

NETWORKS FINAL PROJECT

The provided code is a server program in Python that utilizes the User Datagram Protocol (UDP) to send data to clients in a reliable manner. The program is designed to send files to clients and receive acknowledgement packets from the clients.

Server:

- starts by importing necessary libraries, including socket, threading, hashlib, and time.
- It sets the global variables for the server port, IP address, delimiter, and packet loss flag.
- The program defines a packet class that contains the checksum, sequence number, length, and message fields. The makePacket function calculates the checksum and sets the length and message fields based on the input data.
- The handle_client function is called whenever a client connects to the server. It takes the client address and data as input parameters.
- The function opens the requested file and reads its contents into a variable.
- It sends the file data to the client in chunks of 500 bytes. It sets a timeout of 2 seconds for each packet sent to the client.
- If the client does not acknowledge receipt of a packet within the timeout period, the server resends the packet.
- The program then starts listening on the specified port for incoming client connections. When a client connects, the handle_client function is called on a new thread.

Client:

sends requests to a server over a UDP socket. The client sends a file name to the server and receives the file content if it exists on the server. The code makes use of the hashlib library to calculate the checksum of the data sent and received, and it also uses the requests library to perform GET and POST requests.

The client program first sets up a socket with a timeout of 5 seconds. It then takes the file name as input from the user and sends it to the server using the `sendto()` method. The code uses a while loop to keep waiting for the server response. If the server response is not received within 5 seconds, the program retries up to 3 times before giving up.

When the server response is received, the program verifies the sequence number, checksum, and packet length. If the packets are matching, the program writes the

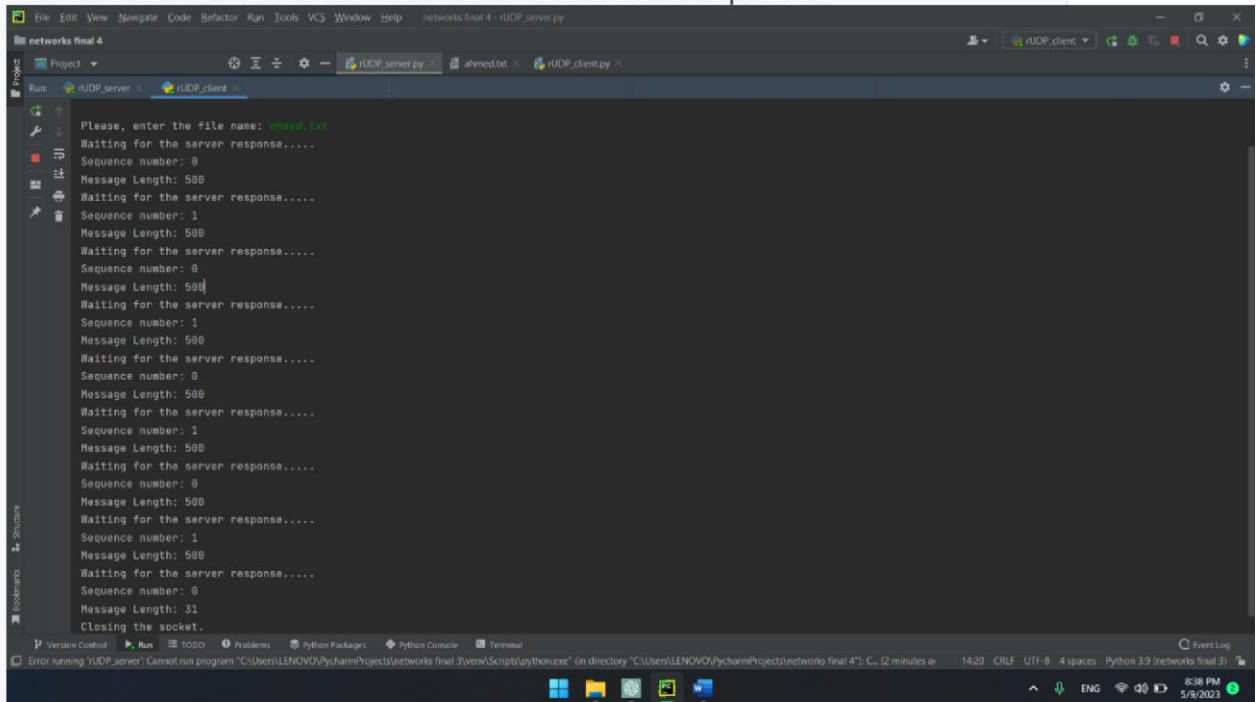
received data to a new file. If the packets do not match, the program ignores the packet and continues waiting for the next packet.

The program sends an acknowledgment (ack) message to the server for each packet received. When the last packet is received, the program breaks out of the loop and closes the socket.

The code also provides two functions, `GETrequest()` and `POSTrequest()`, which can be used to perform GET and POST requests using the requests library. These functions are not used in the main program.

1. Test case for requesting a file that exists:

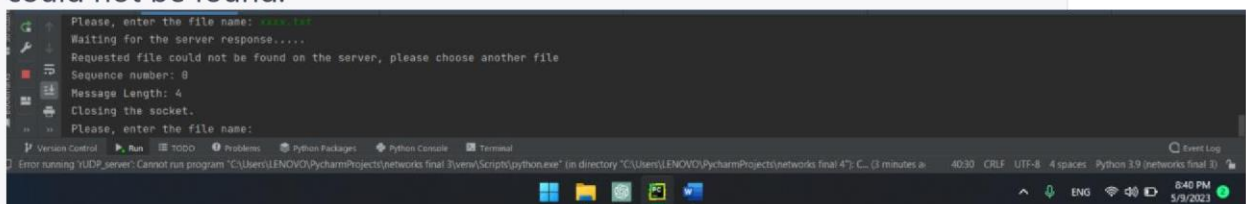
- Send a UDP message to the server containing the name of a text file that exists on the server.
- The server should respond by sending the file data in packets to the client.
- The client should acknowledge each packet received, and the server should re-send any packets that are not acknowledged.
- The test should end when the server sends the last packet.



```
networks final 4 - rUDP_server.py
Please, enter the file name: chase.txt
Waiting for the server response.....
Sequence number: 0
Message Length: 500
Waiting for the server response.....
Sequence number: 1
Message Length: 500
Waiting for the server response.....
Sequence number: 0
Message Length: 500
Waiting for the server response.....
Sequence number: 1
Message Length: 500
Waiting for the server response.....
Sequence number: 0
Message Length: 500
Waiting for the server response.....
Sequence number: 1
Message Length: 500
Waiting for the server response.....
Sequence number: 0
Message Length: 500
Waiting for the server response.....
Sequence number: 1
Message Length: 500
Waiting for the server response.....
Sequence number: 0
Message Length: 500
Waiting for the server response.....
Sequence number: 1
Message Length: 500
Waiting for the server response.....
Sequence number: 0
Message Length: 31
Closing the socket.
```

2. Test case for requesting a file that does not exist:

- Send a UDP message to the server containing the name of a text file that does not exist on the server.
- The server should respond with an error message indicating that the file could not be found.



```
networks final 3
Please, enter the file name: xxx.txt
Waiting for the server response.....
Requested file could not be found on the server, please choose another file
Sequence number: 0
Message Length: 4
Closing the socket.
```

3. Test case for packet loss:

- Simulate packet loss by setting the `packetLoss` variable to `True` before running the test case.



```
"C:\Users\LENOVO\PycharmProjects\networks final 3\venv\Scripts\python.exe" "C:\Users\LENOVO\PycharmProjects\networks final 4\rUDP_client.py"
Checksum mismatch detected or dropping packet
```