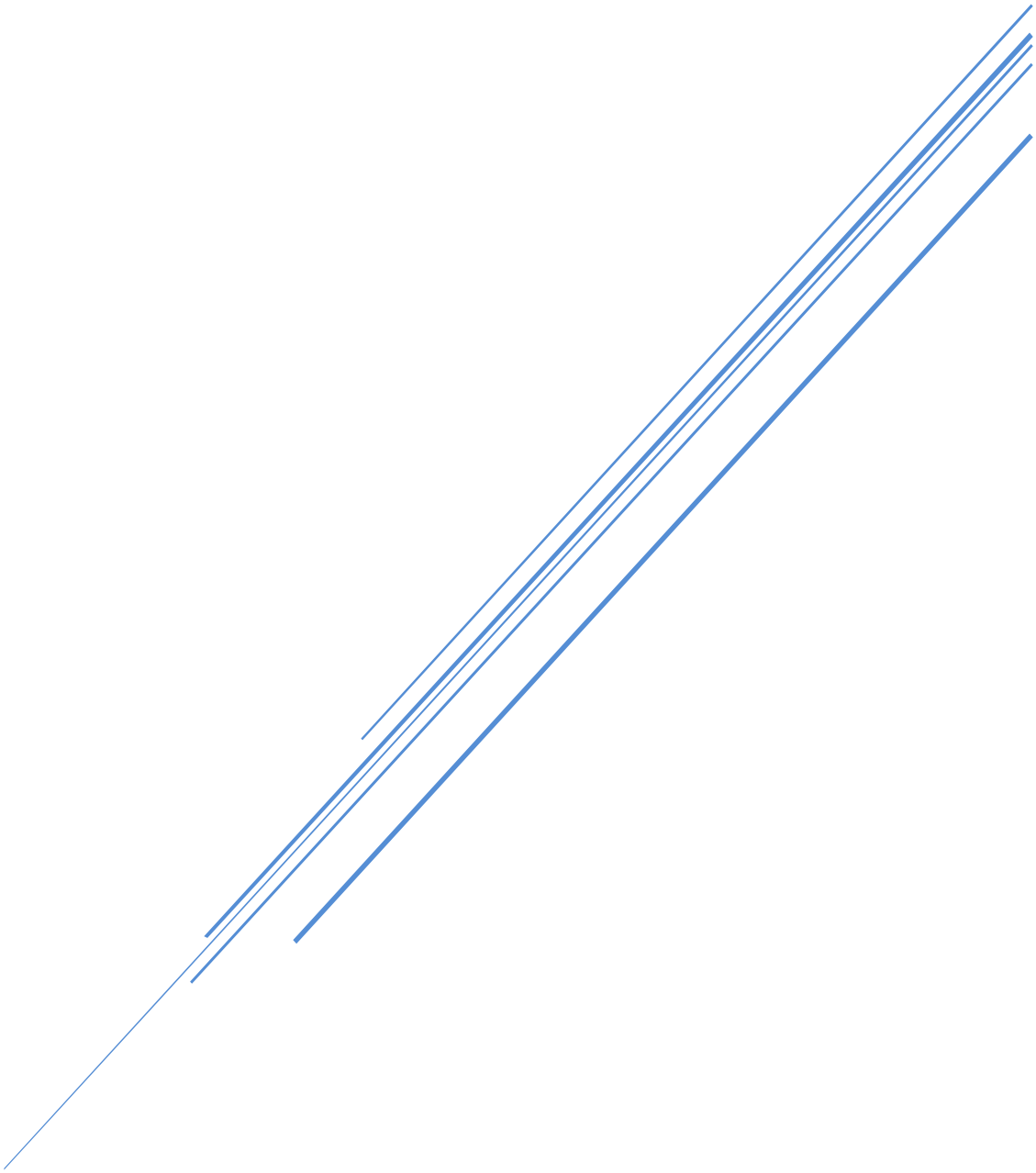
NIHAO SOFTWARE

NIHAO通信软件说明文档

兰天、郭嘉琰、李辉超、沈翰文



Linux系统编程课程大作业

NIHAO通信系统

项目报告书

目录

一、项目简介

本项目是由NIHAO　Software完全使用C语言开发的Linux操作系统下的基于CS架构的即时通信软件，目前的版本号为1.0.0。截止目前，已经实现的基础功能有私聊、群聊、收发文件、通信内容备份以及文件传输；实现的扩展功能有登录、注册、好友查询、好友添加、匿名聊天、改变字体样式等。除此之外，本项目仿照Mac下的QQ的界面进行了界面设计，界面简洁美观，使用体验良好。本项目适合作为团队通信软件来使用，方便团队协作。

二、设计方案

1. 设计方法

1.1. 服务器设计

小组定义服务器的是和客户端的长连接方式通讯，长连接保证了服务器和客户端之间首先建立通讯，然后再进行报文的发送和接收，长连接可以有效的保证和客户端之间的点对点通讯，在其他的服务器设计中，长连接经常用于数据和消息传输使用。

我们小组设计的网络服务器中，采用一个线程服务一个客户端的非阻塞I/O模式实现服务器和客户端之间的信息交互。另一方面，服务器和数据库之间使用的是短连接，因为用户的操作中，对数据库的访问并不是最频繁的操作，因此使用短连接相对比较节省性能。

1.2. 客户端设计

客户端设计中主要分成两个部分，一个部分是前端的页面元素显示，另一个部分就是和数据库的接口的交互。小组在Linux下使用GTK进行界面的可视化开发，并在后端使用Linux基本的socket系统调用建立和服务器之间的连接，实现数据的传输和接收。

1.3. UI设计

UI设计部分我们使用Linux的图形界面库GTK编写，界面仿照了Mac OS版的QQ。这种风格简洁美观，适合办公、生活使用。主界面设计图如下所示：

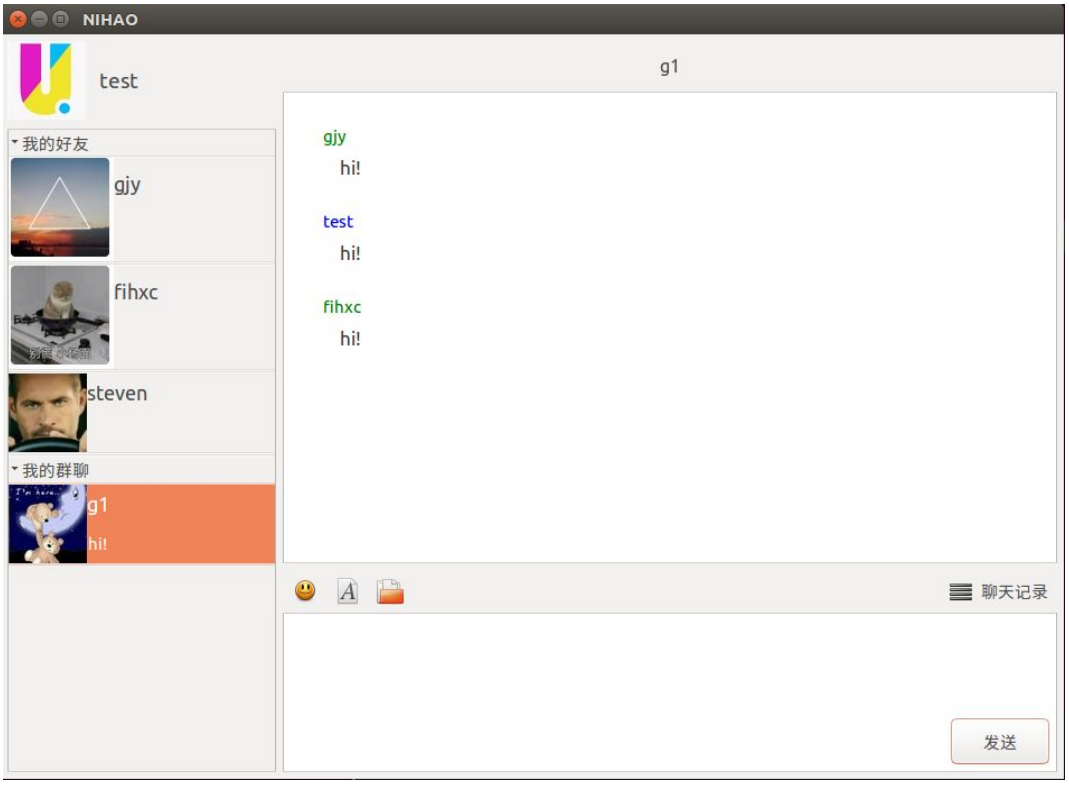
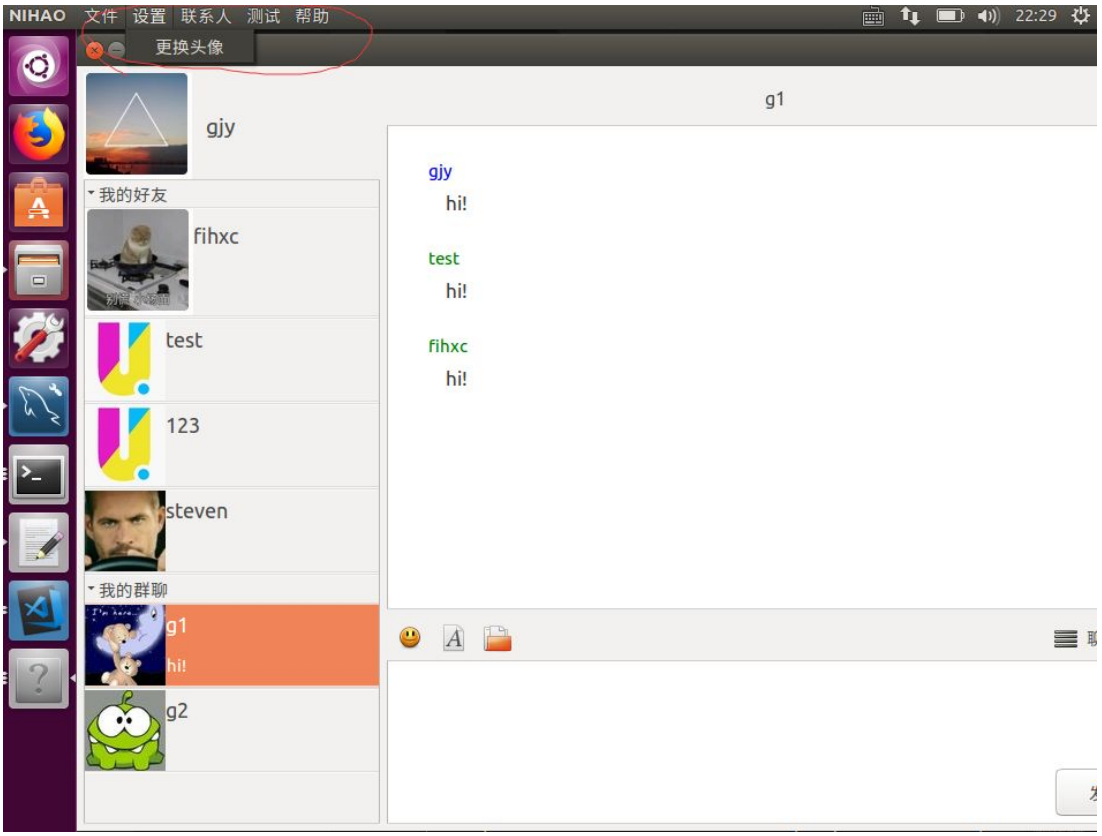


图1: UI界面

主界面左上方是用户自己的个人头像和昵称，下方接着的是好友列表和群聊列表，右侧主体部分是聊天窗口，其中下方是发送栏，上方是消息显示栏。两栏之间还有几个图标分别表示“发送表情”、“调整字体”、“发送文件”和“聊天记录”。

我们的主界面还有一个菜单栏，有“文件”、“设置”等选项如下：



除此以外我们的登录和注册界面等也将在下面展示，此处不做赘述。

2. 设计架构

2.1. 整体设计架构

整体的设计架构采用CS架构，即一种星型结构，客户端首先将不同类型的消息按照一定规则进行编码，然后统一发送给服务器；服务器将收到的消息进行解码然后转发给相应客户端。具体实现时客户端创建后将自动和Server进行连接，Server在接受到客户端的链接请求后将自动创建一个线程来处理与客户端之间的通信。该线程接受客户端通信并将内容进行转发。客户端收到服务器转发过来的消息后进行相应的操作。具体的架构模型如下：

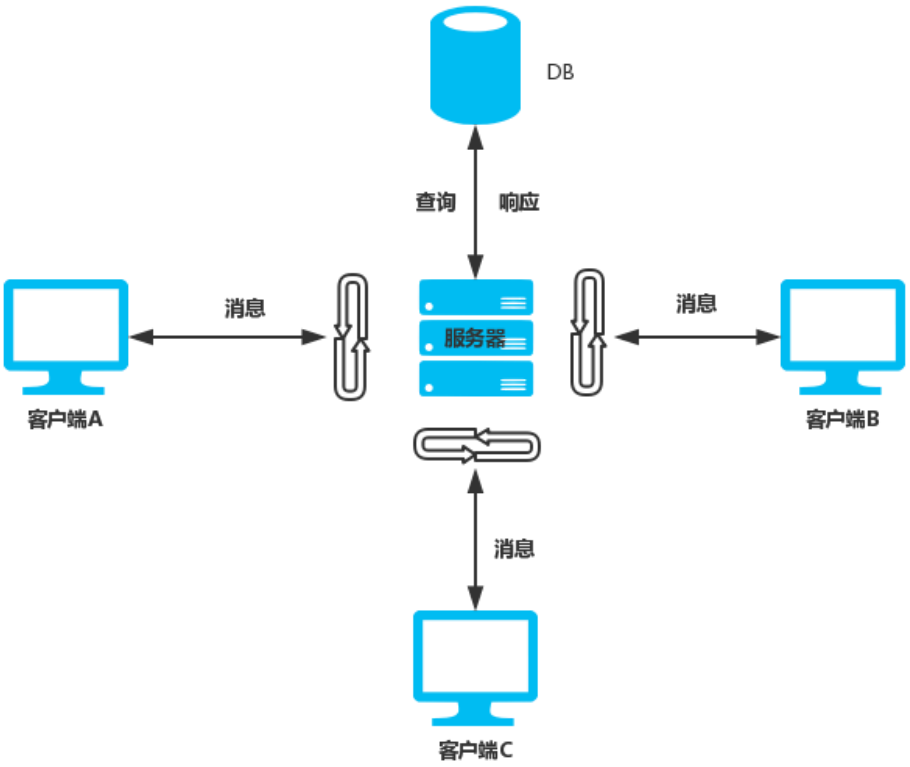


图2: 整体设计架构

2.2. 服务器端架构

服务器端的设计架构采用主线程监听，子线程响应请求的方式，首先主线程循环监听各个端口是否有客户端链接的请求，如果有客户端链接的请求则开启一个子线程来专门处理该客户端的各种请求，主线程继续监听来等待其他客户端的链接。如此类推。其中分裂出的子线程在客户端没有新请求发出时同样处于循环状态，而一旦接受到客户端的请求，则调用函数来进行响应请求。底层架构的设计如下：

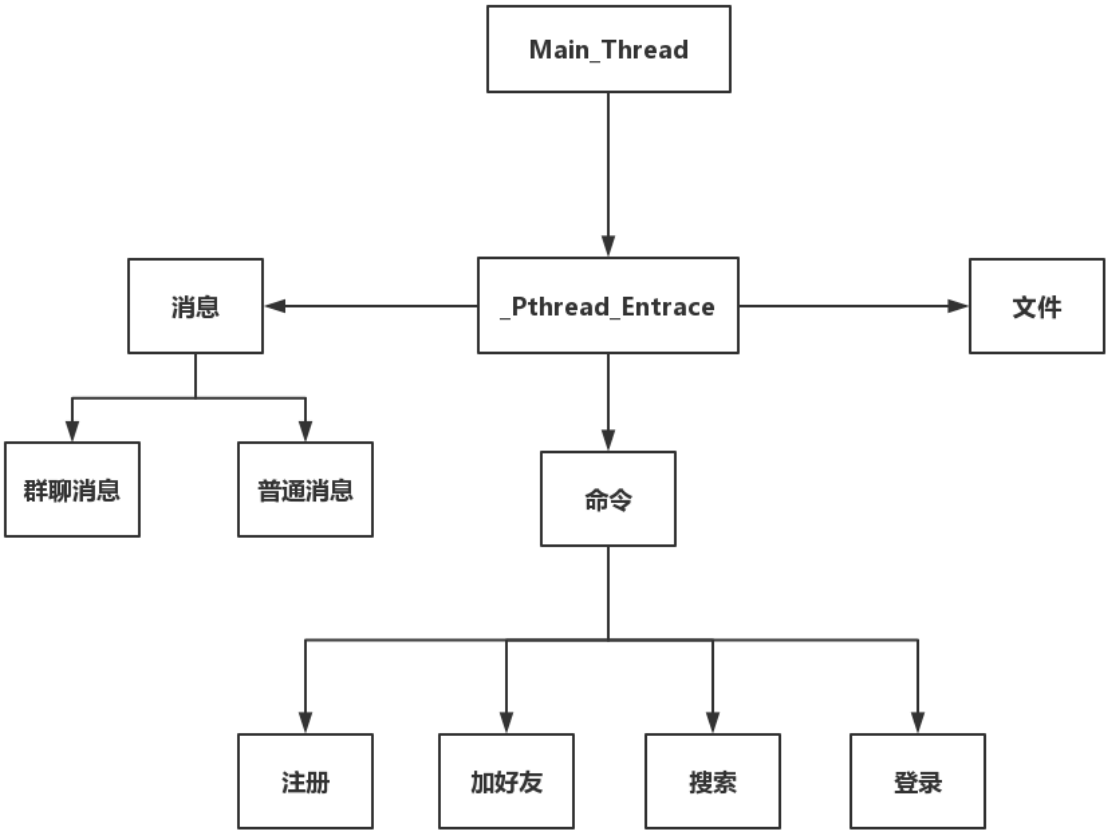


图3: 服务器端架构

整个服务器端将所有的消息类型分为三个大类，七个小类。三个大类分别为消息、文件和命令。其中消息分为群聊消息和普通消息，通过不同的方式进行处理；文件则作为一个统一的消息形式进行处理并转发；而不涉及到转发的命令部分则直接在服务器端进行处理。数据库方面的应用在各个子模块中都有涉及，主要封装成六个接口函数：

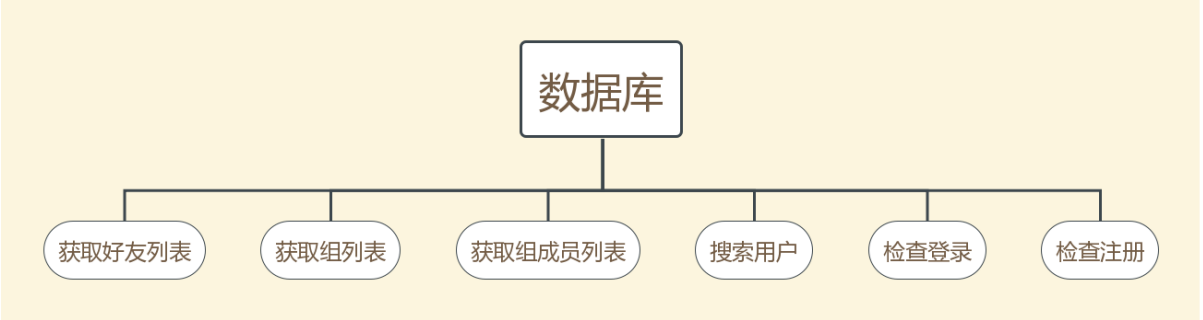


图4: 数据库接口的设计

2.3. 客户端架构

客户端部分采用MCV模型开发，将客户端底层和界面分开进行设计，并使用控制器进行组合。客户端的启动过程主要包括向服务器连接，登录、从服务器获取账号信息，初始化界面，初始化好友列表和组列表等。初始化完成之后，客户端实际上运行着两个线程，主线程响应用户的各种操作，并根据操作调用服务器底层的函数进行处理；子线程接受服务器端的信息，解码信息并实时反馈给客户端的界面。二者紧密结合，从而实现信息的实时响应。子线程根据收到消息类型的不同调用上层界面部分的接口来控制界面的显示，比如收到好友列表信息的改变、收到某人发送的消息以及收到文件等。这部分实现了由底层到界面的转换。整个客户端的架构如图：

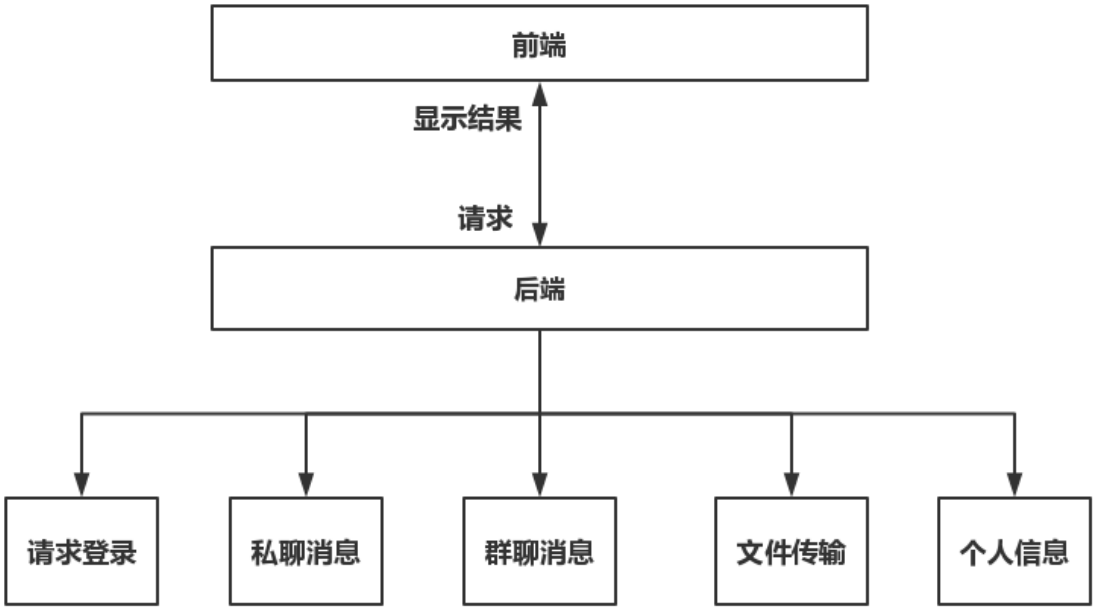


图5: 客户端架构

三、基本功能

1. 私聊

作为聊天软件最基本的功能，私聊功能在本软件中的体验是十分稳定可靠的。首先我们开启服务器，并打开两个客户端，并用提前注册好的账号进行登录，然后选中对方，进行消息的发送。这里我们用gjy和fihxc的通话进行演示。效果如图：

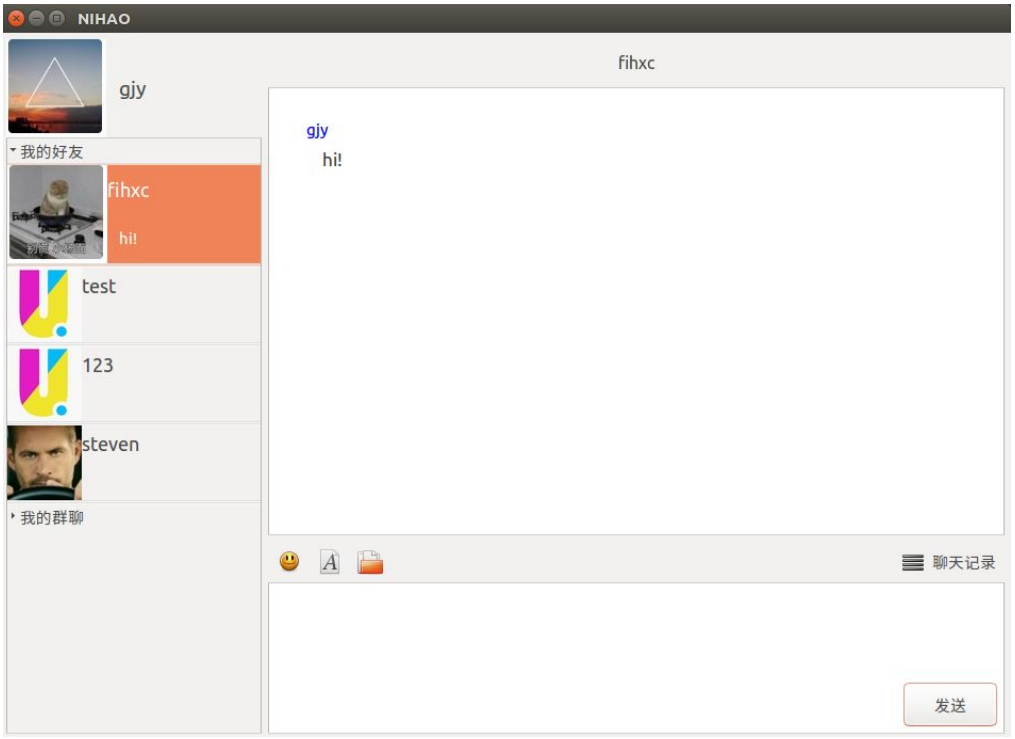


图6: 私聊效果图（1）

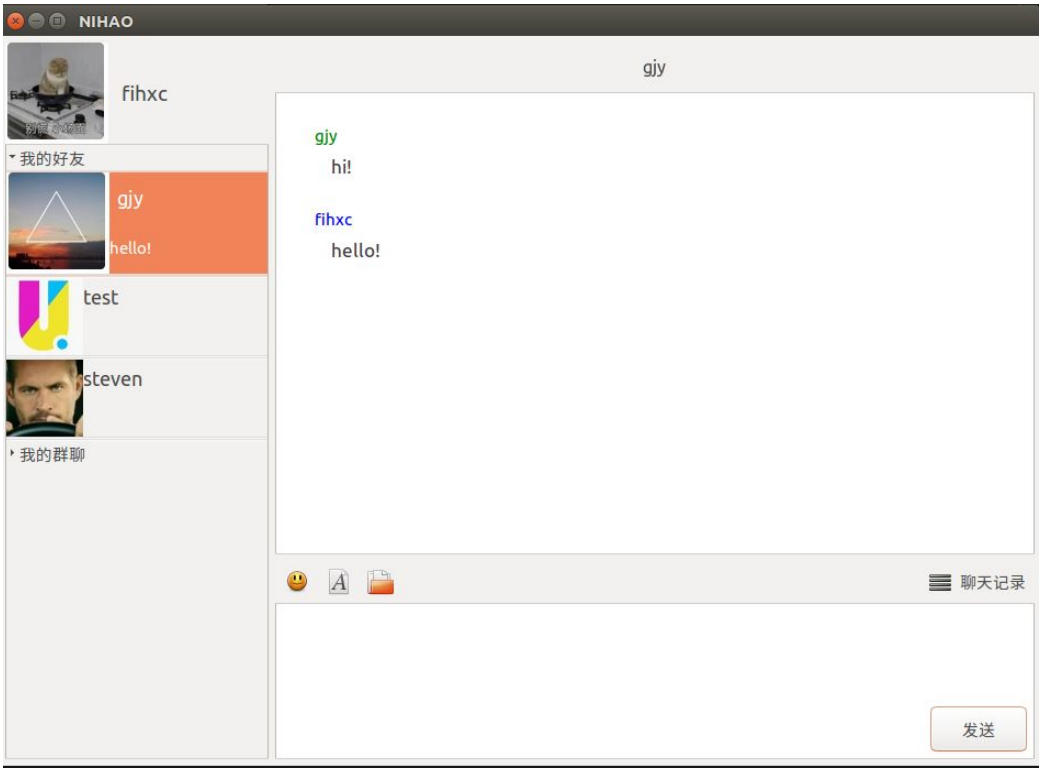


图7: 私聊效果图（2）

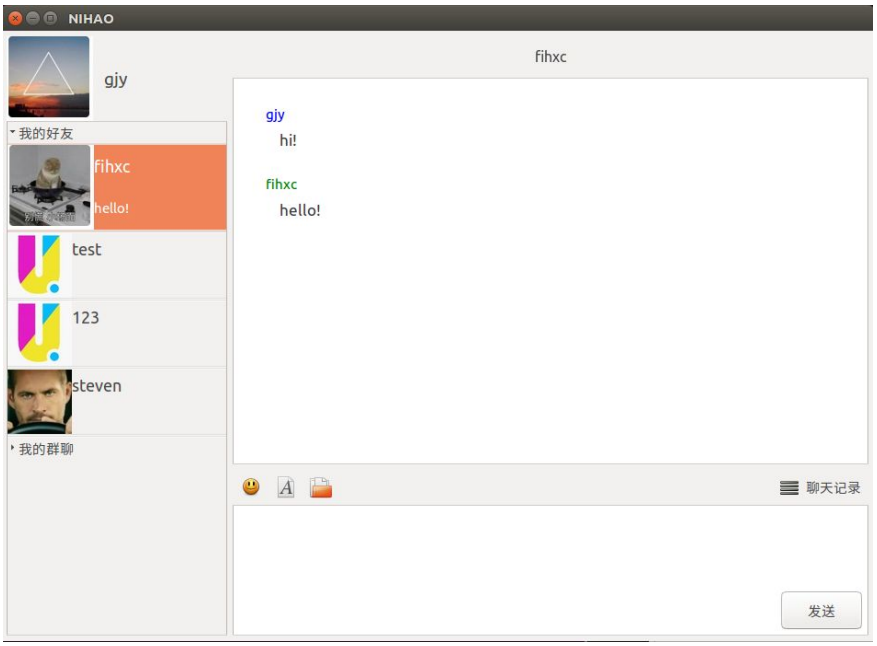


图8: 私聊效果图（3）

可见看到我们的私聊是全双工的，且仔细观察还能发现用户名颜色的不同，在自己的界面看自己的用户名是蓝色的，他人的用户名是绿色的。

2. 群聊

群聊部分我们首先打开服务器，并打开三个客户端，进行登录。登录之后使用提前设置好的群g1进行群聊天消息的测试。该群具有群成员fihxc、gjy和test。测试结果如图：

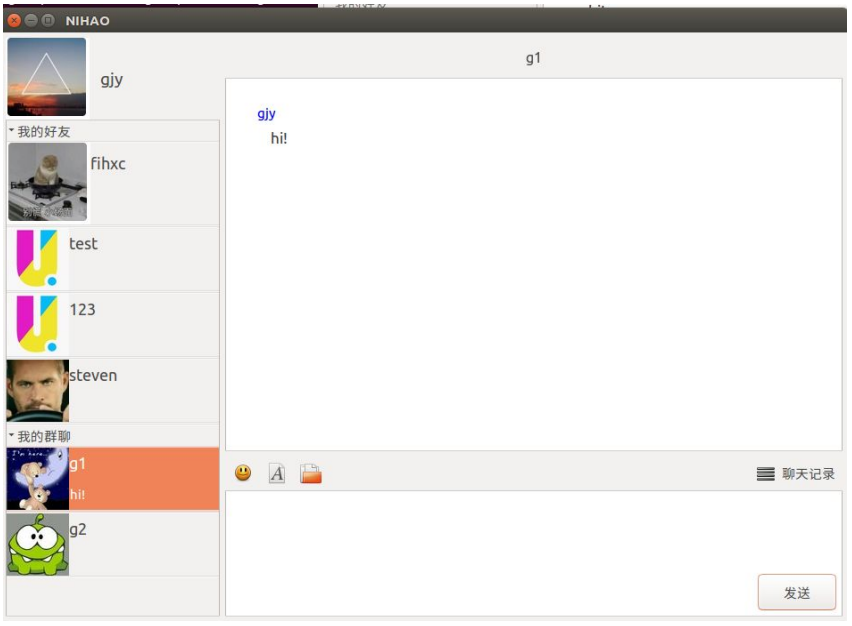


图9: 群聊效果图（1）

先由gjy在群g1里发送消息比如“hi!”然后可以看到fihxc和test都接受到了该消息。

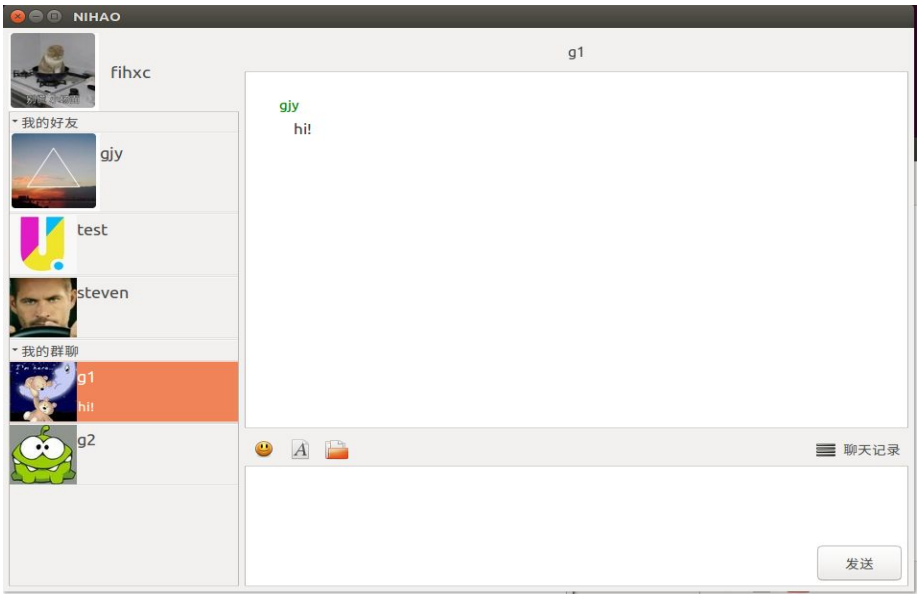


图10: 群聊效果图（2）

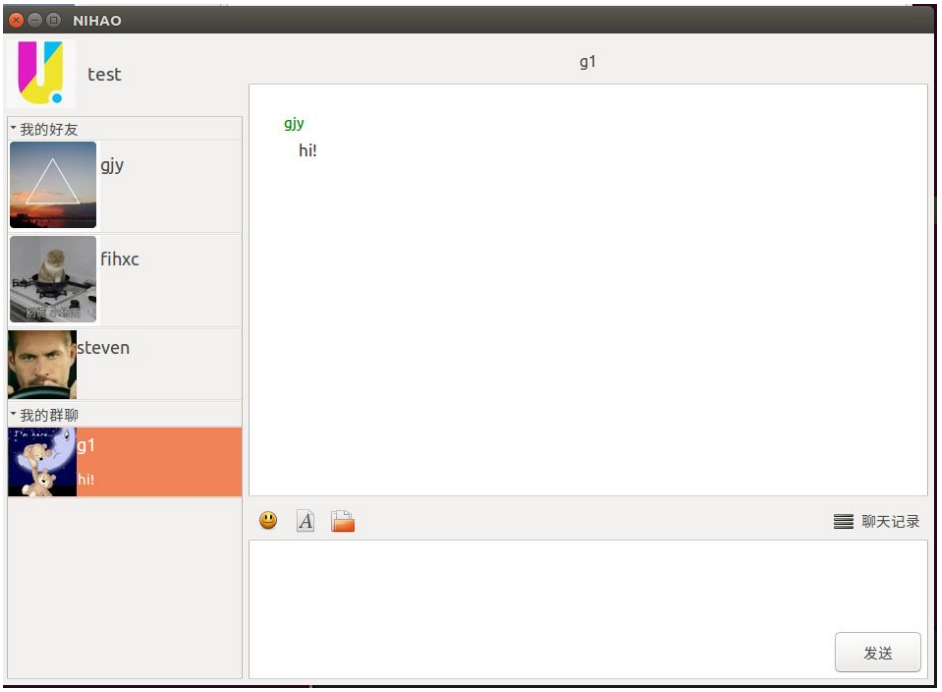


图11: 群聊效果图（3）

下面再由fihxc和test各发一条消息，最终三个账户的显示结果如下三图所示：

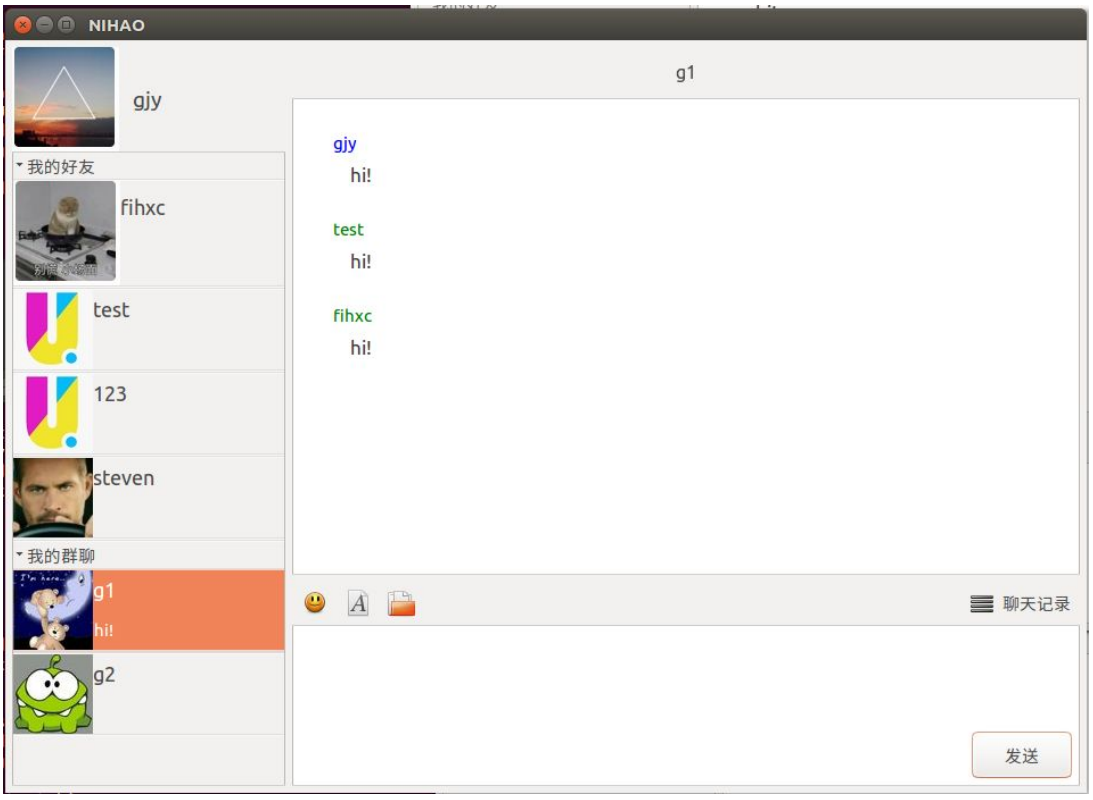


图12: 群聊效果图（4）

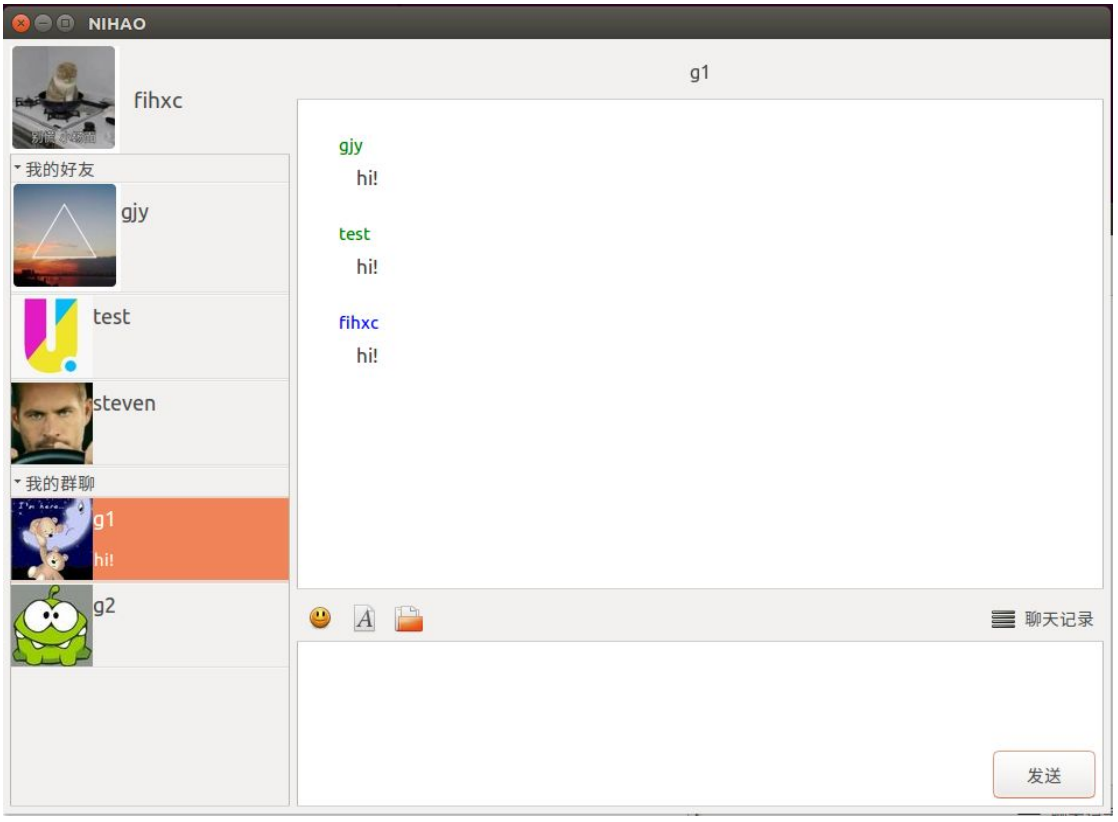


图13: 群聊效果图（4）

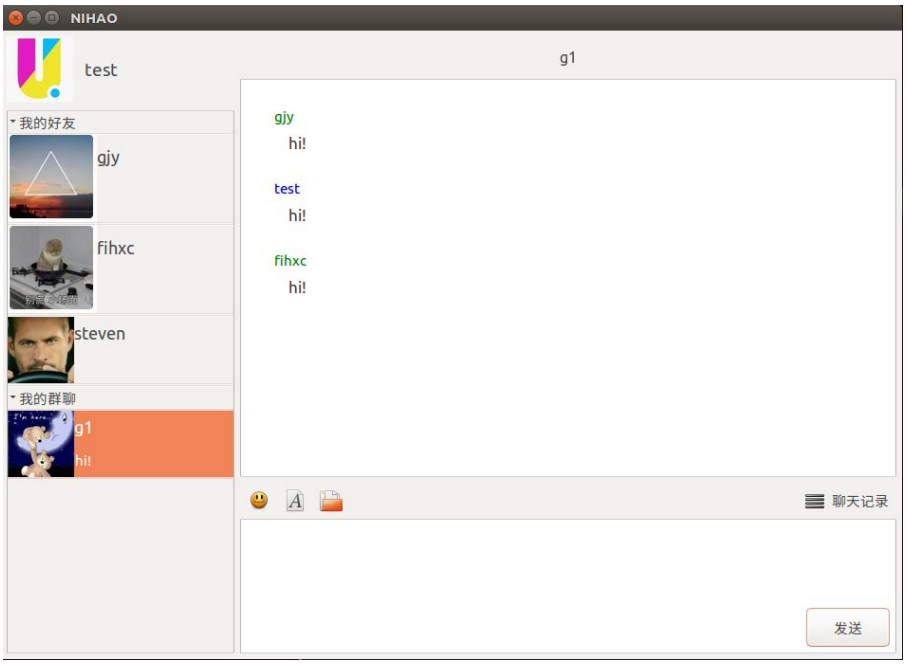


图14: 群聊效果图（5）

可以看到三人都能接收到群聊的消息，且三人聊天界面里自己的用户名都是蓝色显示的，其他人的用户名都是绿色显示的。

3. 文件传输

我们实现了中小型文件的传输功能。只需在主聊天界面选中好友列表中的好友点击像文件的图标，即可进行文件的发送，如下图：

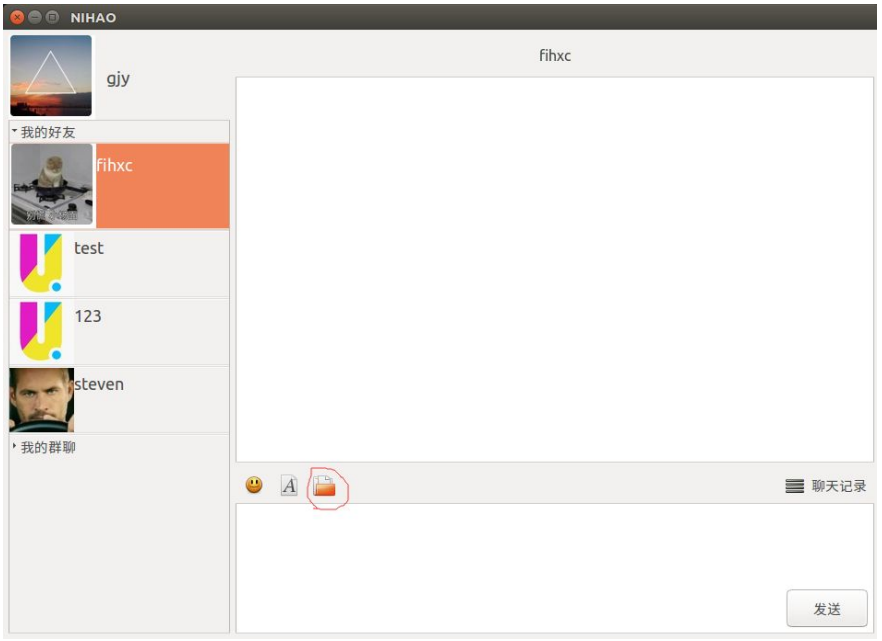


图15: 文件传输效果图（1）

点击文件图标后弹出打开文件对话框，如下图：

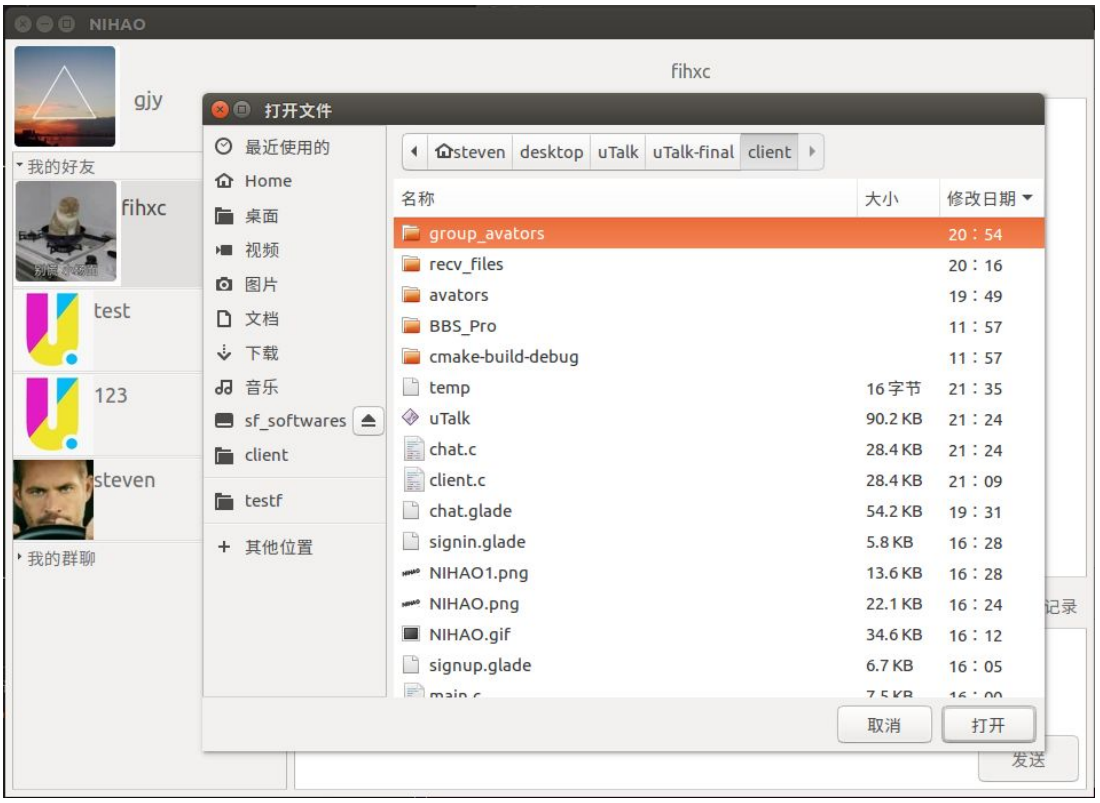


图16: 文件传输效果图（2）

这时好友的聊天窗和你的聊天窗就都会显示出有关文件收发的消息，如下图：

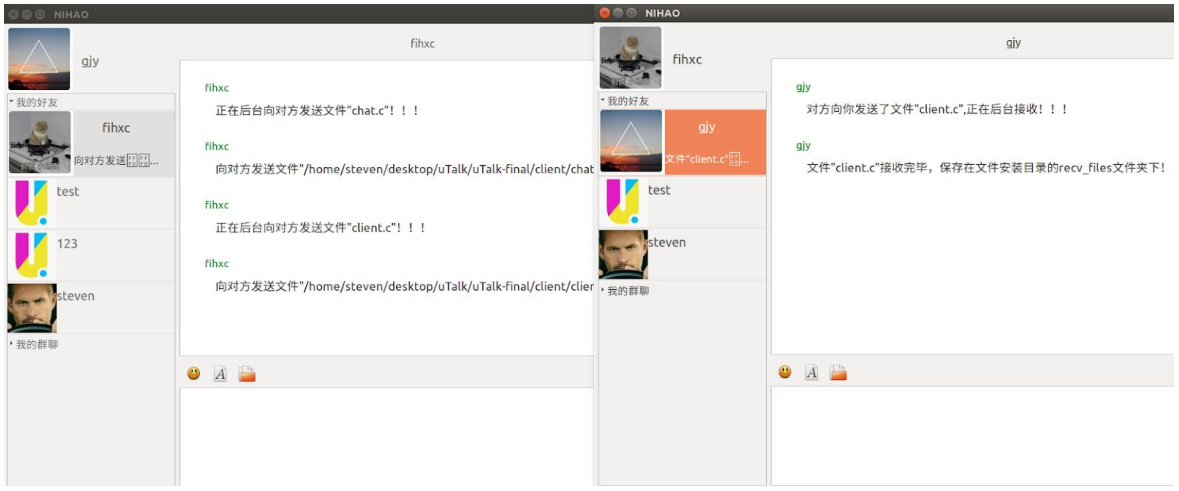


图17: 文件传输效果图（3）

4. 消息备份

历史消息部分我们使用文件来进行存储。首先进行消息的发送。这里我们使用gjy向fihxc发消息，如下图所示：

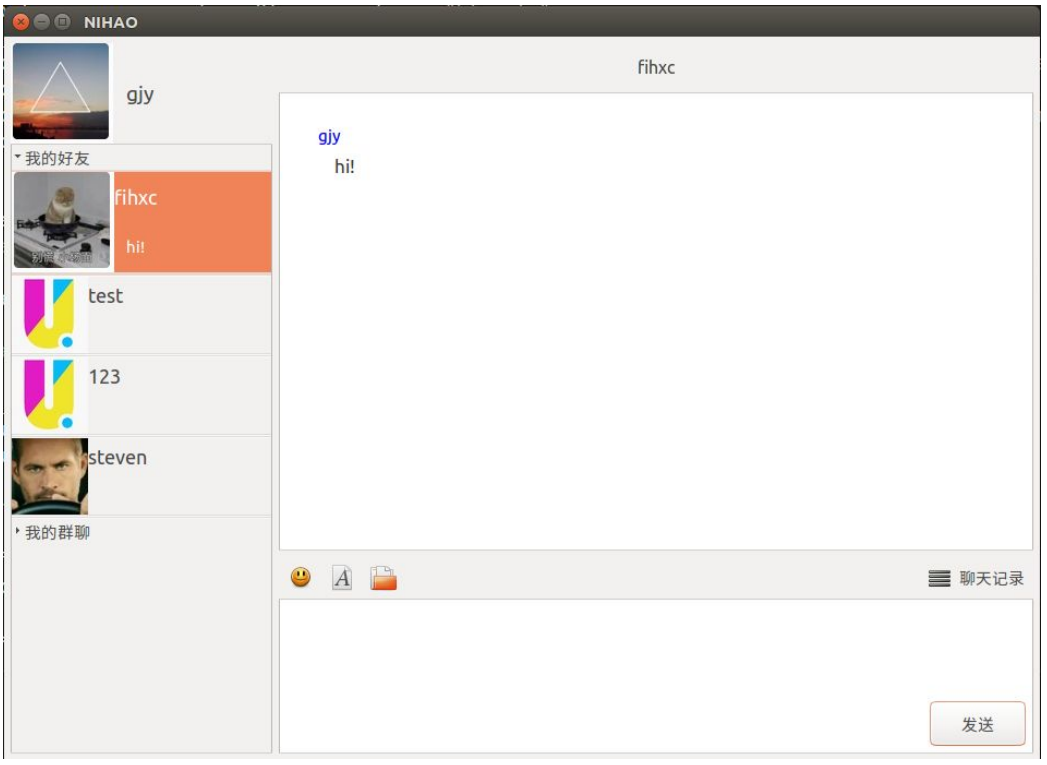


图18: 消息备份效果图（1）

gjy向fihxc发了消息“hi!”。

之后查看服务器端的消息备份情况，如下图：

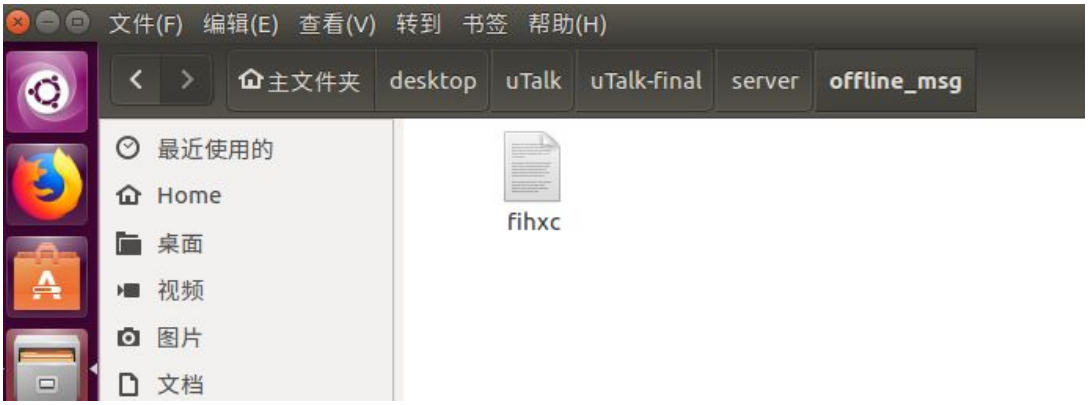


图19: 消息备份效果图（2）

可以看到确实产生了名为“fihxc”的消息记录，里面存储了发给“fihxc”的信息。我们可以打开看一下：

图20: 消息备份效果图（3）

可以看到，该文件确实存储了刚才gjy发给fihxc的信息。

四、特色和创新

1. 登陆模块

在登陆模块中，客户端在登陆界面中输入相应的用户名和密码，并发送给服务器。服务器通过查询数据库的有关数据决定是否登录成功。登录模块和注册模块共享一个界面。具体界面如下图：

图21: 登录界面效果图（1）

我们针对用户在“用户名”和“密码”字段所填内容不同，产生不同的提示信息。

如果用户没有填用户名，则弹出“用户名不能为空！”的警告框，如下图：

图22: 登录界面效果图（2）

如果用户填写了用户名，却没有填密码，则弹出“密码不能为空！”的警告框，如下图：

图23: 登录界面效果图（3）

下面尝试让用户填写错误密码。在这个案例中，用户steven的密码是“123456”。如果我填入“123”，则会弹出“用户名或密码错误！”的警告框，如下图：

图24: 登录界面效果图（4）

若用户填入正确密码，则会正常登录，弹出主界面，如下图：

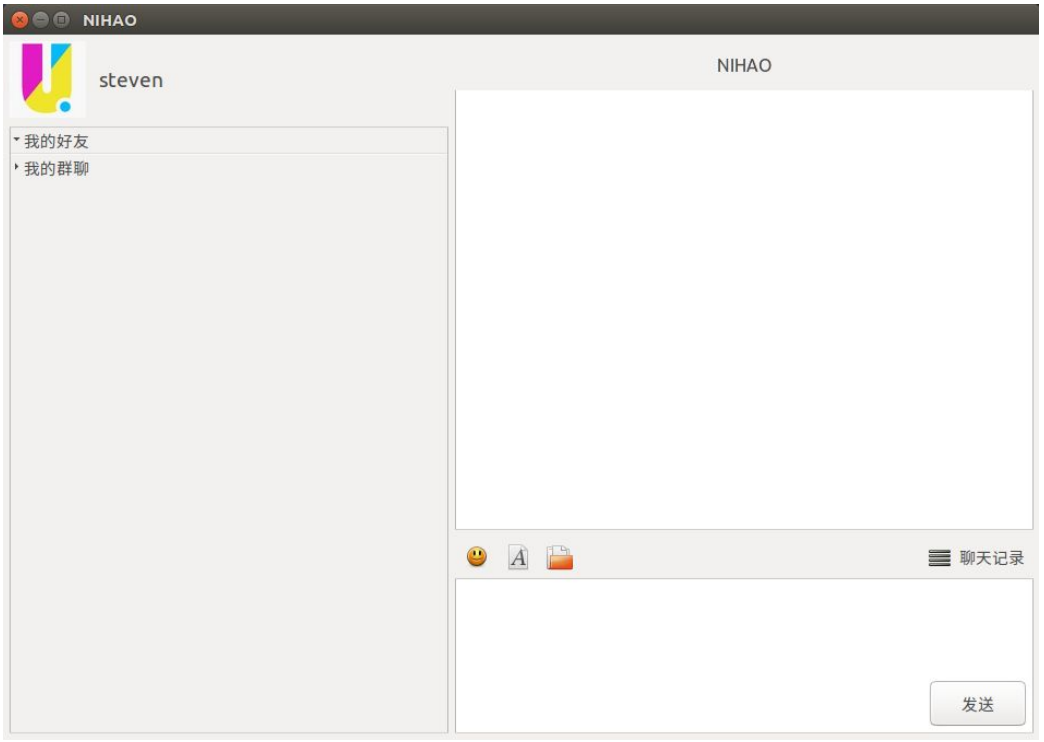


图25: 登录界面效果图（5）

其中用户头像用的是默认图片，可选择“头像更换”功能进行替换。

2. 注册模块

注册模块和登录模块共享的界面之前已经展示过，当用户点击“注册”按钮时，弹出注册对话框如下图：

图26: 注册界面效果图（1）

同样，客户端把用户填写的信息发送给服务器，服务器针对返回的信息做相应反馈。在此处，我们也针对不同的场景设计了不同的反馈。

如果用户注册时没有填写用户名，则会弹出“用户名不能为空！”的警告框，如下图所示：

图27: 注册界面效果图（2）

若用户填写了用户名但没有填密码，则会弹出“密码不能为空！”的警告框，如下图所示：

图28: 注册界面效果图（3）

若用户填写了密码但重复密码未填写或填写得不对，则会弹出“两次密码不匹配！”的警告框，如下图所示：

图29: 注册界面效果图（4）

若用户两次填写的密码相同，但用户名已被注册，则会弹出“用户名已经存在！”的警告框，如下图所示：

图30: 注册界面效果图（5）

若用户输入的是之前未注册过的用户名，且两次密码输入相同，点击“注册”按钮，则会弹出“注册成功！返回登录界面。”的成功框，如下图所示：

图31: 注册界面效果图（6）

点击“确定”后会返回登陆界面。

3. 头像更换

头像更换模块是我们的一个特色功能。其基本实现方法是点击菜单栏上的“设置”按钮下的“更换头像”，然后会访问客户端本地存储头像图片的文件夹中指定序号的图片。这部分和数据库没有交互。呼出菜单栏上的“更换头像”功能如下图所示：

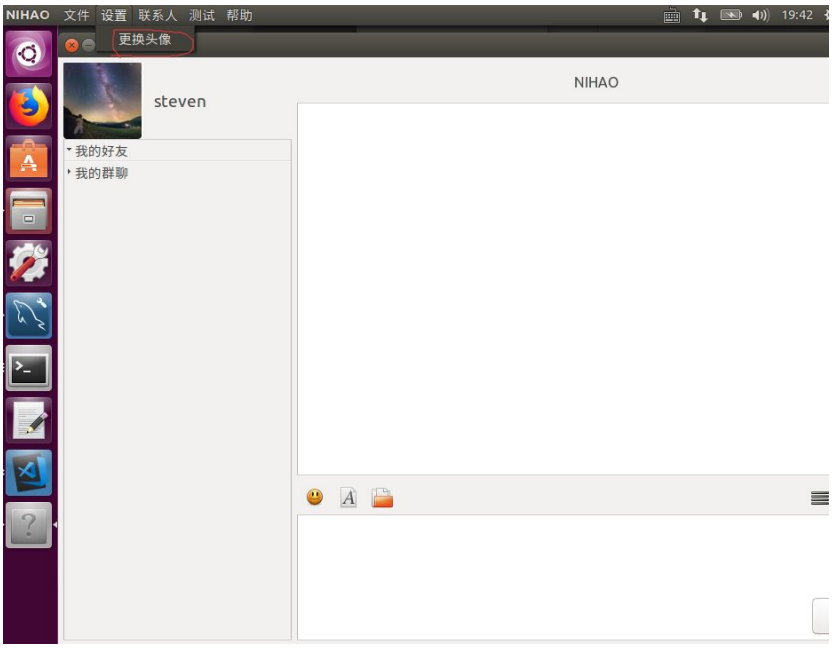


图32: 头像更换效果图（1）

进一步点击“更换头像”后弹出对话框，如下图所示。该对话框也是用Gtk的对话框类去实现，可以呈现9张图片，想要呈现更多图片可以再修改想应的glade框架。

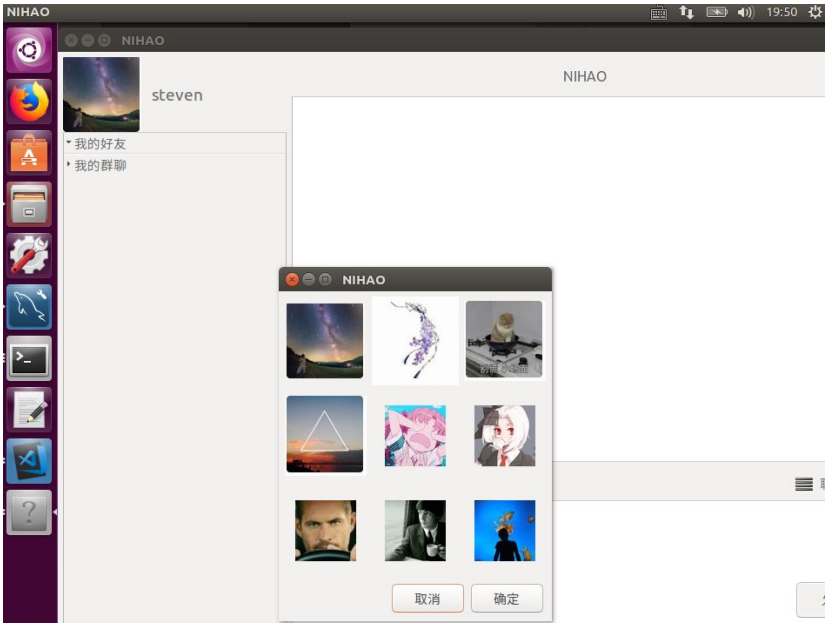


图33: 头像更换效果图（6）

如果我点击左下角的头像，再点击确定，即可达到更换头像的效果，如下图所示：（此处没有gif，但确实可以实现该功能）

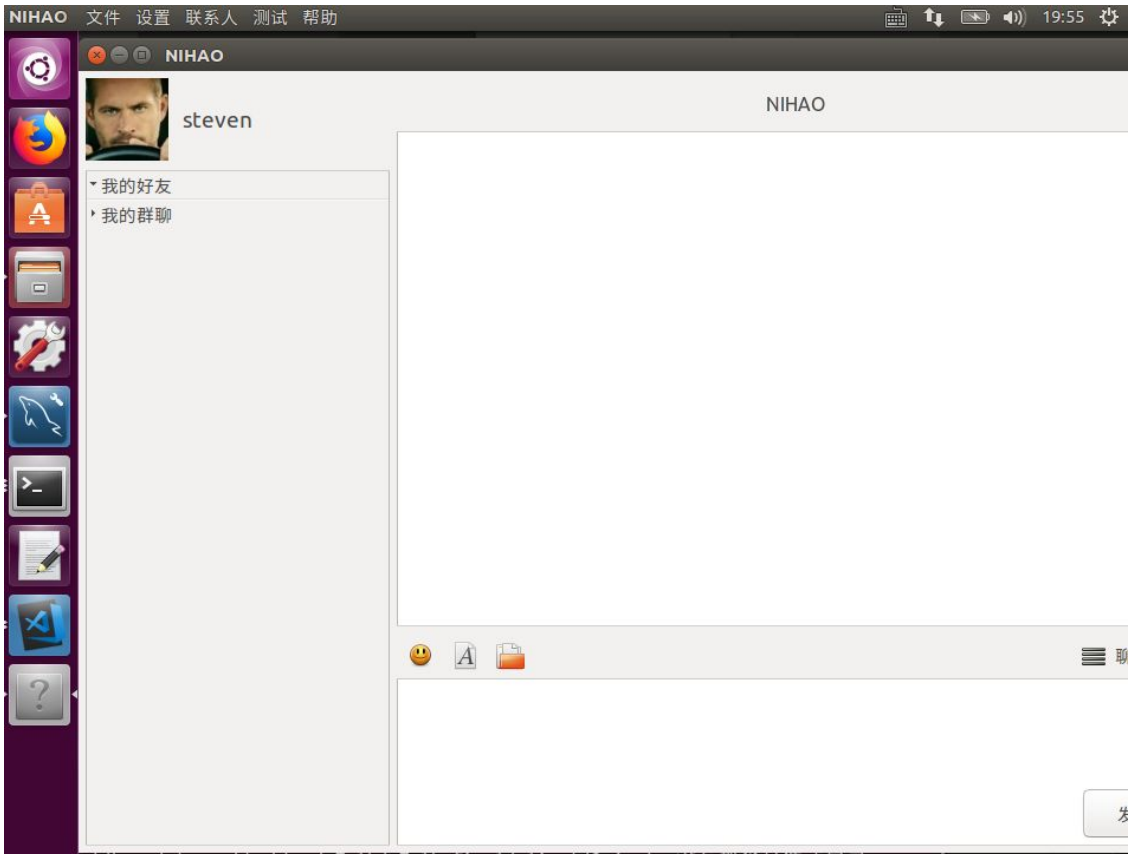


图34: 头像更换效果图（7）

更换头像完成后，会与数据库连接，将原账户对应的头像文件的编号做相应的更改，这样下次登录到主界面时头像就是更新后的结果。

4. 好友模块

4.1. 好友搜索

本系统具有好友搜索的功能，即输入好友的全部或部分名称系统将在数据库中进行匹配并将符合搜索条件的结果返回。搜索部分采用模糊搜索，即当输入的内容与数据库中已有的用户不完全一致时，系统将把最为符合的用户信息给出。此时用户就可以根据搜索出的好友的信息来进行好友的添加。

4.2. 好友添加

本系统具有好友添加功能。首先根据好友搜索部分得到的结果选定欲加好友的用户，之后点击添加即可将对方添加到自己的好友列表中。

5. 新消息提醒

新消息提醒功能就是当两个客户端都在线，一个客户端（用户）A给另一个客户端（用户）B发消息时，若客户端B在与另一个客户端C对话的界面中，客户端B能在自己的主界面窗口下发现好友列表里A的头像旁多出一行小字，该行字正是客户端A给客户端B发过去的消息。如下图所示：

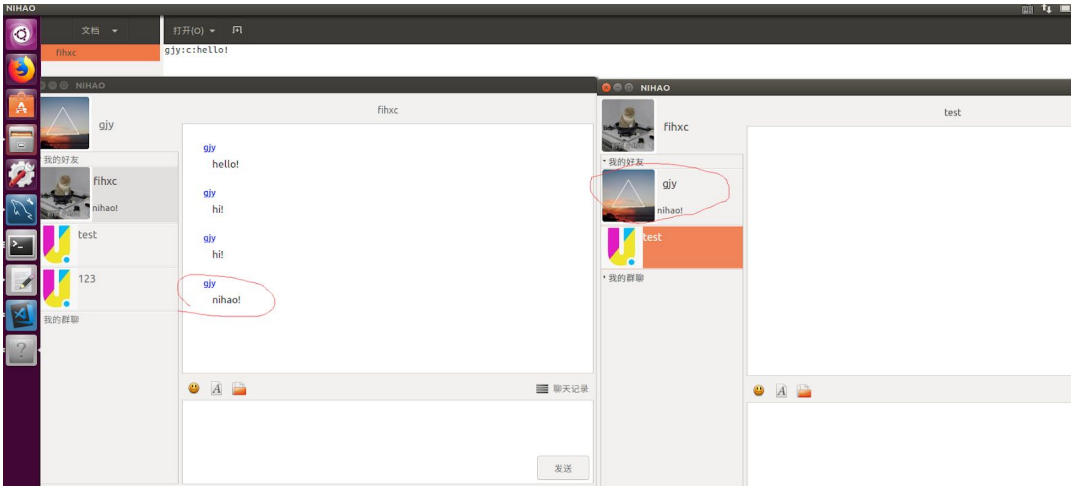


图35: 消息提醒效果图（1）

这样客户端B就能够知道客户端A给自己发送了消息，当客户端B重新点击好友列表里的A时，就能在聊天界面看到相应的消息了，如下图所示：

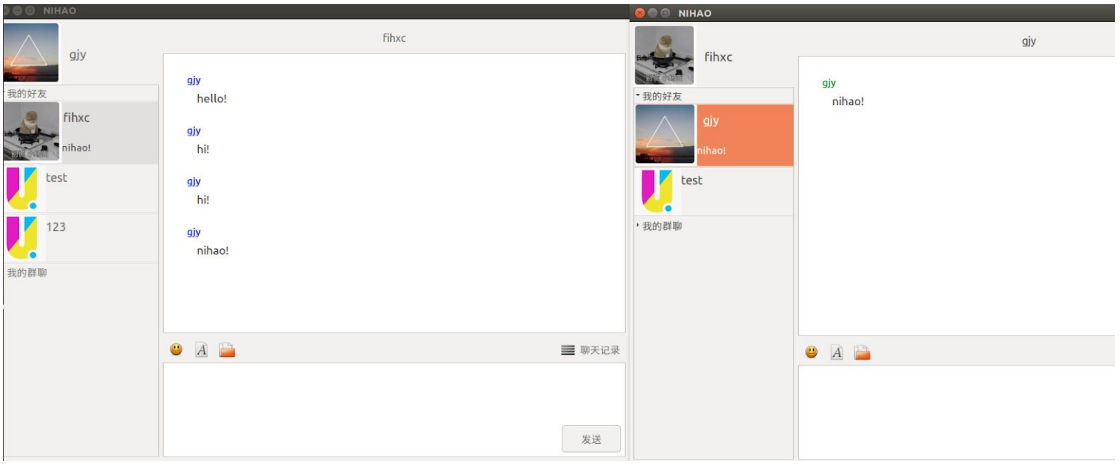


图36: 消息提醒效果图（2）

6. 历史消息的展示

这部分功能的实现主要依靠消息在服务端的存储。一旦客户端选择历史消息，服务器端便会打开对应用户的历史信息文件，并将于对应用户聊天的内容截取并发送回客户端。客户端再在特定的窗口内进行展示。

五、困难和解决

1. 粘包问题

本聊天系统的底层实现采用TCP协议，在最初进行消息传输测试时出现过粘包问题，即客户端连续发送的两次消息在服务器端被视为一次发送所接收。这种情况的出现主要原因在于Linux系统(本质上几乎所有的系统上)实现的关于网络的系统调用中，TCP协议发送数据都使用了Nagle算法处理。因为在实际的应用场合下，TCP-IP协议的报文头部的负载过大，发送过少的数据将会极大的降低数据传输的效率，因此Nagle算法力图避免转发过小的包，要求TCP连接上只能有一个未被确认的小分组，在分组的确认没有到达之前不能发送其他的分组，并有效的收集大量的小分组，在下一次分组确认到达之后一次性发送出去，提高TCP发送数据的效率。

但是在我们的项目中，因为每一次用户发送的数据并不是很多，但是我们为了保证消息的实时性一定需要将数据一条一条的发射出去，Nagle算法造成的TCP粘包问题就成了一定要客服的困难。

Linux系统调用中存在有相关的禁用Nagle算法的接口，但是在实际测试中效果并不是很好Nagle算法有时候仍然存在影响，通过在网络上查询资料得知可能和TCP本身协议有关，所以我们决定使用 sleep 系统调用对 Nagle 算法实现禁用。

通过在一次发送消息之后，在一定的时间内休眠系统强迫send函数立即发送对应的缓冲区中的数据，实现及时的消息传输，经过测试，0.001s的睡眠负载已经可以满足要求。

2. 消息类型的规定和区分

在通信过程中对消息进行编码对判断消息类型以及进行正常的通信交流是相当关键

的。我们在实现时对客户端发送到服务器的消息以及服务器发送到客户端的消息的前缀分别进行了编码。具体的编码情况见下表：

客户端到服务端	
登录	/0
注册	/1
获得好友列表	/2
加好友	/3
获得离线消息	/4
查找好友	/5
设置头像	/7
查找头像	/8
获得群列表	/9
普通消息	无
群聊消息	$$
文件	#

表

可以看出，消息主要被分为四种，命令，普通消息，群聊消息和文件。每种消息里面再根据具体内容的不同又进行了新的编码。这样编码得到的数据可以在服务器和客户端之间进行完善的区分和处理，从而确保系统高效有序地运行。

3. 多线程调试

小组在该项目中大量的使用了多线程的方式实现程序的并行化，但是有效的掌握对多线程的调试能力是非常必须的，在这里我们尝试了大量的编辑器实现对多线程的调试都不是非常的有效，最终我们还是选定了GDB作为调试的基本工具。通过使用GDB对项目进行调试，小组有效的抓住了问题的本质，有力的定位到了bug的位置并快速的加以改正。

这让我们认识到了在调试方面，GDB和其他的调试工具相比虽然界面并不是那么的优美，但是因为GDB大量和有针对性的命令和完善的信息处理，使得我们对于系统

的调试和认识方式比其他的可视化的调试工具变得更加容易。并且GDB还有非常成功的终端界面库可以使用。

六、扩展和后续工作

1. 加密通讯部分

在进行通信时，本系统主要使用明文进行传输。然而对于安全可靠的通信而言，直接使用明文进行传输是不被允许的。因此下一步本系统将考虑在安全传输方面着力，来是实现消息通讯的安全。目前主要考虑的加密工具是openSSL，来通过非对称加密系统以及对称加密系统共同完成消息传输的可靠性、私密性等。

2. 建立内嵌的BBS

本系统目前使用的BBS系统是独立于聊天系统而单独存在的。接下来的开发中将进行两部分的合并，即将BBS的服务器部分和界面与聊天系统的服务器部分和前端整合在一起，这样能够使系统更加紧密，用户使用体验更好。主要的实现我们考虑到两种方式，第一种方式直接使用前端的gtk控件来完成，第二种方式是通过HTML文件的形式内嵌与客户端来完成。

3. 群组管理

本系统的群组管理模块还待加强。目前群组管理只实现了后台的模拟。下一步准备扩展群组管理的功能，建立群主和管理员机制，来更好的完成群的创建和管理。实现这部分需要进行数据库群组部分的重新设计，服务端对群组消息的设计以及客户端对群组消息类型的重新编码。

七、分工和合作

郭嘉琰	客户端界面和底层实现的设计和编写
沈翰文	服务器端和数据库的设计和编写

八、心得体会

郭嘉琰：

本次大作业的收获和心得主要如下：

- 进行了客户端的开发，熟悉前端和后端的协调操作

本次大作业完成了Linux下的NIHAO聊天软件的编写。我主要负责客户端的部分。客户端涉及到界面的实现以及底层的实现。对于界面的实现，难点在于快速学习界面编写库并进行使用。考虑到Linux的环境以及自己有过简单的gtk库的编写经验，最终的界面部分采用gtk来进行编写。对于底层，就少了很多编写前端的限制，可以充分利用课上所学的内容来进行与服务器方面的通信。难点在于要和服务器端约定好信息的编码格式，同时还要考虑到进程之间的相互关系。以上两个部分单独来进行实现的话都不是很难，最难的地方在于将两者结合起来。这就涉及到消息从客户端导服务器，以及从服务器再到客户端的过程。前一个过程需要前端调用底层的接口来实现，后一个过程需要底层调用前端的接口来实现。因此接口的设计相当重要。除了客户端本身的一些挑战性的工作外，与服务器的在进行连接测试的过程中也出了很多问题，比如消息的改变、内存泄漏等。这些问题的解决都很耗时间。但我们小组成员密切合作，最终将客户端与服务器端的通信实现的很好。

- 熟悉了通信软件的运作机制

通信软件的运作机制主要还是以客户端——服务器模型为主，即服务端处理并转发客户端的各种消息和请求。这种方式对客户端相当友好，只需要开启两个线程即可保持通信，而对服务器的要求就相对很高。不过对于大公司商业化的今天，服务器的运作不成问题，因此这种模型也就成为目前最有效的模型。除此之外，近些时间随着区块链的发展也出现了分布式聊天软件的概念，即去掉服务器端，而仅保留客户端进行通信，且使用区块链的方法来保证消息的可靠性和不刻抵赖性。对于大数据时代的快速发展，信息量的进一步增加，这种方式可能会具有比较理想的前景。

- 良好的沟通对于软件开发很重要

团队的良好沟通对于软件的顺利开发相当重要。本次大作业我和沈翰文同学主要负责聊天软件的部分，在进行客户端和服务器的对接过程中我们就消息编码的格式和和通信完成的机制进行了讨论，并共同确定了编码的标准。除此之外，在进行测试和

调试的过程中小组成员之间也不断进行交流和讨论，最终使得产品能够顺利完成。可以看出良好的团队沟通具有重要的意义。

沈翰文：

- 如何搭建一个健壮的服务器

本次大作业我们完成了Linux下的NIHAO聊天软件的编写。我主要负责服务器和数据库部分。相比较数据库部分，服务器部分对我来说要难得多，由于先前没有相关开发的经验，只是在Linux选修实验课上接触过一些。通过这次大作业，我深切地认识到最难的还是编码，如何对于客户端发送过来的消息进行剖析分解，再返还相应的消息回去需要仔细考虑。只有与客户端确立了相关编码规范，才能有效地接收和处理客户端的请求。这一点让我想起了计网课上学过的一些协议，比如TCP/IP等，我们这次大作业开发也是一个确立协议的过程。另外，当服务器和多个客户端完成对接进行测试时，debug也成了一个难题，因为设立了断点以后，程序会在服务器和客户端之前来回跳转，如果思路不清晰也容易找不到bug所在。

- 数据库的设计与测试，关于触发器的应用

数据库设计这块由于我有一定小学期软件工程开发的经验，没有遇到太多的困难，对于C语言与mysql的一些API，拿过来直接使用，相对来说比较方便。以我个人经验来说，在表与表之间增加一些触发器能很好地增强数据库的鲁棒性。尽管如此，我觉得还有一些可以优化的地方，比如在存储个人信息时，一些属性是否可以合并到一张表里；以及客户端的文件路径该如何存储等。