计算机通信网络课程作业

姓名：李铁
学号：518030910061
选题：聊天程序
指导老师：姚立红
日期：2020 年 12 月 19 日

目录

1 项目概述

本项目实现了聊天程序的基本功能，分为客户端和服务器端，功能包括：一对一聊天、群组聊天、发送表情、发送语音、发送文件。客户端实现了较为友好的用户界面。项目基于 python3.7 ，使用了各种 python 内置与第三方库，其中最核心的库为 Socket。

项目 UI 基于 PyQt5，实现设计风格仿照微信。UI 组件主要分三个部分：聊天界面，登录界面，注册界面。程序开始时运行登录界面，根据用户的选择进入注册界面进行新用户的注册或者聊天界面与其他用户进行信息交互。

1.1 环境配置

项目基于 python3.7 实现。依赖的库如下：

表 1: 程序中所需要的库

库	说明
os	操作系统接口模块
socket	底层网络接口
time	用于时间的显示
threading	给予线程的并行
sys	系统相关的参数和函数
base64	对数据进行编码的库
PyQt5	Python 的 GUI 库
sounddevice	音频处理库
numpy	处理 sounddevice 采样产生的数组的库

1.2 程序文件列表

表 2: 程序文件列表

soruce/	源代码文件夹
source/img/	程序中涉及到的图片
source/client.py	客户端程序
source/sever.py	服务器端程序
Readme.txt	个人相关信息
a	储存用户 a 下载的文件的文件夹

2 主要算法

2.1 信息的处理

客户端与服务器端相互通信，发送的信息是一个字典，接收方接收到信息后根据 type 所对应的数据进行判断消息的类型，接着进行消息的处理。其中由于每个用户的本地都拥有表情的信息，所以在实际通信过程中并不会发送表情，而是发送表情的编号，客户端接收到表情的编号后，调用本地的表情显示在聊天窗口中。

```
elif (msg["mtype"] == "file"): #文件消息处理
    for item in self.connection:
        if (item.getpeername() == self.add_user[dname]):
            filehin = sname+" has sent "+msg["fname"]+" to you."
            item.send(str({"type": "msg",
                           "mtype": "file",
                           "dname": dname,
                           "sname": sname,
                           "ssname": sname,
                           "time": time.time(),
                           "msg": filehin,
                           "fname":msg["fname"],
                           "file":msg["msg"]}).encode())
        if (item.getpeername() == self.add_user[sname]):
            filehin = "you" + " has sent " + msg["fname"]+" to "+dname+". "
            item.send(str({"type": "msg",
                           "mtype": "file",
                           "dname": sname,
                           "sname": dname,
                           "ssname": sname,
                           "time": time.time(),
                           "msg": filehin,
                           "fname": msg["fname"],
                           "file":msg["msg"]}).encode())
```

图 1: 服务器对文件消息的处理

```
elif(msg["mtype"] == "msg"): #文字消息处理
    for item in self.connection :
        if (item.getpeername() == self.add_user[dname]):
            item.send(str({"type": "msg",
                           "mtype": "msg",
                           "dname": dname,
                           "sname": sname,
                           "ssname":sname,
                           "time": time.time(),
                           "msg": msg["msg"]}).encode())
        if (item.getpeername() == self.add_user[sname]):
            item.send(str({"type": "msg",
                           "mtype": "msg",
                           "dname": sname,
                           "sname": dname,
                           "ssname": sname,
                           "time": time.time(),
                           "msg": msg["msg"]}).encode())
```

图 2: 服务器对文字消息的处理

2.2 客户端消息的存储与调用

服务器发送给客户端的消息以列表的列表的形式存储在 connect 类中，当界面中需要显示的时候对列表进行遍历，挑选出需要的信息进行显示。

```
if (message["type"] == "msg" and message["mtype"] == "file"): #存储文件
    ftemp = []
    ftemp.append(message["sname"])
    ftemp.append(message["ssname"])
    ftemp.append(rtime)
    ftemp.append(message["file"])
    ftemp.append(message["fname"])
    self.recfile.append(ftemp)
if (message["type"] == "msg" and message["mtype"] == "voice"): #存储语音
    vtemp = []
    vtemp.append(message["sname"])
    vtemp.append(message["ssname"])
    lt = rtime+' '+message["len"]+"s"
    vtemp.append(lt)
    vtemp.append(message["voice"])
    self.recvoice.append(vtemp)
```

图 3: 客户端对音频、文件消息的存储

```
for item in con.recmsg:
    if (item[0] == t):
        self.grprecvText.append(item[1]+" "+item[2]+" :")
        if(str(type(item[3]) ) == "<class 'str'>"):
            self.grprecvText.append(item[3])
```

图 4: 客户端对文字消息的调用

2.3 客户端消息框的更新

一般情况下，在两个用户进行通信的时候，各自的消息框显示的都是与对方通信的消息，如果客户端不能根据自动更新，用户只得手动更新消息，这对用户来说是不友好的。所以在聊天窗口类中维护了 chatuser 的变量，如果接收到来自 chatuser 的信息，则进行消息框的更新。

```
if login.pr.chatuser == message["sname"]: #更新对话框
    login.pr.grprecvText.append(message["ssname"]+" "+rtime+" :")
    if (str(type(message["msg"])) == "<class 'str'>"):
        login.pr.grprecvText.append(message["msg"])
    elif (str(type(message["msg"])) == "<class 'int'>"):
        img = QtGui.QTextImageFormat()
        login.pr.grprecvText.insertPlainText("\n")
        path = "./" + str(message["msg"]) + ".png"
        tcursor = login.pr.grprecvText.textCursor()
        img = QtGui.QTextImageFormat()
        img.setName(path)
        img.setHeight(30)
        img.setWidth(30)
        tcursor.insertImage(img)
```

图 5: 客户端对消息框的更新

2.4 线程的应用

服务器端采用多线程的模式，其中一个子线程用于侦听是否有新的客户端与服务器建立连接，而每当建立一个新的连接的时候，则新增一个子线程，用于与处理该客户端发送的消息并与该客户端进行通信。

```
def plisten(self):
    """
    进入侦听模式 处理来自客户端的连接
    :return:
    """
    self.s.listen(128)
    while True:
        sock, address = self.s.accept()
        self.connection.append(sock)
        fun = threading.Thread(target=self.receive,args=(sock,)) #多线程，为每一个连接的客户端分配线程
        fun.start()
```

图 6: 服务器用于与客户端进行通信的多个子线程

```
def run(self):
    fun1 = threading.Thread(target=self.plisten) #多线程，为侦听的套接字分配线程
    fun1.start()
```

图 7: 服务器用于侦听的子线程

3 数据结构

3.1 PostgreSQL 数据库

PostgreSQL 是一款开源的轻量级关系数据库，比较适合学生学习研究时使用。本次课程作业中，创建了 chatroom 的数据库与 chatroom 表格。chatroom 的具体内容如下图所示：

```
                数据表 "public.chatroom"
  栏位    |     类型       | 校对规则 |  可空的  | 预设
----------+----------------+----------+----------+------
 username | character(50)  |          | not null |
 password | character(50)  |          | not null |
 state    | boolean        |          |          |
索引:
    "chatroom_pkey" PRIMARY KEY, btree (username)
```

图 8: chatroom 表格

由上图可以看出，一条关于用户的记录共有三个属性，分别为用户名、密码、登录状态。其中用户名是主码，不可重复。由于实现好友关系、群组关系、消息的存储较为复杂，需要多个建立多个表格，故在本次作业中进行简化。即只要注册过的用户都可以进行通信，都处于群组 group 中，且不可与不在线的用户进行通信。

3.2 使用 socket 类进行通信

客户端程序包含四个类，其中 connect 类用于与服务器端建立连接并存储相关信息。而剩下三个类分别为登录界面、注册界面、聊天界面对应的类。他们是对于 QtWidgets.QDialog 类的继承，通过调用 connect 类中存储的各种信息来把消息通过界面传递给用户。

而服务器端包含一个 sever 类主要用于连接的建立、数据的处理与发送。

```python
class connect(object):
    def __init__(self):
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)#创建套接字
        self.s.connect(("127.0.0.1", 7879))
        self.logflag = False  #登录状态
        self.signflag = False #注册状态
        self.senderror = False #发送状态
        self.recmsg = [] #存储消息的列表
        self.recfile = [] #存储文件的列表
        self.recvoice = [] #存储语音的列表
```

图 9: 客户端的 connect 类

```python
class server(object):
    def __init__(self, addr, port):
        self.addr = addr
        self.port = port
        self.connection = []
        self.userlist = []
        self.add_user = {}
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.conn = psycopg2.connect(database="chatroom", user="postgres", password="litie2000", host="127.0.0.1",port="5432")#连接数
        self.cursor = self.conn.cursor()
        self.cursor.execute("UPDATE chatroom set state = false")
        self.conn.commit()
        self.s.bind((self.addr,self.port))
```

图 10: 服务器端的 sever 类

4 程序测试

4.1 登录测试

运行 client.py 得到结果如下所示：

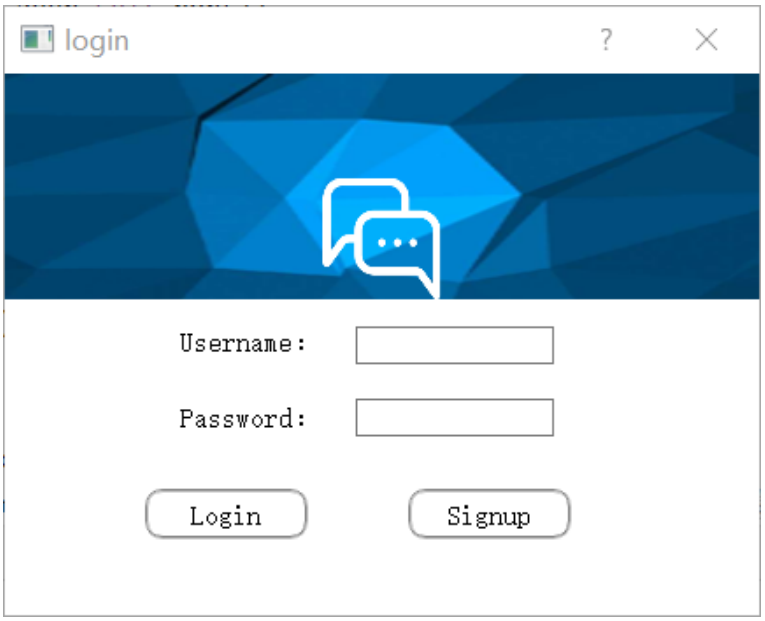

图 11: 登录界面

输入信息错误会出现如下界面：

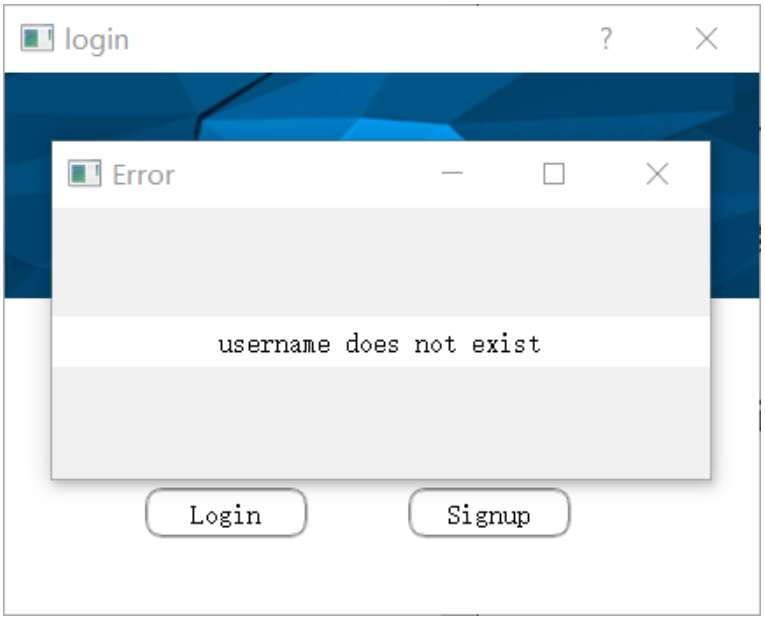

图 12: 用户名不存在

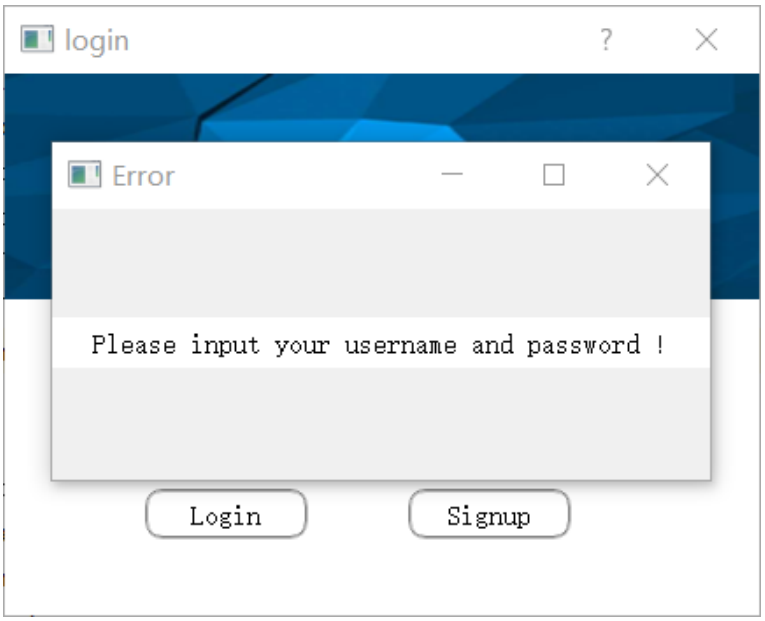

图 13: 未输入用户名或密码

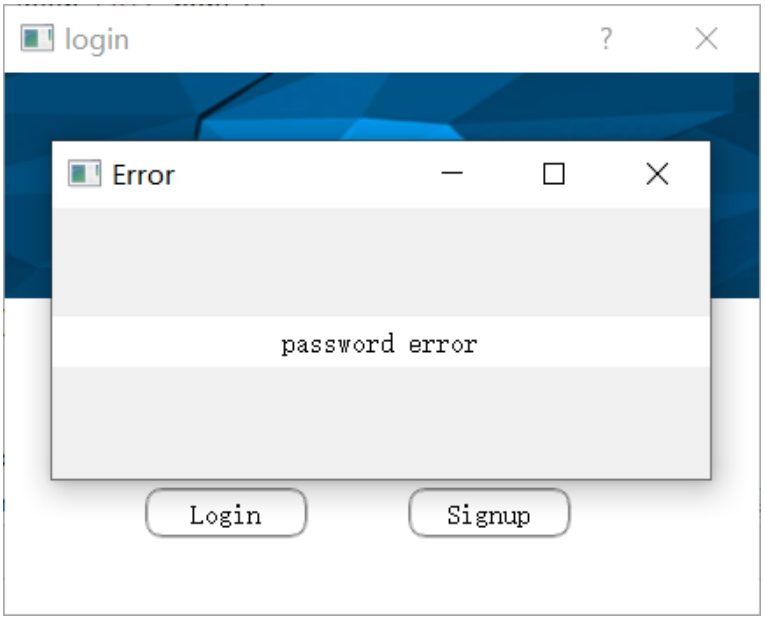

图 14: 密码输入错误

若用户名密码均正确，则进入聊天界面。若按下 signup 按钮，则进入注册界面

4.2 注册测试

注册界面如下所示：

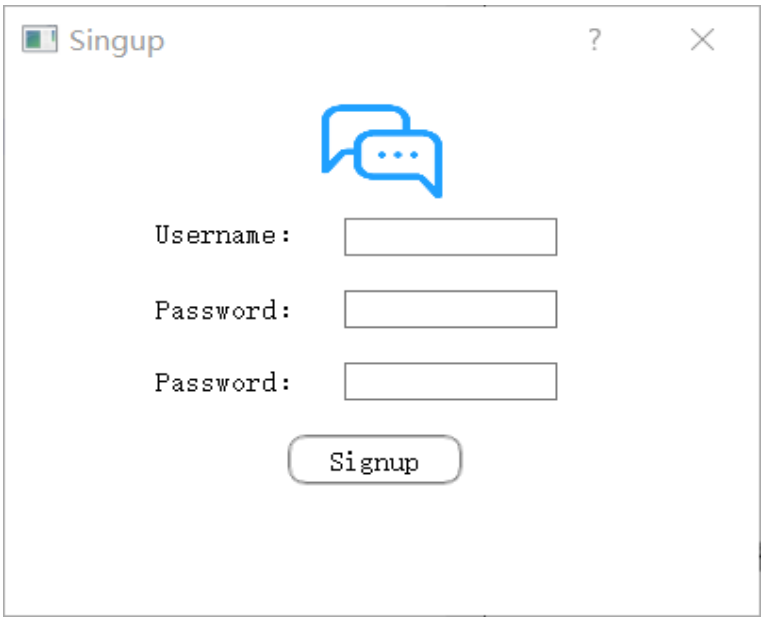

图 15: 注册界面

若未输入用户名或密码：

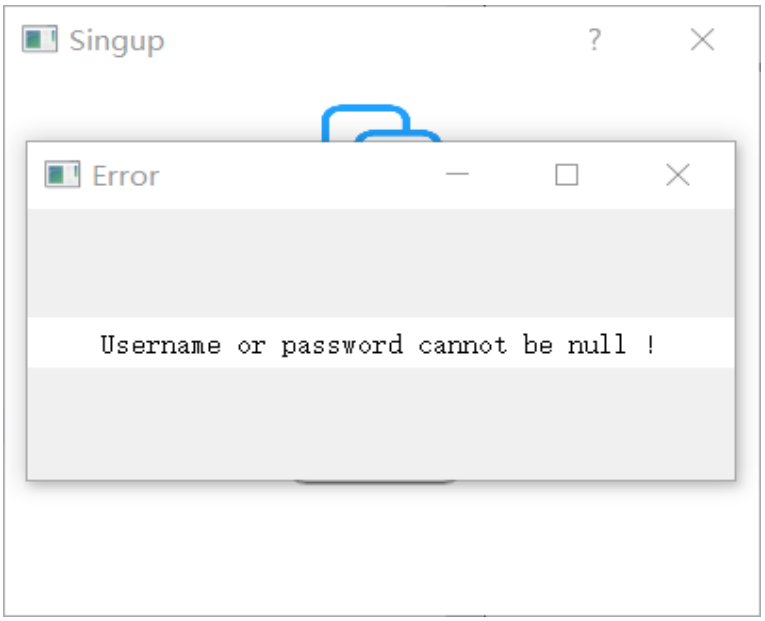

图 16: 未输入用户名或密码

若两次输入的密码不一致：

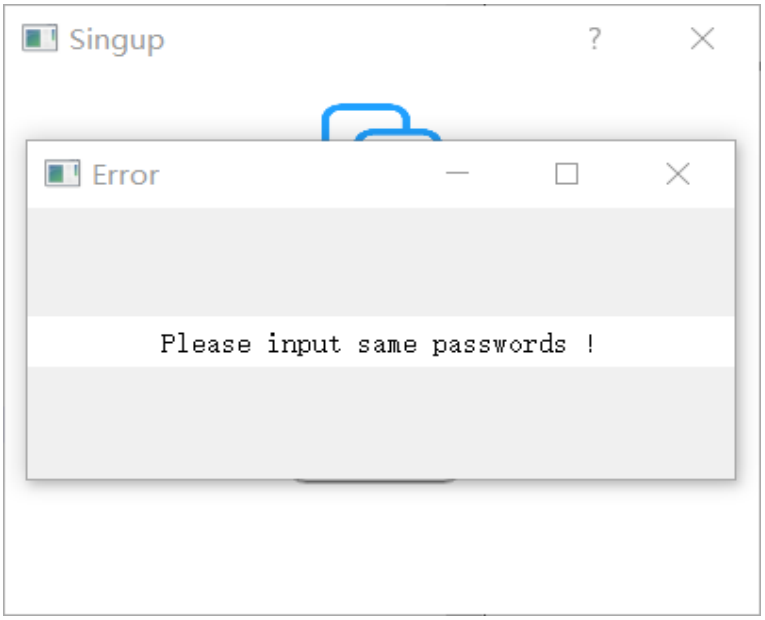

图 17: 密码输入不一致

若注册的用户名已经存在：

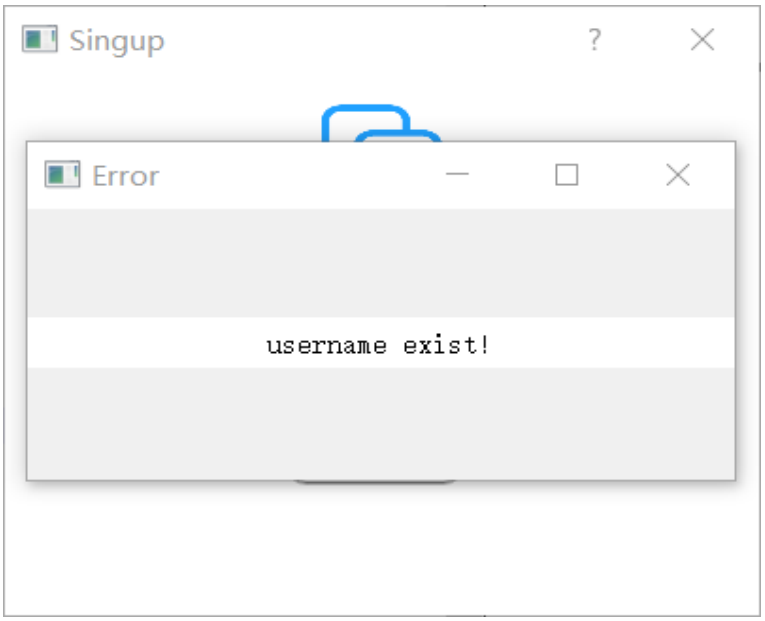

图 18: 注册已存在用户名

注册成功后，将返回登录界面，可以进行登录操作。

4.3 聊天界面

登录成功后界面如下所示：

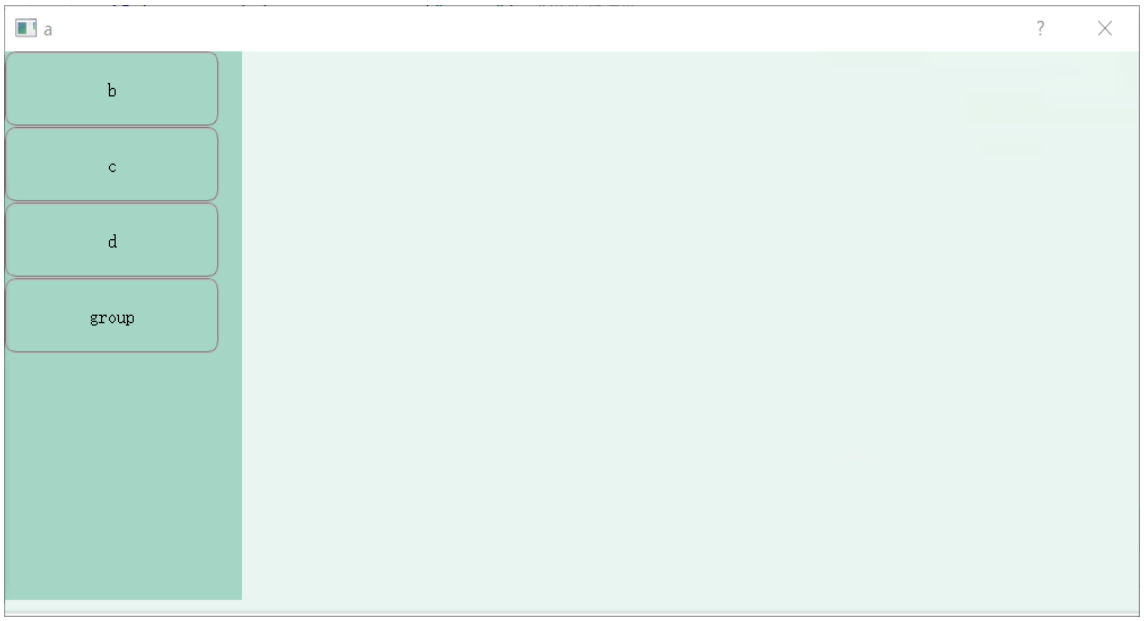

图 19: 初始聊天界面

选择好友后，出现消息框和输入框：

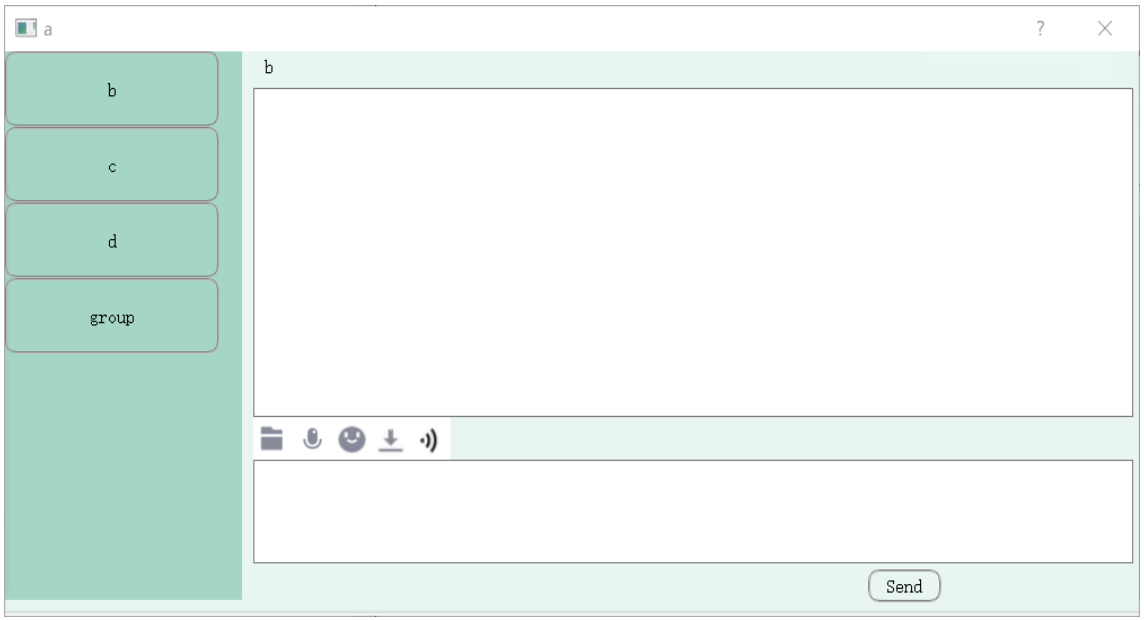

图 20: 选择好友后的聊天界面

若给一位不在线的好友发送消息：

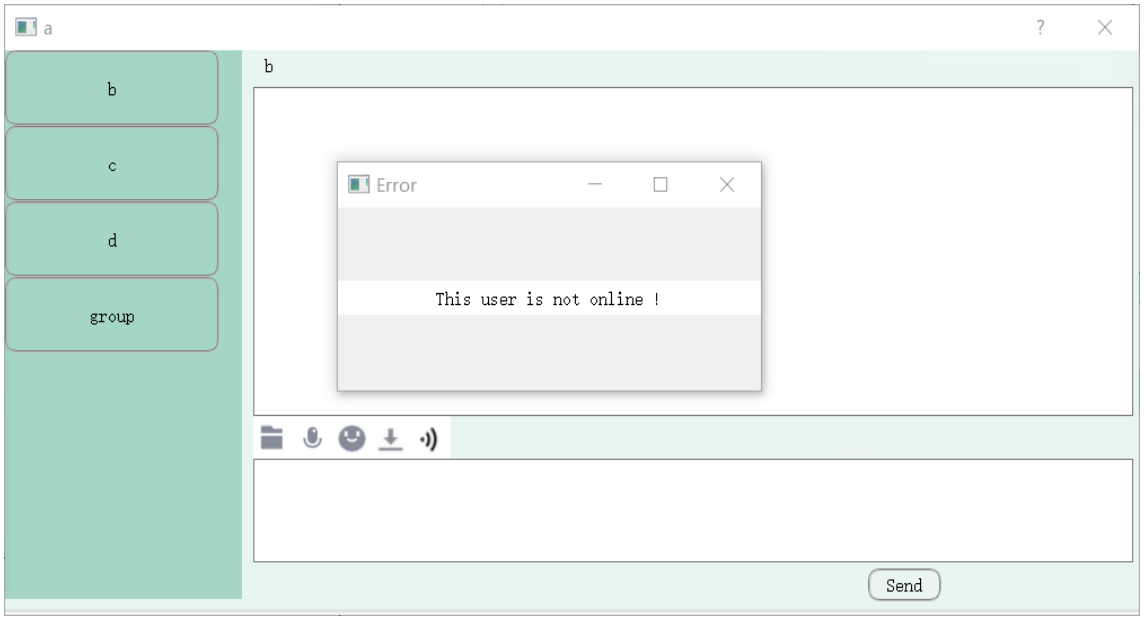

图 21: 与不在线的好友进行通信

好友间进行通信：（以 a 和 d 为例）

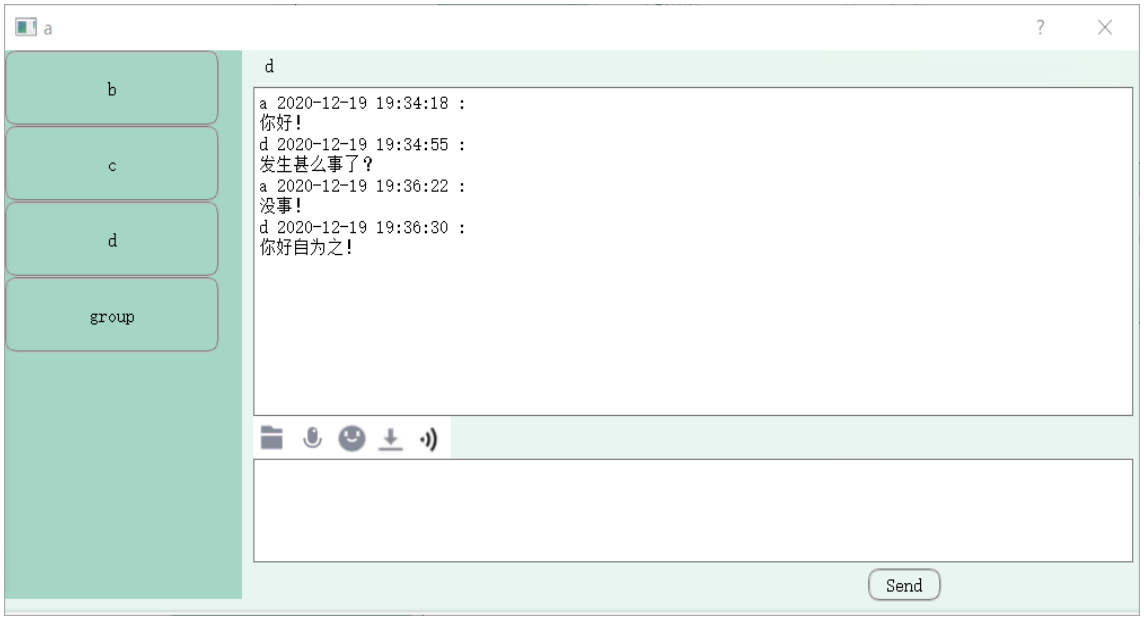

图 22: a、d 间通信 a 的界面

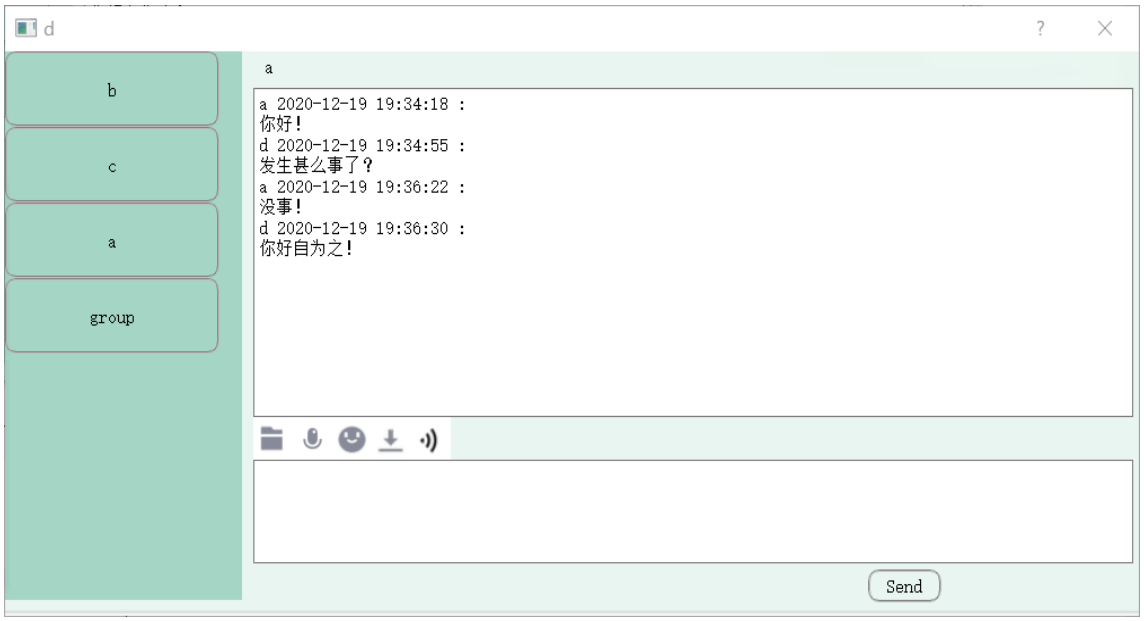

图 23: a、d 间通信 d 的界面

同样地，a 与 d 之间可以发送表情、文件、语音消息。结果如下图所示：

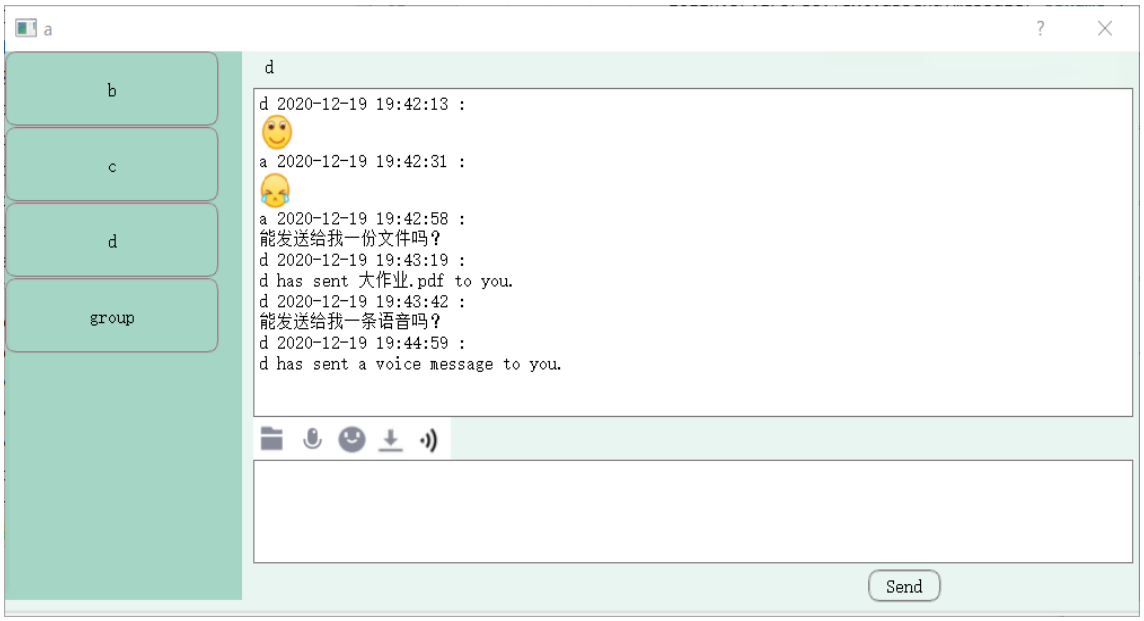

图 24: 其他类型消息的发送 a 的界面

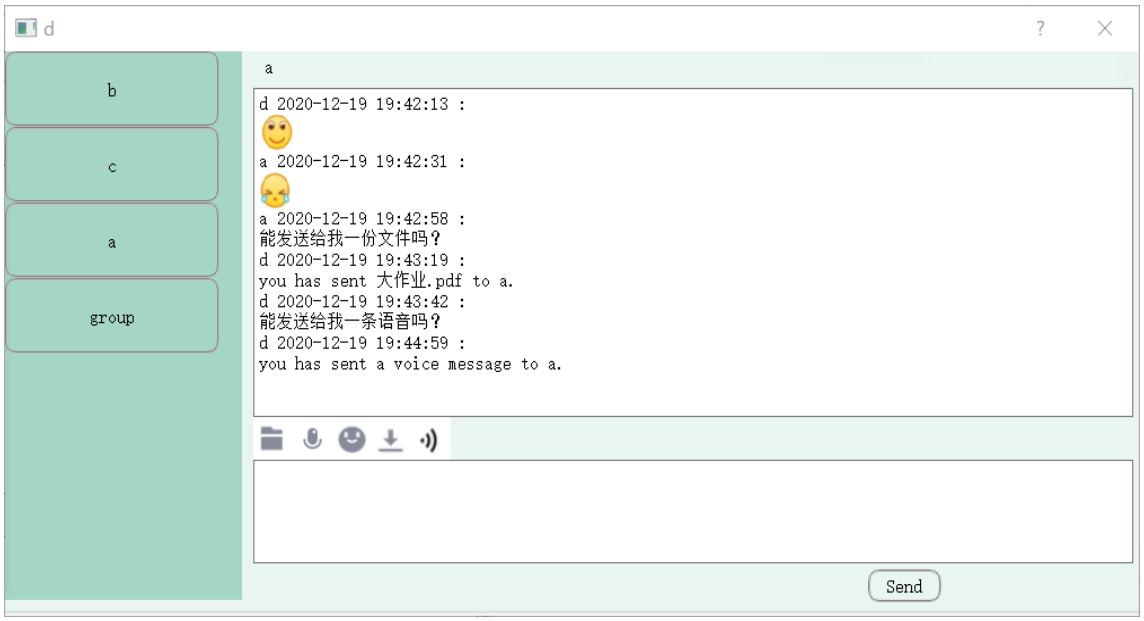

图 25: 其他类型消息的发送 d 的界面

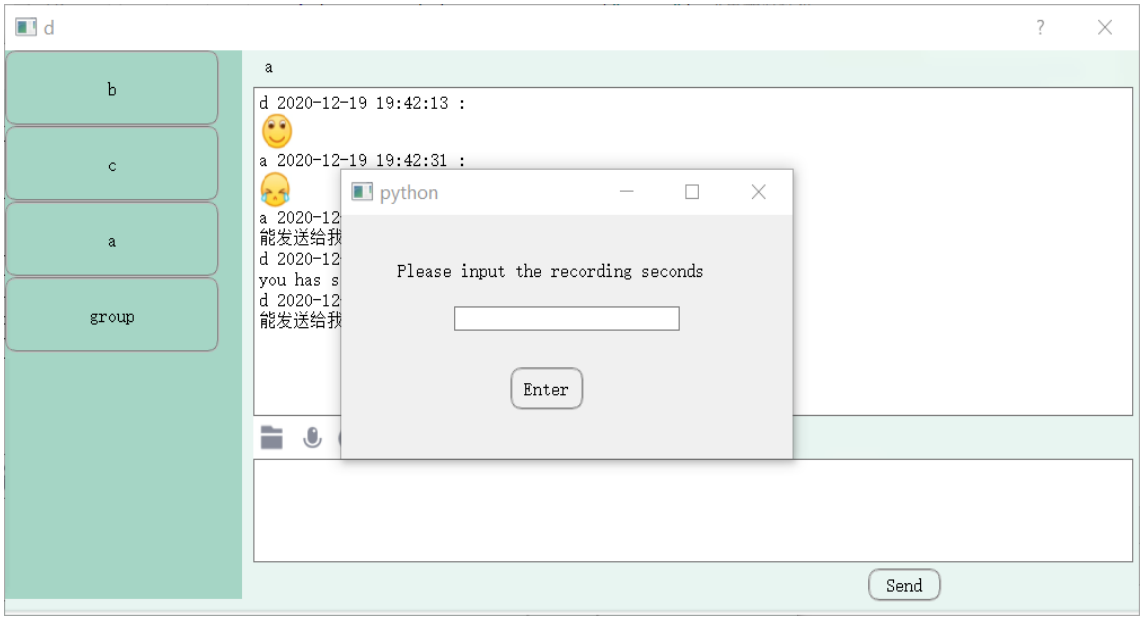

图 26: 输入语音消息时长的界面

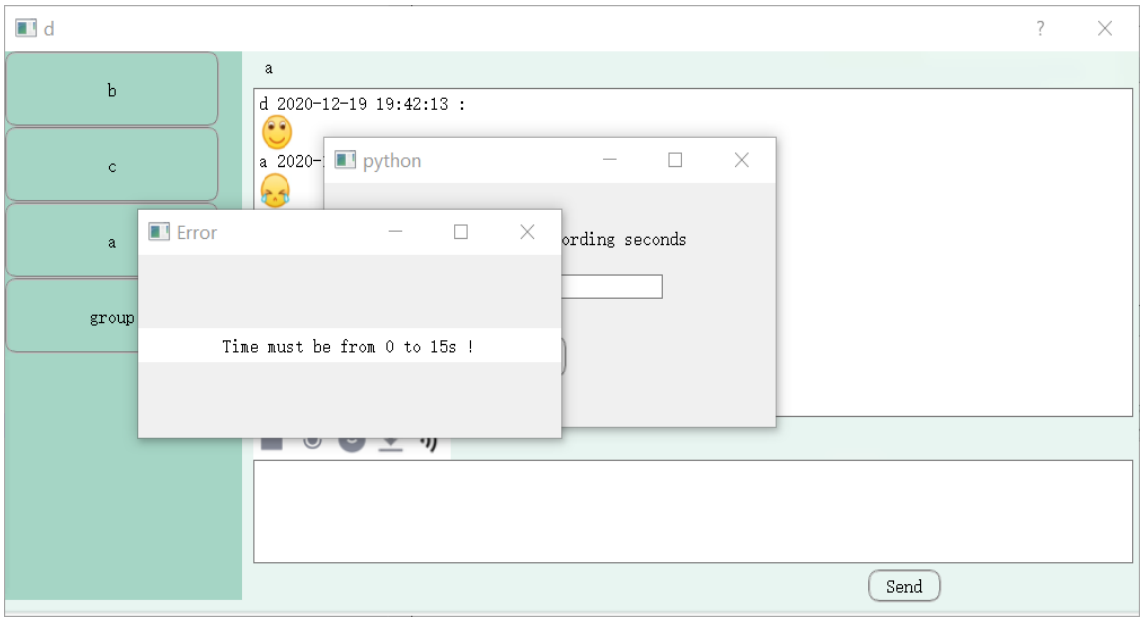

图 27: 输入时长不合法

点击播放按钮，得到所有语音消息的列表，双击可以播放语音消息，如图所示：

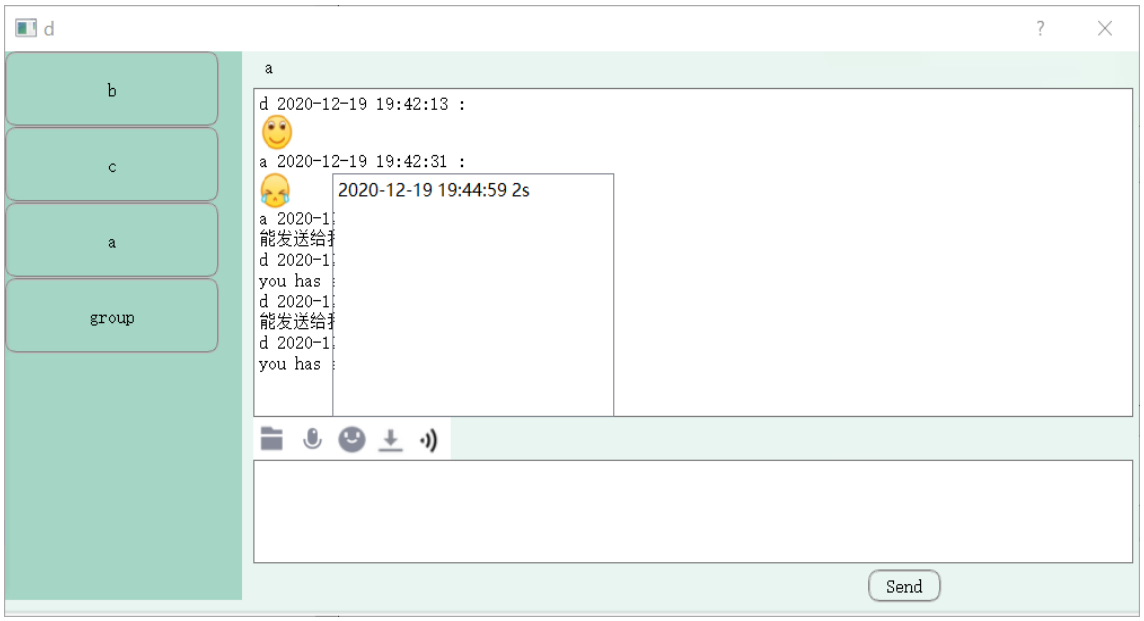

图 28: 语音消息列表

点击下载按钮，得到所有文件的列表，双击可以下载，如图所示：

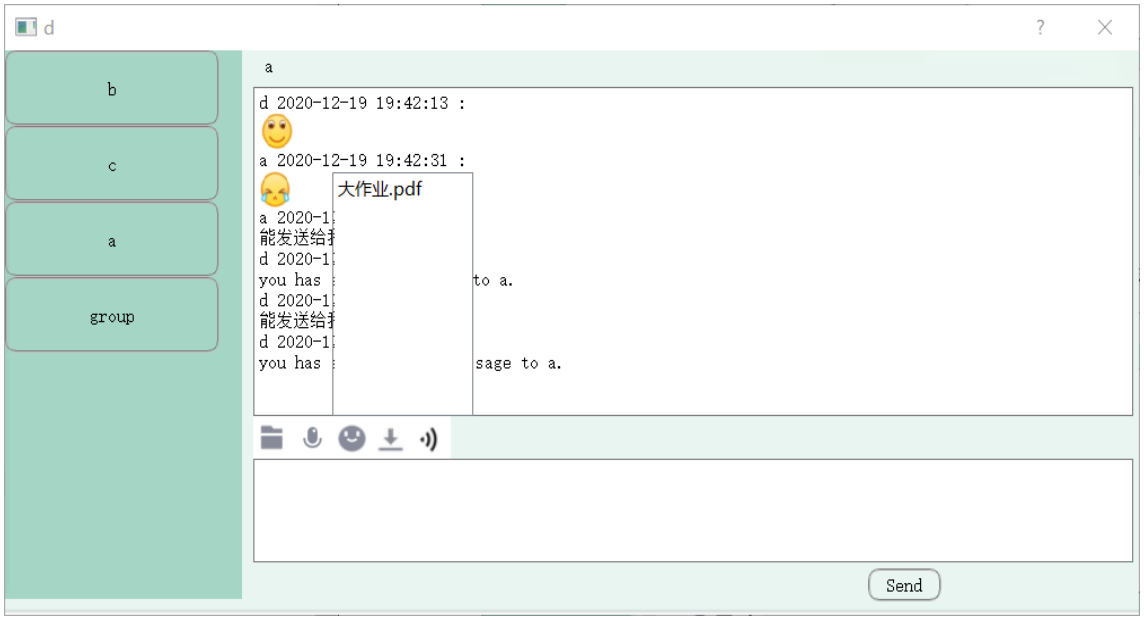

图 29: 文件下载列表

下载结果如图所示：

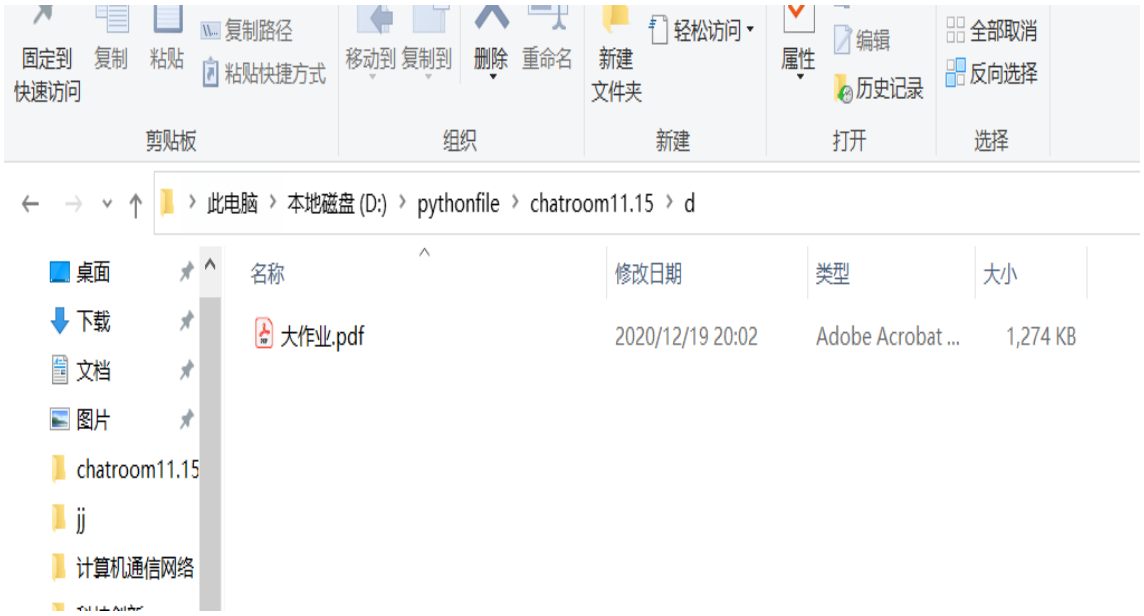

图 30: 文件大作业.pdf 已经被下载

同样的，在 group 群组中，所有在线在用户都可以进行各种类型的通信，结果如下图所示：（以 a、c、d 为例）

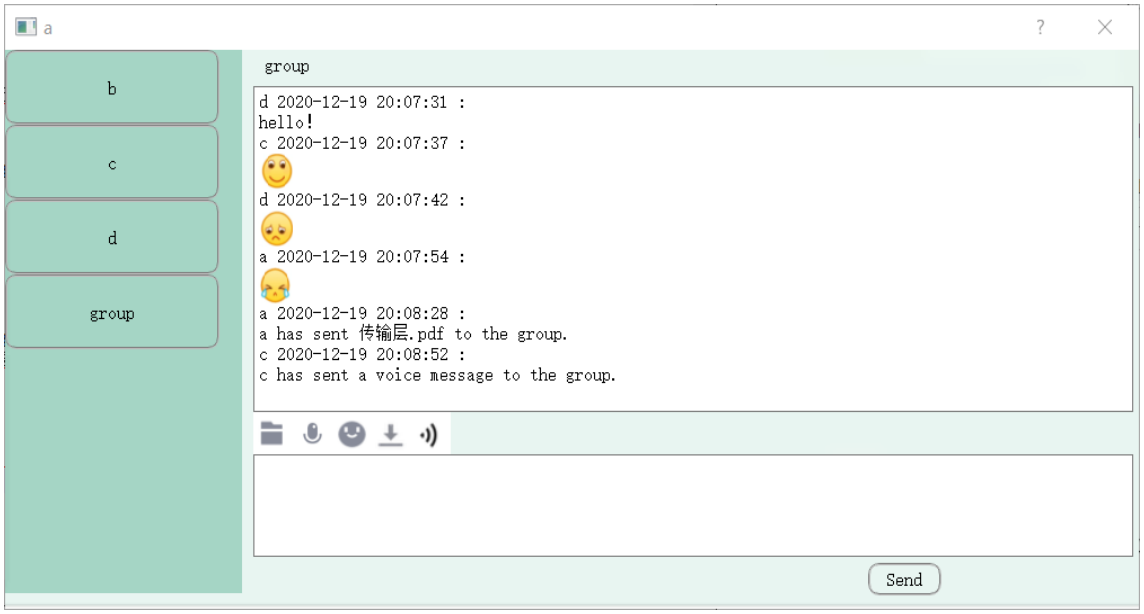

图 31: 群组聊天 a 的界面

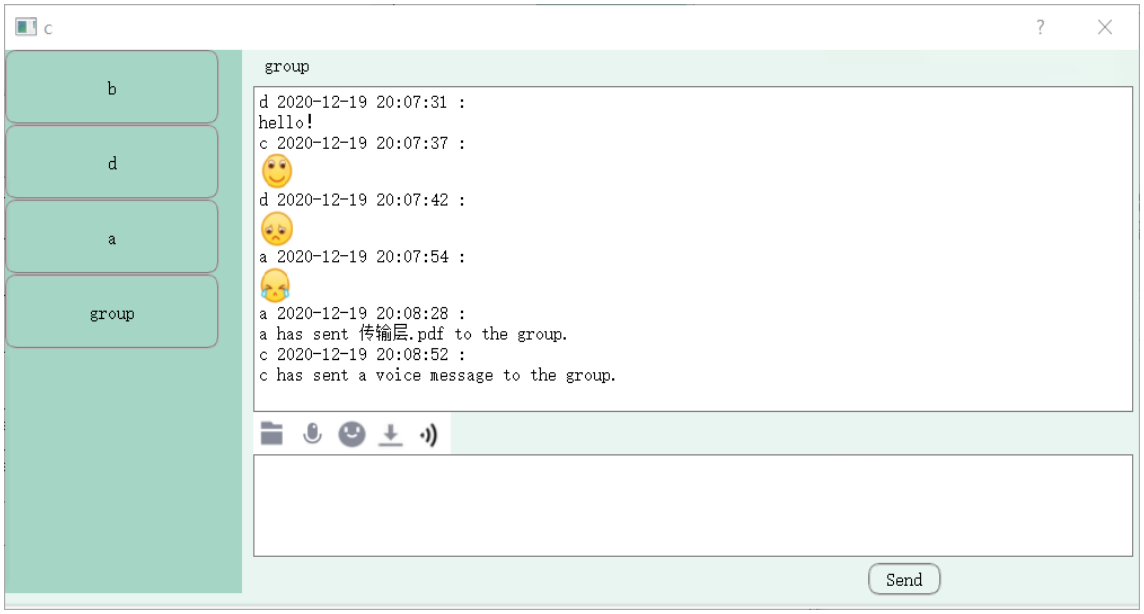

图 32: 群组聊天 c 的界面

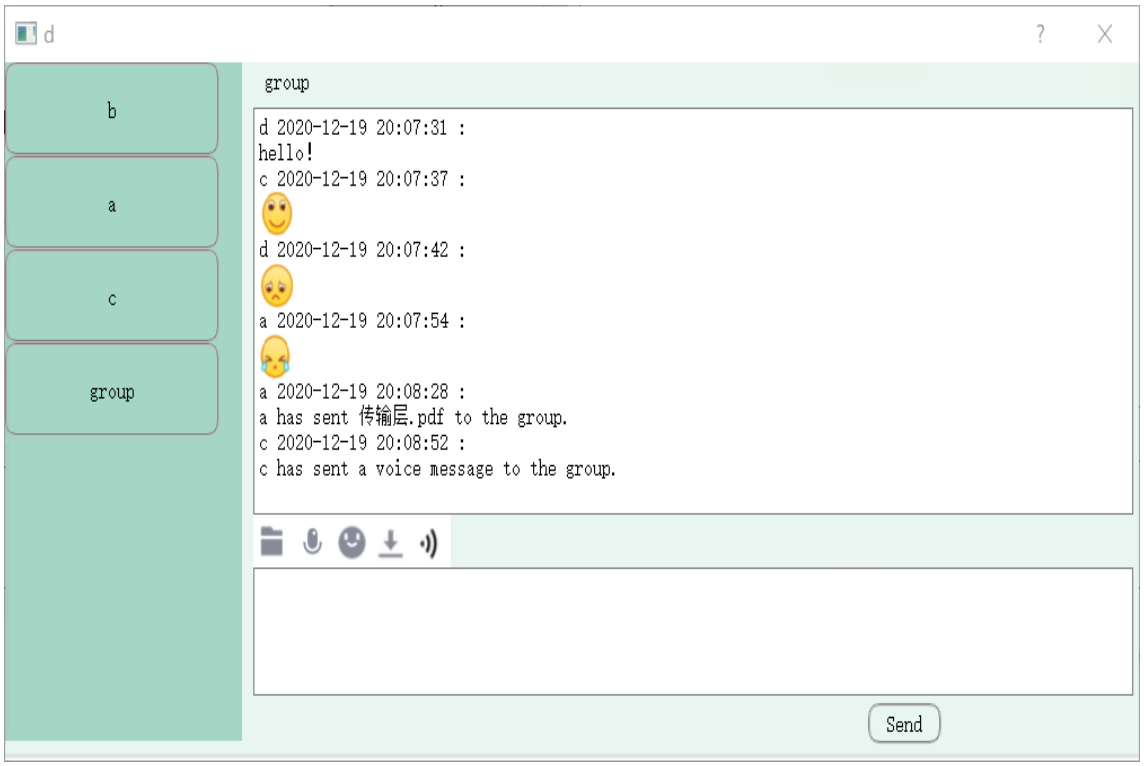

图 33: 群组聊天 d 的界面

5 遇到的问题以及解决方法

5.1 滚动区域

由于客户端获取用户列表的无法预知的，会随着注册人数的增加而增加。当用户较多时，采用滚动区域。具体的思路为，先构造一个 QWeight 类的对象，将其中放入给定数量的按钮，一个用户对应一个按钮，再使用 setWeight 函数将窗口变成滚动区域。但是在实际操作过程中，滚动区域一直没有显示，我上网搜索了许多博客，发现自己思路没有错，但是在构造滚动区域的时候并没有把 self 做为参数传入，导致滚动区域找不到显示的窗口，从而造成了不显示的错误。这看似小小的 bug 却耗费了我大量的精力去解决。

5.2 开发过程中的时序问题

在处理消息的过程中，是采用登录界面、注册界面、聊天界面的类调用 connect 类对象中的状态信息，来对用户进行反馈。在最开始的时候，我发现服务器端可以接受到来自客户端的消息，但是客户端接收不到服务器的消息，客户端总是会崩溃。我一直不能理解这其中的原因。当我接收消息的代码段移到发送消息之后、调用状态变量之前时候，我却发现客户端没有崩溃，可以接收到来自服务器的消息。我这才明白，服务器处理消息和反馈消息是有时延的，而之前客户端发送完数据后，就直接在调用状态信息，而此时客户端还没有接受到来

自服务器的反馈，导致了状态变量为被定义而先被调用，最终客户端程序崩溃。具体的解决办法是调用 time.sleep 函数，让客户端停滞一段时间，等待服务器消息的到来。如此，问题得以解决。

5.3 用户列表按钮的问题

由于用户列表的长度是未知的，故采用 for 循环进行创建按钮。然而按钮个数的不确定性导致了无法为每一个按钮创立一个触发函数，只能所有按钮共享一个触发函数。然而普通的触发函数是没有参数的，通过上网搜索，采用了嵌套函数的方法解决着这个问题。

```
def cuser(self, t):
    def calluser():
        """
        好友按钮触发函数
        """
        self.user.setText(str(t))
        self.chatuser = t
        self.sendText.show()
        self.sendtxtButton.show()
        self.fileButton.show()
        self.voiceButton.show()
        self.emojiButton.show()
        self.downloadButton.show()
        self.playButton.show()
        self.grprecvText.clear()
        #根据选择的好友改变消息框的信息
        for item in con.recmsg:...
        self.grprecvText.show()
    return calluser
```

图 34: 函数的嵌套

5.4 文件过长的问题

一般情况下（各种教程）把 socket 类的 recv 函数的参数设置为 1024，然而当传输较大文件是会出现问题，即已经读取了 1024 字节仍未出现结束符 EOF，使得服务器端报错，此时可以将参数设置得大一些（不可超过最大值）。经过调整后，聊天程序可传输近 100M 的文件。当文件更大时，可采用分段传输，由于其中的逻辑过于复杂，本次作业并未涉及。

6 程序的不足之处

由于时间的限制，聊天程序也有很多不完善的地方，例如好友问题、群组太过单一、只能和在线用户通信、语音消息播放不够方便等。对于这些问题，我会在后续的工作中继续完善与解决。

7 体会与建议

本次课程作业中，我独立完成了项目架构，项目实现，项目测试，报告撰写等过程。尽管项目实现过程中遇到许多困难，但好在最终都一一解决，收获颇丰。现在做出如下总结：

1. 独立思考解决问题的能力得到提升。在以前的 C++ 和数据结构等课程的学习中，编程遇到问题我都是请教一位身边的大神同学。然而这位同学对 socket 库了解甚少，所以这次课程作业，我遇到了 bug 学会自己在 CSDN 或者 stackoverflow 中找到答案。在今后的学习中，我也会先独立思考，查阅资料解决问题，而非直接向同学请求帮助。

2. 初步了解了 Python 网络编程的基本过程。由于之前的学习的大都是理论课程，并没有太多的具体实践，而上学期的软件工程课程的微信小程序也是并未付诸实际运用。然而，我已经和室友成功地通过这个聊天程序进行通讯。即使这个程序功能、界面都仍有很多不足，但这种从 0 到 1 的实践过程依然带给我许多快乐。

3. 纸上得来终觉浅。这次开发的整体的逻辑很容易构思，但实际上做的过程中才知道会有许多的 bug，比如滚动区域不显示的问题。漏掉 self 参数是课堂上、书本上都不会涉及的，只有在实际过程中才遇到的问题。只有多动手才能真正认识到自己的不足之处。

4. 代码规范问题。客户端的时序问题导致了调用为定义变量这一问题的出现，程序最终崩溃。这也不仅是时序错误，也反映了我代码不规范的问题。诸如此类的状态变量应该在类的构造函数中先定义并赋予初值，这样即使时序有问题，程序也不会崩溃，在 debug 时也会更加方便。

8 致谢

感谢姚老师在在课堂上对网络知识和大作业相关要求的讲解，对项目开发帮助很大。

感谢其他同学对我的帮助，这次项目的成功离不开他们的帮助。