计算机网络课题设计

Fellow Chat

学院：信息学院

专业：计算机科学与技术

团队成员：

覃龙虎　22920172204209

赵轩　　22920172204300

李展昆　22920172204151

目录

一、 项目简述

Fellow Chat(群协作工作间)的实现。

我们小组想利用 java 语言以及 socket 编程的相关思想，结合计算机网络课程上所学到的相关网络传输的知识，通过数据库（用户数据库、群组数据库和文件数据库），实现一个类似聊天室的小工具。逻辑上，实现聊天的信息发送，并通过数据库实现多人群聊功能的实现，比较新颖的想法是在群聊中加入类似 github 代码管理的协作功能，即每个群组成员都可以上传项目文件并实时更新，其他群组成员也可以对上传的文件进行一个修改或是下载。

根据我们在计算机网络课程上学习的课程知识，我们可以知道，在计算机网络的边缘部分的通信过程中，存在两种连接方式。一种是客户-服务器方式，另一种是 P2P 对等连接方式，在我们的课程项目中，这两种连接方式我们都有涉及。在用户和用户进行一对一的聊天时，我们采用了类似 P2P 对等连接的方式,而在多人聊天以及登陆注册等方面，我们采用了客户-服务器方式（C/S 方式）。在下面的部分会进行更详尽地解释。

同时，结合 javafx 的相关知识，对这个协作间进行一个界面上的优化，增强我们这个项目的可观性，更加方便用户使用我们的协作工具。

二、 项目原理

1. C/S 交互（控制连接）

作为中央服务器的客户端，连接中央服务器(x.x.x.x,6666），同时告知服务器自己的点对点服务端口 X，这个 X 号端口是随机分配的，但在用户登录后就一直不变，所以又是固定的。

2. P2P 聊天

作为 P2P 服务端，使用 X 号端口进行监听其他 P2P 客户请求，这个 X 号端口在用户登录时向中央服务器注册。

作为客户端，可以主动连接目的 P2P 主机的“工作端口 X“

3. C/S 广播和多播（群聊）

聊天主页面可以实现广播，向全体成员发送信息（要指定管理逻辑，只有高层能发公司重要通知，有待优化，目前主要作为一个测试 C/S 的显示模块）

群聊页面可以多播（仅仅在群聊成员中发消息），群聊的消息通信也是以服务器为中介，向群组成员多播发送的，不同于点对点的通信。

4. FTP 模式下的文件传输

文件的发送都是采用类 FTP 的模式，即通过两条连接来实现，包括“控制连接”和“数据连接”。发送方先建立一个随机端口，通过“控制连接”告诉接收方，接收方再主动的连接发送方，发送方随即通过数据连接发送文件。

~~~~~~

在这里，重点解释一下我们项目的 TCP 连接方式。

根据计算机网络课程上学习的知识，我们可以知道，在互联网的边缘部分的通信过程中，存在两大类通信方式，一种是客户-服务器（C/S）连接方式，另一种是对等连接 P2P 连接方式。而这两种连接方式各有各的特征特点和用途，而在我们的项目设计中，这两种连接方式可以说是都有涉及，而这两种连接方式也有一起结合使用的情况。下面我将结合图示进行讲解。

首先，我们的项目设计的大体结构包括一个储存信息的数据库，一个接受用户请求服务的服务器，（还有一个客户文件存储区），还有若干个客户端。如下图所示：
~~~~~~

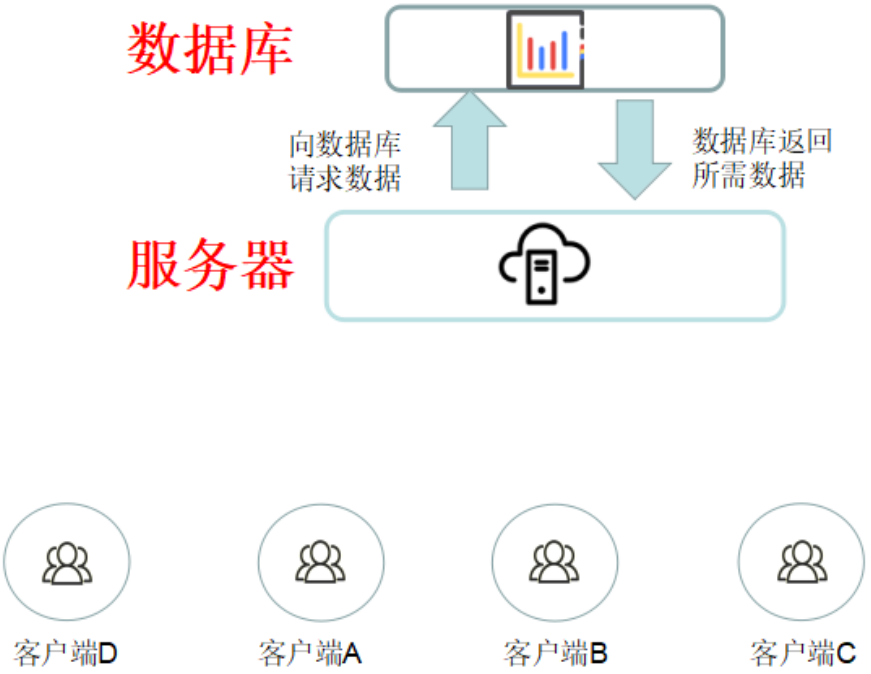


图 2-1　C/S 原理图

下面是进行一对一通信的两张示意图，包括一对一进行聊天信息交换的连接示意图，另一张是一对一进行文件传输的连接示意图。这两种类型的通信有一点不同

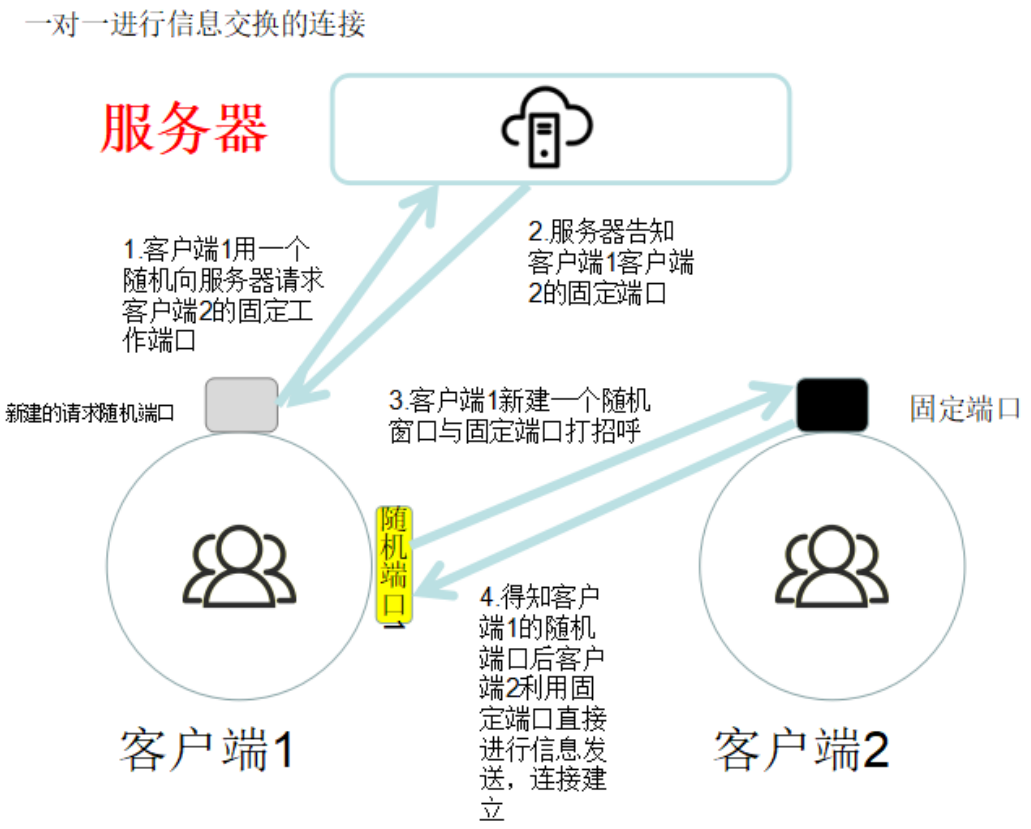


图 2-2　P2P 原理图

根据示意图，我们可以了解一对一进行聊天通信的大致流程。

首先，每个用户在注册的时候都会在服务器中登记一个固定的 P2P 工作端口，这个端口会保存在服务器连接的数据库中。用户在想与其他用户进行通信的时候，可以通过服务器获得其他用户的固定 P2P 工作端口。

在每一次进行一对一通信的时候，比如说客户端 1 想与客户端 2 进行通信，就会通过自己生成一个随机端口向服务器发出请求，请求查询客户端 2 的固定工作端口号。

服务器收到请求后，会去连接的数据库上查询客户端 2 的用户信息，查到 2 的固定工作端口后会向客户端 1 发送回答，回答客户端 2 的固定端口。

然后客户端 1 会根据客户端 2 的固定端口向客户端 2 发起连接请求，即新建一个随机的会话端口 1 来主动去连接客户端 2，客户端 2 监听到连接请求后建立连接，此后，客户

端 1 与 2 开始信息的沟通。

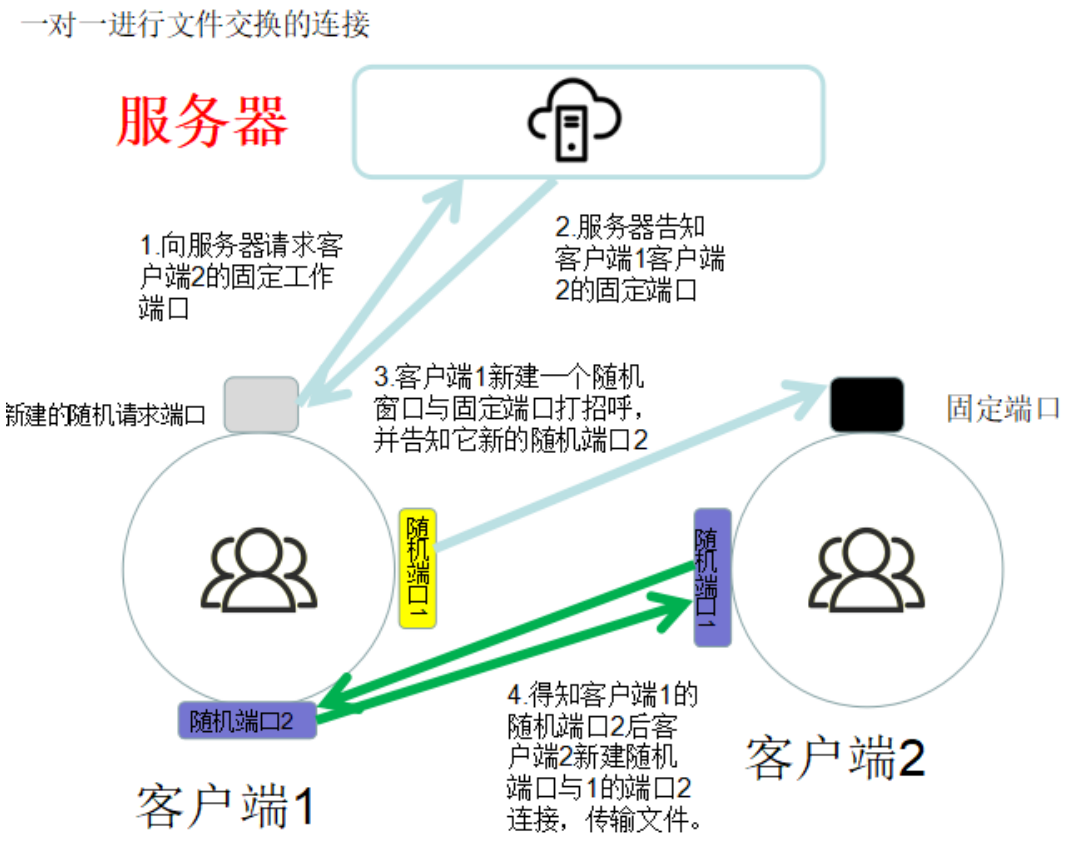


图 2-3　文件传输原理图

根据上面的一对一文件传输的相关示意图以及上面已经分析过的一对一聊天信息传输的过程，可知：

前面客户端 1 向服务器请求客户端 2 的固定工作端口的过程是一样。

不同的在于在客户端 1 从服务器得知客户端 2 的固定工作端口之后，生成了另一个随机文件端口 2，在打招呼的时候将该随机端口 2 的相关信息一并传给客户端 2。

客户端 2 的固定工作端口在收到客户端 1 随机端口 1 的相关招呼信息之后，生成一个随机文件传输端口 1，然后通过该随机端口 1 向客户端 1 的随机文件传输端口 2 发送信息，建立连接，传输文件。

最后关于一对一通信，我们认为我们项目中的这种通信方式其实是对等连接 P2P 连接的一种，因为客户端 1 和客户端 2 在功用地位上都是平等的。但是客户端 1 在寻找客户端 2 的时候，其实是又通过服务器的数据库来查找的，所以在这个通信的过程中，其实又包括了客户-服务器 C/S 通信方式的一些内容。

关于多人群聊，示意图如下：

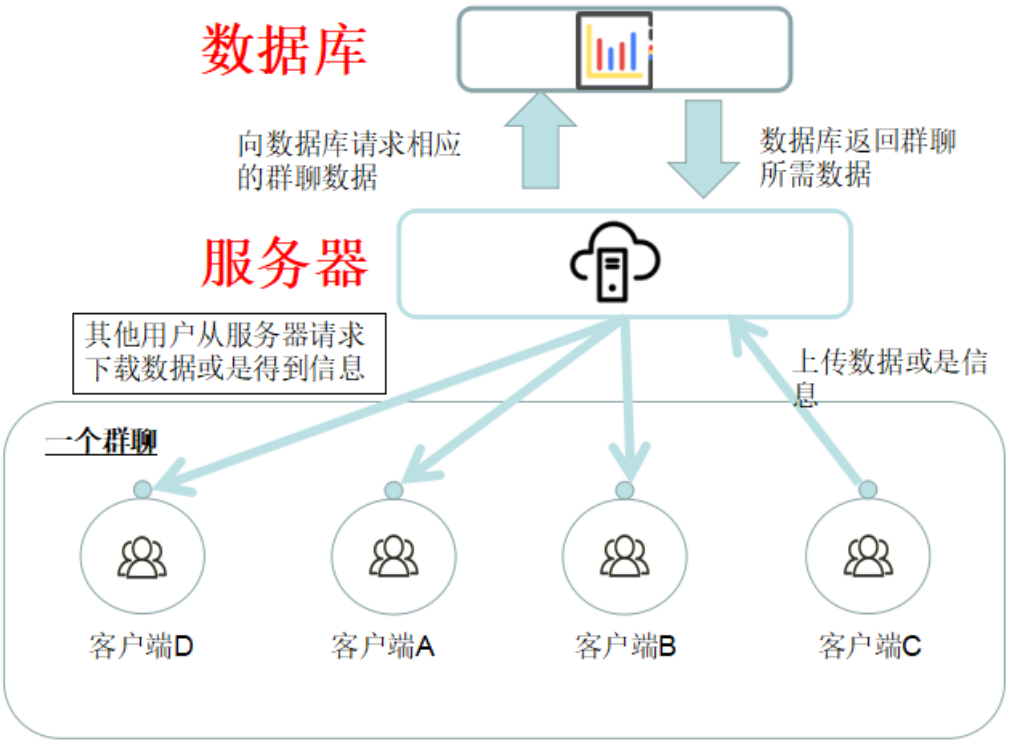


图 2-4　群聊原理图

通过这个示意图，我们可以了解多人群聊的大致原理。

首先是创建群聊，某个用户选中一些特定的用户发送给服务器，服务器将在负责记录

群组的数据库中新建一个群组，自动将相关用户的信息加入到相应的群组信息中去。这样，一个新的群组就创建成功了。

群组创建成功了以后，用户就可以在群组中发送信息或是传输文件（包括上传、更新、下载）。

服务器收到相应的数据后：

1. 若是用户发来的聊天信息，则将该信息转发给群组的其他成员

2. 若是上传的数据，则将文件保存在服务器的主机上面，同时更新数据库中的群组信息以及文件信息。

讲完了连接的方式和原理，我们来看看我们用到的数据库的结构和保存的信息。

数据库是用来反馈服务器请求的信息的，根据我们项目简述中的功能，我们可以知道服务器向数据库请求的数据大致有三类，一类是用户的相关信息，一类是群组的相关信息，还有一类是上传的文件的信息。所以，相对应的，就是三个相对应的数据库，分别是用户信息数据库、群组信息数据库和文件信息数据库。当服务器需要查询信息的时候，就从相应的数据库中查询相应的信息返回给服务器。

下面是三个数据库的相关信息结构以及截图。

1. 用户信息数据库：

Id	username	password	mail	message	state	serverPort	ipAddress	GIDs	G_num	G_next	G_message
1	aa	Ebwj2xospw3jqmyt2VIujg==	aa@qq.com		1	52428	127.0.0.1	#101	1	2	
2	bb	mPs0zYH78+zw85GLC6y6Qg==	bb@qq.com		0	0			0	1	
3	cc	JFQadoayKnijwe7Zxl766w==	cc@qq.com		0	0			0	1	
4	Kenelm	WsKnR0Zmum8xpbEX6NLz+w==	1097824882@qq.com		1	52436	127.0.0.1	#101	1	1	
5	dd	NShcJK5I6VpMG9oBVAaL6A==	dd@qq.com		0	0			0	1	
6	lzk	Cv2N9P+DwhOrgrng2dYu+Q==	lzk@qq.com		0	0			0	1	
7	ee	HMtZRI7ISh57AS2bj86Gig==	ee@qq.com		0	50970	127.0.0.1		0	1	
8	zx	4QrcOUm6Wau+VuBX8g+IPg==	105460737@qq.com		0	0			0	1	
9	QLH	WsKnR0Zmum8xpbEX6NLz+w==	mail		0	0			0	1	
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 2-5

从上面这张截图可以看出用户信息数据库中包括如下信息：用户的昵称 username、用户的密码 password、用户的注册邮箱 mail、用户发送的信息 message、用户的在线状态 state（在线为 1，离线为 0）、用户 P2P 通信的服务器端口 serverPort、IP 地址 ipAddress，用户所在的群组 ID--GIDS、用户所在的群组数量 G_num、G_next 以及在群组中发送的信息 G_message.

说明：

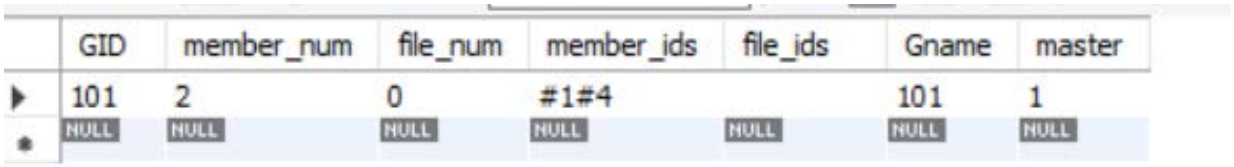
（1）目前 message 和 G_message 只能实现短期保存群组离线的消息，并在用户登陆后清除，在线消息不保存

（2）密码是通过加密后的保存的

（3）G_next 是为了方便知道用户创建下一个群聊时，GID 应该如何确定

（4）G_num 包括用户创建的群组和用户所属的群组

2. 群组信息数据库



GID	member_num	file_num	member_ids	file_ids	Gname	master
101	2	0	#1#4		101	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 2-6

包括如下类型的信息：

群组 ID-GID、群组成员数量 member_num、群组中文件的数量 file_num、群组中成员的 ID 序列 member_ids、群组中文件的 ID 序列 file_ids、组名 Gname、创建者 master.

说明：

GID = uid * 10 + G_next

3. 文件信息数据库

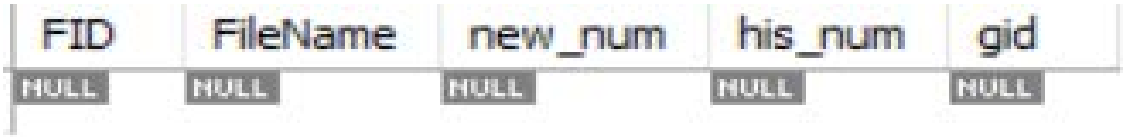


图 2-7

包括如下文件信息：

FID、FileName、new_num、his_num 和 gid。

说明：

FID = GID * 100 + file_num + 1

new_num 是新上传的版本的数量

his_num 是旧的版本的数量

gid 是文件所属的群组 ID

三、 项目设计

在具体的项目上，我们可以分为三个大部分，分别是：登陆注册、一对一聊天和多人群聊。下面我将对这三个部分进行一个笼统地介绍。

- ***登陆注册部分***

在这个界面，包括登录、注册和找回密码三个功能。

1. 登录

在客户端登陆界面，用户需要输入自己已经创建用户的用户名、密码以及自己主机的端口号以及 IP 地址。然后向服务器发送用户的基本信息，然后等待服务器的响应。

在服务器方面，在代码中有一部分专门用来处理用户请求的方法，登陆时调用user_login()方法，传入的参数包括用户名，密码，服务器 IP 地址以及客户端口号。根据数据库中存储的用户信息比对后，若是登录合法，则更新用户 IP、PORT 和状态。最后，调用 user_name_update()方法，更新用户状态。还有就是当有用户下线的时候，调用user_offline()方法改变其在线状态使其下线。

2. 注册

注册过程同登录过程大同小异。首先是在客户端注册界面，将注册所需的用户名邮箱等信息按照客户端发送线程发送给服务器，按下注册按钮后等待服务器的响应。

在服务器方面，调用 user_register()方法，首先判断传入的参数中的用户名是否已存在，如果不存在，就在数据库中添加一条新的记录，记录中包括新用户的 ID,用户名，密码，注册邮箱以及当前状态，完成新用户的注册。

3. 找回密码

在登录界面点击找回密码按钮，输入相应的用户名，注册邮箱以及新定义的密码发送给服务器，等待服务器响应。

在服务器方面，调用 user_forget()方法，根据传入的用户名和注册邮箱在数据库中的已有信息进行比对，找到该用户的记录。根据传入的新密码，更改该条记录的 passwd，最后实现密码找回并修改的要求。

- 一对一聊天部分

根据前面的项目简述我们已经知道，在这一部分我们采用的是对等连接 P2P 方式。采用 P2P 方式的特征就于不区分哪一个是服务提供方哪一个是请求方，只要运行了对等连接软件，就可以进行平等的对等的连接通信。

在这里进行一对一的通信消息传递时，这里是是两个用户直接通过登陆时登记的 IP 地址和主机的工作端口号进行主机之间的通信。但是这并不意味着整个过程都不使用服务器的服务，比如一开始在用户界面，就需要通过服务器来获取用户的登录状态信息，又比如在 P2P 通信时要通过服务器查找通信对象的固定服务端口号和 IP 地址，才能建立 P2P 连接。

值得注意的是，这里 P2P 连接的建立，包括主动连接和被动连接。若自己没有和通信对方建立连接，才会主动连接对方；但自己若已经被动接受，则直接使用这个连接。即本项目的点对点本质上还是 C/S 模式，不过主要通信过程在两点之间完成。

- ***多人群聊部分***

由前面的项目简述我们可以得知，在这一部分我们采用的是客户-服务器方式，这是因为，因为多人聊天的存在，不可能通过简单的一条连接完成群聊的操作，何况我们还添加了代码管理上传共享的功能。这样看来，在这一部分的实现上，有很多数据是需要服务器存储的，比如传递的消息，需要先交给服务器，然后服务器执行相应的操作，比如将消息转发给群组中的其他所有人，将上传的文件先保存在数据库中等待用户下载或是更新等等。很明显，在这里采用客户-服务器 C/S 方式是比较合适的。

群聊中具体的功能包括多人聊天、上传新文件、下载文件、上传文件的新版本、下载整个项目。

- ***文件传输部分***

本项目的文件传输采用的是类 FTP 协议，即维护两条 TCP 连接，一条控制连接，一条数据连接。在发送文件时，发送方通过控制连接来向接收方来传达指令，包括文件的信息和自己的临时端口，接收方即刻主动的连接该临时端口，建立数据连接。

主服务器和客户的文件传输、客户和客户的文件传输、客户在群组中的上传和下载都是采取这种类 FTP 协议的模式。

在客户向服务器发出 3 中类型的文件发送请求时（包括上传、更新、下载），服务器需要有不同的响应处理。

四、 项目代码

（一）代码结构简述:

项目代码包括两个部分，一个是服务器端的代码，另一个是客户端的代码。

只有服务器直接访问数据库和存储区，几乎所有的客户操作都先必须向服务器发送请求才能完成。

在服务器端逻辑区包括四个文件，分别是 S_ReceiveFileThread,S_SendFileThread,Server 和 User。前面两个是用来实现服务器的接收文件线程以及发送文件线程的。后面的 User 用来保存用户信息包括 IP、昵称和状态等的。主要的结构逻辑在 Server 文件中，它同时也会调用这三个文件来实现服务器的相关功能，后面会详细地解释。

在客户端逻辑区也包括四个文件，分别是 C_ReceiveFileThread,C_SendFileThread,Main 和 User。同上面的服务器端的基本类似，有一个接收文件线程和一个发送文件的线程，以及获取用户信息的代码以及一个核心的 CLient 客户端文件实现客户端的相关功能。

此外，在 UI 部分，服务器采用 swing 框架，只提供一个相对简单的页面；客户端采用 JAVAFX 构造用户页面，界面组件较为丰富，且使用 CSS 文件修改样式。

目录结构如图 4-1-1 和图 4-1-2

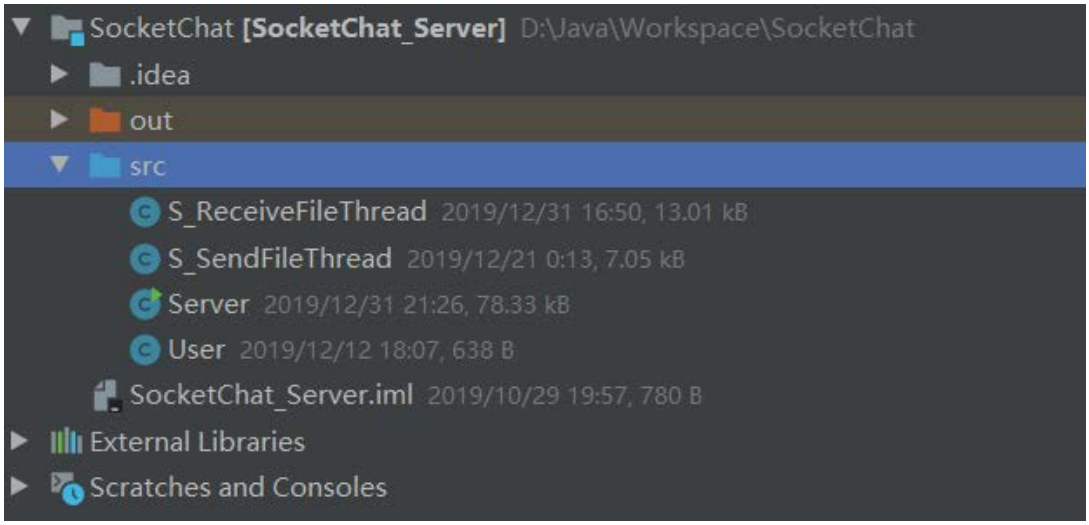


图 4-0-1 服务器

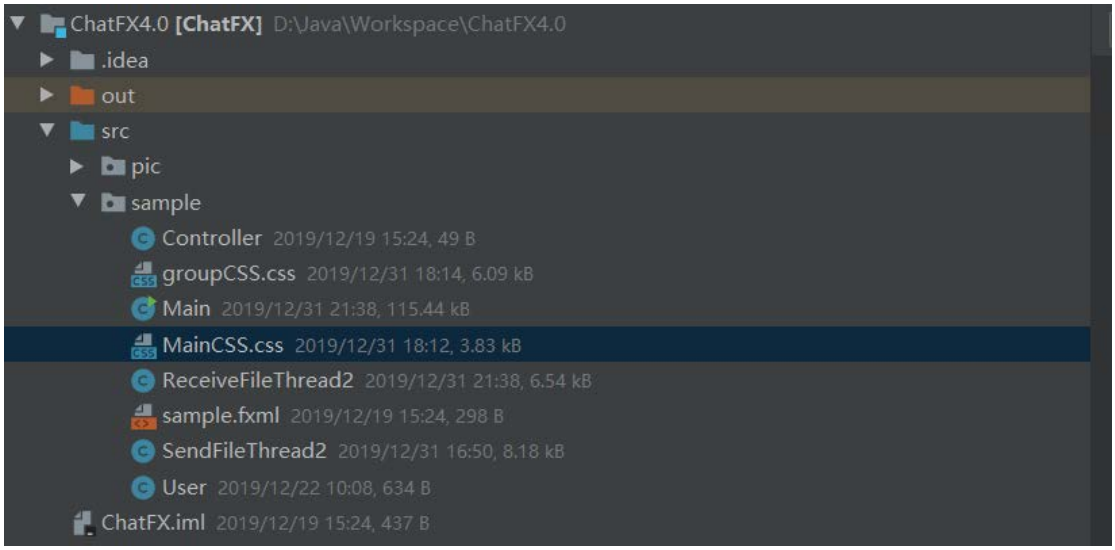


图 4-0-2 客户端

（二）服务器核心代码：Server.java

作为一个中央服务器，开启固定端口 6666 监听客户连接请求，并开始服务。

为每个客户建立“C/S 主命令连接”，客户所有的操作请求命令都将通过自己的这条“主连接”来告知服务器。

客户发来的请求命令如图表 4-2-1：

用户命令	命令头	附属信息
断开连接	CLOSE	
注册	USERZHUCE	昵称+密码+邮箱
登录	USERLOGIN	昵称+密码+P2P 端口+IP
忘记密码	USERFORGET	昵称+邮箱+新密码
保存离线消息	LIXIAN	发送方+接收方+信息
请求 P2P 用户信息	P2P	P2P 对象
向服务器传输文件	FILE_P2P	文件传输端口+文件名+文件大小+IP+昵称
创建群组	CREATE_GROUP	组 ID+组名+群主+成员 ID 表+
打开群组	OPEN_GROUP	组 ID
上传群文件(add)	FILE_G_ADD_NEW	组 ID+文件传输端口+文件大小+IP+昵称
上传群文件(push)	FILE_G_PUSH	组 ID+文件传输端口+文件大小+IP+昵称
下载群文件	FILE_G_DOWNLOAD	组 ID+昵称+文件大小
发送群消息	GROUP_MESSAGE	组 ID+昵称+消息内容
其他	转发广播消息	消息内容

表 4-2-1

对于客户的每一种命令，都有对应的操作。

即中央服务器在收到客户相应的请求后，若成功处理请求（包括处理 3 个数据库和处理本地文件存储等内容），便通过该客户对应的“主连接”，向客户反馈信息。

客户收到的服务器反馈以及其他服务器命令如图表 4-2-2：

服务器命令	命令头	附属信息
服务器关闭	CLOSE	无（所有客户需要强制下线）
新用户上线	ADD	昵称 + IP
新用户下线	DELETE	昵称
用户列表	USERLIST	列表项：昵称+状态
用户 ID 列表	USER_ID_LIST	列表项：昵称+ID
已达人数上限	MAX	无（该客户不允许登录）
群组列表	GROUPLIST	数量+G_next+群组列表（列表项：组名+组 ID)
服务器发文件	FILE_P2P	文件传输端口+文件名+文件大小+IP+昵称
登录反馈	USERLOGIN	登录是否允许+离线消息+离线群组消息
注册反馈	USERZHUCE	注册是否允许
更改密码反馈	USERFORGET	更改是否允许

P2P 服务信息	P2P	查询者的昵称+P2P 端口+IP
创建群组成功	CREATE_GROUP_OK	无（客户可进行下一步动作）
打开群组成功	OPEN_GROUP_OK	无（客户可进行下一步动作）
上传文件成功	FILE_UPLOAD_FINISH	无（客户可进行下一步动作）
允许下载文件	FILE_DOWNLOAD_OK	文件传输端口+文件名+文件大小+IP
群组信息	GROUP_MESSAGE	群组信息内容

表 4-2-2

```
/*----------主要服务框架----------*/

/** 启动服务器 ...*/
public void serverStart(int max, int port) throws java.net.BindException {...}
/** 关闭服务器 ...*/
/deprecation/
public void closeServer() {...}


/** 服务器线程 ...*/
class ServerThread extends Thread {...}
/** 为一个客户端服务的线程 ...*/
class ClientThread extends Thread {...}
```

图 4-1-1

```
/** 为一个客户端服务的线程 ...*/
class ClientThread extends Thread {
    private Socket socket;
    private BufferedReader reader;
    private PrintWriter writer;
    private User user;
    /deprecation/
    public void run() {// 不断接收客户端的消息，进行处理。
        String message = null;
        while (true) {
            try {
                message = reader.readLine();// 接收客户端消息
                System.out.println(message);
                StringTokenizer stringTokenizer = new StringTokenizer(
                        message,  delim: "#/");
                String command = stringTokenizer.nextToken();// 命令
                if (command.equals("CLOSE")) {...}
                else if (command.equals("USERLOGIN")) {...}
                else if (command.equals("USERZHUCE")) {...}
                else if (command.equals("USERFORGET")) {...}
                else if(command.equals("LIXIAN")) {...}
                //可能待改
                else if (command.equals("P2P")) {...}
                else if(command.equals("FILE_P2P")){...} //点对点传文件（这里是C/S的P2P）
```

```
else if(command.equals("CREATE_GROUP")){...}
else if(command.equals("OPEN_GROUP")){...}

else if(command.equals("FILE_G_ADD_NEW")){...}
else if(command.equals("FILE_G_PUSH")){...}
else if(command.equals("FILE_G_DOWNLOAD")){...}
else if(command.equals("GROUP_MESSAGE")){...}
else {...}//初始连接消息
```

图 4-1-2

这些操作的辅助函数如下

```
/*----------------------处理用户 登录页面请求-----------------------------*/

/** 找回密码模块 ...*/
public int user_forget(String username, String mail, String new_password) {...}

/** 注册模块 ...*/
public int user_register(String username, String password, String mail) {...}

/** 当有用户下线时，在服务器改变状态 ...*/
public int user_offLine(String name) {...}

/** 登陆模块 ...*/
//12_16
public int user_login(String username, String password,int serverPort,String myIP) {...}
/**利用MD5进行加密 ...*/
public String EncoderByMd5(String str) throws NoSuchAlgorithmException, UnsupportedEncodingException {...}
/**判断用户密码是否正确 ...*/
public boolean checkpassword(String newpasswd,String oldpasswd) throws NoSuchAlgorithmException, UnsupportedEncodingException{...}
```

图 4-1-3

```
/*-----------------------处理用户 主页面请求-------------------------------*/

/** 获取指定群组的信息 ...*/
public static String open_group_for_info(int c_gid) {...}
/** 上传 新的文件 ...*/
public int add_new(String msg){...}
/** 上传 某一文件的新的版本 ...*/
public int push(String msg){...}
/** 下载 指定文件 ...*/

/** 下载 全部文件 ...*/
```

图 4-1-4

```
/*------------------------处理用户 群聊页面请求------------------------------*/
/** 用户不在线时，保存 群组 的离线消息 ...*/
//待改12_16
public int G_send_messageTo_board(String send_from, String send_for , int gid, String message) {...}
/** 获取指定群组的信息 ...*/
public static String open_group_for_info(int c_gid) {...}
/** 上传 新的文件 ...*/
public int add_new(String msg){...}
/** 上传 某一文件的新的版本 ...*/
public int push(String msg){...}
/** 下载 指定文件 ...*/

/** 下载 全部文件 ...*/
```

图 4-1-5

（三）客户机核心代码（Main.java）

客户部分主要是根据三个页面分别展开结构部署，包括登录页面、主页面和群聊页面。主要的命令请求都是通过和服务器建立的主连接完成，如图 4-3-1，具体命令表 4-1-1 和 4-1-2 已经说明，不再赘述。

如图 4-3-2 所示，登录页面的登录注册请求通过“主连接”发送到服务器后，经过服务器的反馈，若允许登录，则通过反馈信息来初始化主页面。

如图 4-3-3 所示，主页面中，提供 P2P 聊天和 P2P 文件传输功能（不选择指定用户时可实现广播），创建群聊和打开群聊功能。其中，对于 P2P 模块，客户端既运行 P2P 服务，又要运行 P2P 客户端代码，如图 4-3-4 所示。

如图 4-3-4 所示为群聊页面。通过主页面中的指定群聊，经主服务器反馈后，可打开群聊，完成群聊信息的初始化。群聊页面也包括两个模块，分别是群文件模块和群聊模块。

```
//发送消息（向服务器发消息）
public void sendMessage(String message) {...}

// 不断接收消息的线程（服务器发来的消息）
class MessageThread extends Thread {...}

//连接服务器
public boolean connectServer(int port, String Ip_of_server, String name) {...}
// 客户端主动关闭连接
/deprecation/
public synchronized boolean closeConnection() {...}
```

图 4-3-1 主客户线程

```
//-----------------------登录 页面-------------------------//
public void LoginPage() {...}
```

图 4-3-2 登录页面

```
//----------------------聊天 主页面-------------------------//

public void MainPage() {...}
//private final Desktop desktop = Desktop.getDesktop();
//发送文件给选定的用户 （或者 给服务器，可能待改）
public void sendFile() {...}

//执行文本信息
public void send() {...}

//创建群聊 --->tell server
public void create_group() {...}

//打开群聊 --->tell server
public void open_group() {...}
```

图 4-3-3 主页面

```
//----------------------点对点连接（我作为客户端）主动连接----------------------------------//

/** 连接p2p ...*/
public boolean connectServer_p2p(int port, String hostIp, String name) {...}

// 不断接收p2p消息的线程
class MessageThread_P2P extends Thread {...}


//----------------------点对点连接（我作为服务端）被动监听----------------------------------//

//为另一个主动链接的客户端提供服务的线程
class ClientThread extends Thread {...}

//服务线程
class ServerThread extends Thread {...}

//启动服务器（p2p）
public void serverStart(int port) throws java.net.BindException {...}

/** 关闭服务 ...*/
/deprecation/
public void closeServer() {...}
```

图 4-3-4 P2P 模块

```
//-----------------------------群组页面-------------------------------------
private void GroupPage_Left(){...}
private void GroupPage_Right(){...}
public void Q_GroupPage() {...}

class singleFilePane extends GridPane{...}

// 自定义列表中已经实现
public void group_init(String msg){...}

public boolean updateFileBox(){...}
//有待开发一键上传整个文件夹的功能（或整个项目）@@@@@@@@12_19
/*...*/
//群组发消息
public void G_send(){...}
File file_to_upload;
public void G_upload(){...}

public void G_push(){...}
public void G_download(){...}
```

图 4-3-5 群组页面

（四）其他辅助文件

除了 CSS 文件，和用户类，其余主要是提供文件的发送和接受功能的辅助类，解释从略。

本项目的文件辅助类由于多次修改，多次增添模块，较为复杂，只是需要了解，文件的发送都是采用类 FTP 的模式，发送方先建立一个随机端口，通过控制连接告诉接收方，接收方再主动的连接发送方，发送方随机发送文件。

此外，文件的发送涉及到 4 种主要的模式，一个是基础的 P2P 传文件，一个是新文件的传送，一个是新版本的传送，一个是响应下载请求。

所以客户端的文件发送需要区分 3 种 MODE，服务端的文件发送需要区分 2 种 MODE。

与此相对应，接受文件也是分 MODE 进行的。

五、 项目演示

（1）数据库后台：

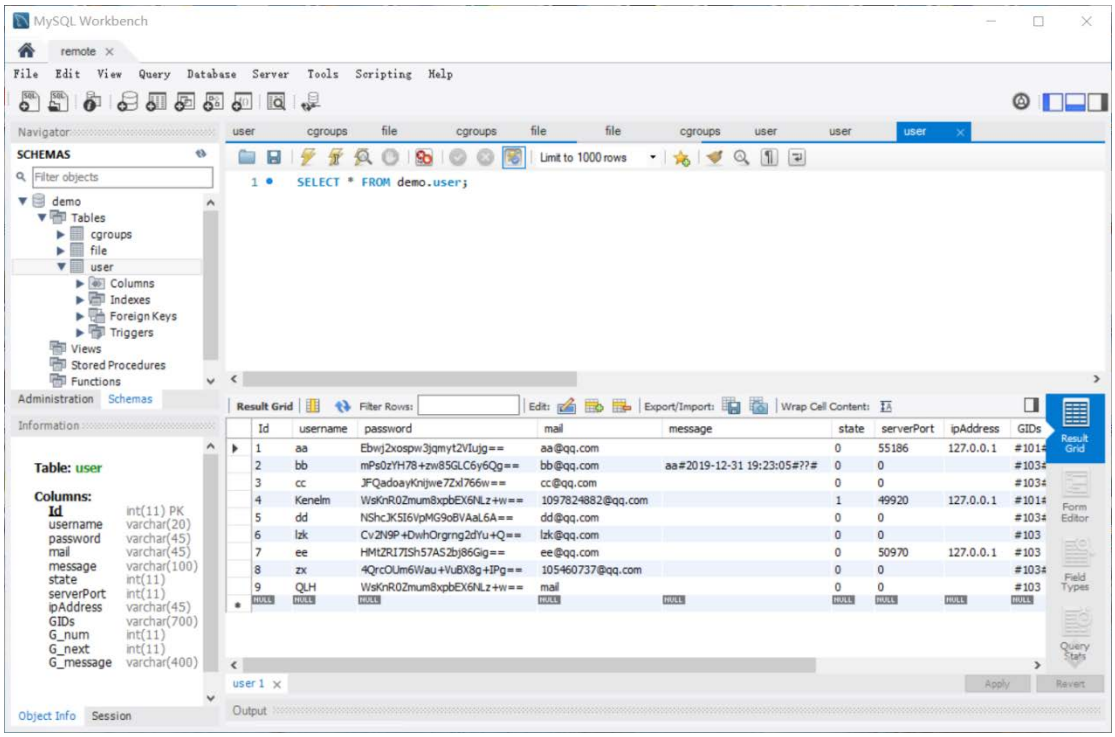


图 5-1-1 数据库

（2）服务端：

服务器界面

图 5-2-1 服务器界面

服务器存储区

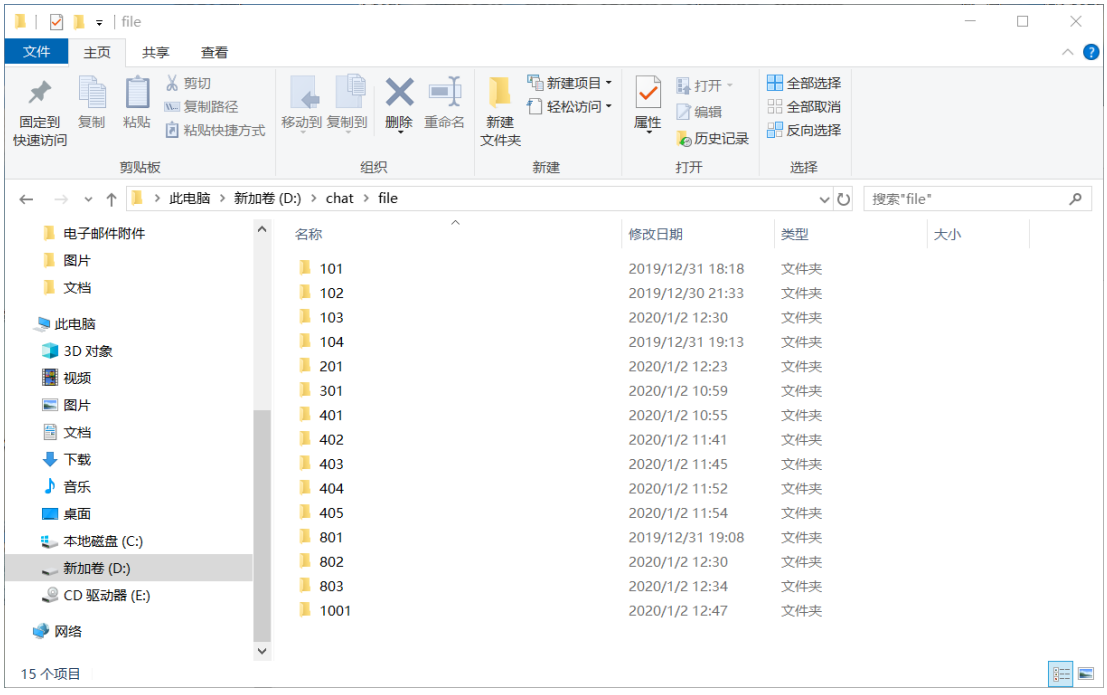


图 5-2-2 服务器文件存储区（每个组 ID 都有对应的文件夹）

（3）客户端:

登录

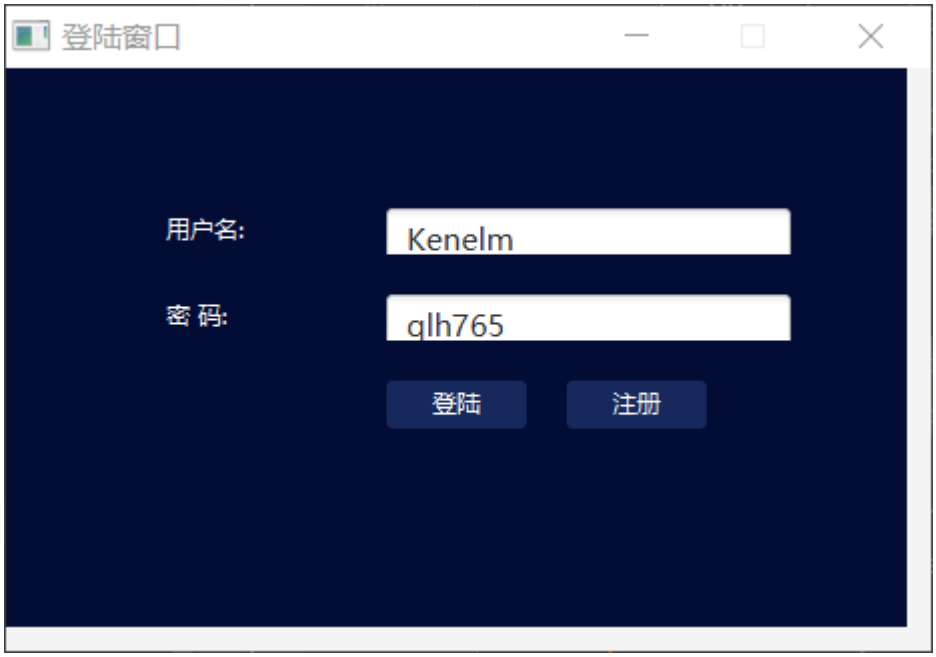


图 5-3-1 登录页面

主页面

图 5-3-2 主页面

文件页面

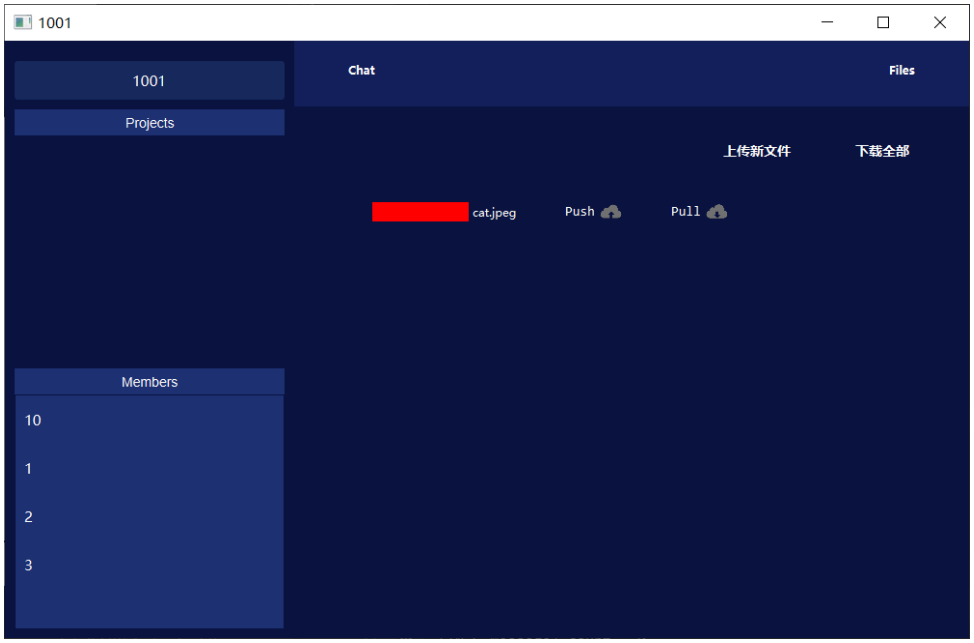


图 5-3-3 文件页面

群聊聊天页面

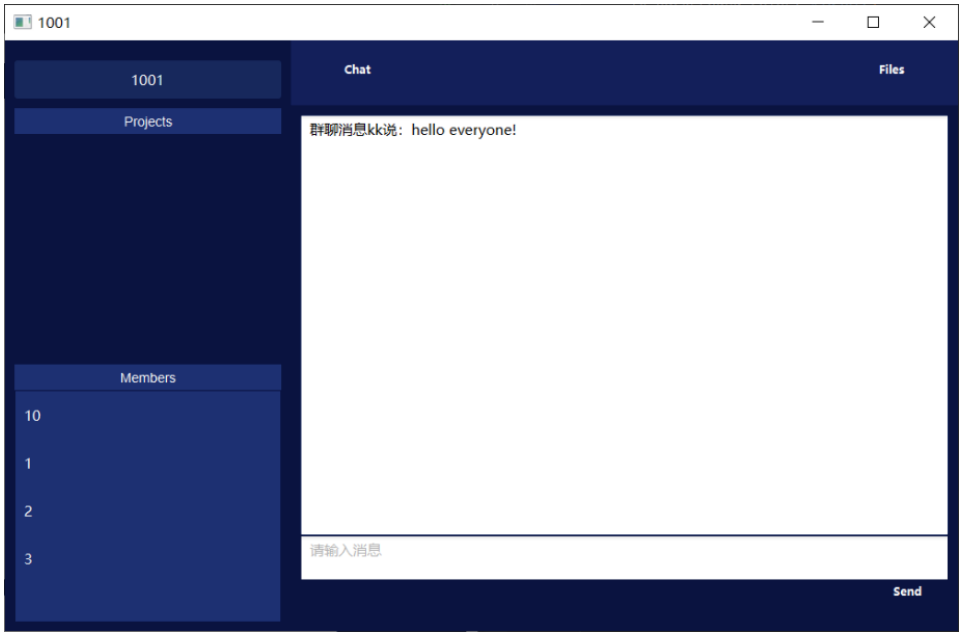


图 5-3-4 群聊页面

六、 反思与不足

首先，在聊天功能上，应该做一些扩充

一方面，目前 P2P 聊天和群组聊天都没有实现保存聊太记录的功能。这可以作为一个扩充模块。

另一方面，由于我们的项目目标是为了促进团队协作，所以在群聊中除了能实现不同的点对点聊天外，还应该添加一个公告模块，用以发布正式任务信息。这些公告信息应该要实现长期存储。

其次，文件管理模块，目前只能实现群组下载群组文件、上传新文件、以及上传文件的新版本。对于一个文件，数据库和服务器只记录了所有的已上传的版本的信息，客户界面仅仅能显示正式版本的文件。

后续的优化可以从下面几个方面进行：

（1）对于整个文件夹的上传和下载还有待解决

（2）加入文件夹管理，方便一个群聊创建多个项目以及对单个项目进行管理。在设计文件管理的时候，可能要重新考虑文件 ID 的规则，以及数据库的字段设计和服务器的存储结构设计。

（3）客户可以以交互式的方式查看当面文件的历史版本、正式版本、新的待处理版本信息。管理员可以对项目文件更新正式版本。

此外，在用户界面上，还有优化的空间，后期可以继续适当的美化界面，增加一些利于用户交互的动画，如文件传输进度信息、文件夹折叠等等。

最后，在代码上，为了更高的用户体验，服务器和客户端代码都需要进一步做性能优化。如相同的数据库查询避免多次查询，优化数据库字段的规则，提高查询效率；以及Aplication 线程的涉及到的多线程冲突；目前在界面交互上也存在逻辑问题。

补充说明：

1. 登录模块界面上没有显示出来，服务器的设置信息以及忘记密码的部分，但在代码中有实现，以后优化 UI 再去补充

2. 本次项目中的群聊名称还没有实现用户指定，暂时用 GID 代替了

七、 团队分工

覃龙虎：项目总架构、代码结构、数据库的设计；P2P 的通信设计，以及群聊的大部分功能和文件传输类的设计（包括客户端服务器两部分）

赵轩：主要负责登录页面、主页面、群聊页面以及服务器的 UI 部分，包括 CSS 样式优化和部分逻辑（为后端提供的操作信息）。

李展昆：登录注册页面的逻辑，部分主页面逻辑，包括主页面和群聊页面的信息初始化模块。课题项目报告和 PPT 的整理。

备注：本项目代码较为庞大，许多分工都有交叉。

八、 附录

见用户手册