以下是根据课程设计要求撰写的"说明文档"框架及内容。由于无法实际执行代码或捕获网络数据包，我将提供完整的文档结构和填充示例内容，您可以根据实际项目情况补充具体细节：

​基于UDP的可靠文件传输协议实现说明文档​

​文件名：​​ udp\_packet\_capture.doc

a. Wireshark包捕获截图

(注：此处需插入实际抓包截图)

​截图示例说明：​​

​握手过程​

图1：客户端发送SYN包（序列号=0）

图2：服务器返回SYN-ACK包（序列号=1000, ACK=1）

图3：客户端确认ACK包（序列号=1, ACK=1001）

​数据传输​

图4：序列号为1-400的数据包传输（窗口大小400字节）

图5：服务器返回ACK 401确认包

图6：序列号401-800的数据包重传（模拟丢包场景）

​连接终止​

图7：FIN标志位交换过程

b. 实现关键点及解决方案

​关键点1：可靠传输模拟​

python

python

复制

# 解决方案：自定义协议头设计

class PacketHeader:

def \_\_init\_\_(self, seq, ack, flags, data):

self.seq = seq # 序列号(4字节)

self.ack = ack # 确认号(4字节)

self.flags = flags # 标志位(2字节: SYN/ACK/FIN)

self.checksum = 0 # 校验和(2字节)

self.\_pack(data)

def \_pack(self, data):

# 计算校验和并封装二进制数据包

header = struct.pack('!IIHH', self.seq, self.ack, self.flags, self.checksum)

...

​关键点2：滑动窗口控制​

python

python

复制

# 解决方案：窗口状态跟踪

class WindowManager:

def \_\_init\_\_(self, size=400):

self.base\_seq = 0 # 窗口起始序列

self.window\_size = size # 字节窗口

self.pending\_acks = {} # {seq: (sent\_time, retry\_count)}

def can\_send(self, next\_seq):

return (next\_seq - self.base\_seq) < self.window\_size

​关键点3：动态超时重传​

python

python

复制

# 解决方案：RTT自适应计算

def update\_timeout(self, sample\_rtt):

# EWMA算法: 新超时 = (α \* 旧超时) + ((1-α) \* 新RTT)

self.timeout = max(0.1,

0.875 \* self.timeout +

0.125 \* sample\_rtt \* 2.5)

# 重传检测线程

for seq, (sent\_time, \_) in self.pending\_acks.items():

if time.time() - sent\_time > self.timeout:

self.\_retransmit(seq)

​关键点4：丢包模拟机制​

python

python

复制

# 服务器端丢包模拟

def receive\_packet(self):

if random.random() < 0.3: # 30%丢包率

log("DROPPED PACKET: Simulating network loss")

return None

...

c. 知识点总结

​已掌握的核心知识：​​

| 知识领域 | 具体技术点 |
| --- | --- |
| 网络协议栈 | UDP协议特性 vs TCP可靠性机制 |
| Socket编程 | Python socket模块高级应用 |
| 协议设计 | 自定义二进制协议头封装/解析 |
| 多线程编程 | 收发线程分离+超时监控线程 |
| 网络诊断工具 | Wireshark抓包与协议分析 |

​通过Task新掌握的知识：​​

​滑动窗口实现​

字节级窗口控制（非报文计数）

累积确认机制优化

​RTT动态计算​

指数加权移动平均(EWMA)算法

超时系数动态调整策略

​传输状态机设计​

连接建立(SYN/SYN-ACK)

优雅关闭(FIN交换)

异常中断处理

​网络模拟技术​

可控丢包率实现

延迟插入仿真

d. Git仓库URL

​项目地址：​​ https://github.com/[YourGitUsername]/Reliable-UDP-Implementation

​代码结构：​​

markdown

markdown

复制

├── client.py # 客户端实现

├── server.py # 服务端实现

├── protocol.py # 协议封装模块

│ ├── PacketHeader # 协议头类

│ ├── checksum\_calculator # 校验和计算

│ └── packet\_parser # 数据包解析

├── window\_manager.py # 滑动窗口控制器

└── docs/

└── udp\_packet\_capture.doc # 本文档

文档使用说明

​截图补充​：请将Wireshark抓包截图按章节插入对应位置

​Git信息​：替换方括号中的Git账号信息

​代码细节​：实际实现的关键函数可补充代码片段

​数据补充​：在测试结果部分添加实际传输的统计指标：

markdown

markdown

复制

### 性能测试

| 测试项 | 值 |

|-------------|----------|

| 平均RTT | 34.2ms |

| 丢包率 | 28.7% |

| 重传率 | 42.3% |

| 最大吞吐量 | 1.2MB/s |

此框架完全遵循课程设计要求，您可直接补充实际项目内容后提交。建议用Word或Markdown工具生成.doc格式文档。