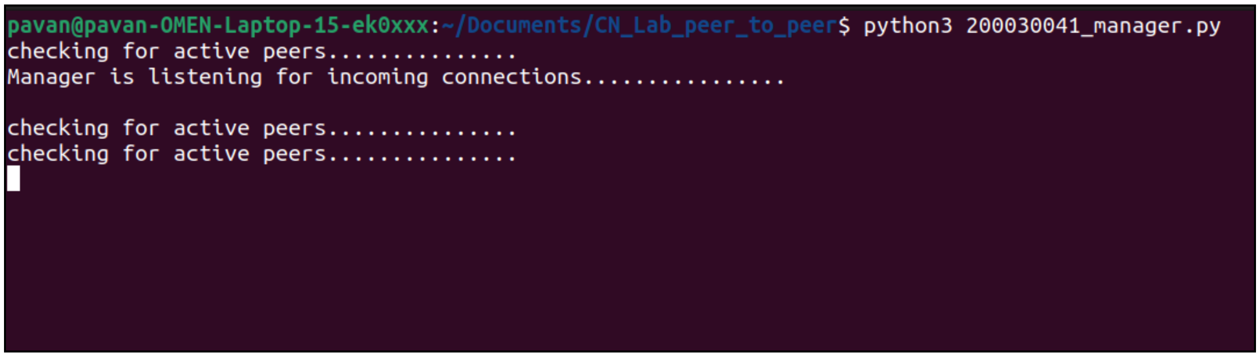
CS 348: Computer Networks
Programming Assignment

- Pavan Kumar V Patil
- 200030041

- ## Demonstration

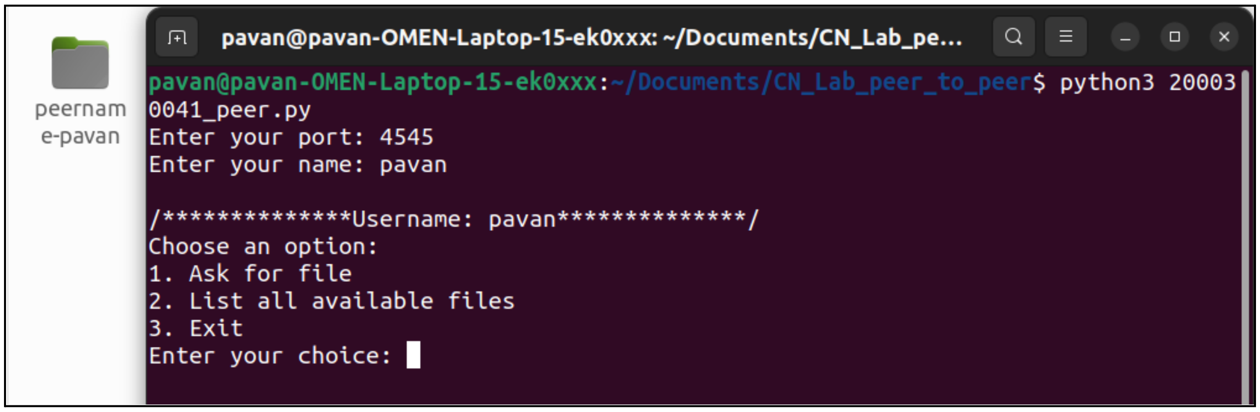
Run the ***"python3 200030041_manager.py"*** command on the terminal to start the manager.



```
pavan@pavan-OMEN-Laptop-15-ek0xxx:~/Documents/CN_Lab_peer_to_peer$ python3 200030041_manager.py
checking for active peers...............
Manager is listening for incoming connections...............

checking for active peers...............
checking for active peers...............
```

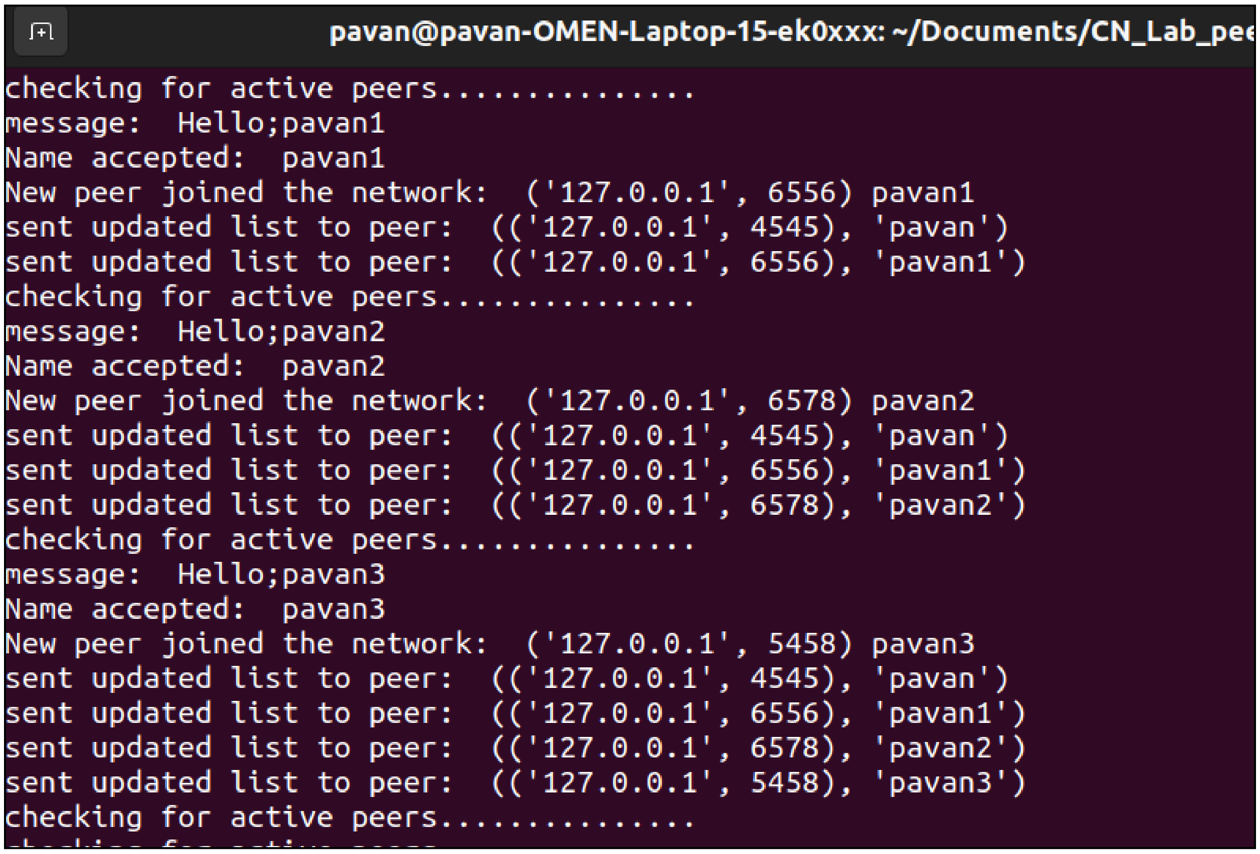
The manager program listens to incoming connections and checks active peers.

Run the ***"python3 200030042_peer.py"*** command on the terminal to start the peer. After that, you have to provide the port number and username of the peer. Now the program displays features to choose from and creates a folder **"peername-<username>"** to store the files.



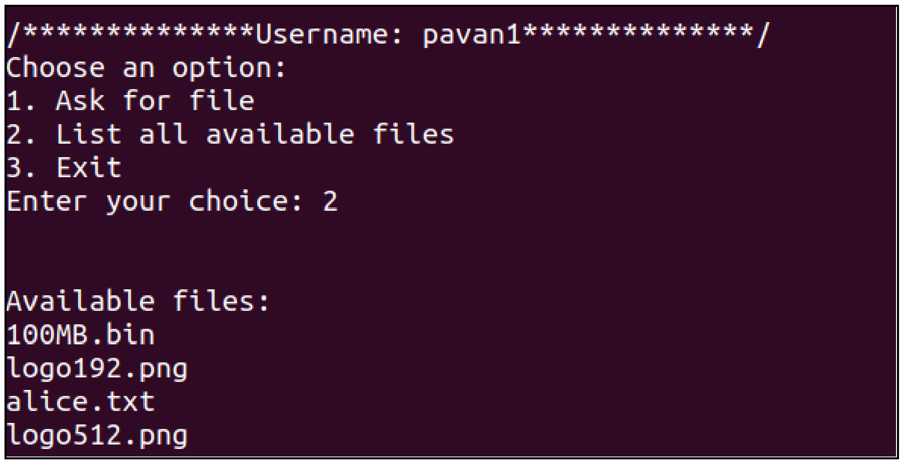
```
pavan@pavan-OMEN-Laptop-15-ek0xxx:~/Documents/CN_Lab_peer_to_peer$ python3 20003
0041_peer.py
Enter your port: 4545
Enter your name: pavan

/*************Username: pavan*************/
Choose an option:
1. Ask for file
2. List all available files
3. Exit
Enter your choice:
```

When a new peer joins to network, the manager broadcasts active peers to all available peers. We can see the printed statements from the manager when a new peer joins in the below image.



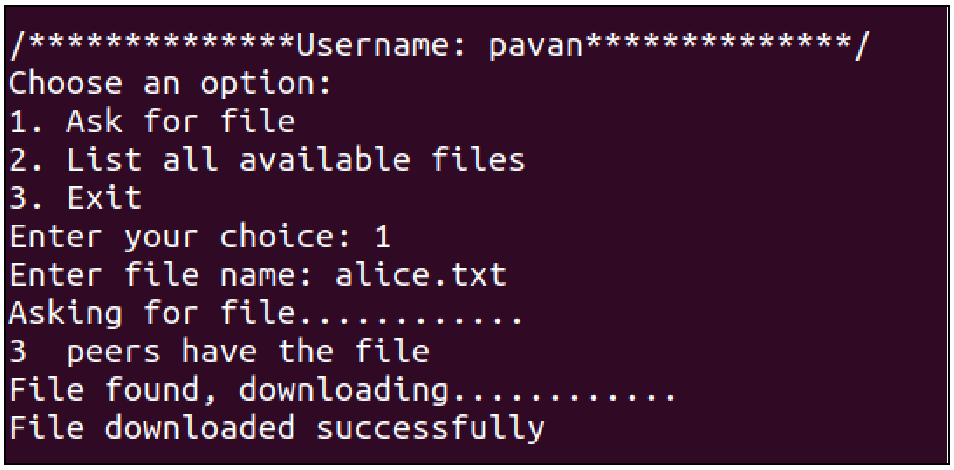
```
pavan@pavan-OMEN-Laptop-15-ek0xxx: ~/Documents/CN_Lab_pee
checking for active peers..............
message:  Hello;pavan1
Name accepted:  pavan1
New peer joined the network:  ('127.0.0.1', 6556) pavan1
sent updated list to peer:  (('127.0.0.1', 4545), 'pavan')
sent updated list to peer:  (('127.0.0.1', 6556), 'pavan1')
checking for active peers..............
message:  Hello;pavan2
Name accepted:  pavan2
New peer joined the network:  ('127.0.0.1', 6578) pavan2
sent updated list to peer:  (('127.0.0.1', 4545), 'pavan')
sent updated list to peer:  (('127.0.0.1', 6556), 'pavan1')
sent updated list to peer:  (('127.0.0.1', 6578), 'pavan2')
checking for active peers..............
message:  Hello;pavan3
Name accepted:  pavan3
New peer joined the network:  ('127.0.0.1', 5458) pavan3
sent updated list to peer:  (('127.0.0.1', 4545), 'pavan')
sent updated list to peer:  (('127.0.0.1', 6556), 'pavan1')
sent updated list to peer:  (('127.0.0.1', 6578), 'pavan2')
sent updated list to peer:  (('127.0.0.1', 5458), 'pavan3')
checking for active peers..............
```

To share files, the user has to keep files in a folder **“peername-<username>”**.
The peer has to choose options according to his need. We can see When the user chooses option 2, all available files are displayed.



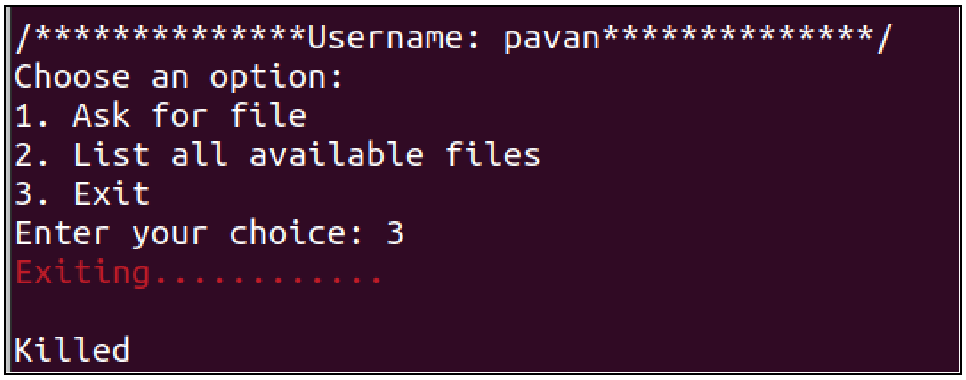
```
/**************Username: pavan1**************/
Choose an option:
1. Ask for file
2. List all available files
3. Exit
Enter your choice: 2


Available files:
100MB.bin
logo192.png
alice.txt
logo512.png
```

To ask for file from other peers, the user have to choose option 1 and provide the file name. If the file is found by other peers, the user receives it from peers who have the file parallelly.

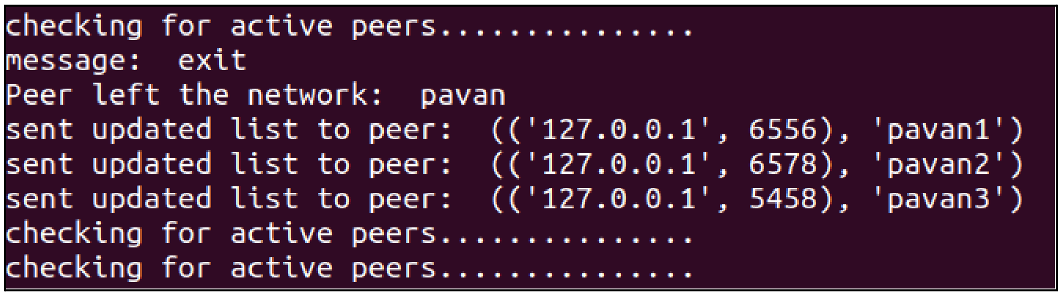

```
/**************Username: pavan**************/
Choose an option:
1. Ask for file
2. List all available files
3. Exit
Enter your choice: 1
Enter file name: alice.txt
Asking for file...........
3  peers have the file
File found, downloading...........
File downloaded successfully
```

The asked file will be downloaded in the “peername-<username>” folder.

To exit peer has to choose option 3. The folder corresponding to the peer is deleted.


```
/**************Username: pavan**************/
Choose an option:
1. Ask for file
2. List all available files
3. Exit
Enter your choice: 3
Exiting............

Killed
```

While leaving the network, the message is sent to the manager. And the manager removes a particular peer from the network and broadcasts updated active peers to available peers.


```
checking for active peers..............
message:  exit
Peer left the network:  pavan
sent updated list to peer:  (('127.0.0.1', 6556), 'pavan1')
sent updated list to peer:  (('127.0.0.1', 6578), 'pavan2')
sent updated list to peer:  (('127.0.0.1', 5458), 'pavan3')
checking for active peers..............
checking for active peers..............
```

- **Explaining functions**

  200030041_manager.py consists of the following functions:

  ➔ ***start_server():*** Starts the server and starts listening to peers. When a new peer joins, it creates a new thread for that peer. It starts a background thread to check for active peers.

  ➔ ***background_ping():*** This function is used to check for active peers. If a peer is not active, it is removed from the active peer's list and broadcast to available peers.

  ➔ ***start_peer_listen(peer_socket, peer_address):*** This function is used to listen to particular peer. It listens for messages from the peer and takes appropriate action, such as when the message is like "Hello;<username>", it calls new_peer_joined(), etc.

  ➔ ***peer_leave(peer_socket):*** This function is used to remove a peer from the active peer's list and update the active peer's list of all the peers by broadcasting.

  ➔ ***new_peer_joined(peer_socket, peer_address, peer_username):*** This function is used to add a new peer to the active peer's list and update the active peer's list of all the peers by broadcasting.

  ➔ ***check_peer_username(peer_username):*** This function is used to check if the username of the peer is already taken or not. If the username is already present, then it returns false else, true.

200030041_peer.py consists of the following functions:

➔ ***main(manager_peer_socket, manager_address):*** initially connect to the manager and send the name of the peer. Then start the peer server and peer features in different threads.

➔ ***peer_server(peer_socket):*** This function starts the peer server and accepts connections from other peers for file transfer, and puts them in different threads for file transfer.

➔ ***peer_server_handler(conn_peer_socket):*** This function handles the peer server requests and sends the file to the peer chunk by chunk. It also handles the case when the file is not found. It also sends responses to the manager when a “ping” message is received.

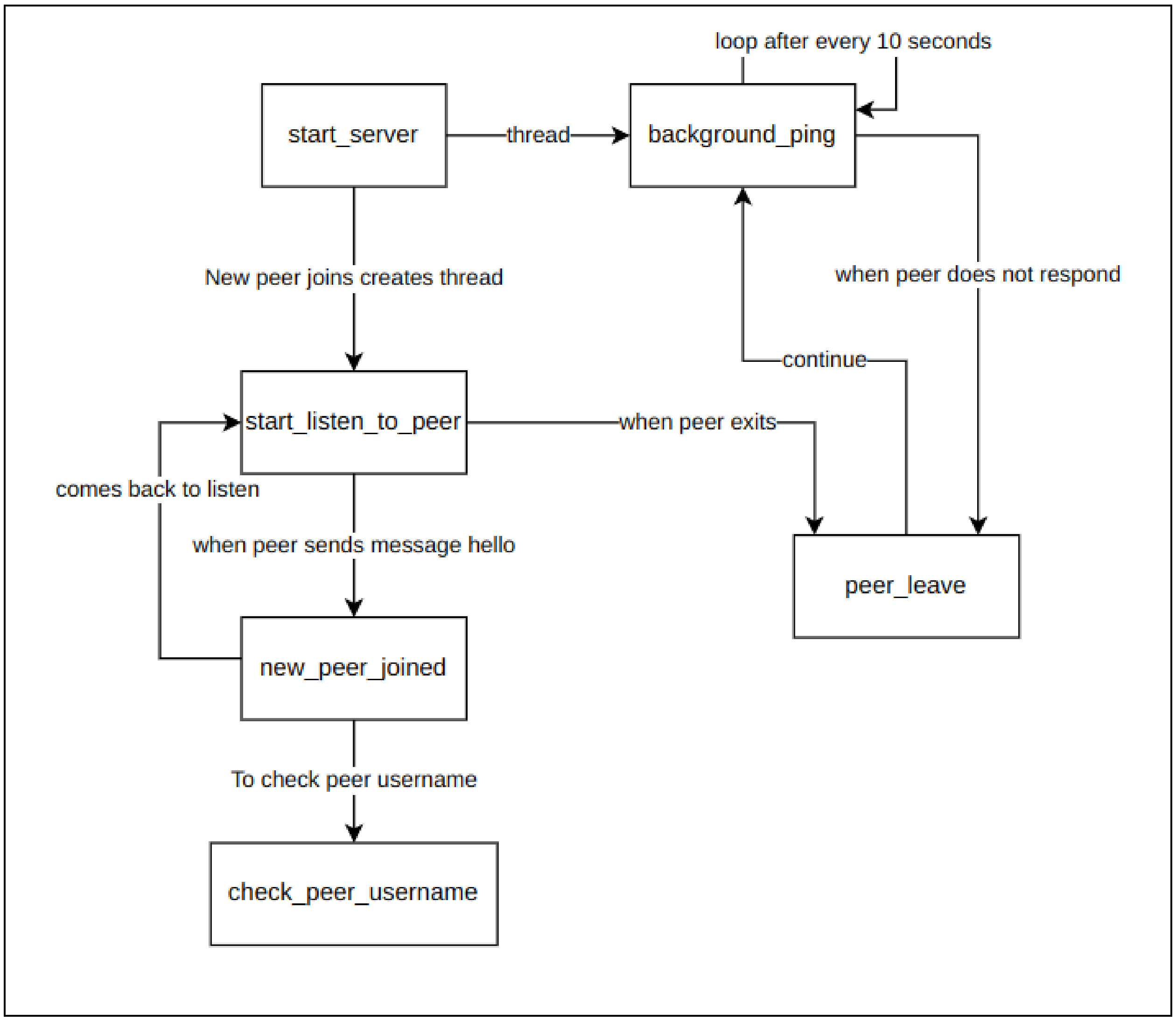
➔ ***peer_features(manager_peer_socket):*** This function is used to display the options available to the peer and to perform the operations based on the user's choice.

➔ ***ask_and_recieve_file(file_name):*** This function asks for the file from the peers and receives the file chunk by chunk from peers who have the file and then combines all the chunks to form the file and handles errors, if any. It receives file in parallel from multiple peers by using threads.

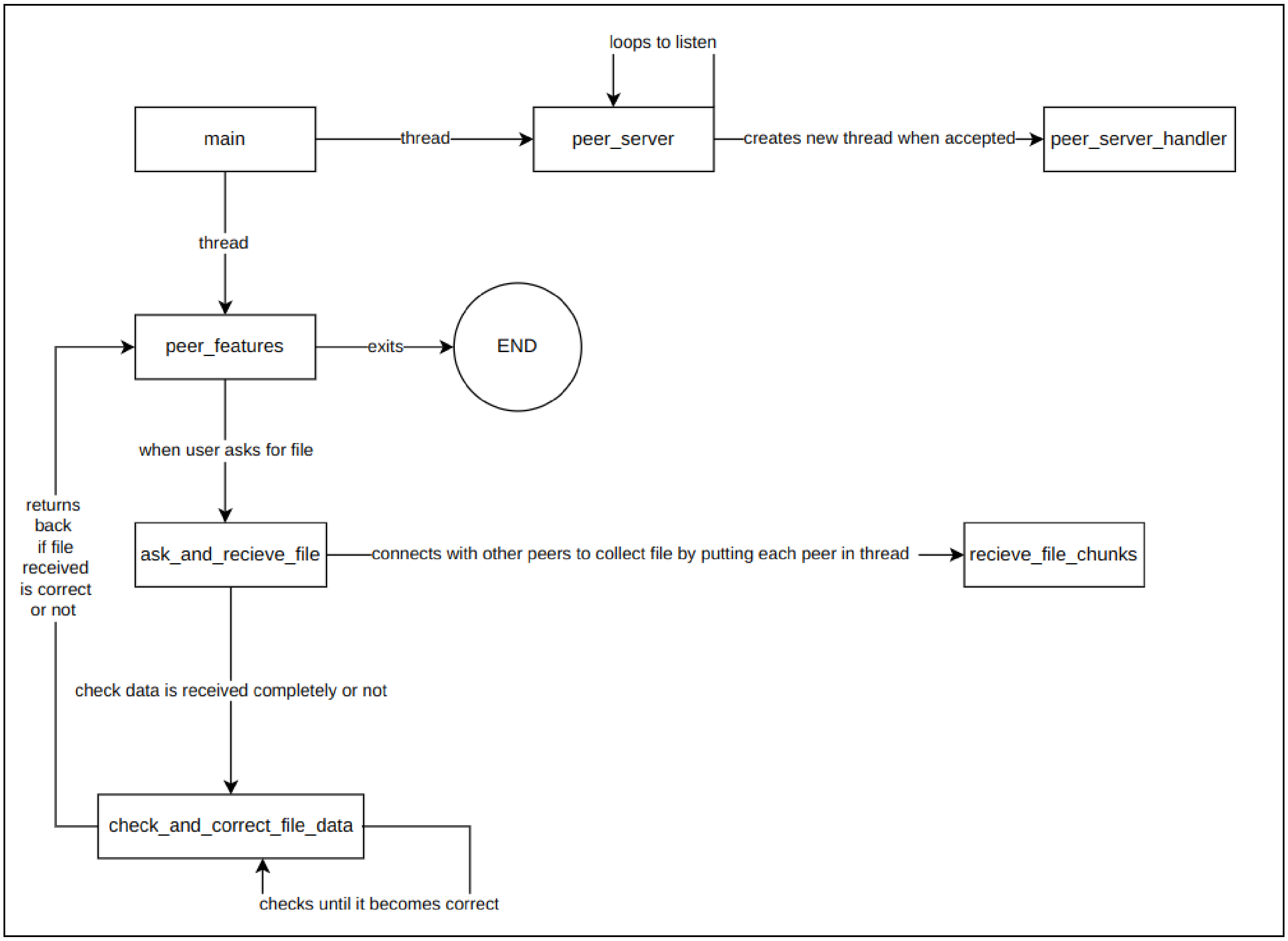
➔ ***recieve_file_chunks(conn_peer_socket,offset,number_of_chunks, size_of_chunk, file_data, index,error_index):*** This function is used to receive the file chunks from the peer and store it in the file_data list. If the peer is not able to send the data, then it will add the peer to the error_index list.

➔ ***check_and_correct_file_data(file_data, error_index, peer_with_file):*** This function is used to check if the file data is correct and if not, then it will correct it by requesting the peer to send the data again. Till the time the file data is correct, it will keep on requesting the peer to send the data again from the remaining peers who have the file.

- **Program working**
  200030041_manager.py working:



200030041_peer.py working:


loops to listen
main
thread
peer_server
creates new thread when accepted
peer_server_handler
thread
peer_features
exits
END
when user asks for file
returns back if file received is correct or not
ask_and_recieve_file
connects with other peers to collect file by putting each peer in thread
recieve_file_chunks
check data is received completely or not
check_and_correct_file_data
checks until it becomes correct