



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

SRM Nagar, Kattankulathur – 603 203,

SCHOOL OF COMPUTING

DEPARTMENT OF NETWORKING & COMMUNICATIONS

Course Project

**Title : Multi-threaded File Transfer
using TCP Socket in Python**

Course Code : 18CSC310J

Course Name : Data Centric Networking and System Design

Faculty : Dr. N. Senthamarai

Team Members:

Nune Nitesh – RA2011028010124

Marla Sai Ruthwik – RA2011028010124

Manthuri Hrithikesh – RA2011028010133

Gatta Venkata Amruth – RA2011028010136

INDEX

No	Particulars	PgNo.
1.	Objective	3
2.	Methodology	3
3.	Codes	5
4.	Demo and screenshot	9
5.	Proof of GitHub upload	10
6.	References	10

OBJECTIVE:

A Multi threaded file transfer client server program build using a python programming language. The server has a capability to handle multiple clients concurrently at the same time by using threading. The server assigns each client a thread to handle the work for the client.

PROJECT OVERVIEW:

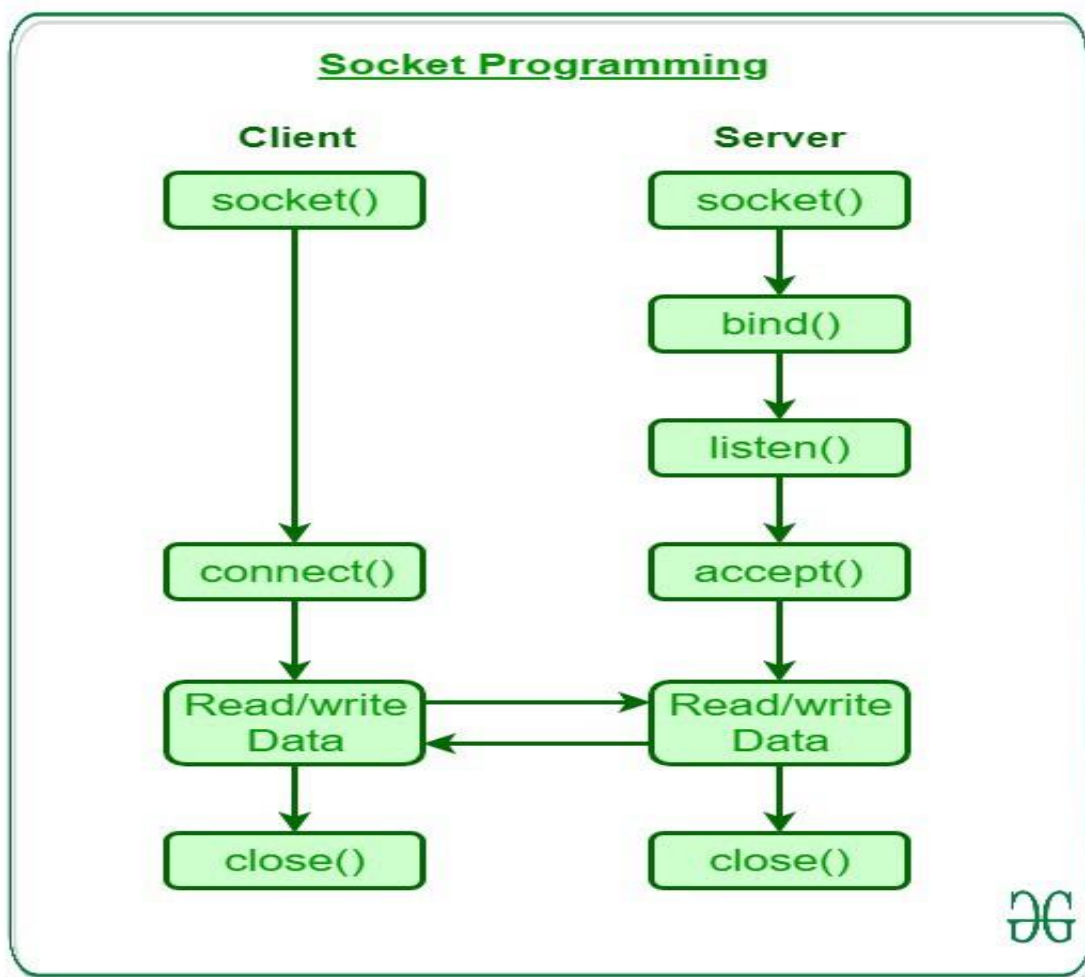
The server program can handle an arbitrary number of concurrent connections and file exchanges, only limited by system configuration or memory. The server is started without any parameters and creates a TCP socket at an OS-assigned port. It prints out the assigned port number and stores it in a local file port, which is used when starting clients. The server listens on its main socket and accepts client connections as they arrive. Clients perform an upload or download operation or instruct the server to terminate.

REQUIREMENT ANALYSIS

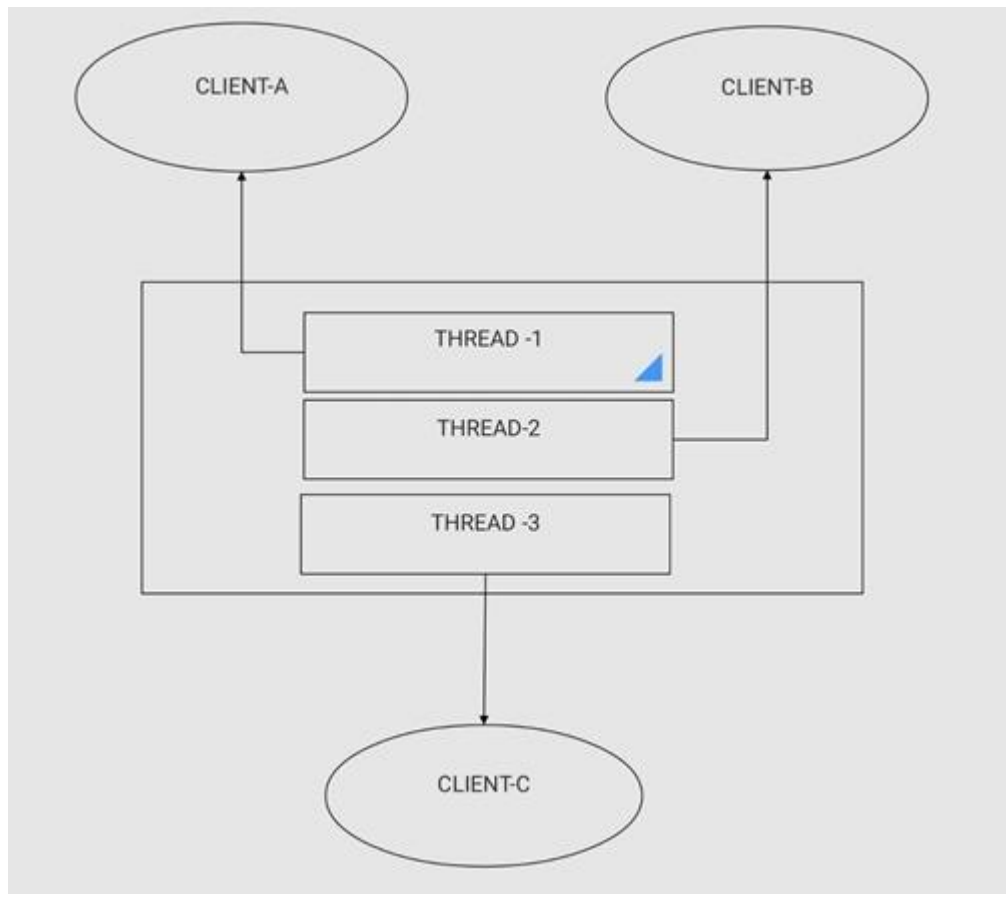
- Socket programming
- Multi-threading
- Programming Language – PYTHON
- Server side and Client side script
- Platform used - ATOM

ARCHITECTURE AND DESIGN

SERVER CLIENT ARCHITECTURE



MULTITHREADING ARCHITECTURE



BASIC FUNCTIONALITIES

- LIST : List all the files from the server data.
- UPLOAD : Upload a file to the server data.
- DOWNLOAD : Download a file to the client data.
- DELETE : Delete a file from the server data.
- LOGOUT : Disconnect from the server.
- HELP : List all the commands.

CODE PARAMETERS

- PORT : Specifies port address, present in both client and server.
- FORMAT : Specifies format of packet that is sent or received over the network.
- SERVER_DATA_PATH : Specifies the server data path in both client and server side.
- CLIENT_DATA_PATH : Specifies the data path in client side.
- ADDR : Holds the information about the IP address and Port address.

CODES:

Client.py

```
import socket
```

```
IP = socket.gethostbyname(socket.gethostname())
```

```
PORT = 4456
```

```
ADDR = (IP, PORT)
```

```
FORMAT = "utf-8"
```

```
SIZE = 1024
```

```
def main():
```

```
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
    client.connect(ADDR)
```

```
    while True:
```

```
        data = client.recv(SIZE).decode(FORMAT)
```

```
        cmd, msg = data.split("@")
```

```
        if cmd == "DISCONNECTED":
```

```
            print(f"[SERVER]: {msg}")
```

```
            break
```

```
        elif cmd == "OK":
```

```
            print(f"{msg}")
```

```
        data = input("> ")
```

```
        data = data.split(" ")
```

```
        cmd = data[0]
```

```

if cmd == "HELP":
    client.send(cmd.encode(FORMAT))
elif cmd == "LOGOUT":
    client.send(cmd.encode(FORMAT))
    break
elif cmd == "LIST":
    client.send(cmd.encode(FORMAT))
elif cmd == "DELETE":
    client.send(f"{cmd}@{data[1]}".encode(FORMAT))
elif cmd == "UPLOAD":
    path = data[1]

    with open(f"{path}", "r") as f:
        text = f.read()

    filename = path.split("/")[-1]
    send_data = f"{cmd}@{filename}@{text}"
    client.send(send_data.encode(FORMAT))

print("Disconnected from the server.")
client.close()

if __name__ == "__main__":
    main()

```

Server.py

```

import os
import socket
import threading

IP = socket.gethostname(socket.gethostname())
PORT = 4456
ADDR = (IP, PORT)
SIZE = 1024
FORMAT = "utf-8"
SERVER_DATA_PATH = "server_data"

```

```
def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")
    conn.send("OK@Welcome to the File Server.".encode(FORMAT))

    while True:
        data = conn.recv(SIZE).decode(FORMAT)
        data = data.split("@")
        cmd = data[0]

        if cmd == "LIST":
            files = os.listdir(SERVER_DATA_PATH)
            send_data = "OK@"

            if len(files) == 0:
                send_data += "The server directory is empty"
            else:
                send_data += "\n".join(f for f in files)
            conn.send(send_data.encode(FORMAT))

        elif cmd == "UPLOAD":
            name, text = data[1], data[2]
            filepath = os.path.join(SERVER_DATA_PATH, name)
            with open(filepath, "w") as f:
                f.write(text)

            send_data = "OK@File uploaded successfully."
            conn.send(send_data.encode(FORMAT))

        elif cmd == "DELETE":
            files = os.listdir(SERVER_DATA_PATH)
            send_data = "OK@"
            filename = data[1]

            if len(files) == 0:
                send_data += "The server directory is empty"
            else:
                if filename in files:
                    os.system(f"rm {SERVER_DATA_PATH}/{filename}")
                    send_data += "File deleted successfully."
                else:
                    send_data += "File not found."
```



```
conn.send(send_data.encode(FORMAT))

elif cmd == "LOGOUT":
    break
elif cmd == "HELP":
    data = "OK@"
    data += "LIST: List all the files from the server.\n"
    data += "UPLOAD <path>: Upload a file to the server.\n"
    data += "DELETE <filename>: Delete a file from the server.\n"
    data += "LOGOUT: Disconnect from the server.\n"
    data += "HELP: List all the commands."

conn.send(data.encode(FORMAT))

print(f"[DISCONNECTED] {addr} disconnected")
conn.close()

def main():
    print("[STARTING] Server is starting")
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(ADDR)
    server.listen()
    print(f"[LISTENING] Server is listening on {IP}:{PORT}.")

    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()
        print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 1}")

if __name__ == "__main__":
    main()
```

DEMO & SCREENSHOT:

Server side code

```
server.py      client.py
1  import os
2  import socket
3  import threading
4
5  IP = socket.gethostbyname(socket.gethostname())
6  PORT = 5544
7  ADDR = (IP ,PORT)
8  SIZE = 1024
9  FORMAT = "utf-8"
10 SERVER_DATA_PATH = "server_data"
11
12 """
13 CMD@Msg
14 """
15 def handle_client(conn,addr):
16     print(f"[NEW CONNECTION] {addr} connected.")
17     conn.send("OK@Welcome to the File Server.".encode(FORMAT))
```

```

server.py
client.py

19 while True:
20     data = conn.recv(SIZE).decode(FORMAT)
21     data = data.split("@")
22     cmd = data[0]
23
24     if cmd == "HELP":
25         send_data = "OK@"
26         send_data += "LIST: List all the files from the server.\n"
27         send_data += "UPLOAD <path>: Upload a file to the server.\n"
28         send_data += "DOWNLOAD <filename>: Download a file from the server.\n"
29         send_data += "DELETE <filename>: Delete a file from the server.\n"
30         send_data += "LOGOUT: Disconnect from the server.\n"
31         send_data += "HELP: List all the commands."
32
33         conn.send(send_data.encode(FORMAT))
34
35     elif cmd == "LOGOUT":
36         break
37
38     elif cmd == "LIST":
39         files = os.listdir(SERVER_DATA_PATH)
40         send_data = "OK@"

```

```

server.py
client.py

38 elif cmd == "LIST":
39     files = os.listdir(SERVER_DATA_PATH)
40     send_data = "OK@"
41
42     if len(files) == 0:
43         send_data += "The server directory is empty."
44     else:
45         send_data += "\n".join(f for f in files)
46         conn.send(send_data.encode(FORMAT))
47
48
49 elif cmd == "UPLOAD":
50     name = data[1]
51     text = data[2]
52
53     filepath = os.path.join(SERVER_DATA_PATH , name)
54     with open(filepath , "w") as f:
55         f.write(text)
56
57     send_data = "OK@File uploaded."
58     conn.send(send_data.encode(FORMAT))

```

server.py

client.py

```
59
60     elif cmd == "DOWNLOAD":
61         name = data[1]
62
63         filepath = os.path.join(SERVER_DATA_PATH, name)
64
65         with open(filepath, "r") as f:
66             text = f.read()
67
68         filepath = data[2]
69
70         with open(filepath, "w") as f:
71             f.write(text)
72
73         send_data = "OK@File downloaded successfully."
74         conn.send(send_data.encode(FORMAT))
75
76     elif cmd == "DELETE":
77         files = os.listdir(SERVER_DATA_PATH)
78         send_data = "OK@"
79         filename = data[1]
```

server.py

client.py

```
81     send_data += "The server directory is empty."
82     else:
83         if filename in files:
84             os.system(f"del {SERVER_DATA_PATH}\\{filename}")
85             send_data += "File deleted."
86         else:
87             send_data += "File not found."
88
89         conn.send(send_data.encode(FORMAT))
90     print(f"[DISCONNECTED] {addr} disconnected")
91 def main():
92     print("[STARTING] Server is starting")
93     server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
94     server.bind(ADDR)
95     server.listen()
96     print("[LISTENING] Server is listening")
97     while True:
98         conn, addr = server.accept()
99         thread = threading.Thread(target = handle_client, args=(conn, addr) )
100         thread.start()
101 if __name__ == "__main__":
102     main()
```

Client side code

```
server.py  client.py
1  import os
2  import socket
3
4  IP = socket.gethostbyname(socket.gethostname())
5  PORT = 5544
6  ADDR = (IP, PORT)
7  SIZE = 1024
8  FORMAT = "utf-8"
9  SERVER_DATA_PATH = "server_data"
10 CLIENT_DATA_PATH = "client_data"
11
12 def main():
13     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
14     client.connect(ADDR)
15     while True:
16         data = client.recv(SIZE).decode(FORMAT)
17         cmd, msg = data.split("@")
18         if cmd == "OK":
19             print(f"{msg}")
20         elif cmd == "DISCONNECTED":
21             print(f"{msg}")
22         break
```

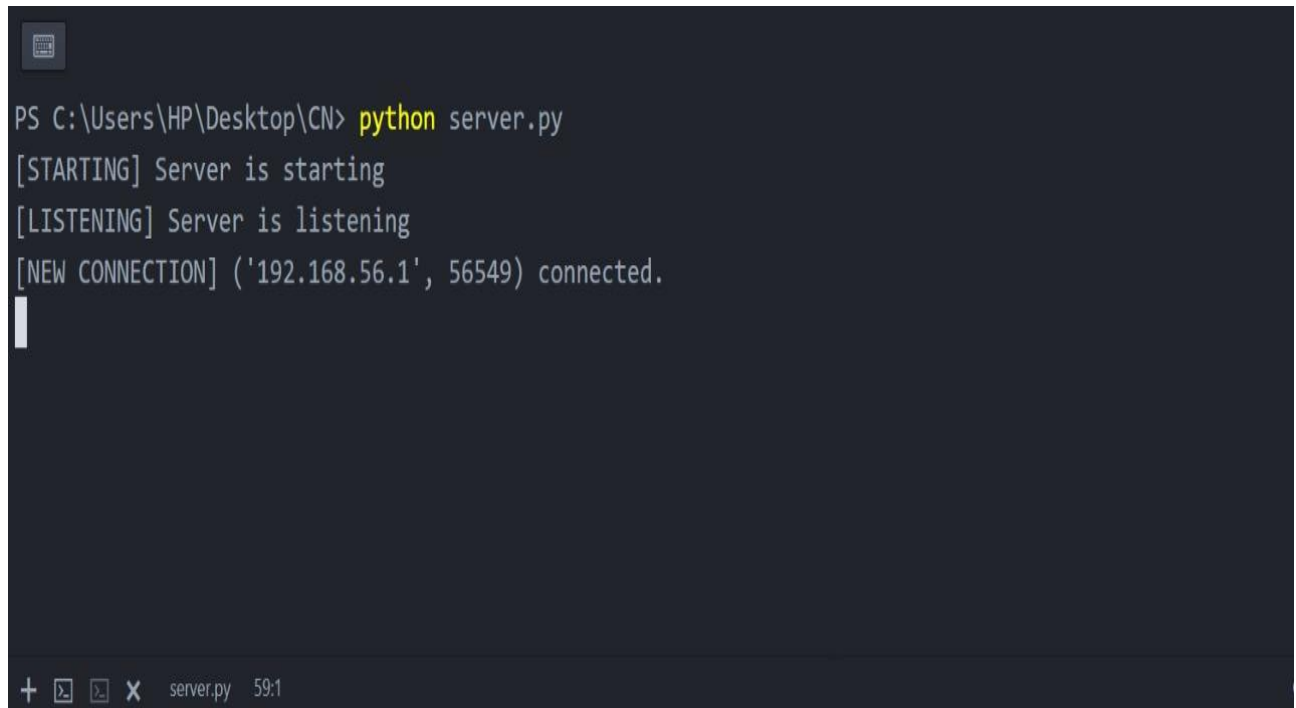
```
server.py  client.py
20  elif cmd == "DISCONNECTED":
21      print(f"{msg}")
22      break
23
24  data = input("> ")
25  data = data.split(" ")
26  cmd = data[0]
27
28  if cmd == "HELP":
29      client.send(cmd.encode(FORMAT))
30
31  elif cmd == "LOGOUT":
32      client.send(cmd.encode(FORMAT))
33      break
34
35  elif cmd == "LIST":
36      client.send(cmd.encode(FORMAT))
37
38  elif cmd == "UPLOAD":
39      ## UPLOAD@filename@text
40      path = data[1]
41      with open(f"{path}", "r") as f:
```

```

server.py      client.py
36
37     elif cmd == "UPLOAD":
38         ## UPLOAD@filename@text
39         path = data[1]
40         with open(f"{path}", "r") as f:
41             text = f.read()
42         ##client_data / data.txt
43         filename = path.split("/")[-1]
44         send_data = f"{cmd}@{filename}@{text}"
45         client.send(send_data.encode(FORMAT))
46
47     elif cmd == "DOWNLOAD":
48         filepath = os.path.join(CLIENT_DATA_PATH, data[1])
49         client.send(f"{cmd}@{data[1]}@{filepath}".encode(FORMAT))
50
51     elif cmd == "DELETE":
52         client.send(f"{cmd}@{data[1]}".encode(FORMAT))
53     print("Disconnected from the server.")
54     client.close()
55
56 if __name__ == "__main__":
57     main()

```

Server side output

A terminal window with a dark background and a small icon in the top-left corner. The text is white and yellow. The output shows the execution of a Python script that starts a server, listens for connections, and successfully connects to a client at 192.168.56.1.

```
PS C:\Users\HP\Desktop\CN> python server.py
[STARTING] Server is starting
[LISTENING] Server is listening
[NEW CONNECTION] ('192.168.56.1', 56549) connected.

```

server.py 59:1

Client side output

A terminal window with a dark background and a small icon in the top-left corner. The text is white and yellow. The output shows the execution of a Python script that starts a client, connects to a server, and displays a help menu with various commands like LIST, UPLOAD, DOWNLOAD, DELETE, and LOGOUT.

```
PS C:\Users\HP\Desktop\CN> python client.py
Welcome to the File Server.
> HELP
LIST: List all the files from the server.
UPLOAD <path>: Upload a file to the server.
DOWNLOAD <filename>: Download a file from the server.
DELETE <filename>: Delete a file from the server.
LOGOUT: Disconnect from the server.
HELP: List all the commands.
>

```

server.py 59:1

PROOF OF GITHUB UPLOAD:

Repository Link: https://github.com/Nunenitesh/THREADS_TCP

The screenshot shows a GitHub repository page for 'Nunenitesh / THREADS_TCP'. The repository is private and has 1 branch (main) and 0 tags. The commit history shows three commits: 'Initial commit' (3 minutes ago), 'Create client.py' (2 minutes ago), and 'Create server.py' (1 minute ago). The README.md file is visible, titled 'THREADS_TCP', with the description 'Multi-threaded File Transfer using TCP Socket in Python'. The right sidebar shows the repository's metadata: 'Multi-threaded File Transfer using TCP Socket in Python', 0 stars, 1 watching, and 0 forks. The 'Releases' and 'Packages' sections indicate no releases or packages have been published.

https://github.com/Nunenitesh/THREADS_TCP

Search or jump to... Pull requests Issues Codespaces Marketplace Explore

Nunenitesh / THREADS_TCP Private Unwatch 1 Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags Go to file Add file <> Code About

Nunenitesh Create server.py c64e3f7 1 minute ago 3 commits

README.md	Initial commit	3 minutes ago
client.py	Create client.py	2 minutes ago
server.py	Create server.py	1 minute ago

README.md

THREADS_TCP

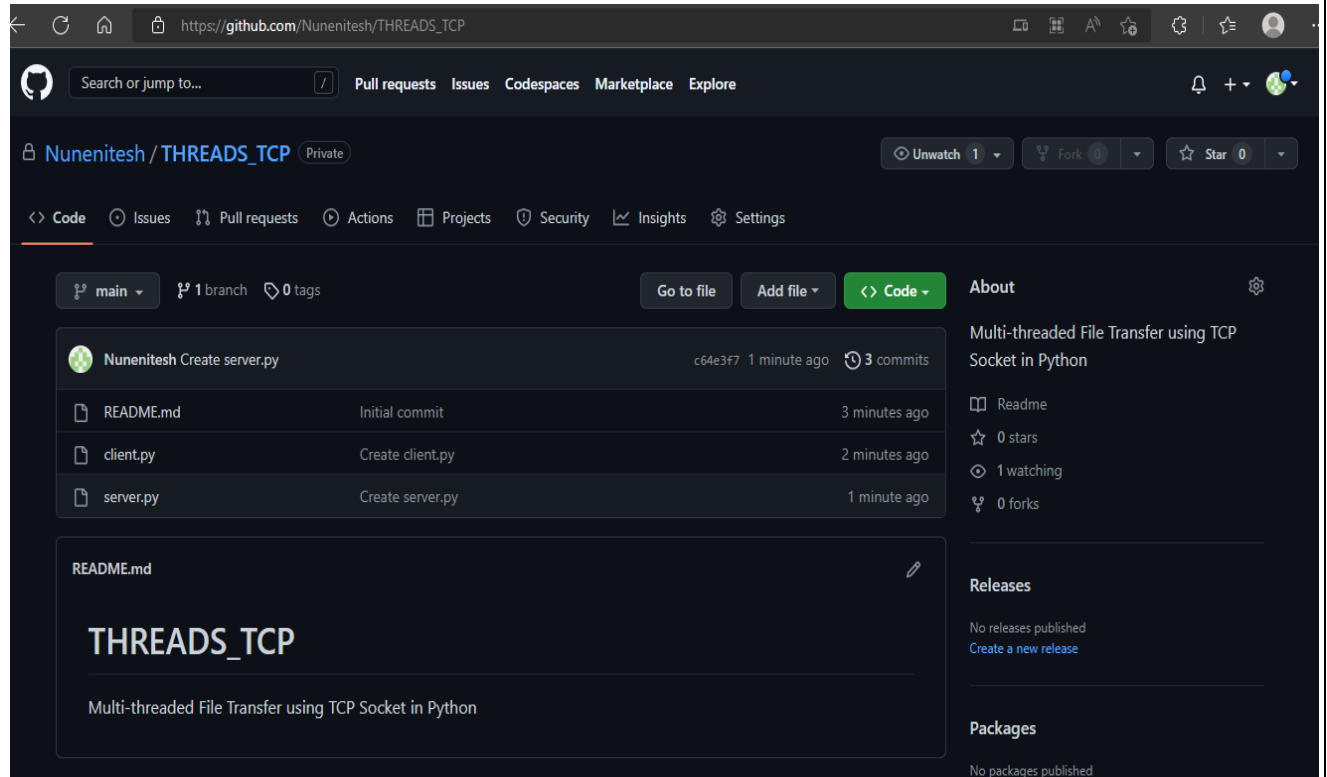
Multi-threaded File Transfer using TCP Socket in Python

Multi-threaded File Transfer using TCP Socket in Python

Readme 0 stars 1 watching 0 forks

Releases No releases published Create a new release

Packages No packages published



REFERENCES:

- <https://github.com/nikhilroxtomar/Multithreaded-File-Transfer-using-TCP-Socket-in-Python>
- <https://www.geeksforgeeks.org/multithreading-python-set-1/>