Peer to Peer File Sharing

Yifei Xiao

2018/10/9

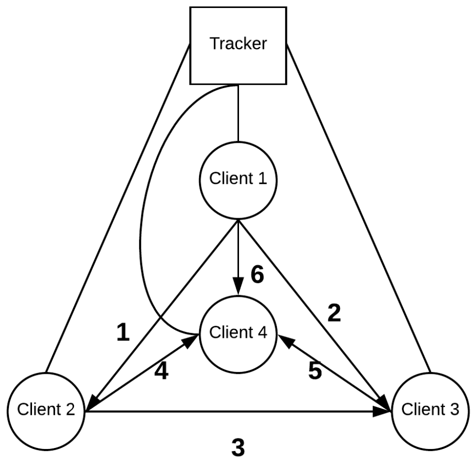
1 Introduction

This peer-to-peer file sharing system implements some basic requirements of P2P system include sending/receiving files by chunks, connecting with peers, sharing chunks to other peers, getting information from trackers, etc. This aims to help better understand the principles of peer-to-peer.

All commits and other related files can be found in my github:
`https://github.com/NomaNomaNoma/Peer-to-Peer`

2 Structure Diagram

The following diagram illustrates how the system works.



1 - Client 2 asks the Tracker for a specific file, which Client 1 has. So Client 1 sends the file to Client 2.

2 - Later on Client 3 asks the same file, and decides to download it from Client 1. Client 1 sends the file to Client 3.

3 - In the meantime, Client 2 will send chunks it has to Client 3 to help it speed up.

4, 5, 6 - After Client 2 and 3 finish downloading, Client 4 wants to download the same file from Client 1, so all of Client 1, 2, 3 will send chunks to Client 4.

3 Some Notice

- This program is written in `Python 2.7` and has been tested on `MacOS`, `Linux CentOS 7`, `Linux Ubuntu 18.04`. There is no guarantee to execute on other systems.
- Make sure install following modules: **socket, threading, json, hashlib, configparser**.

4 The Code

A peer to peer file system can have hundreds or thousands peers, to simplify this model, there are only four clients and one tracker in this project. Also in order to simplify this model, `localhost` is used as IP address. This can be simply modified to other IP addresses if one wants to send files to different end hosts.

`tracker.py:` This is the script for the tracker. It will connect to all clients and send messages between clients. However, the tracker itself **does not** contain files, it **will not** send files to clients either. If the tracker is shut down, all system will be shut down. It is important to keep the tracker running all the time.

`peer.py:` Main script for clients. A client can do three functions inside:

1. Ask the tracker to list all files in the system now, including which clients have these files.
2. Search a specific file from the system.
3. Download a file from a client.

When a client starts, the tracker will add it to the peer list, also the client will automatically register as a server. After that, it can use one of three functions above. When it finishes all its work, it can leave the system and the tracker will remove it from the peer list.

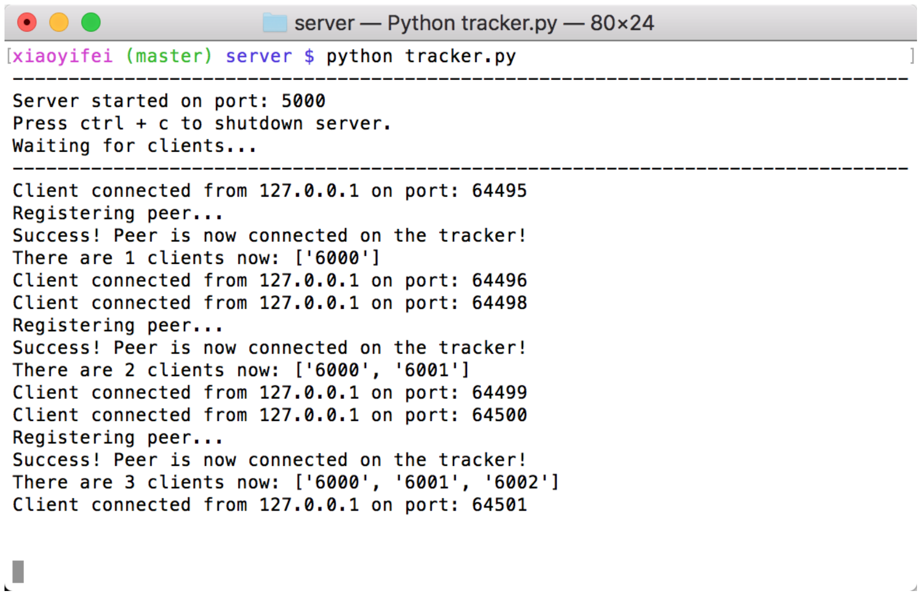
`peer_server.py:` This script is for setting up the client as a peer-server. After a client becomes a peer-server, it can send files to other clients by using TCP connection protocol. It is connected with `peer.py`.

`fileSystem.py:` There is a file monitor in this system, it can help clients send file list to the tracker. A client can get the file list from the tracker also using this script. It also can remove a client from the system when the client is shut down. It is connected with `peer.py`.

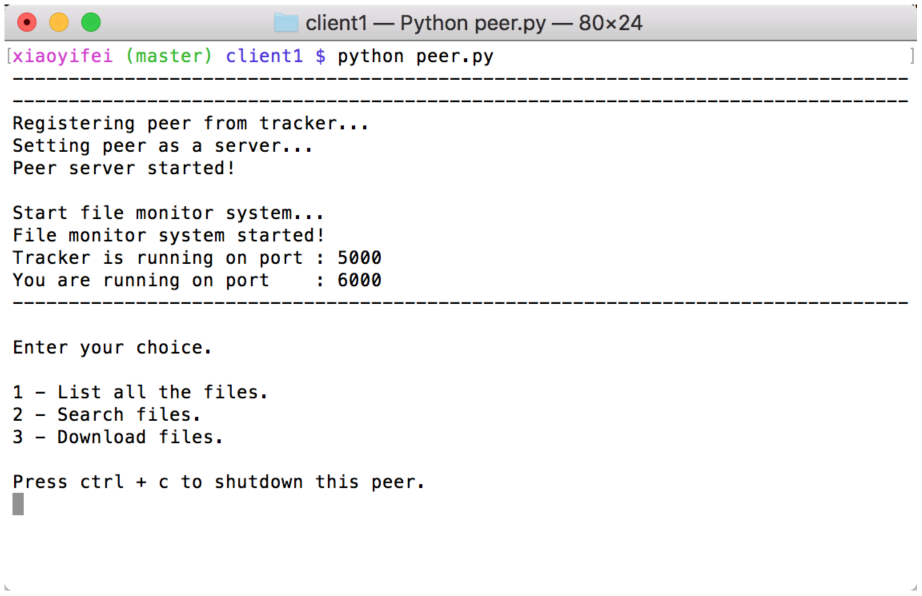
`make_files.py:` This script is for re-making files when a client has received all chunks. After making files, the client will be asked whether it will stay in the system to help others. If the client leaves the system, this script will remove all chunks to clean up space. It is also connected with `peer.py`.

5 Sample Demonstration

1) Start the tracker and clients. The tracker will run on port 5000, the first client will run on port 6000, the second one will run on 6001... The tracker will show the client list and the client will register as a peer-server and start its file monitor system.

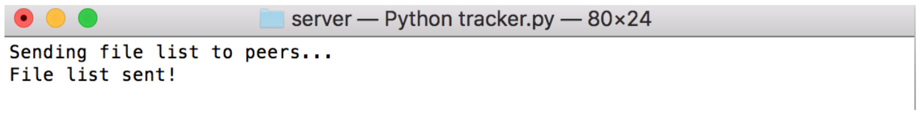


```
[xiaoyifei (master) server $ python tracker.py                                   ]
--------------------------------------------------------------------------------
Server started on port: 5000
Press ctrl + c to shutdown server.
Waiting for clients...
--------------------------------------------------------------------------------
Client connected from 127.0.0.1 on port: 64495
Registering peer...
Success! Peer is now connected on the tracker!
There are 1 clients now: ['6000']
Client connected from 127.0.0.1 on port: 64496
Client connected from 127.0.0.1 on port: 64498
Registering peer...
Success! Peer is now connected on the tracker!
There are 2 clients now: ['6000', '6001']
Client connected from 127.0.0.1 on port: 64499
Client connected from 127.0.0.1 on port: 64500
Registering peer...
Success! Peer is now connected on the tracker!
There are 3 clients now: ['6000', '6001', '6002']
Client connected from 127.0.0.1 on port: 64501
```



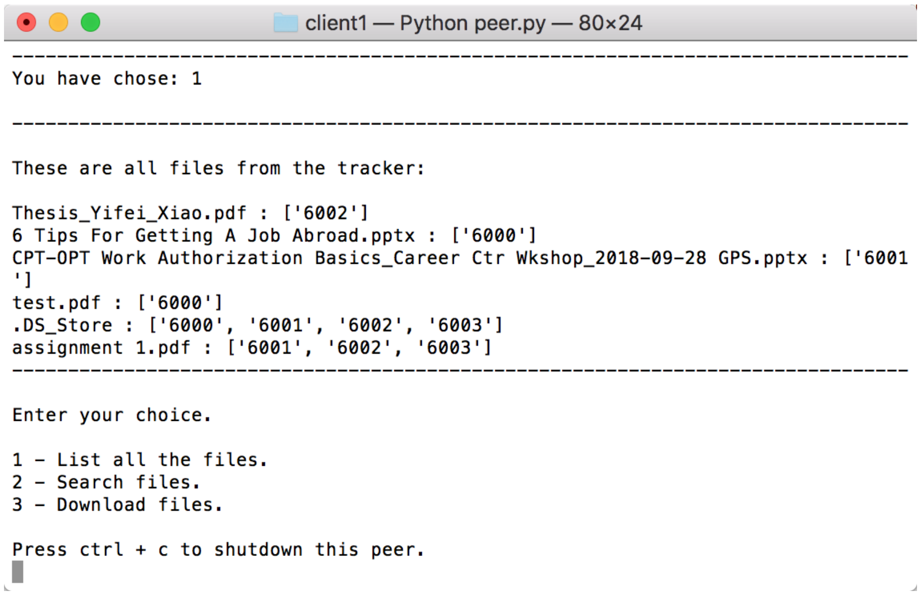
```
[xiaoyifei (master) client1 $ python peer.py                                     ]
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Registering peer from tracker...
Setting peer as a server...
Peer server started!

Start file monitor system...
File monitor system started!
Tracker is running on port : 5000
You are running on port    : 6000
--------------------------------------------------------------------------------

Enter your choice.

1 - List all the files.
2 - Search files.
3 - Download files.

Press ctrl + c to shutdown this peer.
```

2) Choose function 1, the client can get the file list from the tracker.



server — Python tracker.py — 80×24

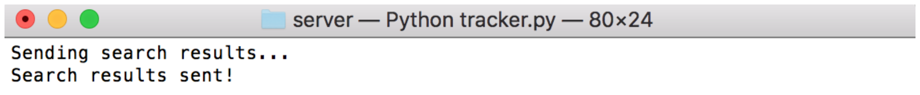
```
Sending file list to peers...
File list sent!
```



client1 — Python peer.py — 80×24

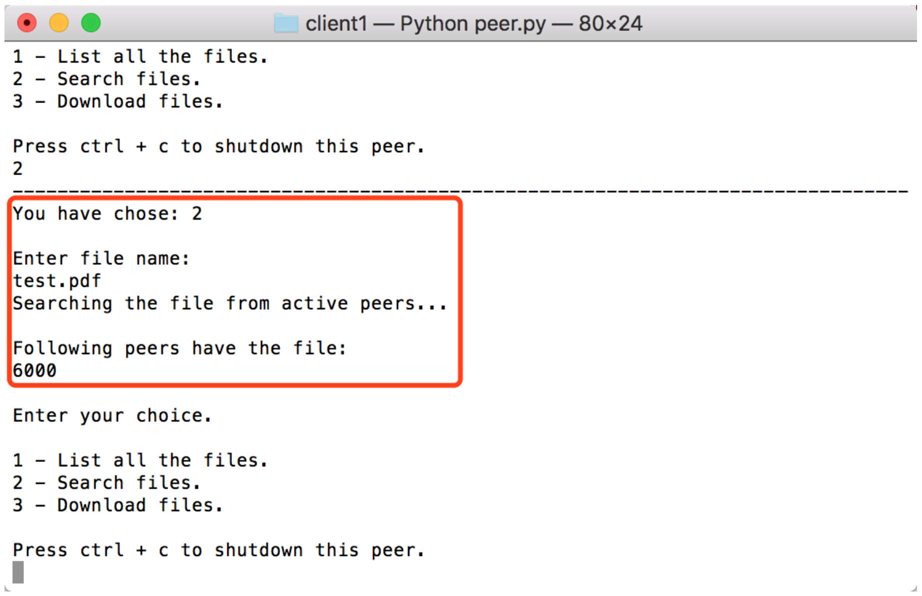
```
--------------------------------------------------------------------------------
You have chose: 1

--------------------------------------------------------------------------------

These are all files from the tracker:

Thesis_Yifei_Xiao.pdf : ['6002']
6 Tips For Getting A Job Abroad.pptx : ['6000']
CPT-OPT Work Authorization Basics_Career Ctr Wkshop_2018-09-28 GPS.pptx : ['6001
']
test.pdf : ['6000']
.DS_Store : ['6000', '6001', '6002', '6003']
assignment 1.pdf : ['6001', '6002', '6003']
--------------------------------------------------------------------------------

Enter your choice.

1 - List all the files.
2 - Search files.
3 - Download files.

Press ctrl + c to shutdown this peer.
```

3) Choose function 2, the client can search a file from the system.



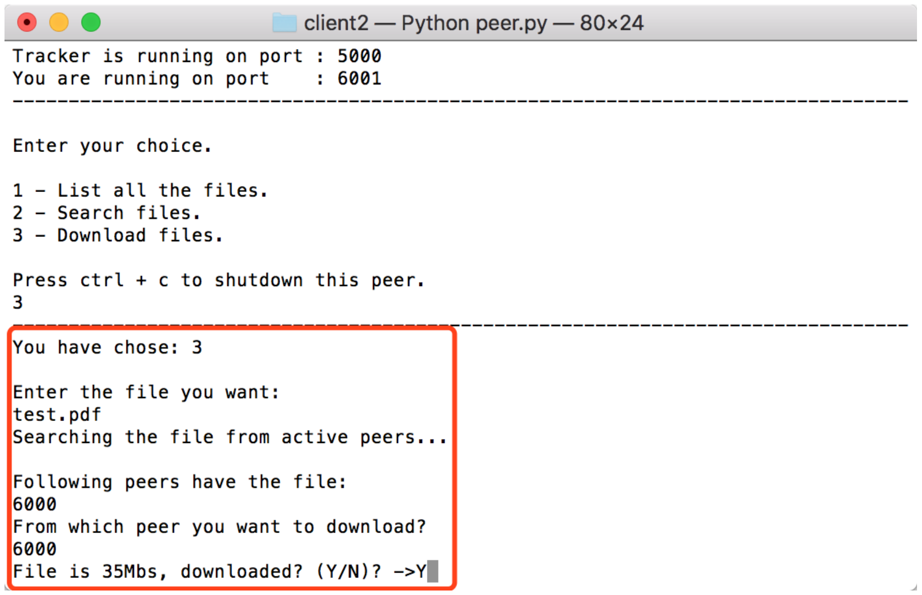
server — Python tracker.py — 80×24

```
Sending search results...
Search results sent!
```



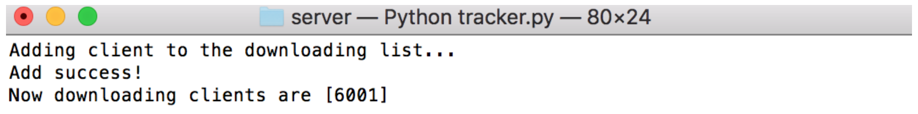
```
1 - List all the files.
2 - Search files.
3 - Download files.

Press ctrl + c to shutdown this peer.
2
--------------------------------------------------------------------------------
You have chose: 2

Enter file name:
test.pdf
Searching the file from active peers...

Following peers have the file:
6000

Enter your choice.

1 - List all the files.
2 - Search files.
3 - Download files.

Press ctrl + c to shutdown this peer.
```

4.1) Choose function 3, the client can download a file from another client.



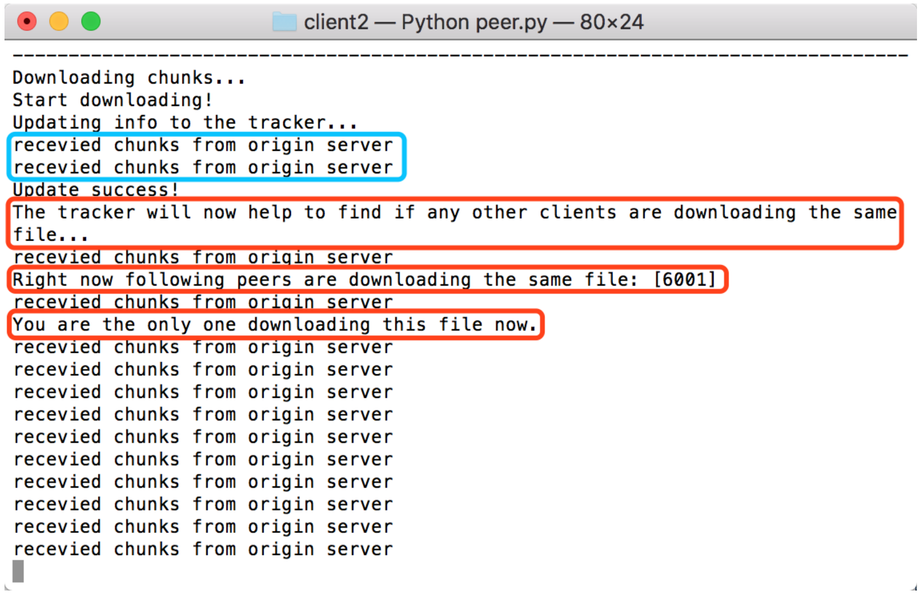
```
Tracker is running on port : 5000
You are running on port    : 6001
--------------------------------------------------------------------------------

Enter your choice.

1 - List all the files.
2 - Search files.
3 - Download files.

Press ctrl + c to shutdown this peer.
3
--------------------------------------------------------------------------------
You have chose: 3

Enter the file you want:
test.pdf
Searching the file from active peers...

Following peers have the file:
6000
From which peer you want to download?
6000
File is 35Mbs, downloaded? (Y/N)? ->Y
```

4.2) When start downloading, the receiver first tells the tracker it is downloading this file, the tracker will add the receiver to the downloading list.



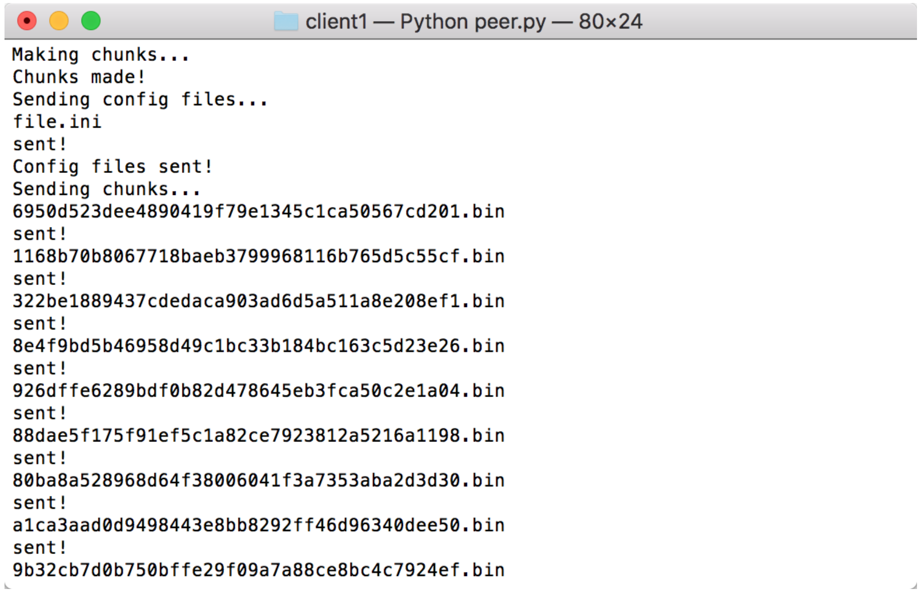
```
Adding client to the downloading list...
Add success!
Now downloading clients are [6001]
```

4.3) The receiver will then connect to the sender client and receive chunks from it (blue part). In the meantime, it will ask the tracker whether there are other clients downloading the same file. If not, it will just receive chunks from origin sender (red part).



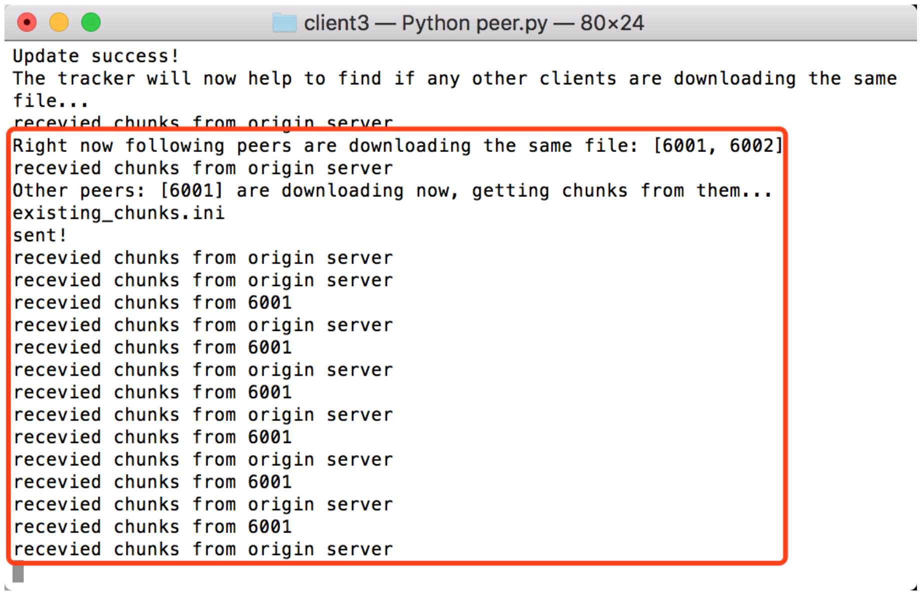
```
--------------------------------------------------------------------------------
Downloading chunks...
Start downloading!
Updating info to the tracker...
recevied chunks from origin server
recevied chunks from origin server
Update success!
The tracker will now help to find if any other clients are downloading the same
file...
recevied chunks from origin server
Right now following peers are downloading the same file: [6001]
recevied chunks from origin server
You are the only one downloading this file now.
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from origin server
```

4.4) On the other hand, the sender client get connection from the receiver and split the file into chunks (1Mb/chunk). Then it will send chunks to the receiver. It will also generate and send a config file called "file.ini". This file includes all chunks info and later will be used by the receiver to re-make the file.



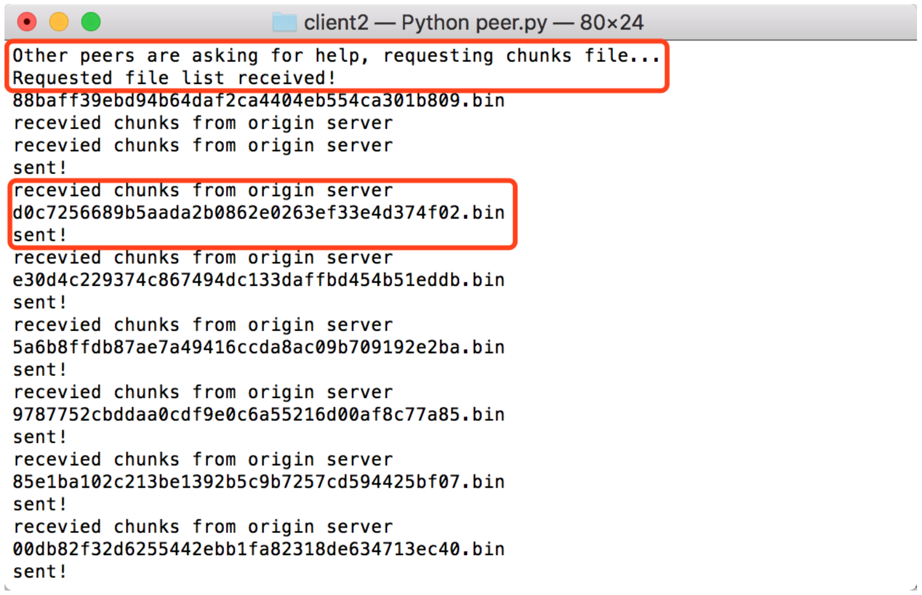
```
Making chunks...
Chunks made!
Sending config files...
file.ini
sent!
Config files sent!
Sending chunks...
6950d523dee4890419f79e1345c1ca50567cd201.bin
sent!
1168b70b8067718baeb3799968116b765d5c55cf.bin
sent!
322be1889437cdedaca903ad6d5a511a8e208ef1.bin
sent!
8e4f9bd5b46958d49c1bc33b184bc163c5d23e26.bin
sent!
926dffe6289bdf0b82d478645eb3fca50c2e1a04.bin
sent!
88dae5f175f91ef5c1a82ce7923812a5216a1198.bin
sent!
80ba8a528968d64f38006041f3a7353aba2d3d30.bin
sent!
a1ca3aad0d9498443e8bb8292ff46d96340dee50.bin
sent!
9b32cb7d0b750bffe29f09a7a88ce8bc4c7924ef.bin
```

4.5) In the mean time, another client joins and wants the same file. It will do the same process as above. However, this time it will generate another config file called "existing_chunks.ini" to send to other downloading peers. This config file contains chunks info which current receiver already has so other peers will not send those chunks. Now it is receiving chunks from the original sender and also the downloading peers.



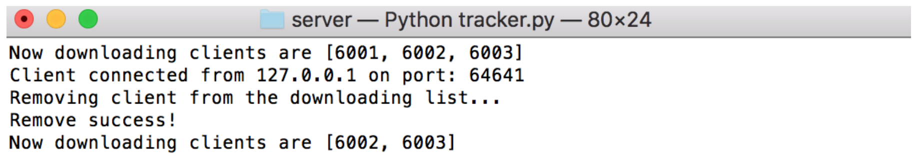
```
Update success!
The tracker will now help to find if any other clients are downloading the same
file...
recevied chunks from origin server
Right now following peers are downloading the same file: [6001, 6002]
recevied chunks from origin server
Other peers: [6001] are downloading now, getting chunks from them...
existing_chunks.ini
sent!
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from 6001
recevied chunks from origin server
recevied chunks from 6001
recevied chunks from origin server
recevied chunks from 6001
recevied chunks from origin server
recevied chunks from 6001
recevied chunks from origin server
recevied chunks from 6001
recevied chunks from origin server
recevied chunks from 6001
recevied chunks from origin server
```

4.6) On the other hand, the first receiver will send chunks to its peer and continue receiving chunks from the original sender.



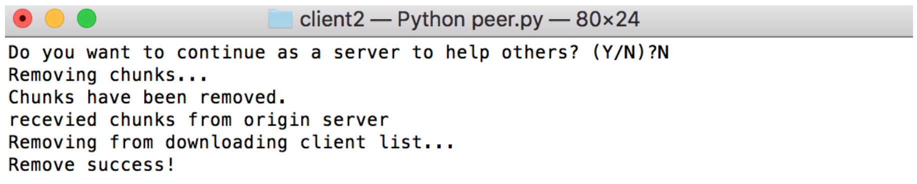
```
Other peers are asking for help, requesting chunks file...
Requested file list received!
88baff39ebd94b64daf2ca4404eb554ca301b809.bin
recevied chunks from origin server
recevied chunks from origin server
sent!
recevied chunks from origin server
d0c7256689b5aada2b0862e0263ef33e4d374f02.bin
sent!
recevied chunks from origin server
e30d4c229374c867494dc133daffbd454b51eddb.bin
sent!
recevied chunks from origin server
5a6b8ffdb87ae7a49416ccda8ac09b709192e2ba.bin
sent!
recevied chunks from origin server
9787752cbddaa0cdf9e0c6a55216d00af8c77a85.bin
sent!
recevied chunks from origin server
85e1ba102c213be1392b5c9b7257cd594425bf07.bin
sent!
recevied chunks from origin server
00db82f32d6255442ebb1fa82318de634713ec40.bin
sent!
```

4.7) When finish downloading, the receiver will be asked whether to share its chunks to other downloading peers or not. If yes, it will keep its chunks until it says no. Then it will be removed from the downloading list and delete all its chunks.



server — Python tracker.py — 80×24

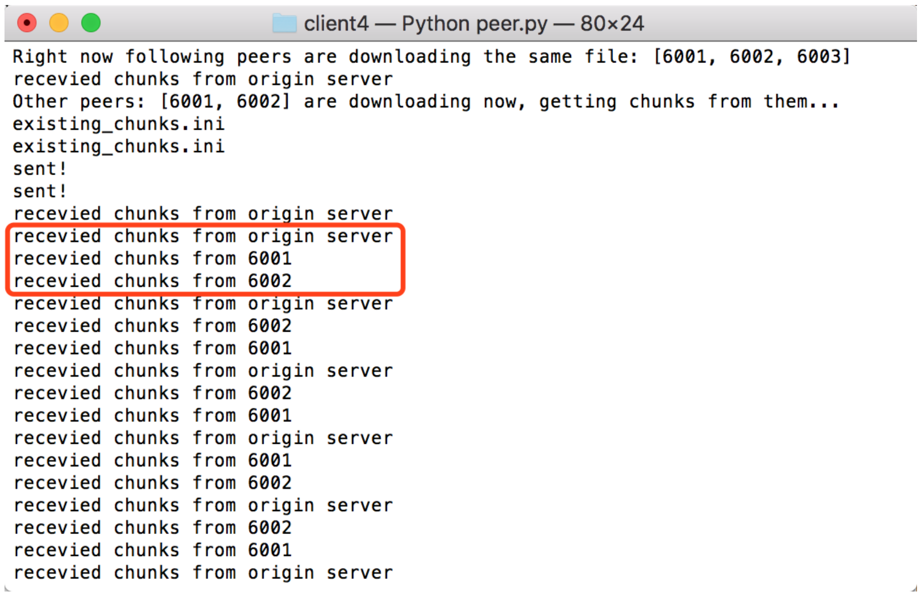
```
Now downloading clients are [6001, 6002, 6003]
Client connected from 127.0.0.1 on port: 64641
Removing client from the downloading list...
Remove success!
Now downloading clients are [6002, 6003]
```



client2 — Python peer.py — 80×24

```
Do you want to continue as a server to help others? (Y/N)?N
Removing chunks...
Chunks have been removed.
recevied chunks from origin server
Removing from downloading client list...
Remove success!
```

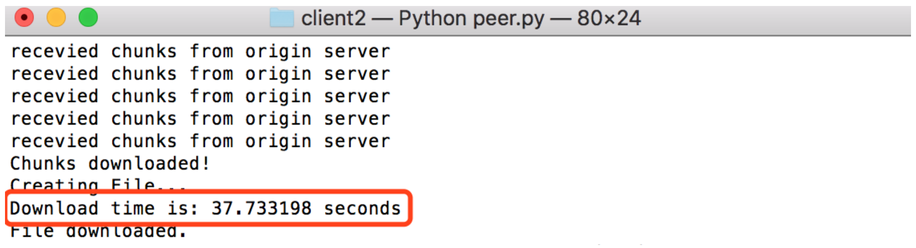
4.8) If later on the forth client joins and wants to download the same file, it will get chunks from all other downloading peers.



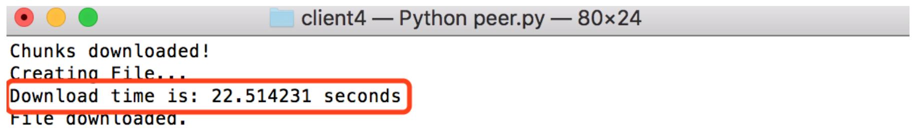
client4 — Python peer.py — 80×24

```
Right now following peers are downloading the same file: [6001, 6002, 6003]
recevied chunks from origin server
Other peers: [6001, 6002] are downloading now, getting chunks from them...
existing_chunks.ini
existing_chunks.ini
sent!
sent!
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from 6001
recevied chunks from 6002
recevied chunks from origin server
recevied chunks from 6002
recevied chunks from 6001
recevied chunks from origin server
recevied chunks from 6002
recevied chunks from 6001
recevied chunks from origin server
recevied chunks from 6001
recevied chunks from 6002
recevied chunks from origin server
recevied chunks from 6002
recevied chunks from 6001
recevied chunks from origin server
```

4.9) We can compare download time between the first receiver (only receives chunks from original sender) and the last receiver (receives chunks from original senders and all peers).

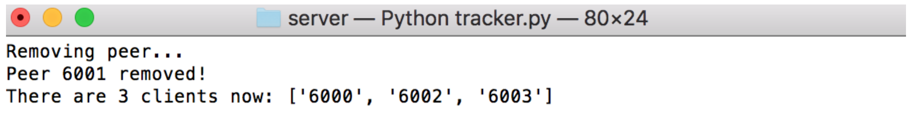


```
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from origin server
recevied chunks from origin server
Chunks downloaded!
Creating File...
Download time is: 37.733198 seconds
File downloaded.
```

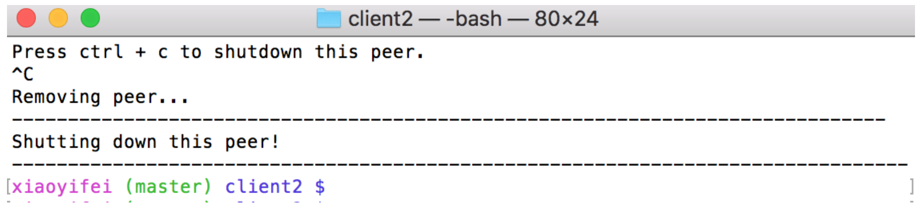


```
Chunks downloaded!
Creating File...
Download time is: 22.514231 seconds
File downloaded.
```

5) Finally, a client can exit the system and will be removed from the tracker.



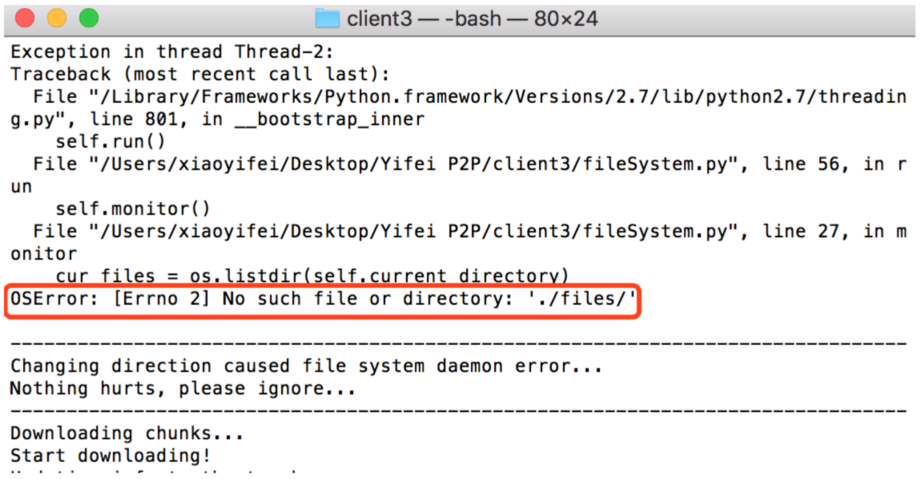
```
Removing peer...
Peer 6001 removed!
There are 3 clients now: ['6000', '6002', '6003']
```



```
Press ctrl + c to shutdown this peer.
^C
Removing peer...
-------------------------------------------------------------------------------
Shutting down this peer!
--------------------------------------------------------------------------------
[xiaoyifei (master) client2 $                                                 ]
```

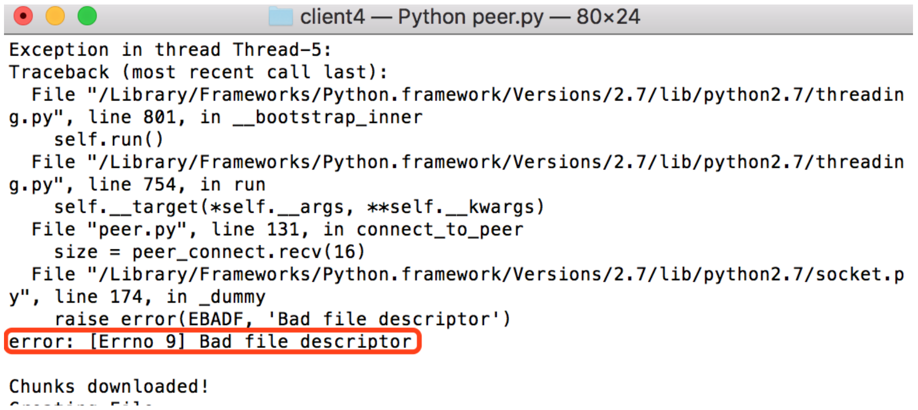
6 Things to Improve

- This system does not support uploading files when it is running, clients must shut down and restart the service if they want to upload files.
- Only one file can be downloaded at a time. Although a client can send multiple chunks at a time, there is only one folder to store chunks. Chunks from different files may cause problems.
- Although other clients can get chunks from downloading peers, the first client cannot, since it does not know about clients who join later. This can be improved by checking downloading list periodically.
- There may be one error when a client starts downloading the file. This error is due to changing directory to `./chunks` to download chunks and there is no directory `./chunks/files`. Since our file monitor system monitors the directory `./files`. This error will not hurt the system. After downloading, the file monitor system will restart by itself.

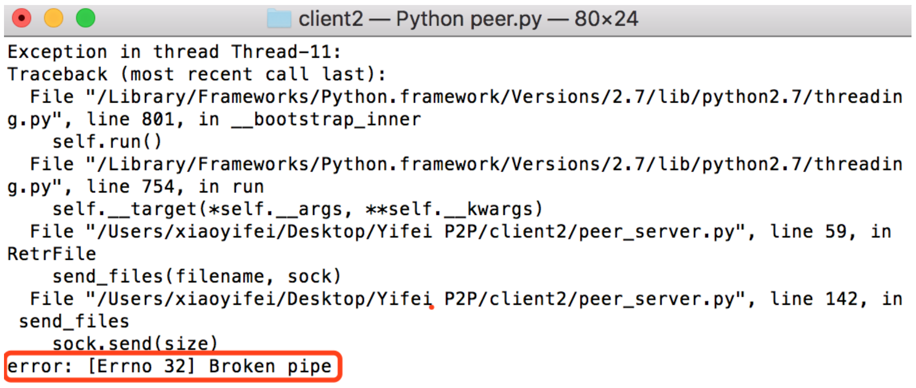


```
Exception in thread Thread-2:
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/threadin
g.py", line 801, in __bootstrap_inner
    self.run()
  File "/Users/xiaoyifei/Desktop/Yifei P2P/client3/fileSystem.py", line 56, in r
un
    self.monitor()
  File "/Users/xiaoyifei/Desktop/Yifei P2P/client3/fileSystem.py", line 27, in m
onitor
    cur_files = os.listdir(self.current_directory)
OSError: [Errno 2] No such file or directory: './files/'

--------------------------------------------------------------------------------
Changing direction caused file system daemon error...
Nothing hurts, please ignore...
--------------------------------------------------------------------------------
Downloading chunks...
Start downloading!
```

- There may some other errors if a client is sharing chunks to others, like `Broken pipe` and `Bad file descriptor`. These errors are raised because once a client finish downloading, it will immediately close its connection. However, the peer-server does not know so it may continue sending chunks but there is no connection. All these errors will not hurt the system. Once the peer-server detects these errors, it will stop sending chunks automatically. Furthermore, if GUI is used for this system, clients will not see any of these errors.



```
Exception in thread Thread-5:
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/threadin
g.py", line 801, in __bootstrap_inner
    self.run()
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/threadin
g.py", line 754, in run
    self.__target(*self.__args, **self.__kwargs)
  File "peer.py", line 131, in connect_to_peer
    size = peer_connect.recv(16)
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/socket.p
y", line 174, in _dummy
    raise error(EBADF, 'Bad file descriptor')
error: [Errno 9] Bad file descriptor

Chunks downloaded!
```



client2 — Python peer.py — 80×24

```
Exception in thread Thread-11:
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/threadin
g.py", line 801, in __bootstrap_inner
    self.run()
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/threadin
g.py", line 754, in run
    self.__target(*self.__args, **self.__kwargs)
  File "/Users/xiaoyifei/Desktop/Yifei P2P/client2/peer_server.py", line 59, in
RetrFile
    send_files(filename, sock)
  File "/Users/xiaoyifei/Desktop/Yifei P2P/client2/peer_server.py", line 142, in
 send_files
    sock.send(size)
error: [Errno 32] Broken pipe
```