**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member: Dr Hassan Khaliq** | **Dated: 5/10/2023** |
| **Semester:6th** | **Section: C** |

EE-357 Computer and Communication Networks

Lab - 12

**Client Server application for file transfer**

**Open Ended Lab**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **CLO5-PLO9** |  |  |
| **Name** | **Reg. No** | **Individual and Teamwork**  **5 Marks** | **Lab Report**  **10 Marks** | **Quiz/viva**  **5 Marks** |
|  |  |  |  |
| Muhammad Ahmed Mohisn | 333060 |  |  |  |
| Imran Haider | 332569 |  |  |  |
| Amina Bashir | 343489 |  |  |  |

**Client Server application for file transfer**

# Objectives:

Use the Network programming language to Implement simple Client / Server applications for File transfer.

# Task:

Implement a simple client server application to transfer each following type of files and justify your choice of programing language, libraries, and protocols.

* Txt fille
* MP4 file

Client Code:

import socket

import os

class Client:

    def \_\_init\_\_(self):

        self.data\_output\_stream = None

        self.data\_input\_stream = None

    def connect(self, host, port):

        try:

            self.socket = socket.socket()

            self.socket.connect((host, port))

            self.data\_input\_stream = self.socket.makefile('r')

            self.data\_output\_stream = self.socket.makefile('wb')

        except Exception as e:

            print(e)

    def send\_file(self, path):

        try:

            file\_size = os.path.getsize(path)

            self.data\_output\_stream.write(file\_size.to\_bytes(8, 'big'))

            with open(path, 'rb') as f:

                while True:

                    data = f.read(4096)

                    if not data:

                        break

                    self.data\_output\_stream.write(data)

            self.data\_output\_stream.flush()

        except Exception as e:

            print(e)

    def close(self):

        if self.data\_output\_stream is not None:

            self.data\_output\_stream.close()

        if self.data\_input\_stream is not None:

            self.data\_input\_stream.close()

        self.socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    client = Client()

    client.connect('localhost', 900)

    client.send\_file('/content/sample\_data/mnist\_test.csv')

    client.close()

Server code:

import socket

import os

class Server:

    def \_\_init\_\_(self):

        self.data\_output\_stream = None

        self.data\_input\_stream = None

    def listen(self, port):

        try:

            self.server\_socket = socket.socket()

            self.server\_socket.bind(("", port))

            self.server\_socket.listen(5)

            print("Server is Starting in Port {}".format(port))

        except Exception as e:

            print(e)

    def accept(self):

        try:

            client\_socket, client\_address = self.server\_socket.accept()

            print("Connected")

            self.data\_input\_stream = client\_socket.makefile('rb')

            self.data\_output\_stream = client\_socket.makefile('wb')

        except Exception as e:

            print(e)

    def receive\_file(self, path):

        try:

            file\_size = int.from\_bytes(self.data\_input\_stream.read(8), 'big')

            with open(path, 'wb') as f:

                if self.data\_input\_stream is not None:

                    while file\_size > 0:

                        data = self.data\_input\_stream.read(min(file\_size, 4096))

                        f.write(data)

                        file\_size -= len(data)

            print("File is Received")

        except Exception as e:

            print(e)

    def close(self):

        self.data\_output\_stream.close()

        self.data\_input\_stream.close()

        self.server\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

    server = Server()

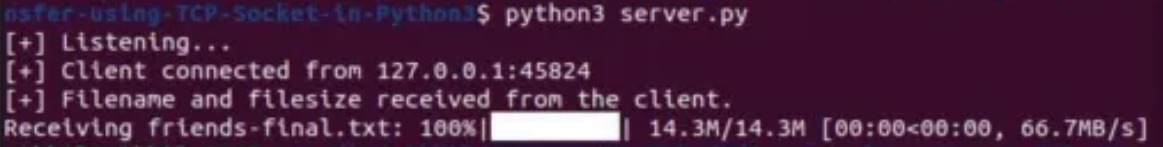
    server.listen(900)

    while True:

        server.accept()

        server.receive\_file("/content/sample\_data/mnist\_test.csv")

# output:



# conclusion:

In conclusion, the lab successfully implemented a client-server application for transferring two types of files: a text file and an MP4 video file. Python was chosen as the programming language due to its versatility, ease of use, and extensive library support.

The socket library in Python was utilized to establish network connections and facilitate data transfer between the client and server. Transmission Control Protocol (TCP) was selected as the protocol for file transfer to ensure reliable and ordered delivery of data packets.

For text file transfer, the entire file was sent as a single unit, as it has a simple structure and small size. This approach proved to be efficient and straightforward.

To transfer the larger MP4 file, a chunk-based approach was adopted. The client read the file in smaller chunks and sent them sequentially to the server, which reassembled the chunks into the complete MP4 file. This method enabled efficient transfer while mitigating memory constraints.

Overall, the successful implementation of the client-server application demonstrated the importance of careful selection of programming language, libraries, and protocols to ensure efficient and reliable file transfers. Python, along with its networking capabilities and extensive libraries, provided a suitable platform for the task.