**总体通讯架构以及程序架构说明书**

**尽量要求一个小组可以在一起写代码，采用结对编程的！**

**网络通讯：**

通讯模块，对命名进行过滤和确认，并调用相关的处理模块

1.客户端的Listener:

class Listener{

tcplistener:

//....初始化....

//开始等待请求

//请求到来...

//解析一下命名

switch(命令){

case:

//do something

//return to client

case:

//do something

}

}

2.客户端的Client：作为基类

class client{

client:

string msg="request:login......" request+usrID+data

sendmessage(void\* buf){

}

}

每一个Client功能类都继承自Client类

class loginClient:client{

string msg="request:login......"

/.....

////

sendmessage();

//recieve; do something }

}

3.服务器端的Listener:

class Listener{

tcplistener:

//....初始化....

//开始等待请求

//请求到来...

//解析一下命名

switch(命令){

case:

//do something login(data);

//return to client

case:

//do something

}

}

1\r\n

userid\r\n

data....\r\n

服务器加上Client;

**备注：**

**a request ...server**

**b request ...server**

1.为实现服务器和客户端的通讯，以及为了避免服务器的进程浪费，因此，通过客户端的一个线程向服务器请求轮询来实现服务器和客户端的通讯。

client....

2.关于双人之间的消息，为了一个人可以在一个时间内发多条消息，采取发消息仅仅为client请求的模式，若是发送消息，即为client。若是接受消息着，则是server,一个人与多人对话采用多线程来实现。

3.UI的显示按照委托和事件模式来完成，当通讯消息需要UI应变，则会触发一个事件的发生。相关的处理者进行相应（处理者一般指UI）。

4.服务器一段要建立一个缓冲机制，用于实现将其他客户的动态变化发送给特定用户。

**命令集：**

**（1）客户端可能发送的请求命令集**

1.登录（给服务器）

2.注册（给服务器）

3.发送单人消息（给服务器发消息时的请求命令）

4.发送单人消息（给另一个客户端发消息时的请求命令）

5.加入群组（给服务器）

6.创建群组（给服务器）

7.添加好友（给服务器）

8.搜索好友（给服务器）

8.删除好友（给服务器）

9,注销、或者程序退出（退到用户登录界面）（给服务器）

10.修改信息（给服务器）

11.密码验证（给服务器）

12.发送群组消息（给服务器）

13.请求发送文件（给服务器）

14.请求发送文件（给客户端）

15.退出群组（）

**（2）服务器端的回复命令集（向客户端发送）**

1.登录：身份信息不正确，或者身份信息正确；

2.注册：注册成功，注册失败...

3.发送单人消息：1.另一个客户在线,2另一个客户不在线。

4.发送单人消息：1.同意建立连接，2.不同意建立连接；

5.加入群组：加入群组成功；失败

6.创建群组：成功失败

7.添加好友：同意、拒绝

8.删除好友：成功、失败

9.搜索好友：返回好友链表...

10.好友不在线：返回好友不在线

11.好友拒绝添加：返回拒绝添加

12.有好友添加请求

13.有离线消息

14.添加好友

**(3)服务器发送命令集**

1.好友上线提醒（给线上的人）

2.群组消息提醒（给相关的人）

3.系统消息(给相关的人：离线或者在线)

**问题记录：**

如果不经过服务器，在非指定ip情况下交流是不可能的事情。