目录

[1 引言 1](#_Toc74757177)

[1.1目的 1](#_Toc74757178)

[1.2需求 1](#_Toc74757179)

[2 设计和构建 2](#_Toc74757180)

[2.1基本设计 2](#_Toc74757181)

[2.2目录和代码 3](#_Toc74757182)

[2.3流程描述 5](#_Toc74757183)

[2.4流程图 7](#_Toc74757184)

[2.5运行情况及截图 7](#_Toc74757185)

[3 问题及解决情况 10](#_Toc74757186)

[3.1问题和解决 10](#_Toc74757187)

1 引言

# 1.1目的

基于UDP和TCP实现一个具有分享文件和文件下载功能的文件分享应用。

# 1.2需求

1.应用应该以图形界面的形式展示，包括各种弹窗放错和进一步的美观设计

2.用户可以选择分享、下载和分享管理实现想要的功能

3.分享时应为主机上所有有权限访问的文件，以绝对路径的形式保存在txt文件里

4.下载时需要在已知网络中搜索文件，搜索为关键字搜索，搜索结果展示给用户供其选择。

5.点击下载后，文件存入同一目录下的download文件夹中，有进度条显示下载进度

6.分享管理可管理已经分享的文件，即存在txt文件中的路径。首先要显示这些文件，然后对每一个文件有打开文件夹和删除操作

2 设计和构建

# 2.1基本设计

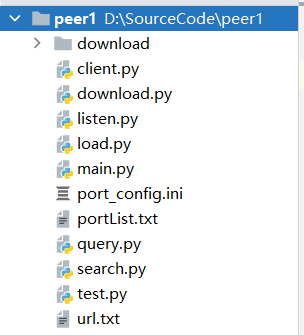
1.使用python的tkinter包实现图形界面编程，按钮与事件绑定执行对应的函数实现功能。

2.使用多线程的方法实现同时监听udp和tcp链接又保持用户界面

3.搜索功能的实现，用udp广播关键字，收到广播的主机解析并在本地搜索，若搜到则返回基于udp的ack确认，返回响应信息。

4.文件下载使用tcp实现。

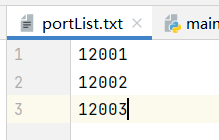
# 2.2目录和代码

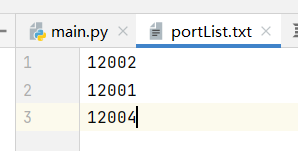


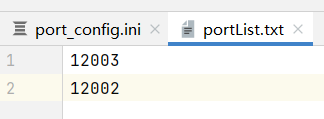
其中**.ini文件**是配置文件，配置本地的udp查询广播端口，udp确认端口以及tcp数据传输端口

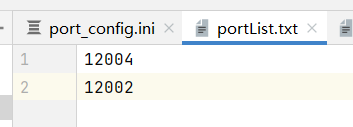
**portList.txt**中包含了一个与这个peer相邻的peer信息 用端口号表示

peer1的：

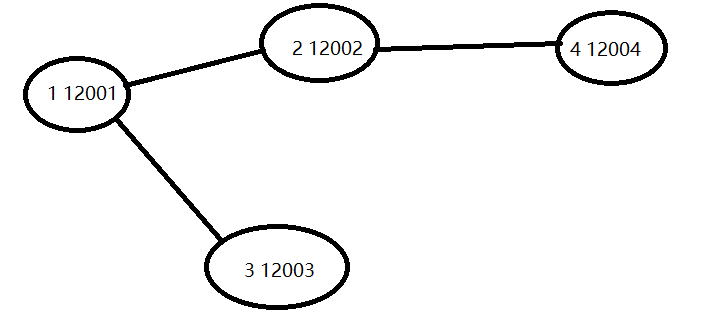
如12001端口表示的机器与自己、12002、12003相临

Peer2的：

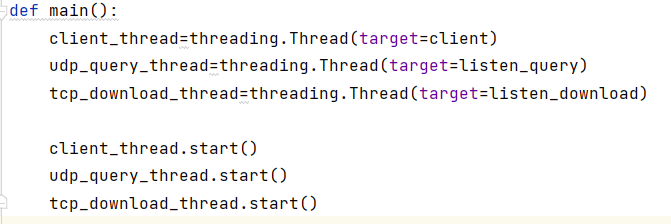
Peer3的：

Peer4的：

在测试案例中公有**4个peer** 它们的链接是这样的：

（编号 端口号）

main.py包含了三个线程



<https://github.com/pcc-CN/Query-flooding-based_Resource_Sharer/tree/main/Query-flooding-based_Resource_Sharer>

存在github上

# 2.3流程描述

启动程序，有客户端、监听udp、监听tcp三个线程同时启动

**客户端**

其中，分享文件要求用户输入本地文件的绝对路径，程序判断文件存在，则以绝对路径的形式加入文本文件url.txt中。

下载文件首先要求用户输入文件关键字，程序通过udp向局域网内广播查询消息，格式为

*message = 'query' + ' ' + keyword + ' ' + ip + ' ' + str(getudp\_ack\_port())+' '+ttl*

其中‘query‘是查询标志，keyword是搜索的关键词，ip是本机ip，udp\_ack的接收端口，以及ttl转发次数。

同时，对udp的ack监听listen\_udp启动。listen\_udp的搜索时间通过settimeout()确定，暂定为2s，如果两秒内未收到任何ack信息，则显示提示消息，若收到，则存储所有消息，绘制一个图形界面显示搜索到的信息。

用户可以选择需要的文件点击下载进行下载。

**监听udp**

监听开头为“query”的查询udp消息，如果收到则提取keyword与本地url.txt文件中分享的文件进行对比，如果找到类似的文件，则将一个或多个类似的文件的绝对路径打包，向（ip,udp\_ack\_port）发送一个确认消息，格式为

ack + filepath1\_file1\_size + filepath2\_file2 + …… + tcp\_port

tcp\_port告诉对方使用这个端口进行tcp连接

当用户选择需要下载的文件时，此线程调用函数download(source,tcp\_port)开始从指定的地址下载指定的文件。文件的下载使用TCP。

Download（）函数建立一个socket与传入的那个tcp\_port链接，根据source这个文件路径，和address这个地址，编辑一条请求消息，消息开头是“down”

格式为：

Message = 'down ' + filepath

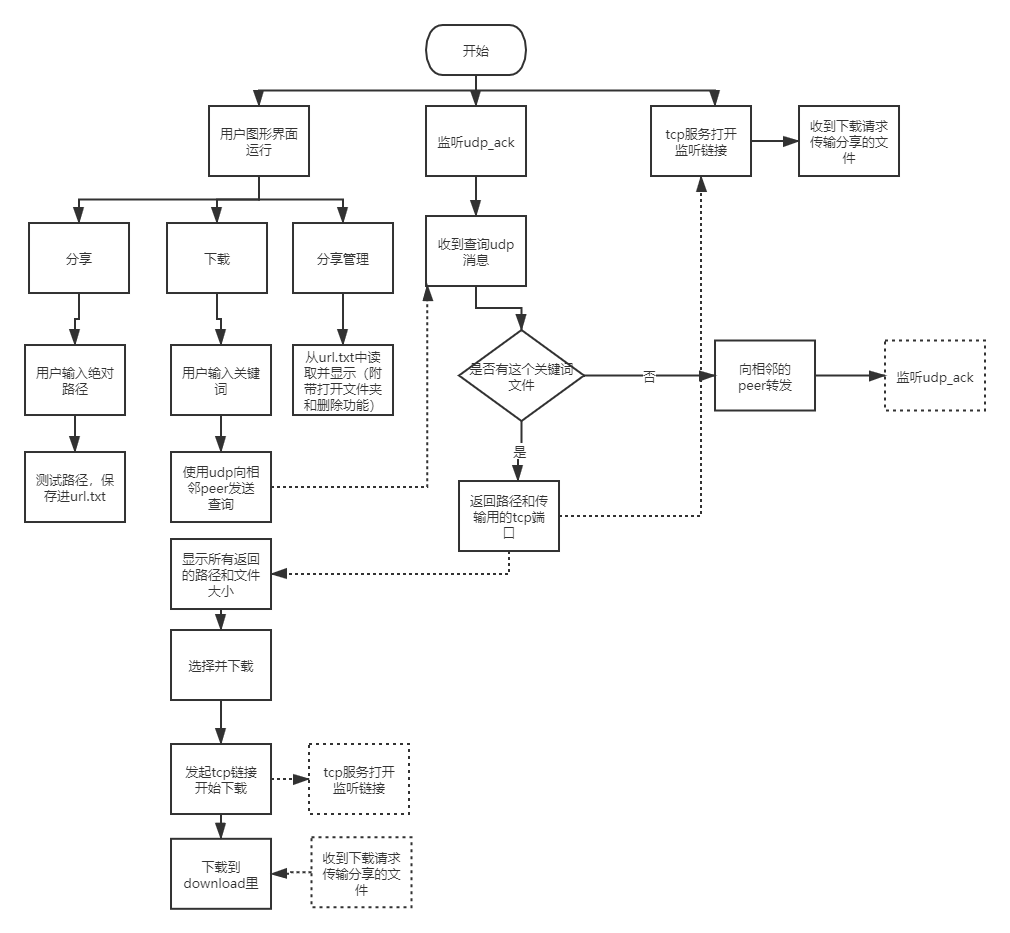
目的主机的返回消息被受到后开始下载，打开文件夹统一目录下的文件夹download开始向里下载对应的文件，同时有进度条显示（为了看到进度条，程序中设置了一个sleep（）实际使用时可以删除）

下载完成后由弹窗提示，文件可以在download文件夹下找到。

**监听tcp**

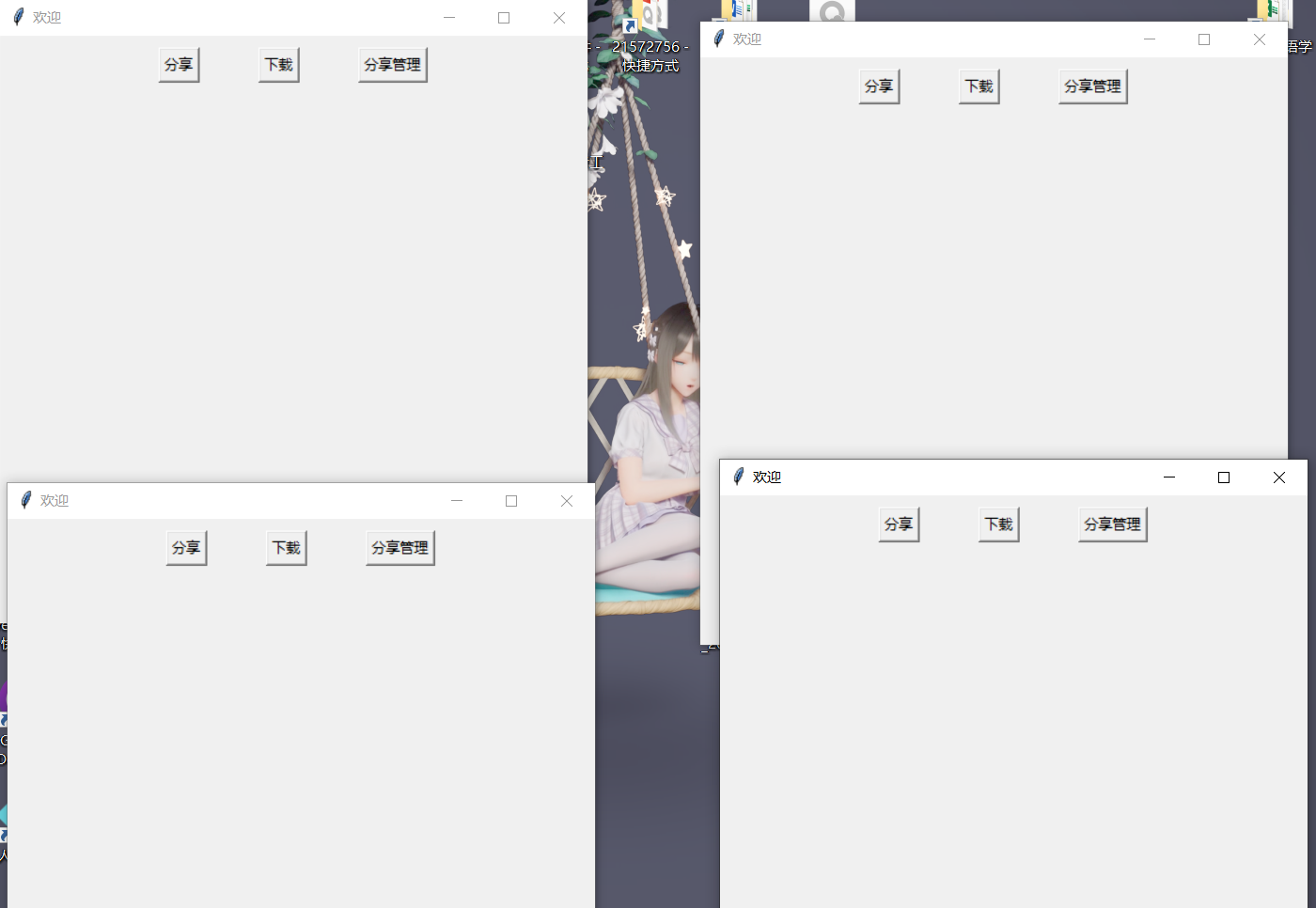
监听tcp线程运行listen\_download()函数，相当于服务器，等待网络中发起的down链接。当连接到请求主机并受到down请求后，解析请求，获取资源路径打开资源并传输。（注意，由于资源的打开使用了open（）打开文件夹和大文件可能出现异常，只获取异常，并没有处理此的问题）每次读取tcp最大的传输数量进行传输。在文件传输结束后发送‘end’信号，继续等待连接。

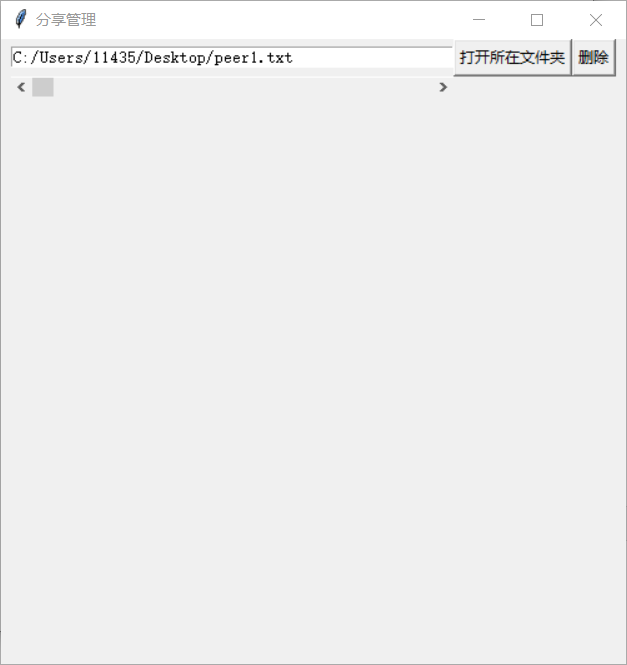
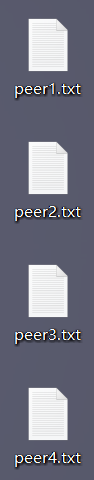
# 2.4流程图



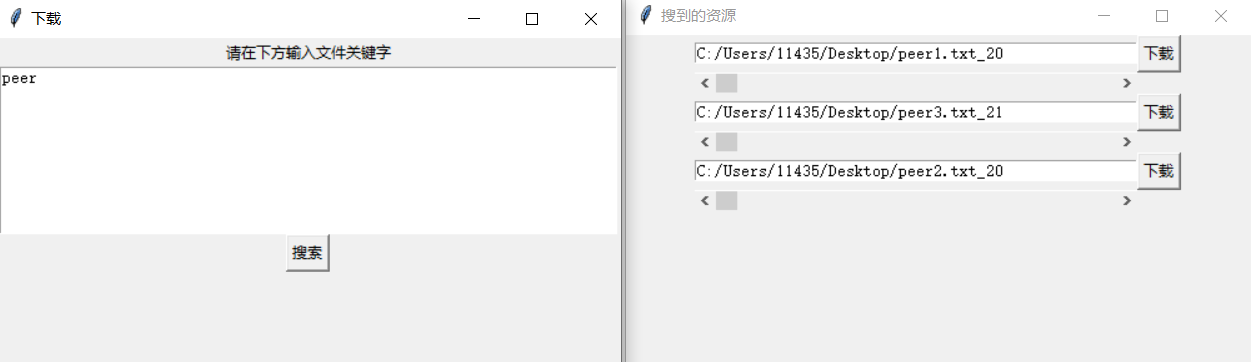
# 2.5运行情况及截图

首先模拟了4台电脑上运行这些peer，

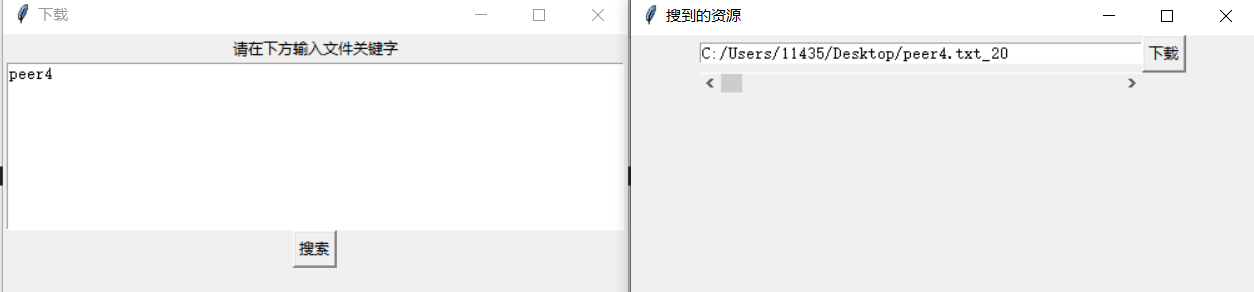
其中 所有的peer都暂时只分享了一个文件peer[i].txt。它们存在桌面上



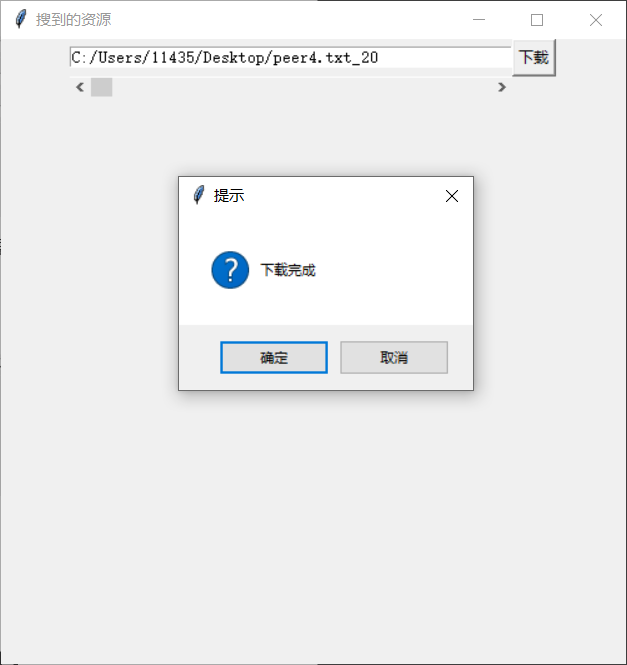
在peer1中搜索“peer”，得到peer1.txt,peer2.txt,peer3.txt及其大小（byte）



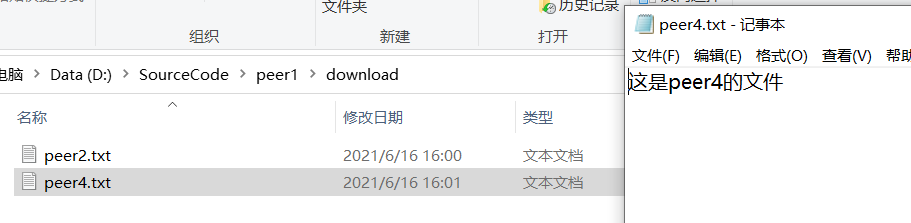
在peer1中搜索“peer4”，仍能得到与其不相连的4中的数据，这是由2转发给4，4再返回给1得到的



点击下载

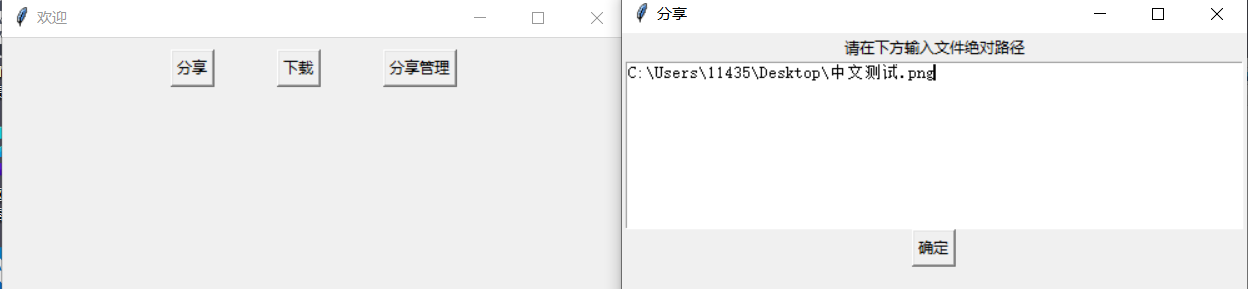


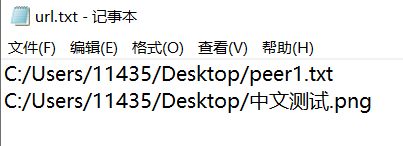
下载成功



如果想**分享**文件，如下操作

点击分享，输入本地某文件的绝对路径以添加



成功加入

3 问题及解决情况

# 3.1问题和解决

遇到的最大问题基本上都是关于图形化编程的，tkinter图形界面的逻辑没有学过，相较于有很多便利框架的js前端开发，显得有些繁杂。从0开始看csdn看文档看谷歌编程，导致出现了很多意想不到的bug和很多难以实现和修正的操作。

主要问题有：

1.button组件的command命令只能绑定无参的函数，如果要传入参数要使用使用StringVar()和Entry textvariable对参数进行绑定或者使用lambda，只能照着文章写但并不明白原理。若用到for in循环，传入的参数并不能绑定到每一个button上，因此还copy了部分事件处理的方法，使用bind绑定每一个按钮的事件，使用了更多内存，且按钮点击效果消失。

====🡺

解决：减少了部分图形界面的逻辑，代码变得冗长复杂。允许图形界面出现部分bug，只要不特别影响核心功能都视为成功。

2.仍有部分页面显示的bug和读入空行的bug没有时间解决，但都不影响应用的主体功能。

3.python文件编译过后，每次打开都会占用端口，并在关闭窗口是不会释放，导致一次只能打开一个文件分享器。

========🡺

不编译文件，将文件源代码分成多个副本，在配置文件中设置每个副本的端口情况。

4.没有足够数量的公网ip供调试使用，泛洪的转发无法追踪调试。

=======🡺

根据和宋老师讨论的结果，将一台机器的多个端口视作多个主机，通过端口号的不同标识不同主机模拟真实情况。

5.泛洪转发可能多次重复相互转发造成网络拥堵。

========🡺

在查询泛洪信息中加入ttl，限制转发次数