0.6，表示服务器监听端口50007，且每次处理数据有0.6的概率发生丢包；客户端输入 python udpclient\_v1.py 127.0.0.1 50007 10 1 0.1，表示与ip=127.0.0.1，port=50007的UDP服务器建立连接，发送10个数据包，报文版本为1，超时间隔为0.1秒。

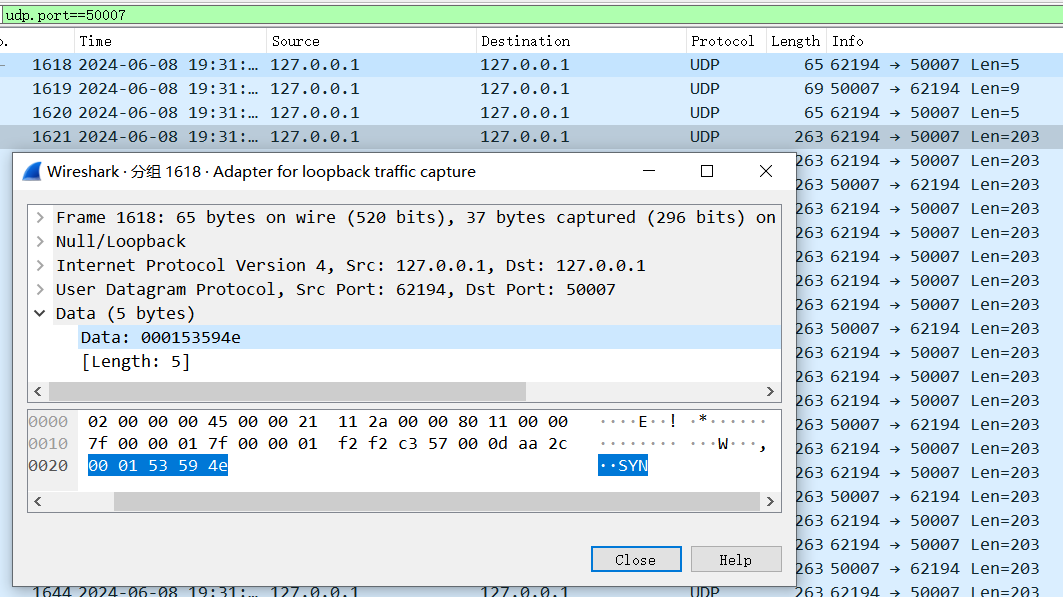
如下图所示，客户端与服务器完成了模拟TCP的连接和建立，服务器丢失了三个数据包。

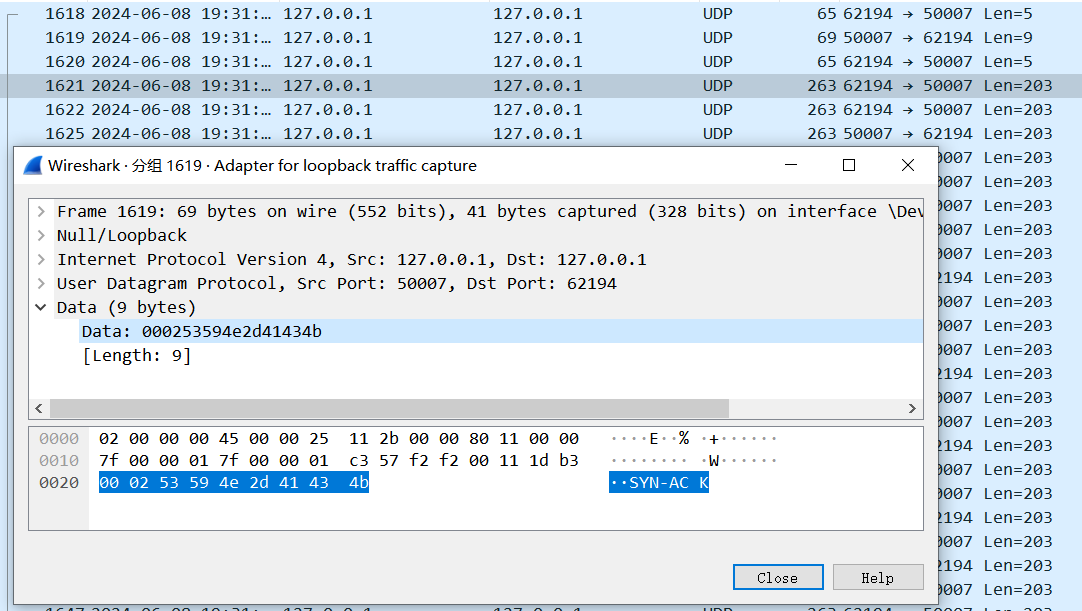


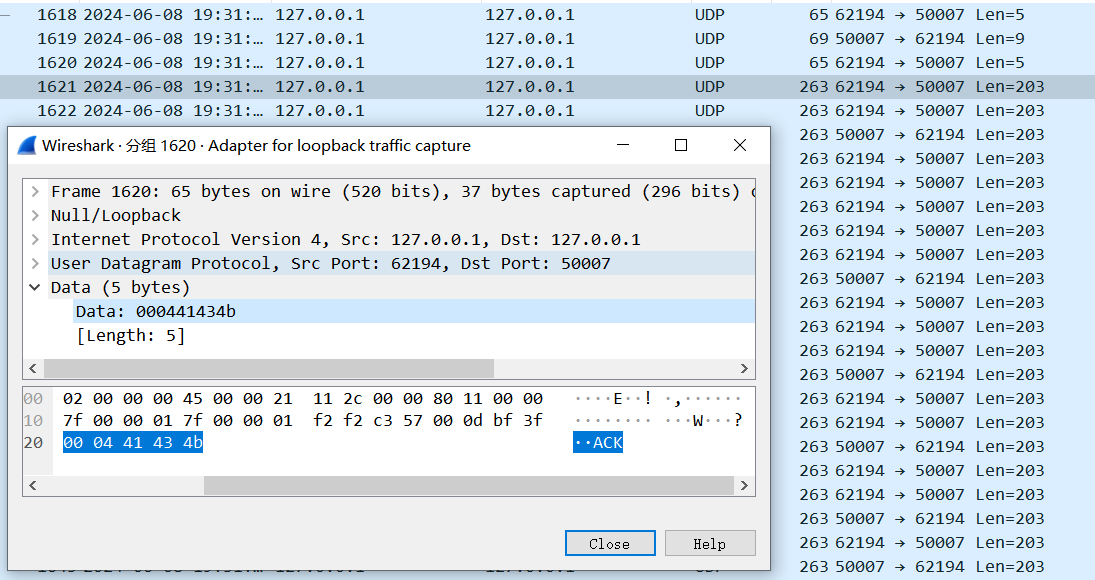
抓包如下：

观察b'\x00\x01' + 'SYN'、b'\x00\x02' + 'SYN-ACK'、b'\x00\x03' + 'FIN'、b'\x00\x04' + 'ACK'、Seq + Ver + ‘内容’ 报文（request、reply）

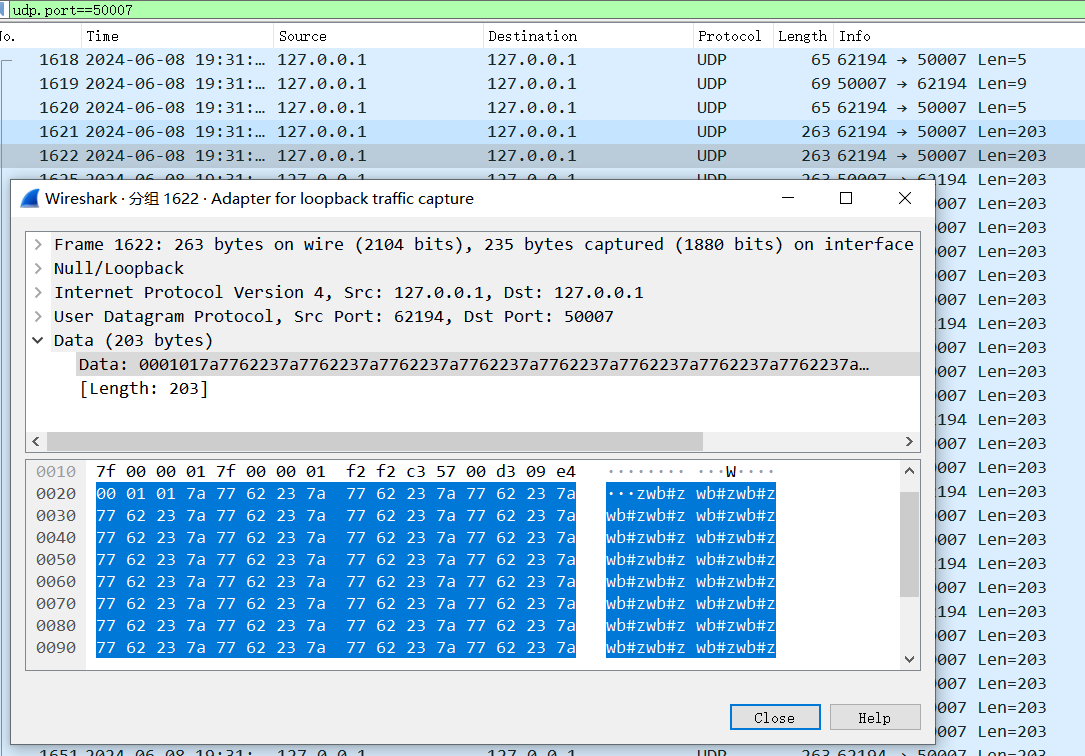
a. 如下图所示，第1个数据包是客户端向服务器发送的SYN；第2个数据包是服务器向客户端发送的SYN-ACK；第3个数据包是客户端向服务器发送的ACK；前两个字节表示类型。自此，模拟三次握手完成。



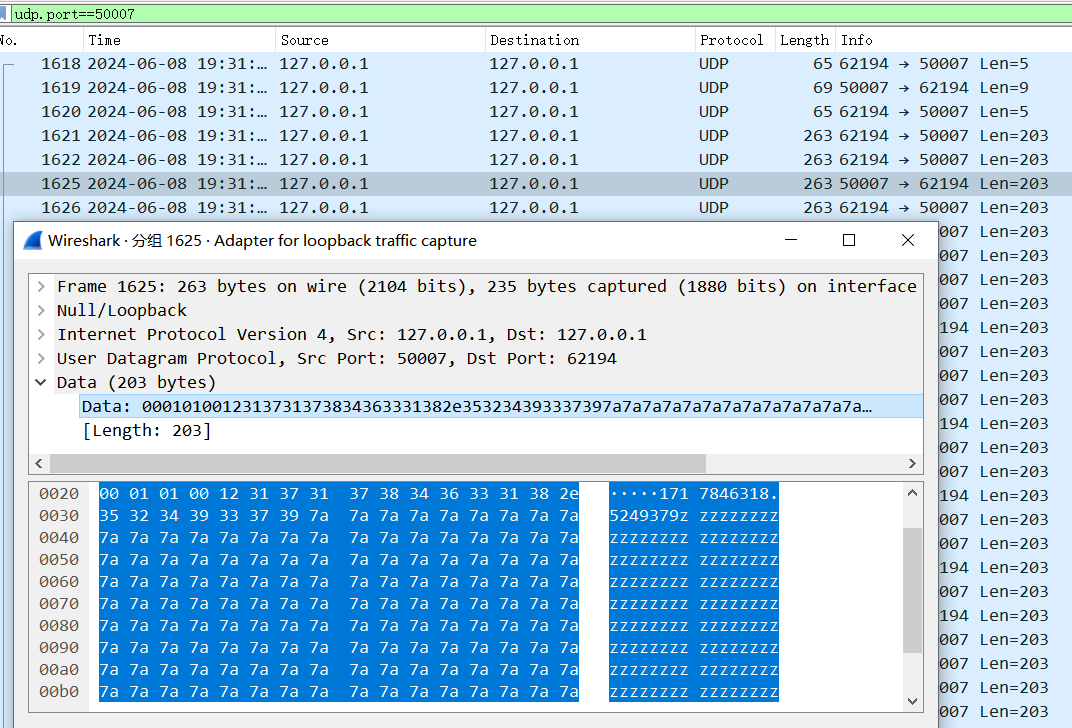




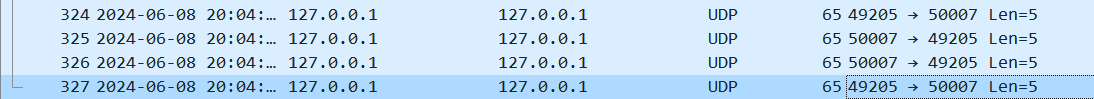
b. 如下图所示，第4个数据包是客户端向服务器发送的request报文，前两个字节0001表示序列号1，后一个字节01表示版本1，后面的字节用“zwb#”填充。Data一共203 bytes。

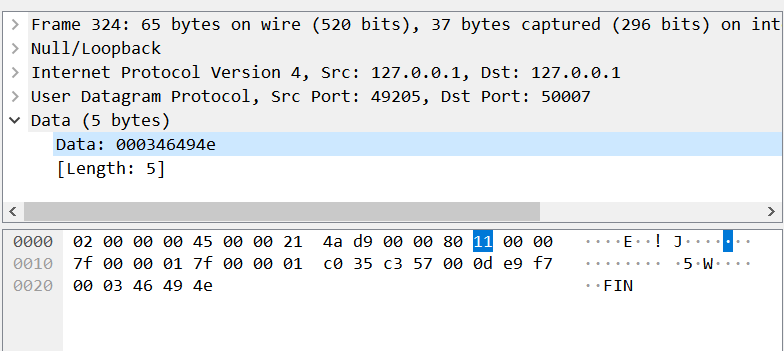


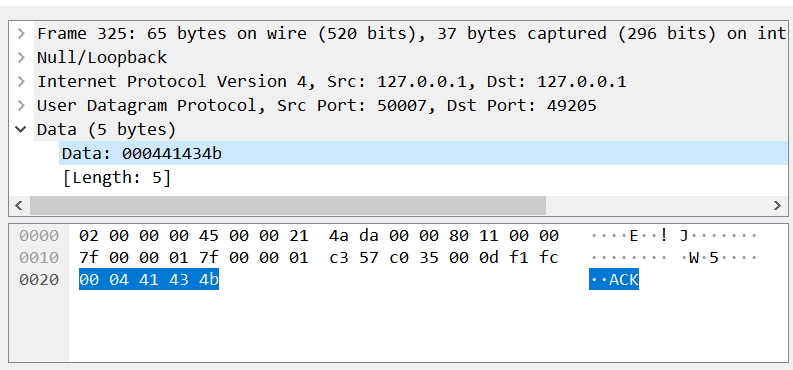
c. 如下图所示，第5个数据包是服务器向客户端发送的reply，前两个字节0001表示序列号1（与request对应），后一个字节01表示版本1，后两个字节表示“时间”字符串的长度，后面的字节用“z”填充。Data一共203 bytes。

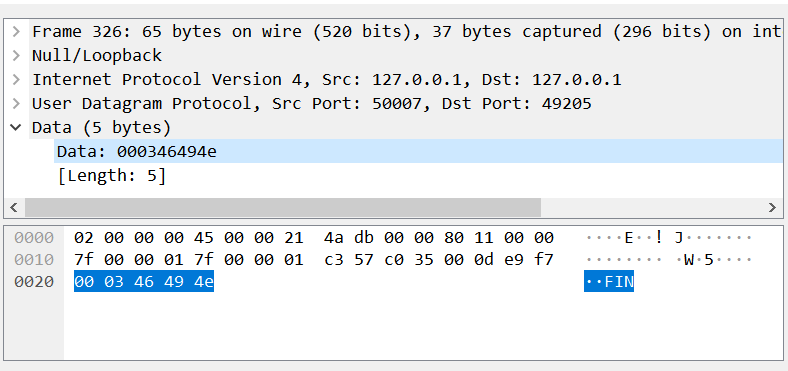


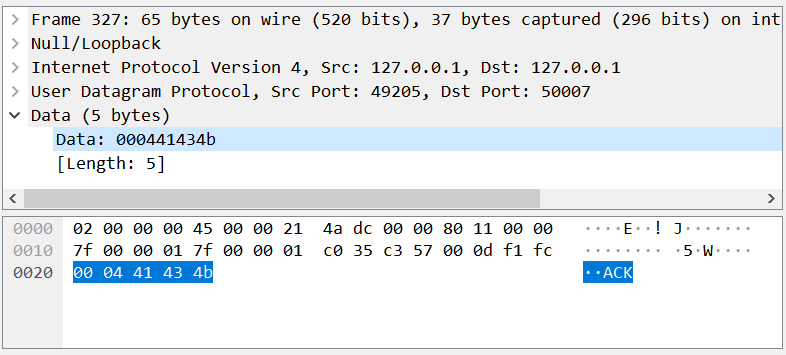
d. 如下图所示，最后4个数据包是客户端与服务器之间的四次挥手，前两个字节表示类型。wireshark意外断开，下面为重新抓包结果：



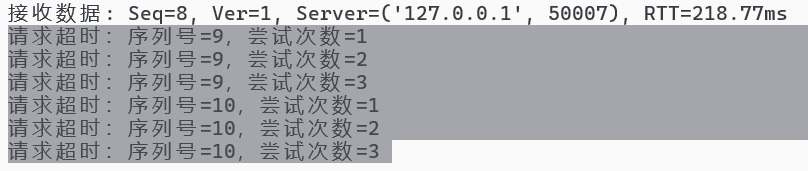




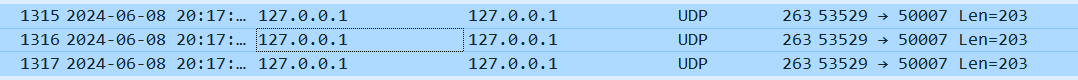


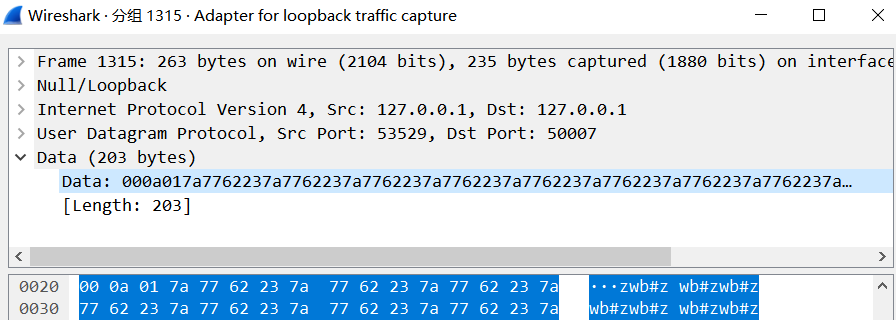


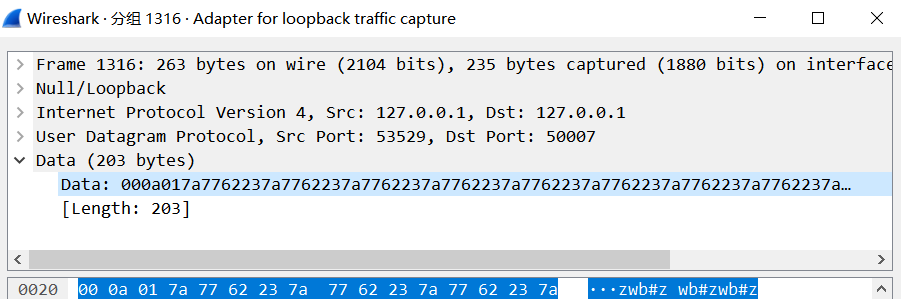
e. 如下图所示，序列号为10的数据包重传了两次，一共发了三次。

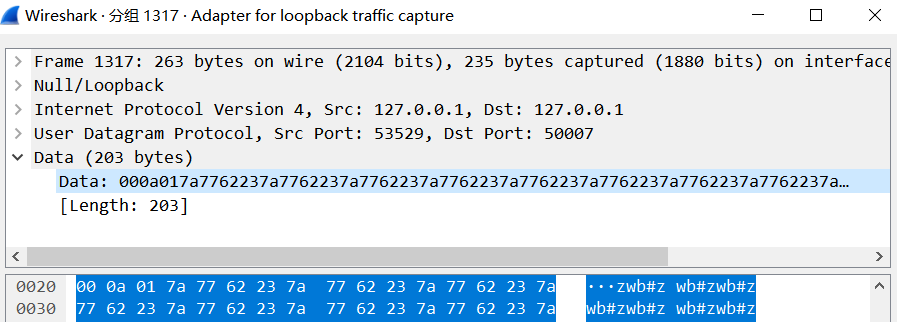


查看发现，1315、1316、1317为序列号为10的三次发送(Data前两个字节均为000a )，后两次为重传。









(4) v2版本下测试UDP的多端连接。第一个客户端的数据包版本为1，第二个为2。如下图所示，可以看到UDP服务器同时处理了两个客户端的连接。

