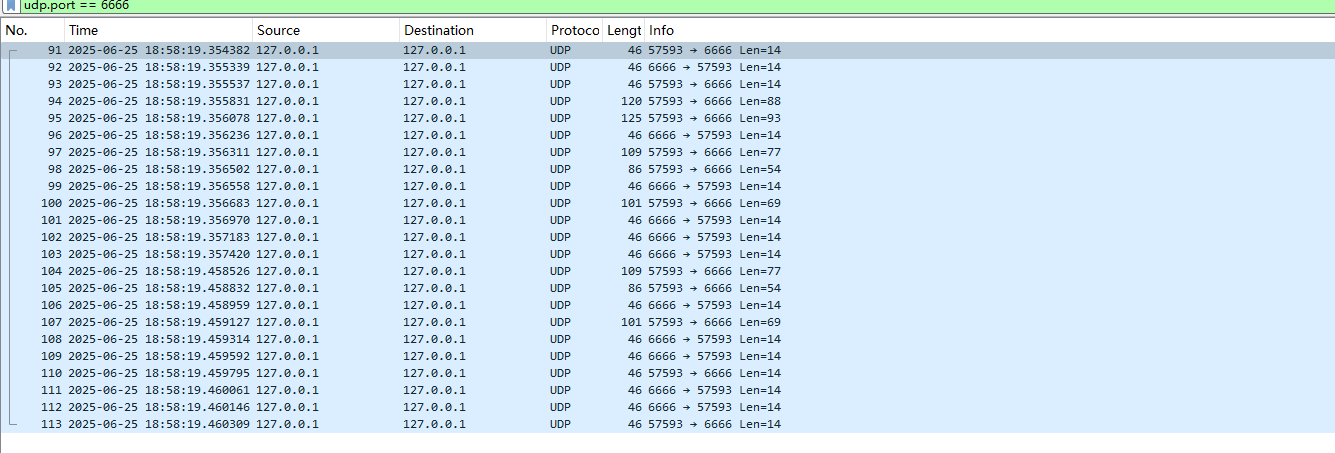
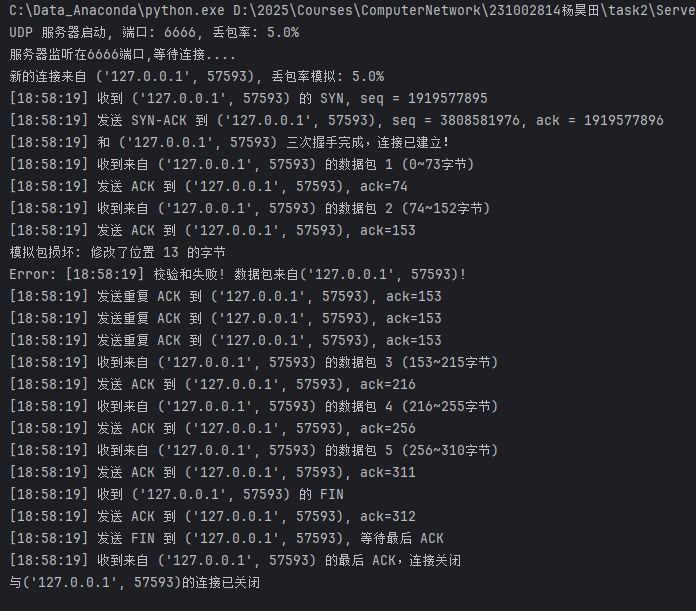
udp\_packet\_capture

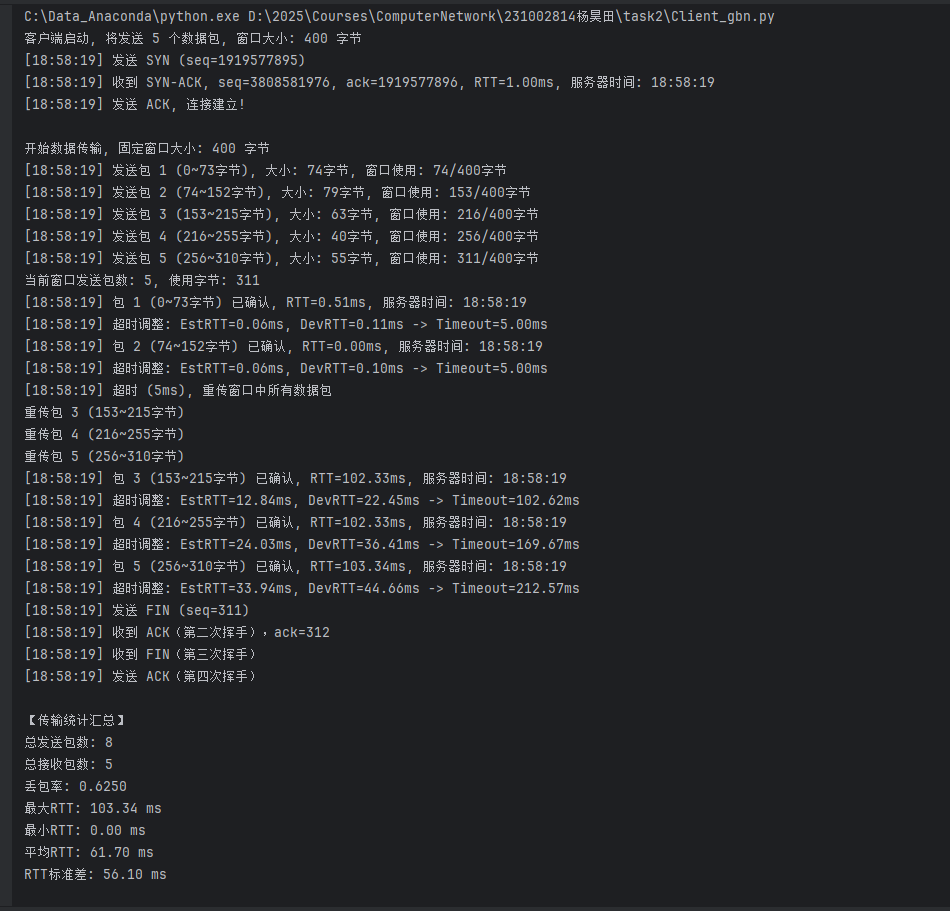
1. Wireshark捕获的部分报文 这里发5个包便于查看细节



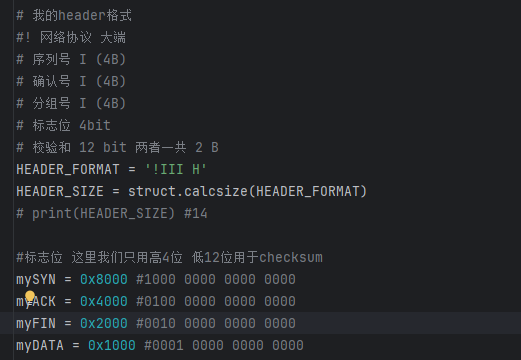
这里展示服务器端的结果



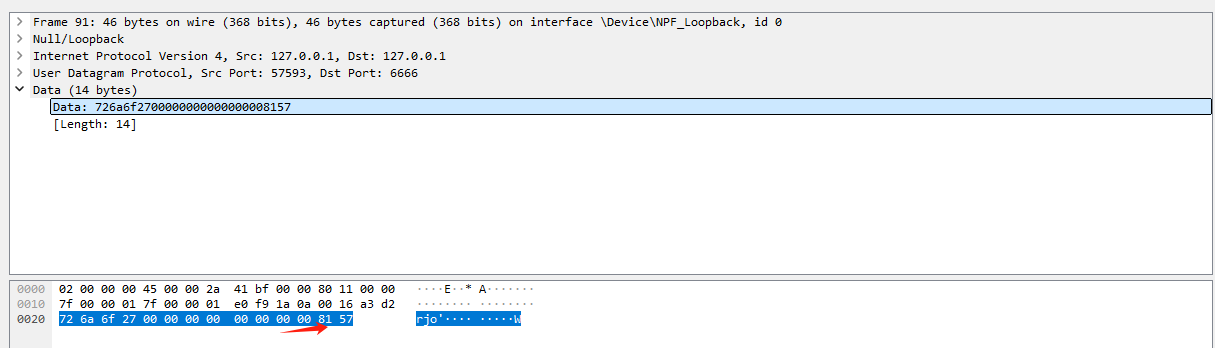
这里展示客户端的结果



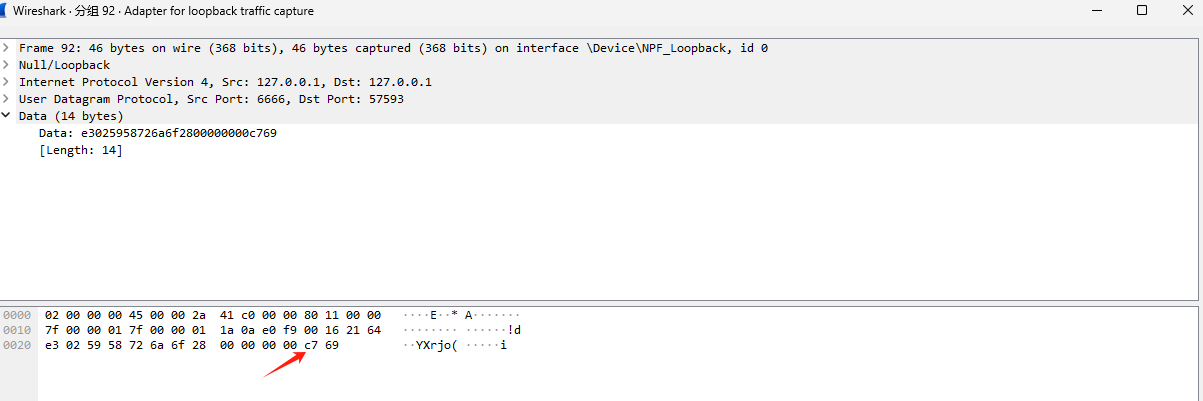
其中Server只向Client发送控制报文,没有具体数据 所以Len = 14



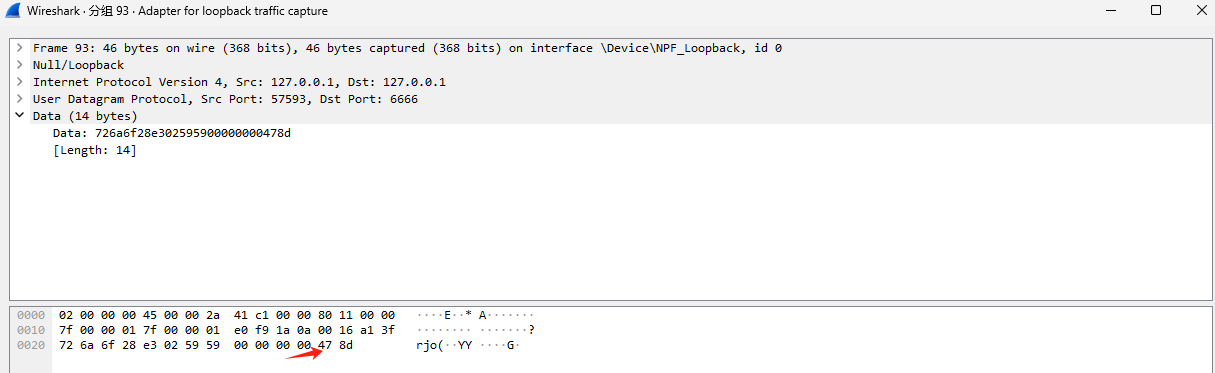
客户端先向服务器发送SYN



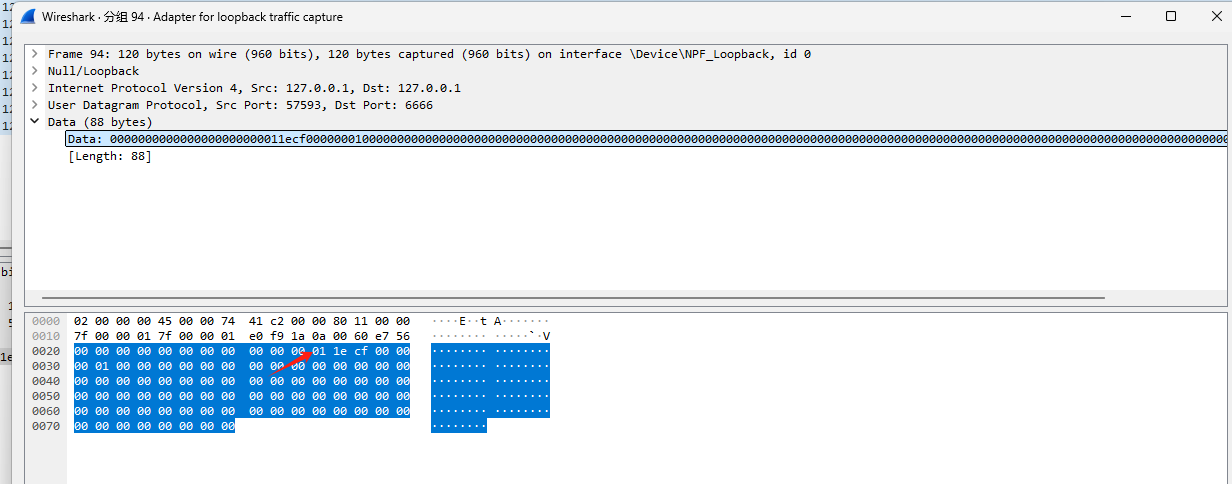
这一位显示是8 及 1000 表示是SYN 第一次握手



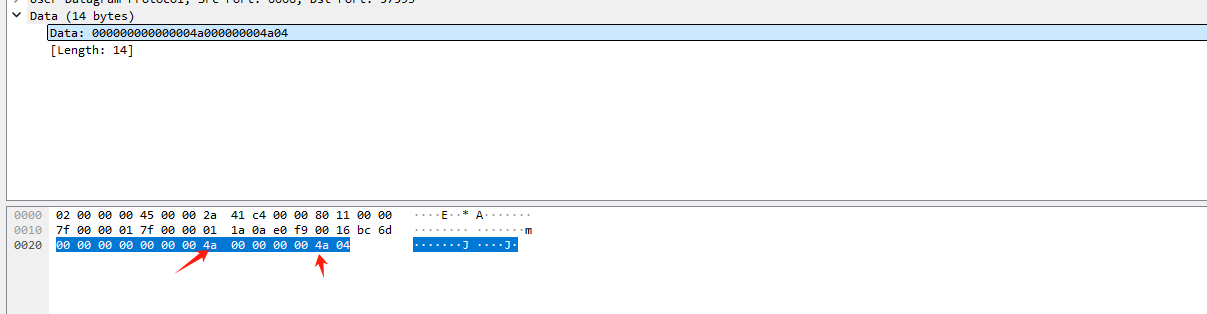
这里服务器向客户端回复C 也就是 1100 也就是SYN和ACK 第二次握手



这里是客户端回复服务器4 也就是 0100 也就是ACK,这是第三次握手 连接成功建立



这是客户端传输的数据报文, 这里数据用packet\_num(4B)和全0补充 代表数据,图中箭头指向00 00 00 01 是header的packet\_num ,是第一个所以packet\_num = 1



这个是服务器回复ack74



回复的这个 数据长度是74 即0~73的都收到了  
模拟数据损坏 这里数据包3损坏了 checksum检测出来 会直接回复重复ack



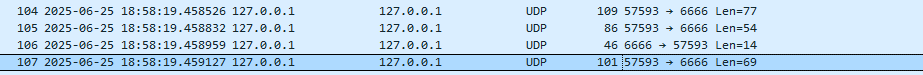
这里回复的是16进制的99 也就是ack153



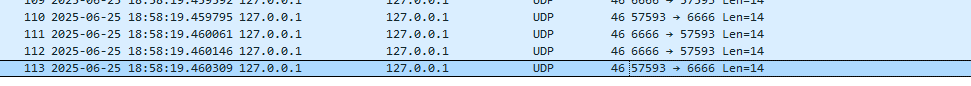
前两个数据包 是 88-14 + 93 -14 = 153 所以我ack153 表示0~152 都收到了



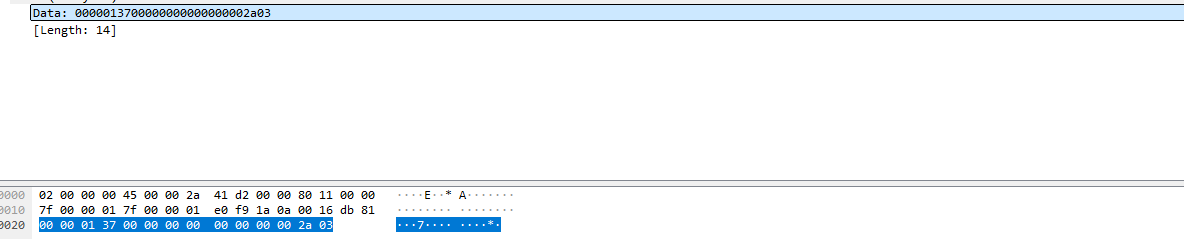
这里是3个重复ack 因为第三个数据包损坏后 4,5就乱序了就发重复ack 153 GBN协议



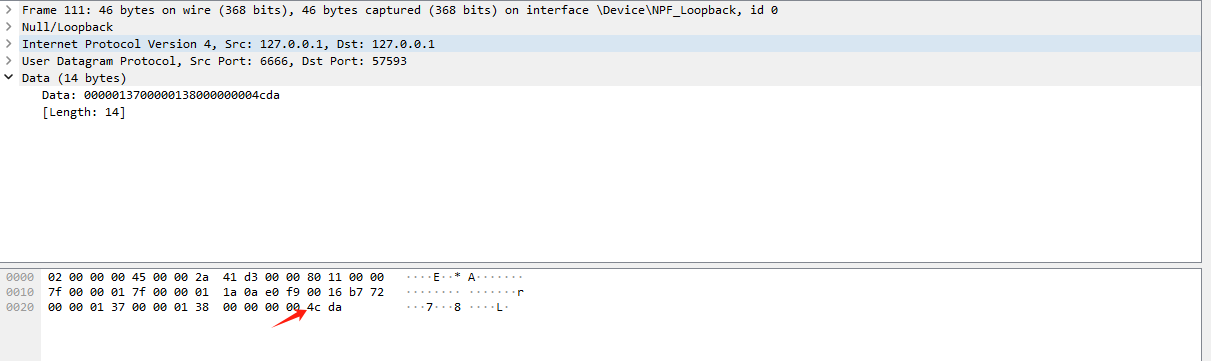
这里触发客户端的重传了



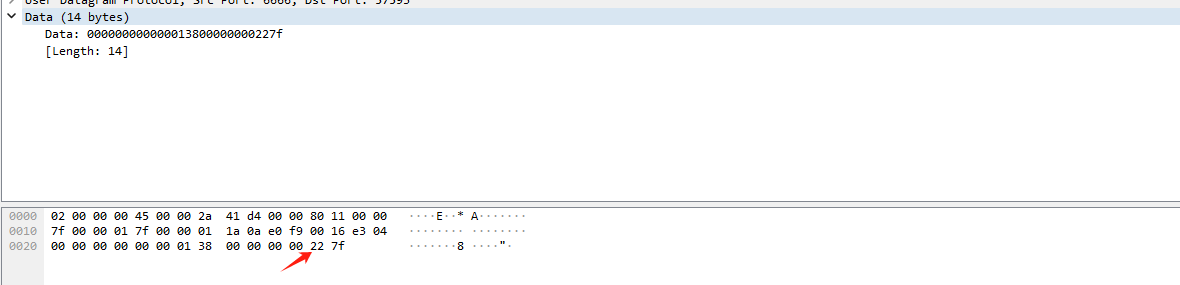
这里是4次挥手



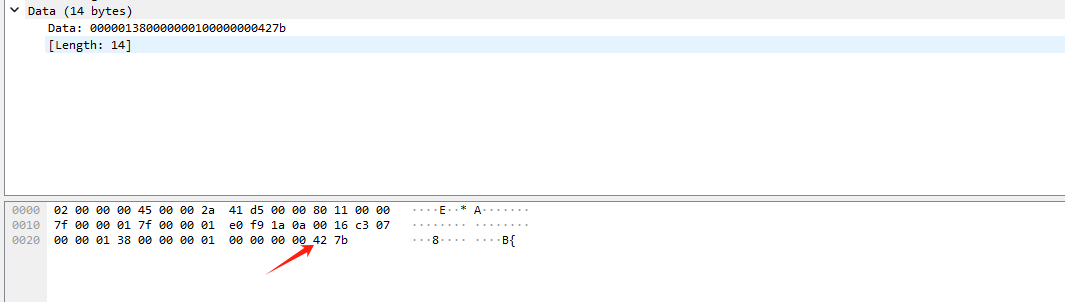
客户端向服务器发送FIN 第一次挥手



服务器发送ACK 第二次挥手



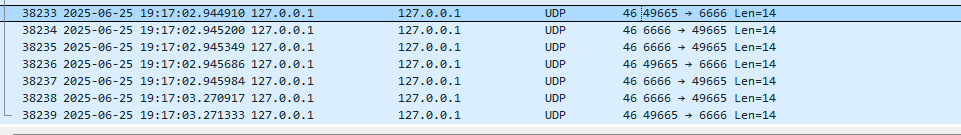
服务器向客户端发送FIN 第三次挥手



客户端回复最后一个ACK 第四次挥手

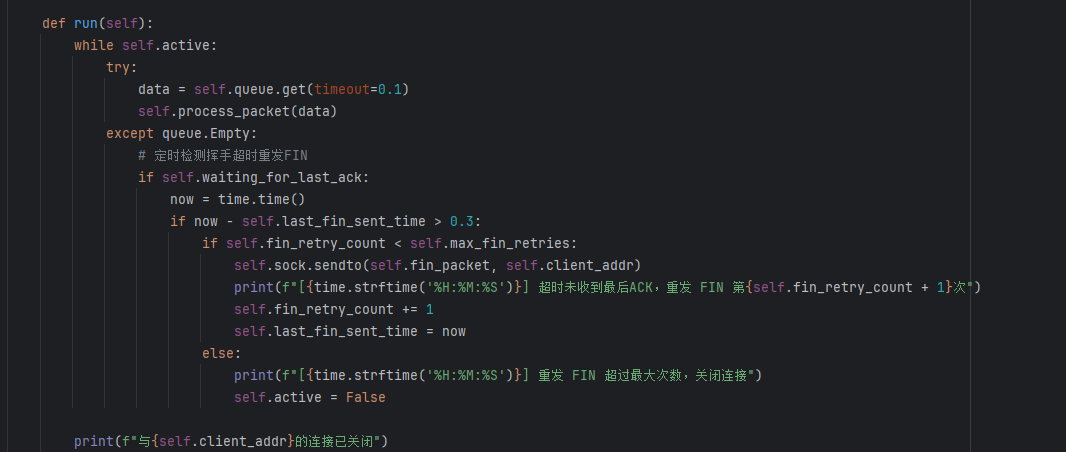
上面展示了三次握手四次挥手 数据损坏重传

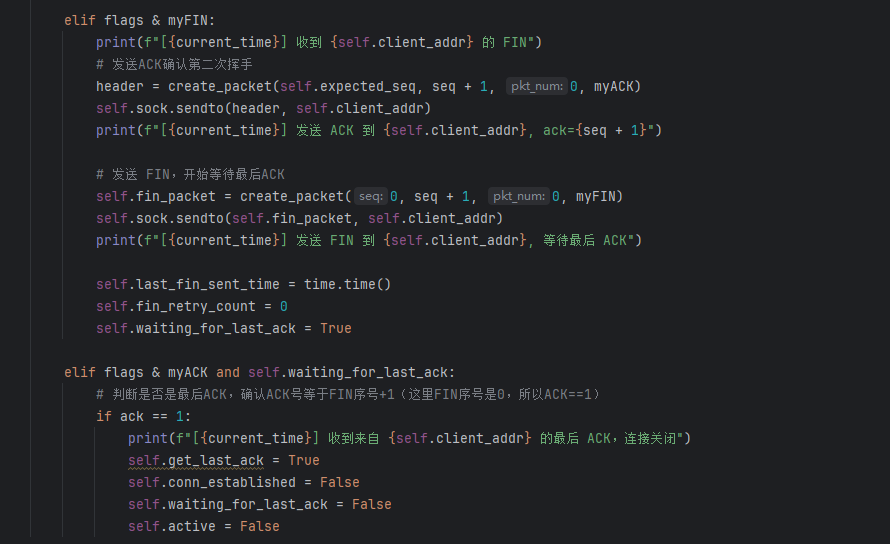
下面是特殊情况 在模拟数据包损坏的情况时 发现如果我的四次挥手的最后一个ACK 丢包或者损坏了怎么办 我的客户端发送了ACK就断开连接了,查阅资料发现 客户端发送ACK后会等待2MSL 这里我使用30s,服务器没收到ACK,就重新发送FIN当客户端再次收到FIN后,再次发送ACK,让服务器可以断开连接 为了模拟这个情况需要把数据包损坏率设置的较大如0.5.

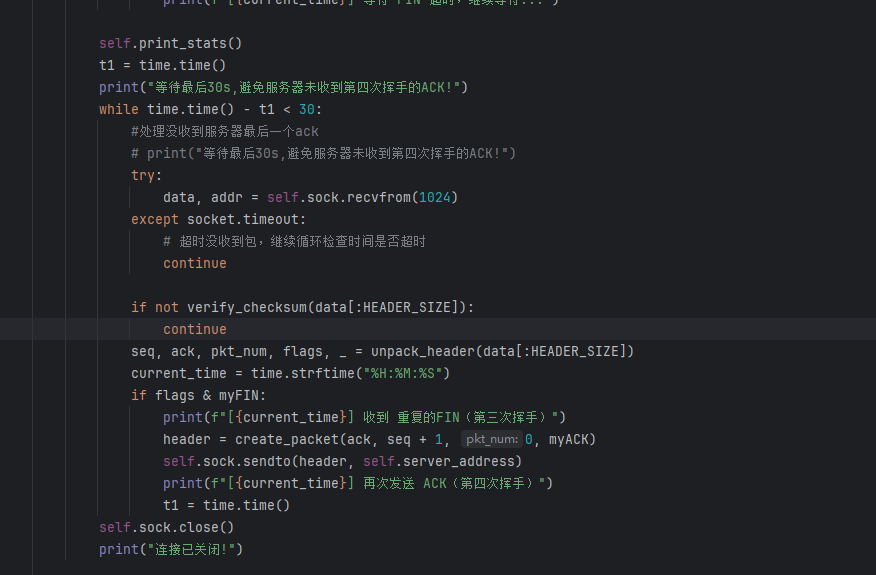


这里是最后的挥手 客户端发送FIN 服务器回复ACK 服务器发送FIN 客户端回复ACK ACK数据包损坏 服务器发送重复ACK(我的实现是如果数据包损坏就发送重复ACK,但是如果是丢包不会发送) 服务器未收到ACK 服务器再发送FIN 客户端再发送ACK 然后收到之后断开连接

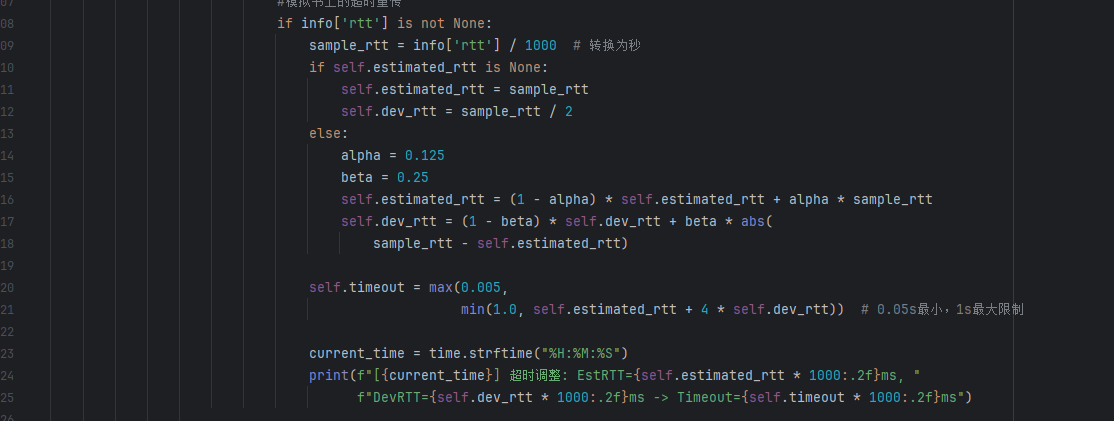
1. 实现上的关键点和代码解决:
2. header的设计 由于需要打印是第几个数据包所以也把packet\_num封装到了header里
3. 三次握手和四次挥手的实现,这里展示4次挥手上述特殊情况的解决办法



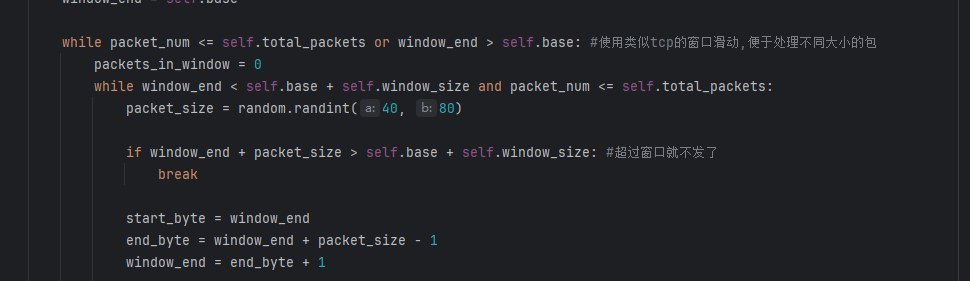




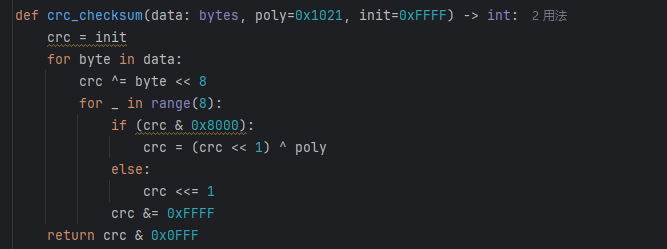
1. 超时重传的时间设置,参考了课本上的P157(第七版)的方法.



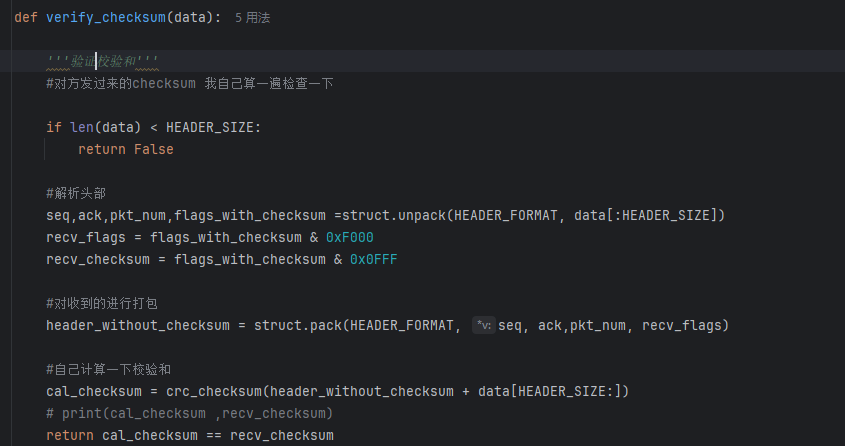
1. 滑动窗口的实现, 由于数据包的大小是随机的这里我采用了字节的滑动窗口,下图是滑动窗口的部分实现



1. 校验码的实现与使用

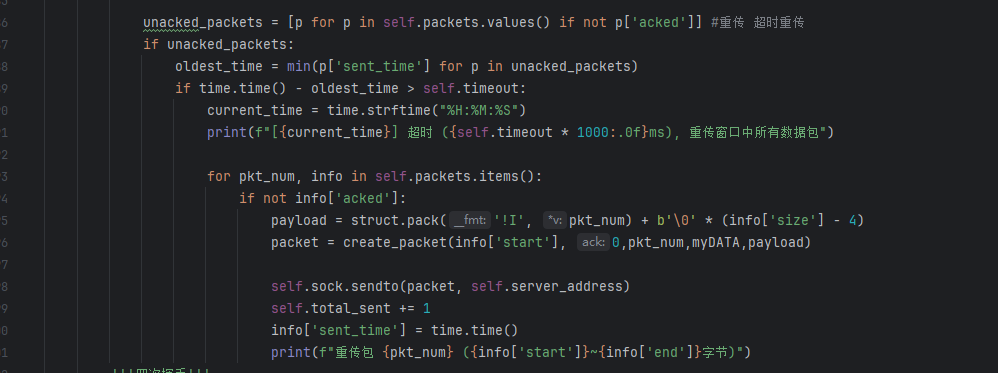


我这里采用CRC冗余校验码



这里是验证校验和

6. 超时重传的实现



使用列表记录那些包还没被ack 然后找出最早的未被ack的然后重传窗口的整个数据包.

1. 掌握的知识
2. 终于比较透彻的理解了三次握手和四次挥手,四次挥手也有简单的实现,ACK和FIN一起发,还有四次挥手的特殊情况的处理办法
3. GBN协议的整个工作原理,滑动窗口的实现
4. 超时重传的实现,
5. 网络编程中的数据发送与接收
6. 有点理解计网的规则是人们的规定,可以自己定义的header,只要对应的解封装就可以明白接收到的数据是什么意思.
7. Checksum的作用,自己实现的checksum竟然真的能对于我模拟损坏的数据包校验出来!
8. 这一整个可靠数据传输流程的实现,在Wrieshark上能够成功抓包并且我可以分析出来整个流程.虽然比task1难不少我感觉,不过确实很不错,学会了好多知识,并且这种抓包数据和程序能够一一对应的感觉很不错!

四.git