



上海海事大学
Shanghai Maritime University

操作系统课程设计 网络聊天室

院(系)名称: 信息工程学院

专业名称: 计算机科学与技术

学生姓名: 孙思进

指导教师: 毕坤 老师

二〇二一年十一月

摘 要

本报告介绍了操作系统课程设计作品：《网络聊天室》的原理组成。本软件通过 *Pythony* 语言编写，使用 *Socket* 通信的方式，实现了点对点、点对面的文本信息传递功能，巧妙利用 *FTP* 服务器的功能实现了图片传输、视频传输、音频传输等功能。最终的成品运行在阿里云主机上，实现真正意义上的网络聊天功能。

关键词: 操作系统; 网络聊天室; *socket*; *C&S*

Key words: Operation System; Internet Chat; *socket*; *C&S*

目 录

摘要	I
1 引言	1
1.1 背景	1
1.2 编程语言	2
1.3 配置环境	2
2 设计分析	3
2.1 性能分析	3
2.2 需求分析	3
2.3 功能分析	4
2.4 数据库	5
3 系统功能实现	6
3.1 登录	6
3.1.1 原理	6
3.1.2 功能设计	6
3.2 注册	7
3.2.1 原理	7
3.2.2 功能设计	7
3.3 文本发送	8
3.3.1 原理	8
3.3.2 功能设计	9
3.4 图片	9
3.4.1 原理	9
3.4.2 功能设计	9
3.5 多媒体影像	10
3.5.1 原理	10
3.5.2 功能设计	10
3.6 队列	10

3.6.1	优先级	10
3.6.2	应用	10
3.7	消息记录	11
3.7.1	存储	11
3.7.2	读取	11
3.8	嵌套字	11
3.9	云服务器	12
3.9.1	概述	12
3.9.2	端口开放	12
4	系统测试	13
4.1	登录界面	13
4.1.1	按键提示	13
4.2	聊天界面	14
5	开源	16
5.1	项目	16
5.2	参考	16
5.3	L ^A T _E X 模板	16
	参考文献	17
	附录 A 代码	18
A.1	客户端	18
A.2	客户端	18

1 引言

1.1 背景

二零二一年正是 5G 通信的商用元年，回顾初代互联网的诞生，通信方式改善了人们的生活，从最初的短信单向通信，到 *Microsoft MSN*，从 *Xiaomi* 米聊，到 *Tencent QQ*，再到 *Tencent WeChat*，大大提高了人们的沟通方式。



图 1.1 *MSN* 中国



图 1.2 *Tencent QQ*



图 1.3 *Tencent WeChat*

1.2 编程语言

本软件主要采用 *Python* 3.7 进行编写，使用的库如表1.1所示

表 1.1 *Python* 配置包

序号	名称	功能
1	<i>Socket</i>	实现点对点通信
2	<i>Tkinter</i>	图形化 <i>GUI</i>
3	<i>openCV</i>	显示图片、视频、音频等多媒体信息
4	<i>ftplib</i>	传输文件
5	<i>json</i>	打包数据，生成嵌套字
6	<i>threading</i>	实现多线程
7	<i>py2neo</i>	<i>Neo4j</i> 在 <i>Python</i> 运行的基础库

1.3 配置环境

本软件的编译和运行均基于表1.2的环境下配置，此外，设计报告撰写使用 \LaTeX 。

表 1.2 系统环境

序号	名称	版本
1	客户端	<i>Windows 10 Professior</i>
2	服务器端	<i>Windows 10 DataBase</i>
3	<i>PyCharm</i>	<i>PyCharm Community</i> 2021.2.2
4	<i>Java</i>	<i>Java JDK</i> 18
5	\LaTeX	<i>TexLive</i> 2020
6	<i>Neo4j</i>	<i>Neo4j Community</i> 4.3.5

2 设计分析

2.1 性能分析

编写一个网络聊天室，对于服务器来说，需要做到如下几个功能：

- 直接
- 快速
- 低延迟
- 安装便捷
- 多媒体化

涉及到点对点通信，*socket* 的通信方式显然是首要选择方式，它基于 *TCP/IP* 协议，实现了点对点的通信，并且传输速度不受其他因素限制。由于采用嵌套字传输方式，在安全方面可能存在一定风险，有被抓包分析的可能性。

对于私人和多人聊天有着不同的功能要求，私人聊天之间需要较高的加密程度，确保信息安全；而多人公开聊天中，则需要考虑多并发的情况，对于一个消息文本，短时间瞬发需求极大，软件需要保证不漏发、不错发的前提下高效率的传输信息。

本软件使用 *python* 语言开发，不涉及库文件的安装情况下，源文件不到 1mb，并且 *python* 的同一个代码源文件可以同时运行在 *Linux MacOS Windows* 系统运行，实现了轻量化的设计需求。

在高带宽的信息化时代下，软件不能仅仅局限于文本的收发，消息的种类更应该多样化，复杂化。因此，不仅要考虑文本的传输，更应该关注到图片、视频、语言的收发功能。

2.2 需求分析

为了做到高效的通信，减少客户端设备的硬件要求，本软件采用 *C/S* 端的假设方式，*C* 端即为 *Client*：客户端，*S* 端即为 *Serve*：服务器端。

本软件初代运行在局域网中，将服务器端假设在局域网的某台电脑中，实现了群聊和单聊的通信功能。考虑到在一定程度上无法每时每刻连接内网的网关，故后续将本软件的服务器端架设在阿里云上，实现了广域网通信。

客户端/服务器模型最终归结为一个“请求/应答”的关系。一个请求总是第一个颁发给客户端，然后服务器总是被动地接收请求，返回结果给客户需要。在客户

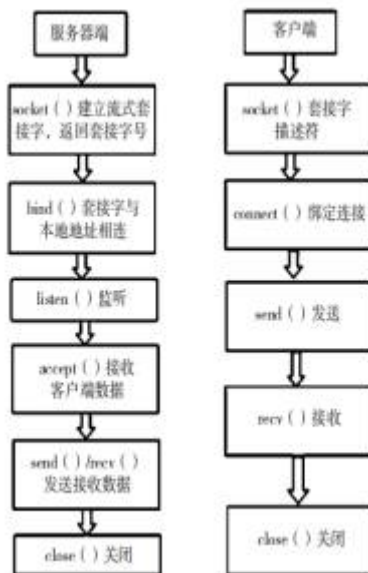


图 2.1 通信原理^[1]

端发送一个请求时, 服务过程一直处于休眠状态。在一个客户端请求时, 服务过程是“唤醒”和为客户提供服务, 根据客户的要求进行回复。^[2] 因此采用 socket 框架。

2.3 功能分析

如图2.2所示, 整个软件的功能主要分为几个方面:



图 2.2 网络聊天室需求分析

对于客户端而言, 主要负责用户的登录和注册, 以及最重要的收发消息。

对于服务器端而言, 负责处理收到的文本, 将其转发到各个用户所在的 *ip* 地址的端口上。

2.4 数据库

数据库使用了 *Neo4j*，一种基于离散数学中图论原理的轻量数据库，普遍被称为知识图谱，相比较传统的 *MySQL*、*SqlLite* 而言，其基本的结构由节点（*Node*）、关系（*Relationship*）和标签（*Label*）组成，通过这三个属性可以构建出传统数据库没有的功能。

如图所示，源文件为一个有关于心理测试的调查信息，通过简单的 *NLP* 处理后，可以将原来的字段拆分成多个关系，形成有向关系网路。

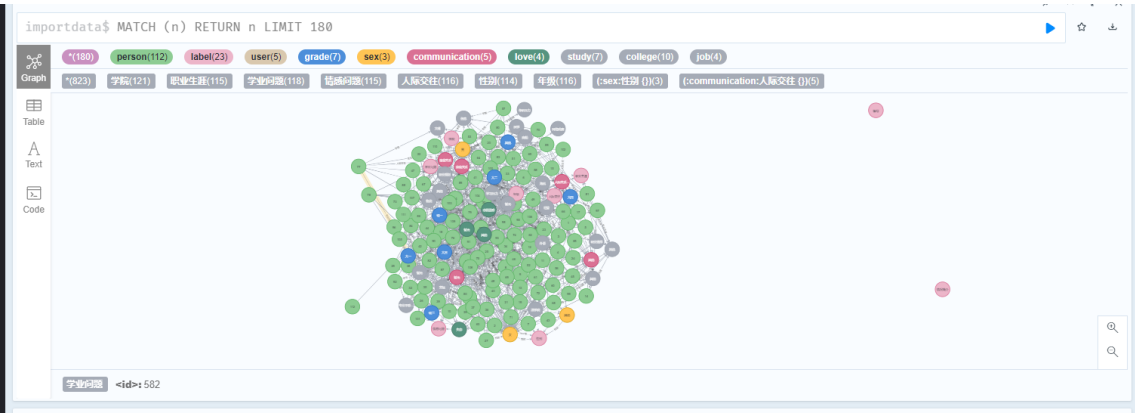


图 2.3 通过 *Neo4j* 形成的复杂关系图

调用 *Neo4j* 数据库的查询极为便捷，其编写语言全部源于 *Java(jdk-12)*，不仅支持在 *Java* 中查询，更能在 *Python*、*C++* 等主流语言中调用，同时也可以以 *jpg*、*png* 和 *json* 等多种格式保存。

此外，*Neo4j* 可以储存的数据极为庞大，如1.2所示，对于一个网络聊天室来说足矣。

表 2.1 *Neo4j* 数据容纳量

S.No	构建基块	数量
1	节点	约 350 亿
2	关系	约 350 亿
3	标签	约 275 亿

3 系统功能实现

3.1 登录

3.1.1 原理

登录部分涉及到验证、确认等操作，流程如图3.1所示：

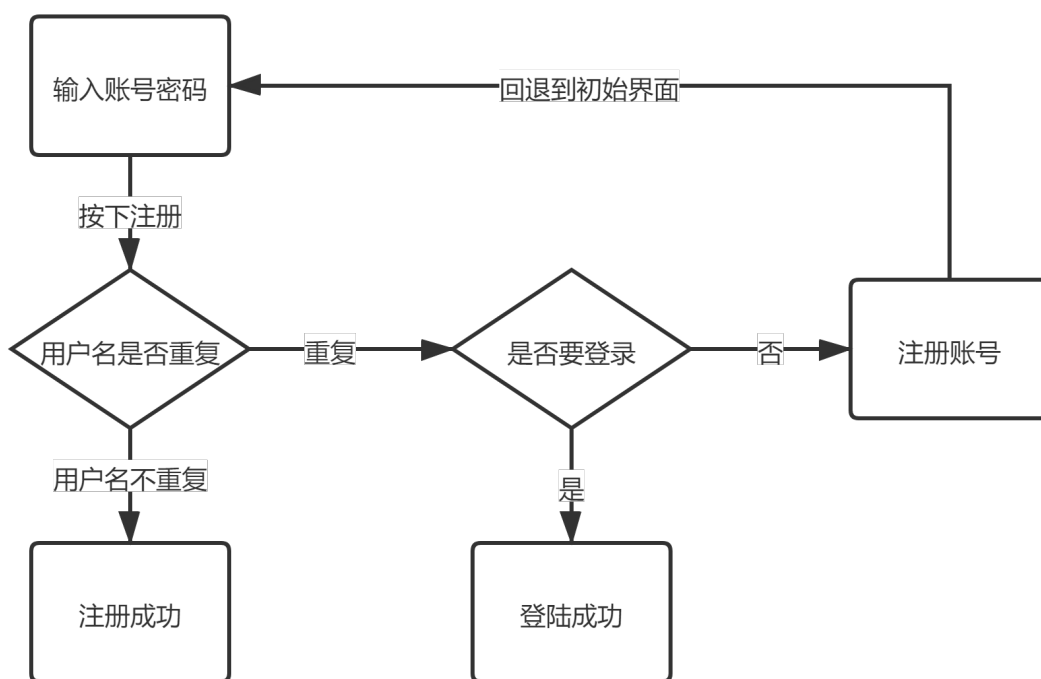


图 3.1 登录系统流程图

3.1.2 功能设计

用户在输入账号和密码以后，按下 *Enter* 或者登录按钮，程序将输入的字符串转化为嵌套字，传输到服务器端，服务器端启动数据库查询功能，如果用户名存在，但是密码错误，返回-1，客户端弹出“密码错误”提示；如果用户名不存在，返回 0，客户端弹出“用户名不存在”提示；如果用户名和密码都匹配，返回 1，客户端弹出“登录成功”提示，跳转到聊天框。包括注册系统在内的返回值均列在了表3.1中。

表 3.1 登录 & 注册返回值

返回值	提示	备注
-1	密码错误	用户输入了存在的账号，但是密码不匹配，保留该界面
0	用户名不存在	用户输入了不存在的账号，不进行密码匹配，保留该界面
1	登录成功	用户输入了匹配的账号和密码，登陆成功，跳转到聊天栏
2	用户名已存在	用户注册的账号已经存在，保留注册界面
3	注册成功	用户注册账号成功，保留注册界面录

3.2 注册

3.2.1 原理

使用流程图3.2简要的画出了注册系统的设计思路。

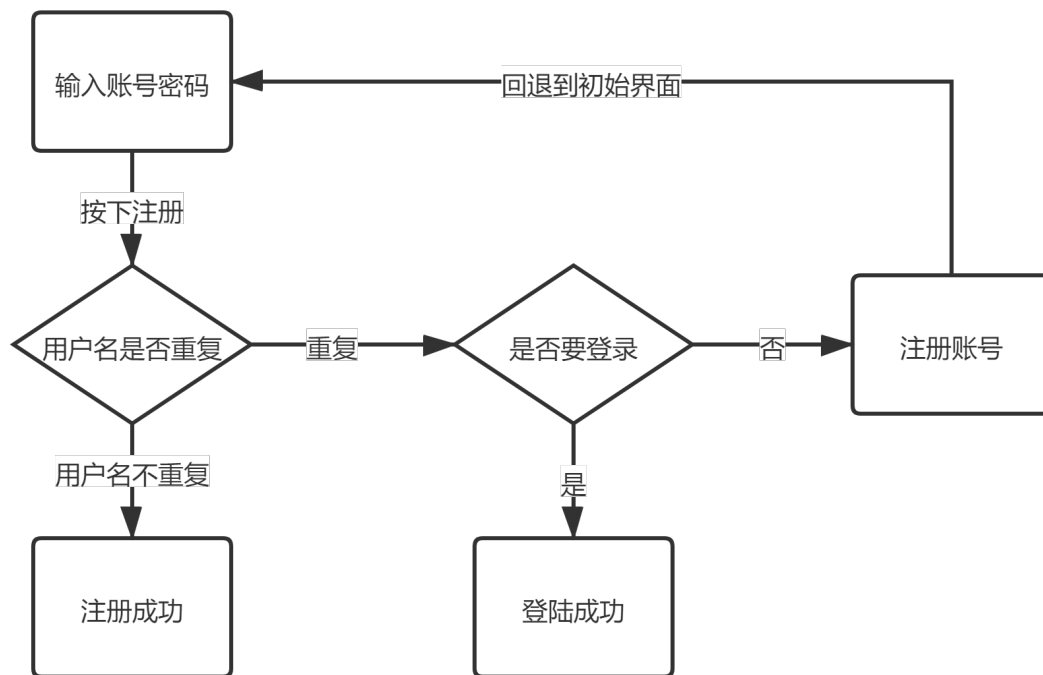


图 3.2 注册系统流程图

3.2.2 功能设计

用户在输入账号和密码以后，按下注册按钮，和登录相同，服务器先匹配用户名是否存在，如果存在，返回 2，客户端提示“用户名已存在”；否则，返回 3，客

户端提示”注册成功“，此时，按下登录即可使用注册的账号进行登录, 返回值对于的效果见表3.1所示。

3.3 文本发送

3.3.1 原理

通过流程图3.3展示了文本发送的编写方法。

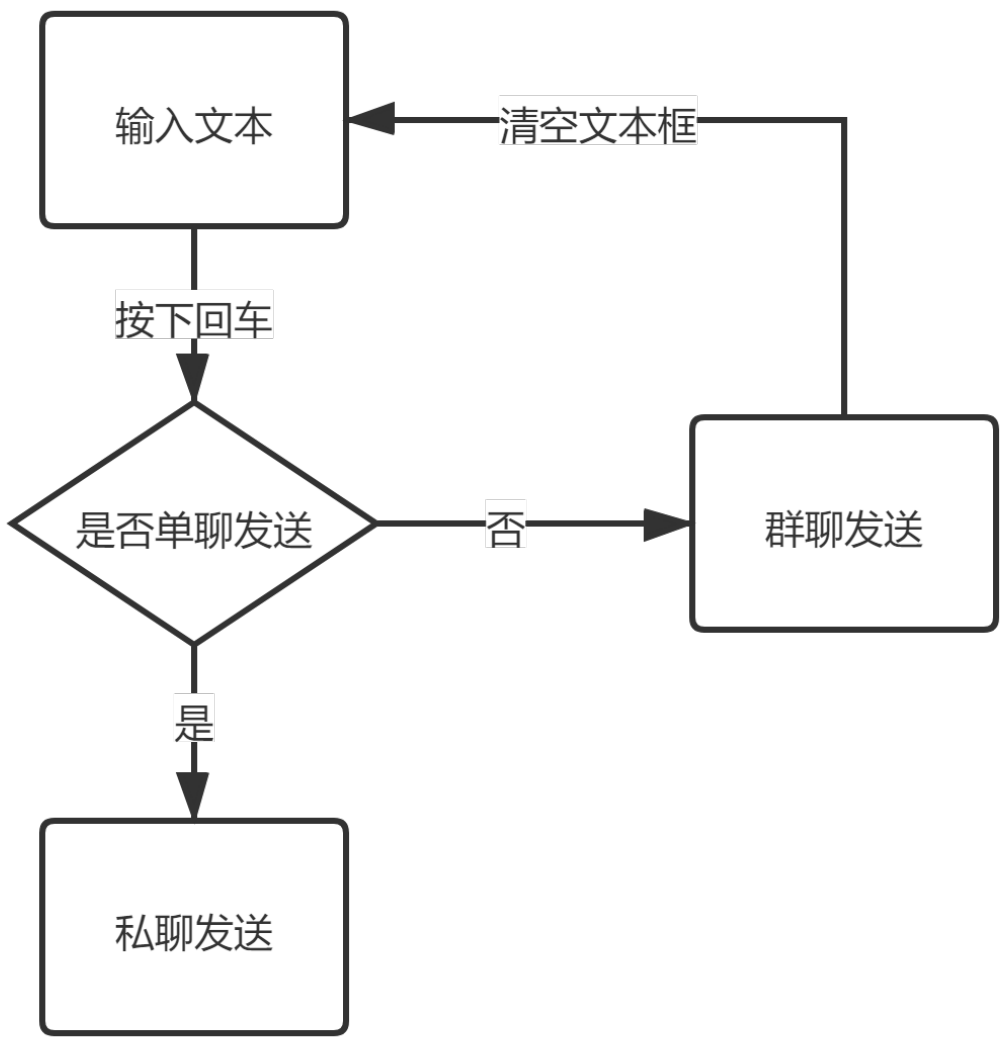


图 3.3 文本发送流程图

3.3.2 功能设计

当登录成功后，将显示聊天窗口，左上角部分为文本显示区域，显示别人发送的群聊、单聊文本；左下角为输入框，输入文本，其右边是发送按键，按下后即可发送；右侧是功能栏，包含发送图片、发送视频等功能。

若用户想要群聊，只需要输入文本 $Text_0$ ，每个人都会接收到文本 $Text_0$ ；若用户想要单聊，则在原先需要输入的 $Text_0$ 后额外输入 ~ 用户名，即字符串3.1：

$$Text = Text_0 + \sim + UserName \quad (3.1)$$

例如，“我”想要和用户“Tom”私聊，对话内容是“你今天吃饭了吗？”，那么“我”应该输入字符串3.2。

$$Text = \text{你今天吃饭了吗} \sim Tom \quad (3.2)$$

3.4 图片

3.4.1 原理

图片发送的实现运用了 *FTP* 服务器，间接实现了图片的传输，同时也能实现聊天记录保存的功能。如流程图3.4所示，图片传输的方式：

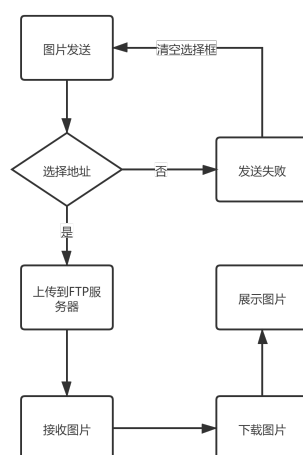


图 3.4 图片发送流程图

3.4.2 功能设计

首先使用 *openCV* 的 *imread* 功能读取图片，转化为点矩阵，再使用 *FtpLib* 模块连接 *FTP* 服务器，输入上传指令 *STORFileName* 上传，输入 *RETR* 下载，再使用 *imshow* 函数显示图像，至此，完成了图片的传输和显示功能。

由于图片的读取和保存都使用 *openCV* 模块，这样可以确保图片的读写方式相同，避免出现 *RGB565* 和 *RBG565* 不同读取方式，提升了程序的鲁棒性。

3.5 多媒体影像

3.5.1 原理

多媒体文件发送的方式和图片发送相类似，也采用了 *FTP* 的间接方法，通过调用 *openCV* 模块实现视频、音频的播放。

3.5.2 功能设计

openCV 提供了视频文件的读取方式 *VideoCapture(filePath)*，通过切割读取的视频每一帧，这时候视频转化为了图片，可以沿用图片的读取方式来显示。

3.6 队列

3.6.1 优先级

当服务器处于高并发状态时，如何才能保证消息的正常收发呢？这时就需要使用队列了。

如图3.5所示，队列的工作原理遵循“先进先出”的原则，这样就可以保证先发送的消息先传输到各个用户，后发送的消息被后接受，从而确保时序的稳定。

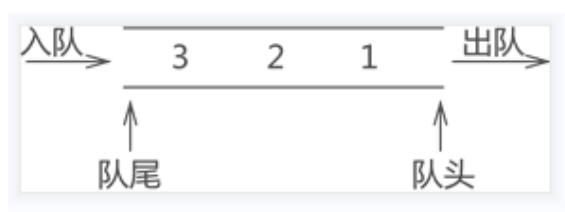


图 3.5 队列优先

3.6.2 应用

在创建完队列后，如果有某个用户发送消息，那么这个消息将传入队列的最前端，记为 Q_0 ，若第一个消息还未被处理，第二个消息又被传入，那么第一个消息的位置将转移到 Q_1 ，第二个消息的位置代替原先第一个消息的位置，即 Q_0 。

3.7 消息记录

3.7.1 存储

消息的存储使用数据库 *Neo4j*, 通过建立节点存放消息, 其标签分别为 *sender*、*receiver* 和 *text*, 使用 *Py2neo* 可以较为快速的创建一个节点, 其语法如代码3.3。

$$\text{CreateRelationship}(\text{graph}, \text{label1}, \text{attrs1}, \text{label2}, \text{attrs2}, \text{name}) \quad (3.3)$$

3.7.2 读取

这时我们可以通过数据库查询代码3.4来获取某个发送者发送的消息, 通过代码3.5来获取某个人接收到的消息。

$$\text{MATCH } (n : \text{sender}) \text{ RETURN } n : \text{text} \quad (3.4)$$
$$\text{MATCH } (n : \text{receiver}) \text{ RETURN } n : \text{text} \quad (3.5)$$

我们既可以通过 *Neo4j* 输入该命令, 也可以在 *Python* 添加这个语段, 输出以 *json* 格式打包, 可以快读在其他的数据读取程序中运行。

3.8 嵌套字

无论使用哪一种地址家族, 嵌套字的类型只有两种。一种是面向连接的套接字, 即: 在通信之前一定要建立一条连接, 就像跟朋友打电话时那样, 这种通信方式也被成为“虚电路”或“流套字节”。面向连接的通信方式提供了顺序的、可靠的、不会重复的数据传输, 而且也不会被加上数据边界。这也意味着, 每一个要发送的信息, 可能会被拆分成多份, 每一份都会不多不少地正确到达目的地。然后被重新安顺序拼装起来, 传给正在等待的应用程序。

无论你使用哪一种地址家族, 嵌套字的类型只有两种。一种是面向连接的套接字, 即: 在通信之前一定要建立一条连接, 就像跟朋友打电话时那样, 这种通信方式也被成为“虚电路”或“流套字节”。面向连接的通信方式提供了顺序的、可靠的、不会重复的数据传输, 而且也不会被加上数据边界。这也意味着, 每一个要发送的信息, 可能会被拆分成多份, 每一份都会不多不少地正确到达目的地。然后被重新安顺序拼装起来, 传给正在等待的应用程序。

3.9 云服务器

3.9.1 概述

本软件运行在阿里云服务器，公网 IP 地址为3.6, 内网 IP 地址为3.7:

47.100.93.63 (3.6)

172.24.12.68 (3.7)

3.9.2 端口开放

阿里云平台涉及到 *Neo4j*、*FTP* 等多个服务器的同时部署，在表3.2中列出了所有需要使用到的端口。

表 3.2 端口及其作用

端口号	用途	备注
21	<i>FTP</i> 服务器 - <i>Port</i> 端口	<i>FTP</i> 源文件位于 C:/FTP/
80	服务器首页端口	<i>IIS</i> 服务提供的主页
443	个人网页端口	个人博客主页的端口
1023-1033	<i>FTP</i> 服务器 - <i>Passive</i>	为了确保连接稳定，开启 <i>FTP</i> 访问模式
6666	网络聊天室端口	提供聊天室对外的端口
7474	<i>Neo4j</i> 端口	知识图谱数据库的对外端口
8888	<i>Jupyter Notebook</i>	<i>Jupyter Notebook</i> 的对外端口

4 系统测试

4.1 登录界面

登录界面如图4.1所示:

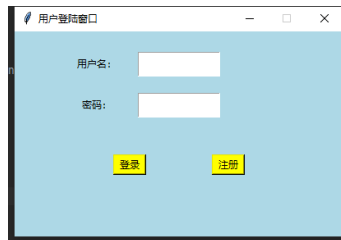


图 4.1 登录 UI

4.1.1 按键提示

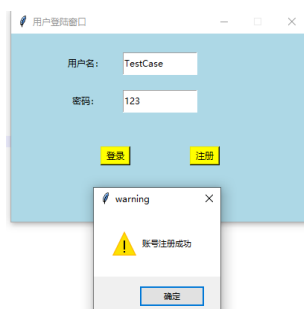


图 4.2 注册成功提示

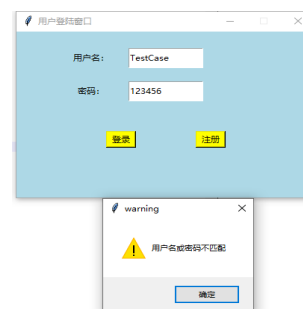


图 4.3 密码错误提示

上述图分别展示了注册成功和登录失败两种状态。



图 4.4 账号不存在提示



图 4.5 登录成功

上述图分别展示了登录成功和账号不存在两种状态。

4.2 聊天界面

聊天界面如图4.6所示:

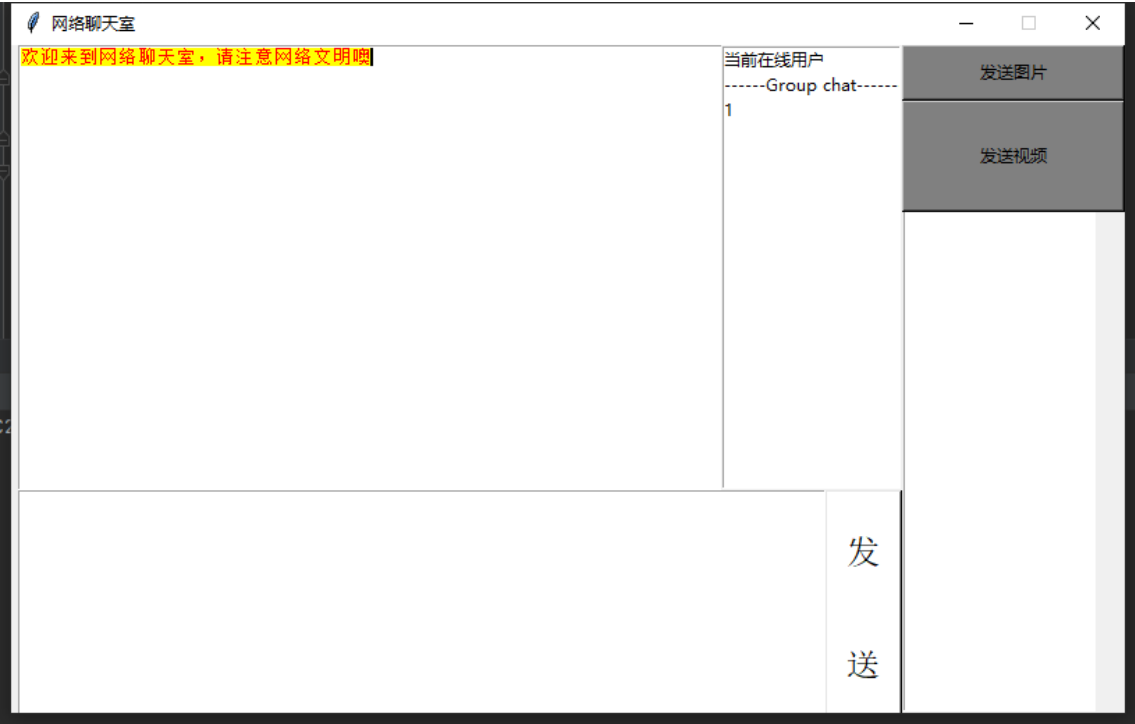


图 4.6 聊天界面 UI

当发送消息以后，如图4.7所示，信息框展示收到的信息。

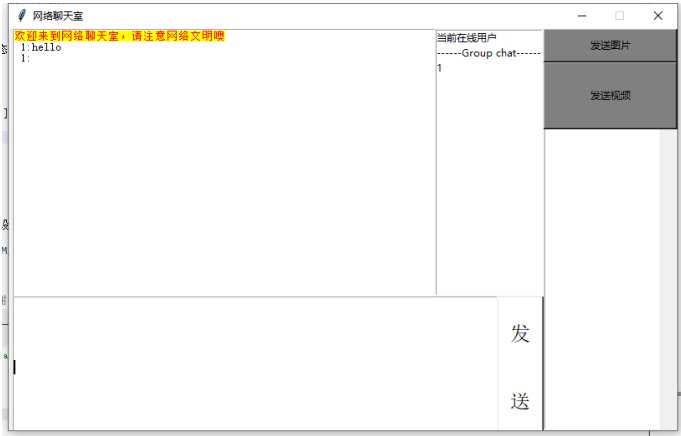


图 4.7 聊天

当发送图片以后，如图 4.8 所示，*openCV* 打开图片。

当发送视频以后，如图 4.9 所示，*openCV* 打开视频。

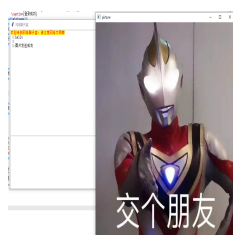


图 4.8 发送图片

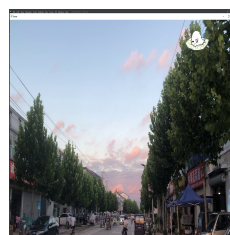


图 4.9 发送视频

5 开源

5.1 项目

本项目开源在 *Github*, 来自 *Stanleysun233*
访问本项目:

$\textit{https : //github.com/StanleySun233/OS_InternetChatRoom}$ (5.1)

5.2 参考

本项目部分 *Neo4j* 代码参考于武汉大学张曙光教授。

$\textit{https : //www.bilibili.com/video/BV1Wt4y1178d}$ (5.2)

5.3 L^AT_EX 模板

本报告使用的 L^AT_EX 模板修改于武汉大学本科学士学位毕业论文, 原作者为黄正华。

$\textit{http : //aff.whu.edu.cn/huangzh/}$ (5.3)

参考文献

- [1] 郭炳均, 王思晗. 网络聊天室的设计与实现 [J]. 信息与电脑 (理论版), 2020, 32(22): 89-90.
- [2] 张淑坤. 基于 SOCKET 嵌套字的 IM 系统设计与实现 [J]. 数字技术与应用, 2013(05): 204.

附录 A 代码

A.1 客户端

```
1      import os
2      import socket
3      import tkinter
4      import tkinter.messagebox
5      import threading
6      import json
7      import tkinter.filedialog
8      from tkinter.scrolledtext import ScrolledText
9      import demo.neo4j.Neo_Fun as NeoFun
10     import cv2
11     import numpy as np
12     import ftplib
```

A.2 客户端

```
1      import os
2      import socket
3      import tkinter
4      import tkinter.messagebox
5      import threading
6      import json
7      import tkinter.filedialog
8      from tkinter.scrolledtext import ScrolledText
9      import demo.neo4j.Neo_Fun as NeoFun
10     import cv2
11     import numpy as np
12     import ftplib
```