项目五软件详细设计文档

# 范围和内容

遵守软件设计方案，满足需求的客户机、服务器的软件详细设计文档。

# 术语与约定

1. 服务器：项目中提供认证服务、接受文件上传、下载请求的软件实体。
2. 客户机：项目中发起连接、认证，并提供文件上传、下载功能的软件实体。
3. 文件：多数情况下可以指代“文件/文件列表”，因为使用停等协议传输的过程中，文件或文件列表都将被分为一份份数据，使用一种PDU类型，在传输前相互确认传输内容，过程中根据对应的方法解读和使用其中的数据即可。

# 功能模块设计概述

# 基本需求

1. 服务器实现同时接入多个客户端
2. 结合项目三，每个客户通过认证，并各自使用不同的帐号
3. 客户端可以看到当前接入用户的列表
4. 结合项目二，客户端可以选择与某个指定用户进行聊天，或者选择将数据发送给所有客户
5. 结合项目三，客户可以上传或下载文件
6. 两个客户之间可以通过服务器的中转在线相互传递文件，即不是一边先上传，完毕后，另一边再下载，而是通过服务器代理中转的“端到端”传输（类似QQ的在线传输）
7. 客户端能够接收服务器的广播命令

# 功能概述

1. 服务器可以服务多个客户端，为每个客户端提供聊天、群聊、文件上传下载、客户端之间文件“直传”（即文件不在服务器端保存）等功能。
2. 客户端输入正确的帐号密码后即可连接服务器，请求文件传输，或者与其他在线客户端进行聊天，也可以选择与客户端直接进行文件“直传”。

# 主要技术指标

1. 服务器同时服务最多64个客户端，针对每个客户端同时服务最多8条数据连接。
2. 文件数据传输时，采用停等协议，设置超时重传。
3. 针对服务器与客户端之间的交互，定义了简单的协议，例如设计了适用于不同类型连接不同请求的多种PDU，以向用户提供良好的聊天信息与数据传输的服务。

# 关键问题分析及解决思路

1. 问题：如果向用户提供良好的服务

解决思路：使用MFC界面程序设计，使用户能进行更良好的交互

1. 问题：如何服务多个用户

解决思路：采用MFC中的异步响应机制，为每个用户的多种套接字注册异步响应事件，当收到用户的请求消息时，立即调用相应函数进行相应。

1. 问题：如何实现群发消息

解决思路：在协议设计时，规定了ChatPDU中的cmd标识位中的值来确定一条聊天信息是否为群发信息。

1. 问题：如何实现在线用户列表的更新

解决思路：当一个新用户上线时，服务器会向所有在线用户广播新用户的ID、昵称（帐号）等信息，并且会向新用户依次发送所有在线用户的信息。

1. 问题：文件传输如何实现

解决思路：在原有的用户用于登录、聊天的控制套接字之外，当用户希望进行文件传输时，由服务器发起的向用户建立的数据套接字，专门用于文件/文件列表数据的传输。

1. 问题：如何用户之间实现文件“直传”

解决思路：在文件传输的基础上，当服务器收到“直传请求”时，根据发送方提供的用户ID找到接收方用户，建立一定的映射关系，并向接收方发送文件“直传”的请求，在收到接收方肯定的回复后，向双方新建数据连接，开启文件中转传输。

# 软件模块划分与层次结构关系

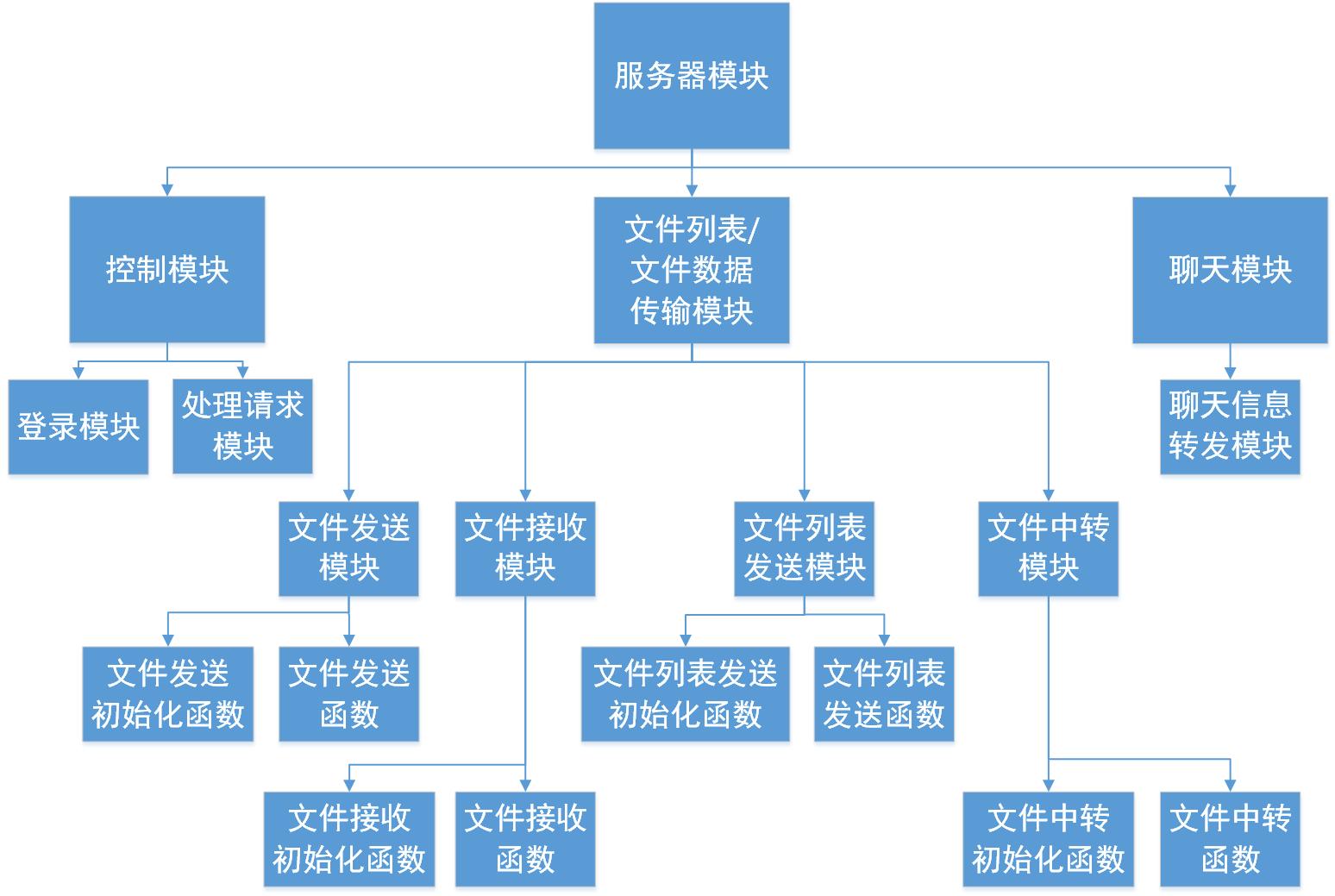


图3-5-1 服务器模块与层次关系

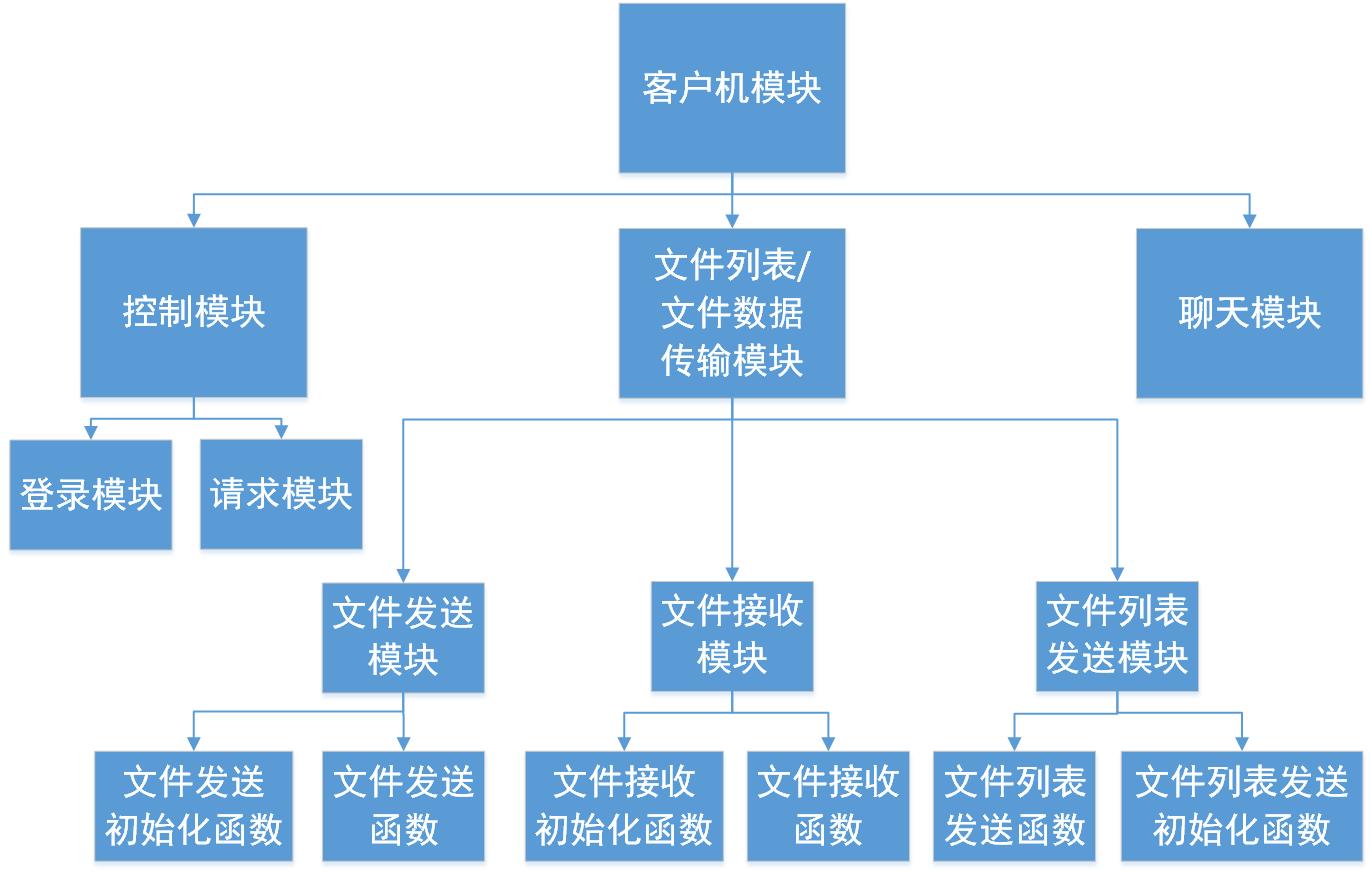


图3-5-2 客户端与层次关系

# 工作原理

1. 控制模块

协议约定控制套接字上传输的PDU中各字段含义，由cmd字段值区分不同的请求，以及服务器收到请求之后的响应。

1. 文件列表/文件数据传输模块

文件数据连接后的第一份文件PDU，其中由No字段标识了该连接类型，开始传输后，No值标识数据包编号，结合停等协议，进行差错重传。

1. 聊天模块

客户端与服务器建立连接，并能够以服务器为桥梁相互通信。该通信的关键在协议报文的设计实现上，为能准确通信，报文内容除了消息本身外，还应包括由frdno字段标识的发送对象（标识符），以及由cmd字段标识消息的类型。

# 各子模块方案设计

1. 控制模块

客户端与服务器建立连接后，服务器通过帐号密码来确认用户身份，若是合法用户，则为其服务。客户端可以向服务器请求文件的上传下载、获取文件列表或者发起客户端之间的文件“直传”，这些都取决于客户端向服务器发送的控制PDU中的命令cmd字段中的值，服务器对其值进行判断后，做出响应，完成客户端的需求。

1. 文件列表/文件数据传输模块

服务器在控制套接字上收到用户的文件数据传输请求后，新建数据套接字，来传输用户所需的文件的数据。并且，在连接建立的第一刻，服务器便会向客户端发送一份文件PDU，其中由No字段标识了该连接是上传文件（UP\_FILE\_START）、下载文件（DOWN\_FILE\_START）或是获取文件列表（GET\_FILELIST\_START），以此让客户端做出相应的反应，如建立文件准备接收，打开文件开始发送，或者是准备接收文件列表项并显示在界面上。最后，由发送文件的一方，发送一份长度为3、len字段值为0的数据包作为结束标识，接收方做出相应的结束动作。

1. 聊天模块

客户端与服务器建立连接，并能够以服务器为桥梁相互通信。该通信的关键在协议报文的设计实现上，为能准确通信，报文内容除了消息本身外，还应包括消息发送对象（标识符）与消息长度。此外，当其他用户上线或下线时，服务器应通知其他用户对其好友列表进行维护。综上，聊天的PDU报文应包括消息、消息长度、发送对象、命令（信令）等这四个要素。

功能函数方面，应包括：

客户端——用户列表维护、聊天消息发送函数、聊天消息处理显示；

服务器——用户上线通知、用户下线通知、聊天消息处理。

# 接口设计

# 外部接口关系

聊天模块，除void Frd\_Online\_Notice(SOCKET sock)，void ChatProcess(SOCKET sock, ChatPDU \*chat\_buff)，void Modify\_Friend\_List()外，其他的消息处理的部分整合于LRESULT WindowProc(UINT message, WPARAM wParam, LPARAM lParam)。

# 内部接口关系

无

# 模块详细设计

* 1. 报文格式及定义

1. **质询报文PDU**

#define INQUIRING\_PDU\_TYPE 1 //质询报文类型代号

typedef struct InquiryPDUS {

char type; //PDU类型，值为INQUIRING\_PDU\_TYPE = 1

char num; //存储质询数个数（范围1-16）

char each\_len[4]; //表达每个质询数的长度。每个表达占2位，值1、2、3、0分别表示质询数长度为1、2、3、4。4个字节32bit，最多能表达16个数。

char inquiry\_buf[INQUIRY\_BUF\_SIZE]; //每个质询数，高位存储在前，低位存储在后，符合网络字节序

} InquiryPDU;

1. **控制命令报文PDU**

#define CTL\_PDU\_TYPE 2

#define CTL\_BUF\_SIZE 255 //一个报文载荷所能存储的最大字节数

//以下为命令位cmd取值，改为2开头的两位数，与CTL\_PDU\_TYPE看齐

#define REQ\_ACCOUNT 20 //服务器向客户端请求用户帐号

#define REQ\_FILELIST 21 //客户端向服务器请求获取文件列表

#define REQ\_DOWNFILE 22 //客户端向服务器请求文件下载

#define REQ\_UPFILE 23 //客户端向服务器请求文件上传

#define REQ\_TRANSLATE 24 //文件中转确认信息

#define RE\_ACCESS 25 //服务器告知客户端登陆成功

#define RE\_TIMEOUT 26 //服务器告知客户端超时

#define RE\_ERR\_ACCOUNT 27 //服务器告知客户端用户名错误

#define RE\_ERR\_INQUIRY 28 //服务器告知客户端质询结果不通过

#define RE\_ERR\_FILENAME 29 //服务器告知客户端文件名错误

typedef struct CtlPDUS{

unsigned char type; //PDU类型，值为CTL\_PDU\_TYPE = 2

unsigned char cmd; //控制命令，可选参数见于上文宏定义

unsigned char len; //报文中载荷的长度，0~255

unsigned char id; //在文件上传时使用，标识文件传输对象为服务器或者其他用户

char buf[CTL\_BUF\_SIZE]; //载荷，根据不同的命令cmd值

//可填用户名、上下载的文件名等

} CtlPDU;

说明：该类型PDU主要用于客户端的控制套接字与服务器交互必要的控制信息。例如登录时，服务器向客户端请求帐号；得到客户端返回后，经验证后再发送质询PDU；得到质询返回值且验证正确后，客户端可以根据需求向服务器发送请求，包括获取文件列表、下载文件、上传文件等。

1. **文件/文件列表传输报文PDU**

#define FILE\_PDU\_TYPE 4 //文件传输类型

#define FILE\_BUF\_SIZE 255 //一个报文载荷所能存储的最大字节数

//数据连接建立后的第一帧皆由服务器发送至客户端

//通过不同No值，告知该连接的功能

#define GET\_FILELIST\_START 2

#define DOWN\_FILE\_START 3

#define UP\_FILE\_START 4

typedef struct FilePDUS{

unsigned char type; //PDU类型，值为FILE\_PDU\_TYPE = 4

unsigned char No; //传输时，PDU编号，可选参数0 或1第一份文件/文件列表PDU，No值可选2、3、4

unsigned char len; //默认为报文长度总长度。当值为0时，表示传输结束

char buf[FILE\_BUF\_SIZE]; //载荷，具体包括文件列表各文件名或文件数据流

} FilePDU;

说明：当服务器在控制套接字上收到客户端的不同的请求之后，向客户端新建数据连接，发送不同类型的文件/文件列表PDU第一帧，具体由PDU中的No值来区分，客户端收到第一帧后，做对应的准备处理，之后开始正常的文件/文件列表传输。

1. **聊天消息报文PDU**

#define CHAT\_PDU\_TYPE 6 //聊天报文类型代号

#define CHAT\_BUF\_SIZE 255 //一个报文载荷所能存储的最大字节数

#define ADD\_FRD 60 //服务器通知客户端，添加聊天用户指令

#define DEL\_FRD 61 //服务器通知客户端，删除聊天用户指令

#define FRD\_MSG 62 //单发聊天信息

#define BROCAST\_MSG 63 //群发聊天信息

typedef struct ChatPDUS{

unsigned char type; // PDU类型，值为CHAT\_PDU\_TYPE = 6

unsigned char cmd; //命令，包括好友上线、下线、单播信息、广播信息

unsigned char frdno; //用户ID（Friend No）

char msg[CHAT\_BUF\_SIZE]; //载荷，包括聊天消息或上下线用户名等

} ChatPDU;

说明：基本沿用了项目二的PDU设计，略作调整，命令cmd位标识不同的命令，好友上下线等都是有服务器发现有变动时主动发给客户端。例如有新用户上线，服务器告知所有已在线用户上线的新用户的情况，并将所有已在线用户的信息告知新用户，下线类同。

* 1. 协议描述
  2. 登录：

客户端连接服务器后，服务器请求帐号，客户端发送帐号，服务器查找用户信息文件（UserList.txt），找到对应帐号，生成发送质询值，或告知帐号错误。客户端计算发送质询返回值，服务器验证返回值通过，告知客户端登录成功，否则告知其登录失败。

* 1. 聊天：

登录成功后，客户端可向服务器发送符合格式的ChatPDU，服务器识别PDU类型，根据cmd字段和frdno字段，选择单播聊天信息、广播聊天信息。每当有新客户端登录成功后，向在线客户端广播新客户端信息，并告知新客户端已在线客户端信息，即登录后初始化，之后动态更新。

* 1. 文件传输：

登录成功后，客户端可向服务器发送符合格式的CtlPDU，以请求上传下载文件或者获取文件列表。服务器根据请求，新建套接字。发送第一份FilePDU声明该连接类型以及传输的文件名（若是文件列表，则无此动作）。之后，采用停等协议，依次发送接收文件。

* 1. 子模块设计
     1. 子模块：控制模块
        1. 子模块流程

1. 客户端连接服务器
2. 服务器请求帐号
3. 客户端回应帐号
4. 服务器查用户信息文件，
   1. 若正确，计算发送质询数
   2. 若错误，发送“帐号错误”，回到步骤2
5. 客户端计算发送质询返回值
6. 服务器接收检验返回值
   1. 若正确，发送“登录成功”
   2. 若错误，发送“登录失败”，回到步骤2
7. 若客户端发送请求
8. 则服务器在收到请求回应“ACK”，启动对应模块处理，回到步骤7
   * + 1. 子模块各相关处理函数描述

void CServerDlg::AcceptNewUser(SOCKET sock);

//函数：接收新用户

//功能：接收新的连接建立，添加新用户，初始化

void CServerDlg::RecvCtl(SOCKET sock, struct CtlPDU \*ctl\_pdu);

//函数：分类CTL报文

//功能：根据Ctl报文中的cmd字段值，分发不同的处理函数

int CServerDlg::RequstAccount(SOCKET sock);

//函数：发送“询问账号”请求

//功能：新建CtlPDU，填入type字段为CTL\_PDU\_TYPE，cmd字段为R\_ACCOUNT，长度len字段为0；向套接字sock对应用户，发送2个字节

short CServerDlg::RecvAccount(SOCKET s, struct CtlPDU \*ctl\_pdu);

//函数：接收用户帐号信息

//功能：检查用户帐号是否存在，若存在，发送质询数；若不存在，发送错误，再次请求帐号

unsigned long UserUnit::f\_producenum(struct InquiryPDU \*p\_pdu, long secret, unsigned char \*bufflen);

//生成质询数函数

//位置：发送方

//功能：生成发送的随机数数组，并计算出质询值用以验证

//输入：

// buff[]——存储待发送的随机数PDU

// bufflen——存储待发送的数组实际长度

//输出：

// 计算得到的质询值

short RecvInquiry(SOCKET s, struct InquiryPDU \*inq\_pdu);

//函数：接收质询值

//功能：比较接收到的质询值与缓存中用户对应的质询值，一致则登录成功，不一致则重新登录

int SendCtlPDU(SOCKET sock, char cmd);

//函数：发送回复

//功能：新建CtlPDU，填入type字段为CTL\_PDU\_TYPE，长度len字段为0；向套接字sock对应用户，发送2个字节

* + 1. 子模块：文件/文件列表数据传输模块
       1. 子模块流程

1. 服务器在控制套接字上收到客户端的文件传输请求
   1. 若是文件“直传”请求，跳转到步骤2
   2. 否则，跳转到步骤4
2. 则向传输对象发送文件“直传”确认请求
   1. 若对方同一，则跳转到步骤3
   2. 否则，跳转到步骤1
3. 服务器向传输对方客户端建立新连接
4. 服务器向发起请求客户端建立新连接
5. 服务器向新连接发送第一份报文，告知该连接功能
6. 根据不同连接的性质，由协议中规定的开始方开始发送
7. 对端做接收存储返回ACK，或者转发动作
8. 发送方收到ACK，再次发送一份数据，回到步骤7，循环直至传输结束
   * + 1. 子模块各相关处理函数描述

short StartSendFilelist(SOCKET s);

//函数：处理文件列表请求函数

//功能：收到文件列表请求后，新建套接字，发送第一份列表中文件信息，设置套接字状态为S\_SNDFL0

short SendFilelist(SOCKET s);

//函数：发送文件列表函数

//功能：收到用户ack成功接收上一份文件列表数据后，发送下一份文件列表数据

short StartSendFile(SOCKET s, char \*filename);

//函数：处理文件下载请求

//功能：收到文件下载请求，检验文件名合法性，会送合法文件名，文件大小

short SendFile(SOCKET sock);

//函数：发送文件

//功能：得到上一帧ACK回复之后，发送下一帧，若文件一道结尾，则发送len=0的帧表示结束

short StartRecvFile(SOCKET s, char \*filename);

//函数：开始接收文件

//功能：处理上传文件请求，新建套接字，发送文件信息

short RecvFile(SOCKET sock);

//函数：接收文件

//功能：接收到一帧合法的数据帧后，发送ACK；若收到len=0的帧，则关闭套接字。

void CServerDlg::RequestTranslateFile(SOCKET sock, struct CtlPDU \*ctl\_pdu);

//函数：请求文件数据转发

//功能：当一个客户端发起文件“直传”请求，服务器向对方询问是否接收文件

short CServerDlg::StartTranslateFile(SOCKET sock, int i, int j);

//函数：开始文件数据转发

short CServerDlg::QuitTranslateFile(SOCKET sock, int i, int j);

//函数：结束文件数据转发

short CServerDlg::TranslateFile(SOCKET sock);

//函数：转发文件数据包

* + 1. 子模块A：（服务器）新用户上线通知
       1. 子模块接口：void Frd\_Online\_Notice(SOCKET sock);

其中，sock：新用户的sockets标识符

* + - 1. 模块流程

1. 通过sock查询改用户用户名和ID；
2. 根据以上信息封装PDU；
3. 发送给该用户以外的所有用户。
   * 1. 子模块B：（服务器）聊天消息处理函数
        1. 子模块接口：void ChatProcess(SOCKET sock, ChatPDU \*chat\_buff);

其中，sock：发送消息用户的sockets标识符；char\_buff 指向接收的聊天消息报文的指针

* + - 1. 子模块流程

1. 通过sock查询改用户用户名和ID；
2. 根据PDU中的cmd字段执行相应的处理。
   * 1. 子模块C：（客户端）好友列表维护
        1. 子模块接口：void Modify\_Friend\_List();
        2. 子模块流程
3. 清空好友列表；
4. 重组好友ID与显示的映射关系；
5. 逐条显示。

# 存在的问题

1. 服务器界面上暂时没有注册用户的选项，只能由管理员手动在用户信息文件中添加。
2. 当客户端收到文件“直传”的请求时，必须立即做出决定，没法等待一段时间后再处理。
3. 没有进行压力测试

# 结语

# 通过本项目对之前项目二和项目三的代码与功能进行了整合重组，并再次基础上，对功能进行了拓展，对界面进行了完善。不过由于能力与时间等因素，本项目并未达到实用的程度，例如并没有进行一定的压力测试等。

# 参考文献

微软官方文档：http://msdn.microsoft.com/