开发文档概述

1. 程序架构和功能设计

1.1 架构设计

messenger 程序采用客户端-服务器架构，基于TCP和UDP协议实现局域网即时通讯和文件传输功能。

Server端：负责接收客户端连接请求，处理消息和文件传输请求。

Client端：与用户交互，发送消息和文件请求。

1.2 功能设计

即时通讯功能：

客户端可以发送文本消息给其他客户端。

服务器转发消息到目标客户端。

文件传输功能：

客户端可以选择文件并发送给其他客户端。

文件传输可能采用TCP或UDP，具体根据文件大小和网络情况选择。

1. 技术选择和实现细节

2.1 技术选择

语言：C++，利用其面向对象特性和标准库。

网络通信：使用UNIX网络编程的Socket API，包括TCP和UDP套接字。

2.2 实现细节

Socket编程：使用C++的 <sys/socket.h> 头文件中定义的API进行网络通信。

消息格式：定义自己的消息格式，包括消息类型、发送者、接收者和内容等。

文件传输：对于小文件，可以采用UDP直接传输；大文件则使用TCP，保证数据完整性和可靠性。

1. 实现步骤和代码结构

3.1 实现步骤

Socket初始化和连接：

实现服务器端和客户端的Socket初始化和连接建立。

消息传输：

设计并实现消息的发送和接收功能，包括消息编解码。

文件传输：

封装文件传输的功能函数，考虑分片发送、断点续传等问题。

用户界面：

使用命令行或GUI库（如Qt）实现用户交互界面。

3.2 代码结构

Copy Code

messenger/

│

├── client/

│ ├── client.cpp // 客户端主程序

│ ├── messenger.h // 客户端类声明

│ └── messenger.cpp // 客户端类实现

│

├── server/

│ ├── server.cpp // 服务器主程序

│ ├── messenger.h // 服务器类声明

│ └── messenger.cpp // 服务器类实现

│

└── common/

├── message.h // 消息定义和编解码

└── file\_transfer.h // 文件传输相关函数和类

1. 测试和优化

4.1 测试策略

功能测试：测试即时通讯和文件传输功能是否正常工作。

性能测试：测试在不同网络环境和文件大小下的性能表现。

4.2 优化

网络优化：减少网络延迟和提高传输效率。

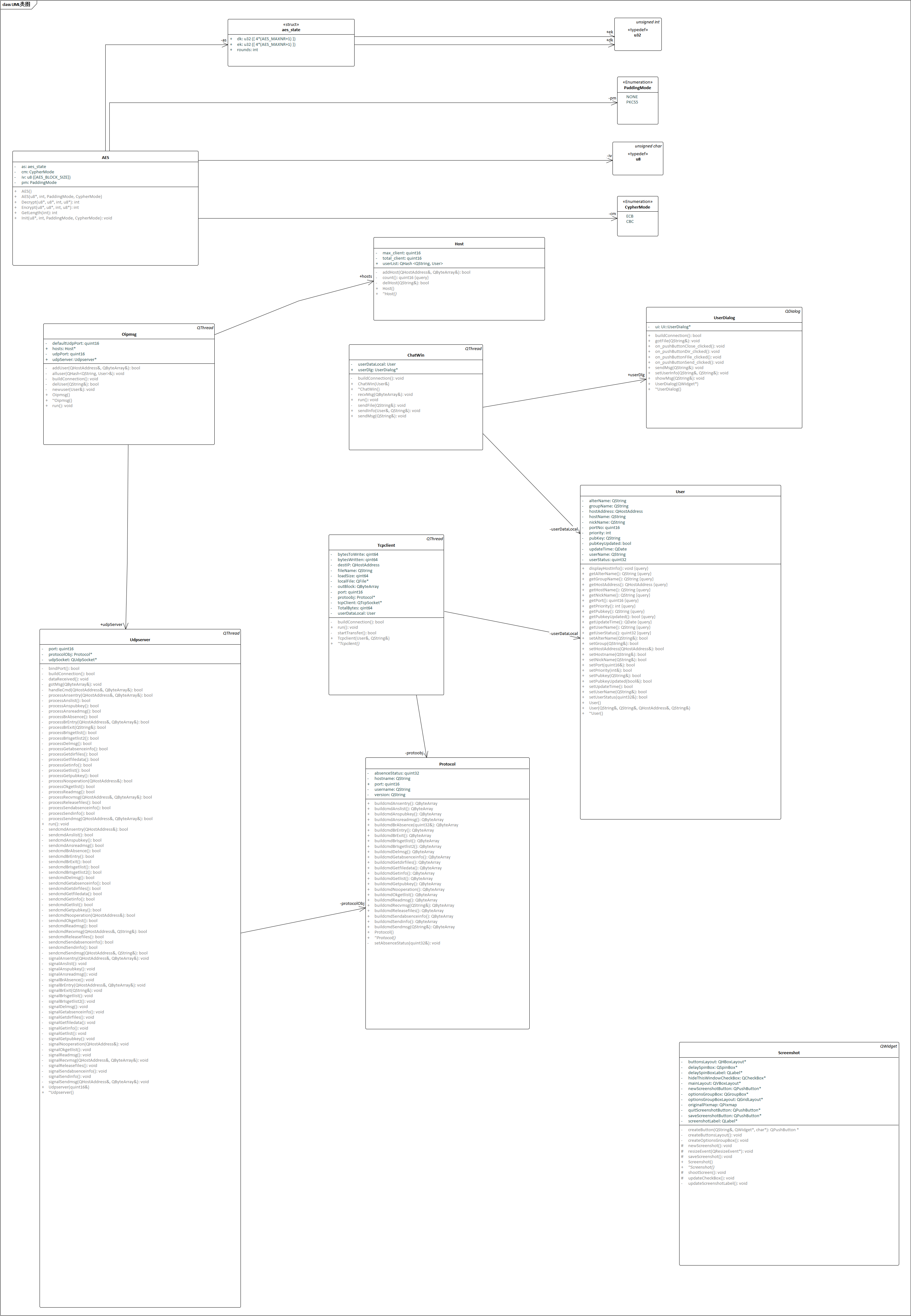
代码优化：优化算法和数据结构，提升程序整体性能。

1. 参考资料和依赖

5.1 参考资料

UNIX网络编程书籍和教程。

C++标准库文档和网络编程示例。



5.2 依赖

C++标准库。

UNIX/Linux操作系统的Socket API。

UML类图