README

Peer-to-peer network, 200010030

Demo video: Screencast 2023-04-04 01:47:40.mp4

RUNNING

To run manager

```
python 200010030_manager.py
```

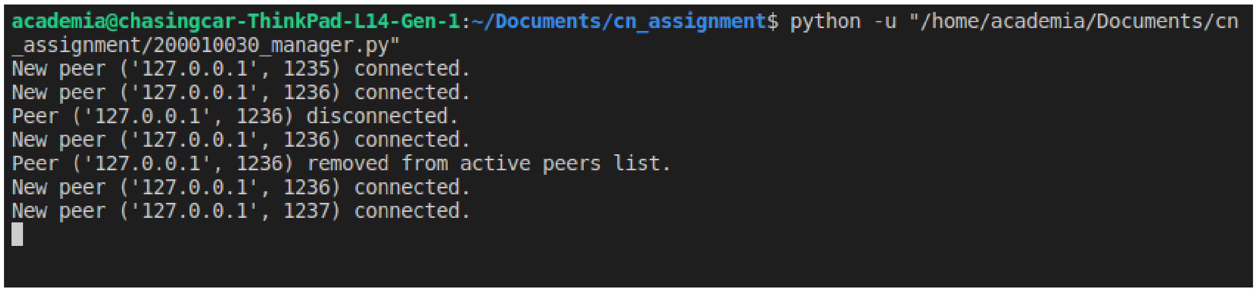
To run peer

```
python 200010030_peer.py
```

Give the manager’s ip as 127.0.0.1 and port 1234. give any free port for peer.
In peer select from options given for appropriate action.

Manager

Manager is always on-server listening for new connections and broadcasting active peer list to the peers. Logs the events such as peers joining the network or leaving the network.



```
academia@chasingcar-ThinkPad-L14-Gen-1:~/Documents/cn_assignment$ python -u "/home/academia/Documents/cn
_assignment/200010030_manager.py"
New peer ('127.0.0.1', 1235) connected.
New peer ('127.0.0.1', 1236) connected.
Peer ('127.0.0.1', 1236) disconnected.
New peer ('127.0.0.1', 1236) connected.
Peer ('127.0.0.1', 1236) removed from active peers list.
New peer ('127.0.0.1', 1236) connected.
New peer ('127.0.0.1', 1237) connected.
```

Active peer-list

A global variable is maintained for storing active peer list in manager. which have tuple ip and port of the peer.

Connect

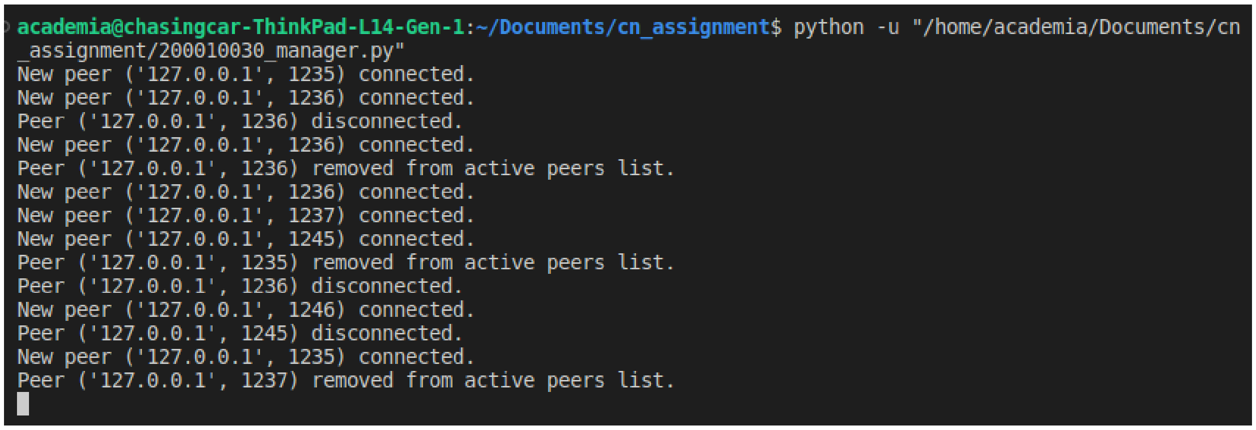
Manager binds to a port which is known to the peer and the peer on entering the network, makes connection to manager address and sends “CONNECT <peer ip> <peer port>” data. which is read by manager and added to active peerlist

Broadcasting

Manager bradcastes the active peer-list to all of the peers using the address in the list itself.

Periodic scan

`check_peets` function handle the periodic checks and time interval is 5 seconds by default. It sends “PING” message command. if the peer doesn’t respond as “PONG”. Manager thinks the peer left the network and removes it from the list and broadcasts the updated list.



```
academia@chasingcar-ThinkPad-L14-Gen-1:~/Documents/cn_assignment$ python -u "/home/academia/Documents/cn
_assignment/200010030_manager.py"
New peer ('127.0.0.1', 1235) connected.
New peer ('127.0.0.1', 1236) connected.
Peer ('127.0.0.1', 1236) disconnected.
New peer ('127.0.0.1', 1236) connected.
Peer ('127.0.0.1', 1236) removed from active peers list.
New peer ('127.0.0.1', 1236) connected.
New peer ('127.0.0.1', 1237) connected.
New peer ('127.0.0.1', 1245) connected.
Peer ('127.0.0.1', 1235) removed from active peers list.
Peer ('127.0.0.1', 1236) disconnected.
New peer ('127.0.0.1', 1246) connected.
Peer ('127.0.0.1', 1245) disconnected.
New peer ('127.0.0.1', 1235) connected.
Peer ('127.0.0.1', 1237) removed from active peers list.
```

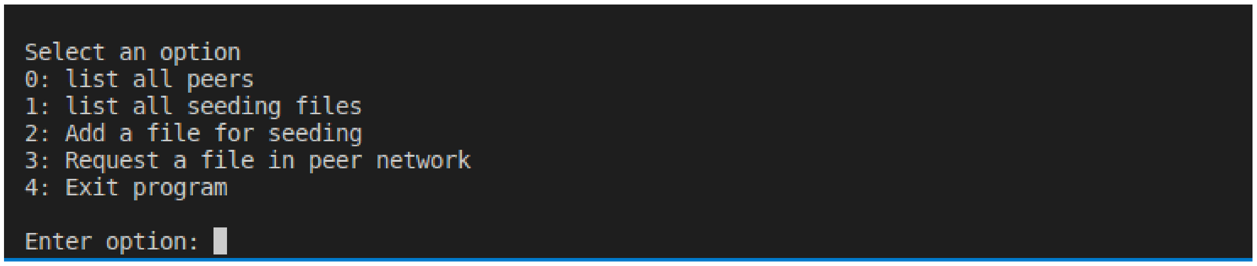
Leaving

When a peer wants to leave the network it sends the message “LEAVE <peer ip> <peer port>” to the manager which in turn removes the corresponding address types in active peer list and braodcastes the same to other peers.

Peer

Peer on start asks the user details of the mager address so that it can send connect command to the manager to register itself. It also takes peer port to which the peer binds to.

Peer has an UI where user can select between the option whether to add a seed file or request the file from peer network. listing the peers in the network and seed files the peer is serving is also available.



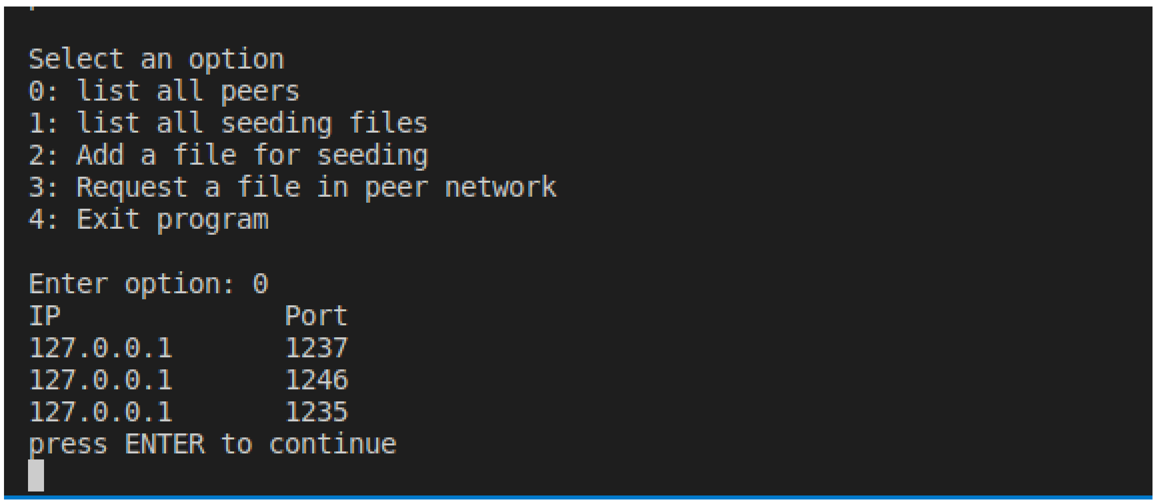
```
Select an option
0: list all peers
1: list all seeding files
2: Add a file for seeding
3: Request a file in peer network
4: Exit program

Enter option:
```

In background the peer server is listening to the commands received from other peers or manager.

`handle_connection` function calls the appropriate function in case of incoming data.

Active peers

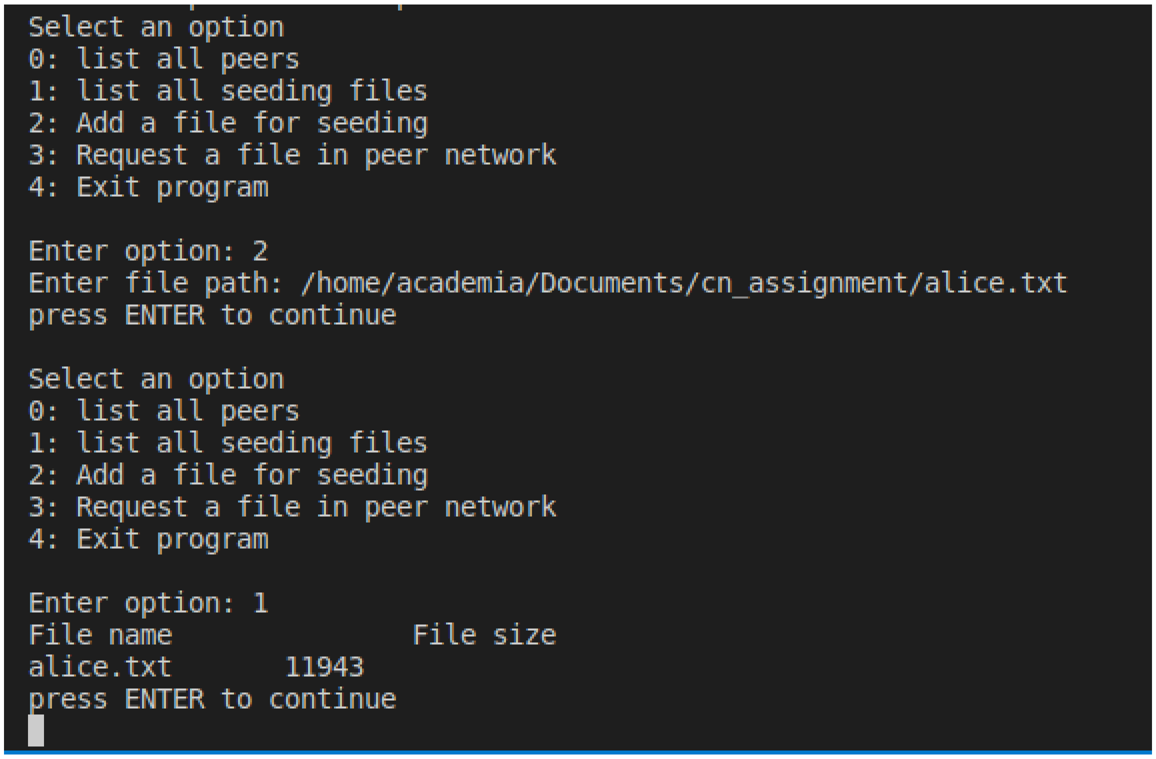
Active peers are updated when “PEERS <list>” data message arrives from manager.



```
Select an option
0: list all peers
1: list all seeding files
2: Add a file for seeding
3: Request a file in peer network
4: Exit program

Enter option: 0
IP              Port
127.0.0.1       1237
127.0.0.1       1246
127.0.0.1       1235
press ENTER to continue
```

List of shareable files

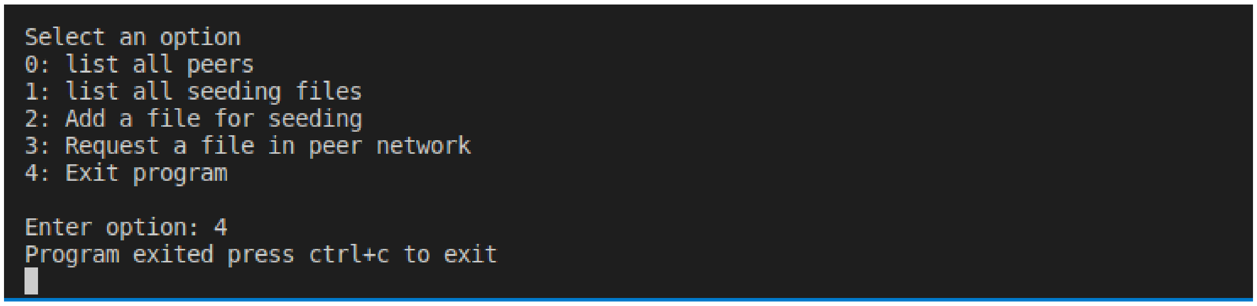
It is stored in the global variable `shared_files` and updated accordingly when user selects option to add a seed file(path or relative path).



```
Select an option
0: list all peers
1: list all seeding files
2: Add a file for seeding
3: Request a file in peer network
4: Exit program

Enter option: 2
Enter file path: /home/academia/Documents/cn_assignment/alice.txt
press ENTER to continue

Select an option
0: list all peers
1: list all seeding files
2: Add a file for seeding
3: Request a file in peer network
4: Exit program

Enter option: 1
File name               File size
alice.txt       11943
press ENTER to continue
```

Leaving the network

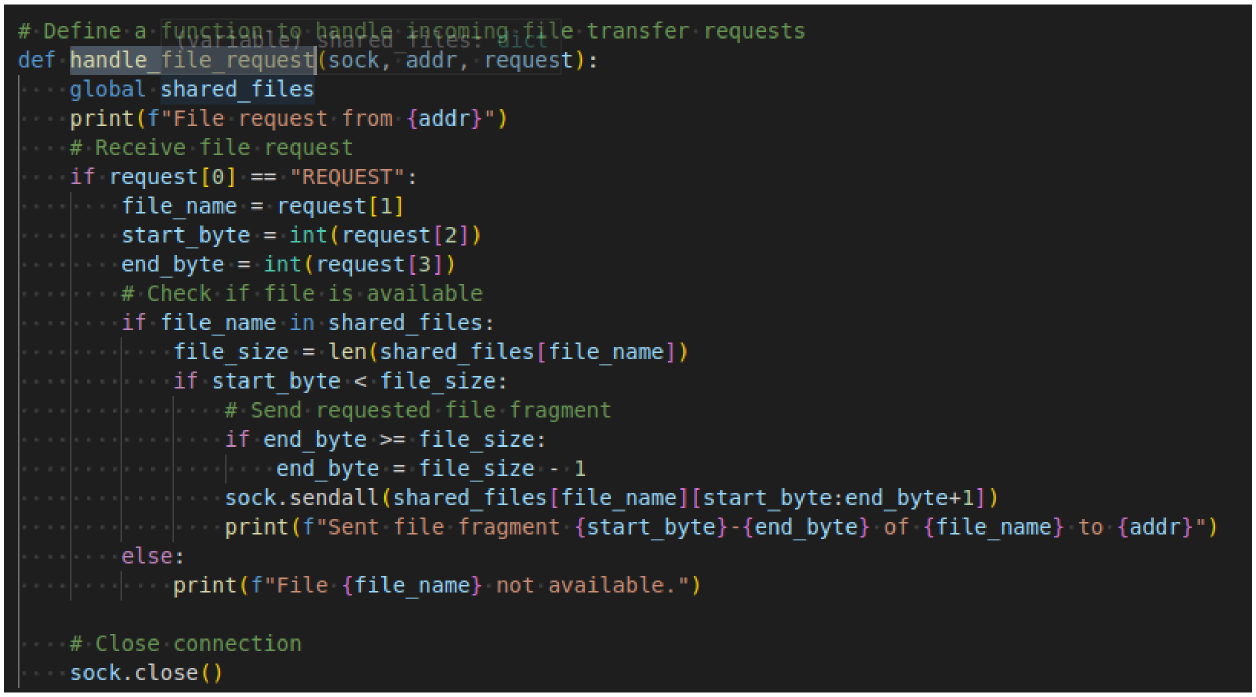
when exit program is chosen in ui by user peer sends “LEAVE <peer ip> <peer port>” and the manager removes the peer from list and broadcasts the list.



```
Select an option
0: list all peers
1: list all seeding files
2: Add a file for seeding
3: Request a file in peer network
4: Exit program

Enter option: 4
Program exited press ctrl+c to exit
```

Serving requested files

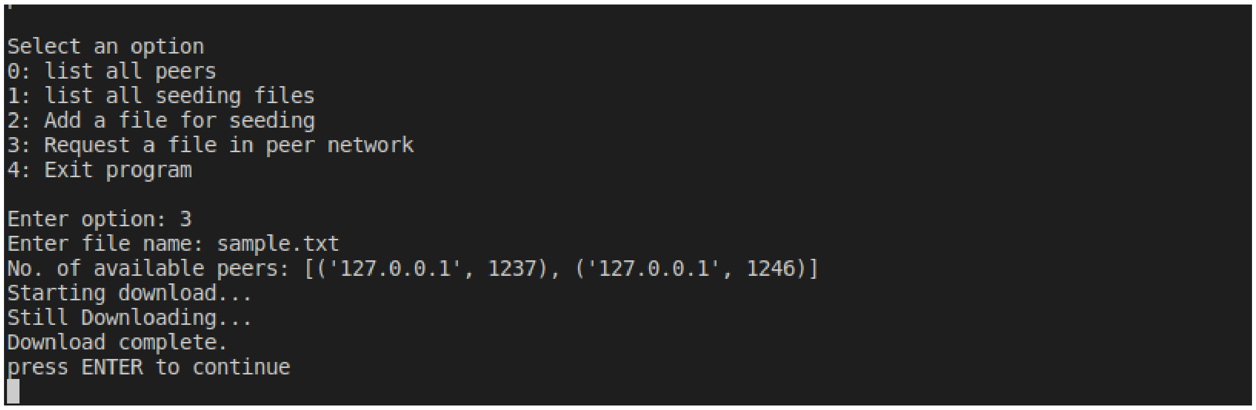
It handled by the `handle_file_request` function.



```python
# Define a function to handle incoming file transfer requests
def handle_file_request(sock, addr, request):
    global shared_files
    print(f"File request from {addr}")
    # Receive file request
    if request[0] == "REQUEST":
        file_name = request[1]
        start_byte = int(request[2])
        end_byte = int(request[3])
        # Check if file is available
        if file_name in shared_files:
            file_size = len(shared_files[file_name])
            if start_byte < file_size:
                # Send requested file fragment
                if end_byte >= file_size:
                    end_byte = file_size - 1
                sock.sendall(shared_files[file_name][start_byte:end_byte+1])
                print(f"Sent file fragment {start_byte}-{end_byte} of {file_name} to {addr}")
        else:
            print(f"File {file_name} not available.")

    # Close connection
    sock.close()
```

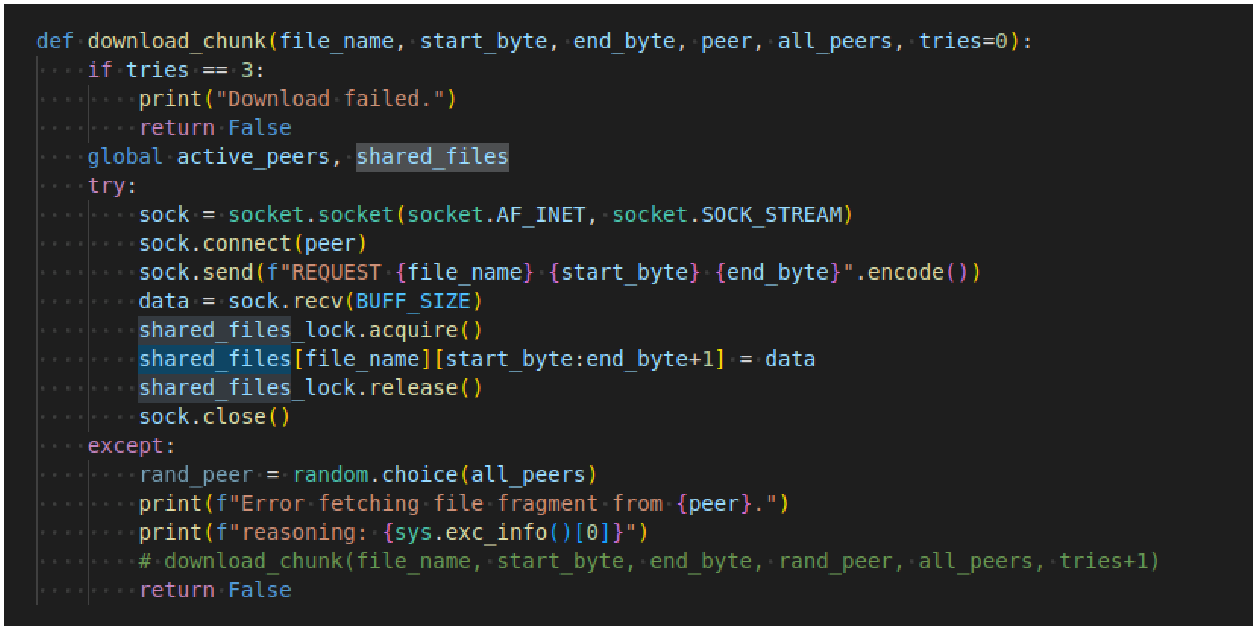
Requesting files

When user request a file to download from peer network in ui, every peer is sent a message “FIND <file basename>” where in return a peer checks if the filename is available then it sends “FOUND <lenght of file>” and “NOT FOUND” if not found.



```
Select an option
0: list all peers
1: list all seeding files
2: Add a file for seeding
3: Request a file in peer network
4: Exit program

Enter option: 3
Enter file name: sample.txt
No. of available peers: [('127.0.0.1', 1237), ('127.0.0.1', 1246)]
Starting download...
Still Downloading...
Download complete.
press ENTER to continue
```

If Found peers with file then the peer tries to fetch the chunks of 1kb from the peers parallely. The newly downloaded file will be placed in the `peer_<peer_port>_downloads` directory and it will also be present in `shared_files` variable so that it can be seeded for other peers.



```python
def download_chunk(file_name, start_byte, end_byte, peer, all_peers, tries=0):
    if tries == 3:
        print("Download failed.")
        return False
    global active_peers, shared_files
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect(peer)
        sock.send(f"REQUEST {file_name} {start_byte} {end_byte}".encode())
        data = sock.recv(BUFF_SIZE)
        shared_files_lock.acquire()
        shared_files[file_name][start_byte:end_byte+1] = data
        shared_files_lock.release()
        sock.close()
    except:
        rand_peer = random.choice(all_peers)
        print(f"Error fetching file fragment from {peer}.")
        print(f"reasoning: {sys.exc_info()[0]}")
        # download_chunk(file_name, start_byte, end_byte, rand_peer, all_peers, tries+1)
        return False
```