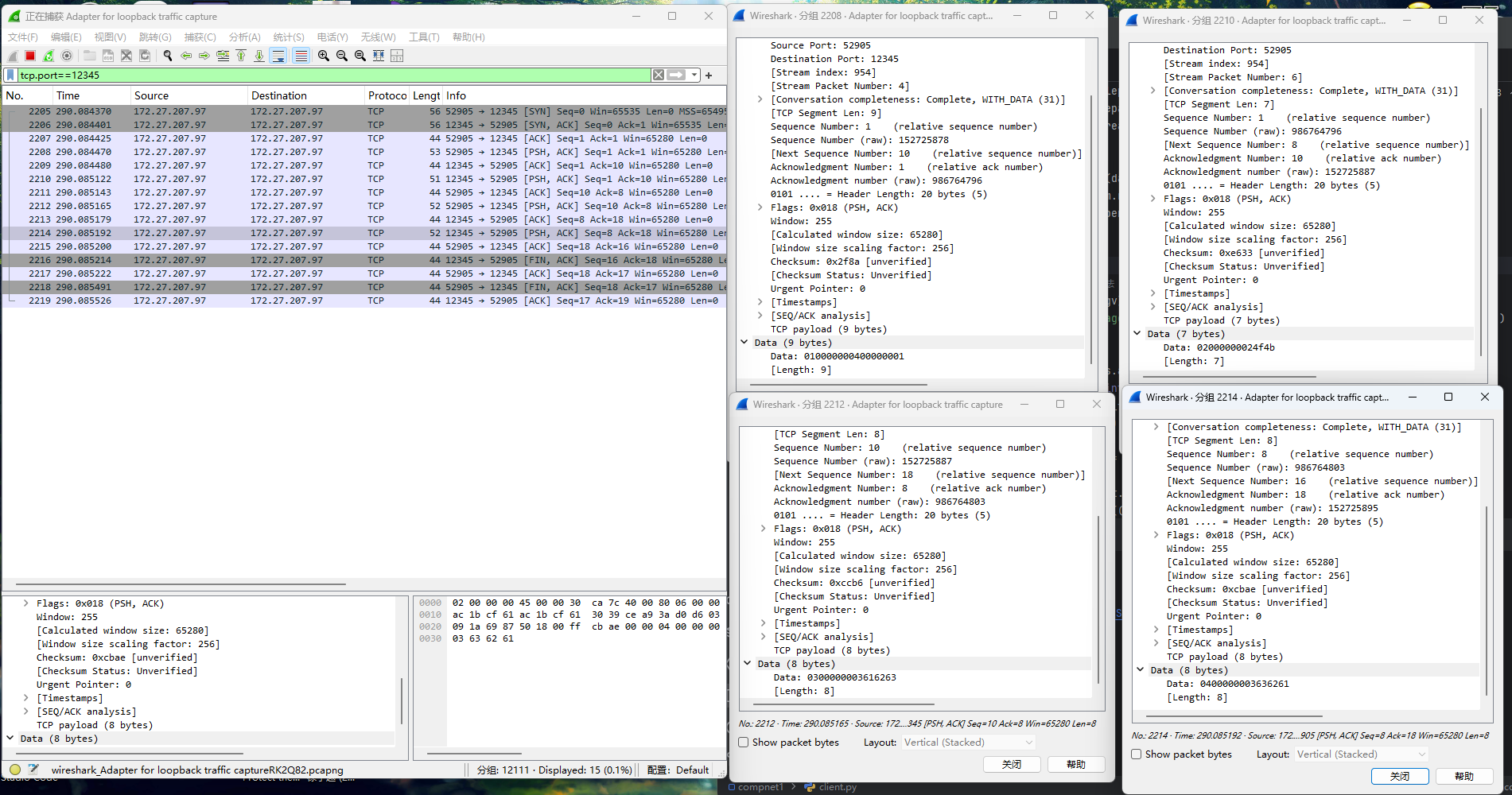
1. 抓包截图如下，明确显示了4种报文  
   

b. 实现的关键点和代码解决方案

可靠消息传输（解决TCP粘包问题）

关键点：确保完整接收变长文本块

解决方案：

def unpack\_message(sock):

header = sock.recv(5) # 先读取固定长度头部

msg\_type, length = struct.unpack('!BI', header)

data = b''

while len(data) < length: # 循环读取直到收满数据

packet = sock.recv(length - len(data))

data += packet

动态文件分块处理

关键点：生成随机长度的文本块

解决方案：

def split\_file(filepath, Lmin, Lmax):

i = 0

while i < len(data):

L = random.randint(Lmin, Lmax) # 随机块长度

blocks.append(data[i:i+L])

i += L # 移动指针

并发连接处理

关键点：同时服务多个客户端

解决方案：

while True:

conn, addr = server.accept()

threading.Thread(target=handle\_client, args=(conn, addr)).start()

逆向结果拼接

关键点：按接收顺序逆序组合结果

解决方案：

reversed\_full = '' # 初始化空字符串for block in blocks:

# 新接收的反转块添加到已有内容前面

reversed\_full = reversed\_text + reversed\_full

协议状态机控制

关键点：确保消息顺序正确

解决方案：

# 服务端严格校验消息顺序if msg\_type != TYPE\_INIT: returnif msg\_type != TYPE\_REQUEST: break

c. 掌握的知识点

网络编程核心技能

TCP socket通信（socket模块）

客户端/服务器架构实现

多线程并发处理（threading）

协议设计与实现

自定义二进制协议格式（头部+载荷）

消息类型状态机管理

端到端数据序列化（struct打包/解包）

高级文件处理

动态文件分块算法

随机块生成（random.randint）

大文件流式处理技术

字符串操作技巧

高效字符串反转（[::-1]语法）

编码转换（encode()/decode()）

内存拼接优化

工程实践能力

命令行参数解析（sys.argv）

异常处理与资源释放（try-finally）

调试日志记录

通过本任务新掌握的知识：

解决TCP粘包问题的标准模式

二进制协议头部设计（类型+长度）

服务端多线程并发响应架构

动态文件分块与重组技术

网络程序的状态机控制逻辑

d.URL: