

CN Lab 11

AIM: Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Lab 11

1. Using TCP/IP sockets, write a client-server program to make client sending the file name & the server to send back the contents of the requested file if present.

client.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("Enter file name")

clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print("From Server", filecontents)
clientSocket.close()
```

Server.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
print("The server is ready to receive")
while 1:
    connectionSocket, addr = serverSocket.accept()
```

```

Sentence = connectionSocket.recv(1024).decode()

file = open(Sentence, "r")
l = file.read(1024)

connectionSocket.send(l.encode())
file.close()
connectionSocket.close()

Output:
The server is ready to receive
Enter file name: example.txt
This is the message sent

example.txt
This is the message sent

```

Code:

ClientTCP.py

```

from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input('Enter file name: ')
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print('From Server:')
print(filecontents)
clientSocket.close()

```

ServerTCP.py

```

from socket import *
serverName = '127.0.0.1'
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
    print('The server is ready to receive')
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file = open(sentence, 'r')
    l = file.read(1024)
    connectionSocket.send(l.encode())
    print('Sent contents of ' + sentence)
    file.close()
    connectionSocket.close()

```

Output

```
import os
import subprocess
import time

# Folder and file details
folder_path = "/content/drive/MyDrive/CN_tcp_udp" # Change this to the full path of your folder
requested_file = "example.txt" # File to request from the server

# Verify the folder exists
if not os.path.isdir(folder_path):
    raise FileNotFoundError(f"Folder '{folder_path}' not found. Ensure the drive is mounted.")

# Change the working directory
os.chdir(folder_path)

# Function to run a script and capture output
def run_script(script, args=None):
    process = subprocess.Popen(
        ["python", script] + (args or []),
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE
    )
    stdout, stderr = process.communicate()
    return stdout.decode(), stderr.decode()

try:
    # Start the server
    print("Starting server.py...")
    server_process = subprocess.Popen(["python", "server.py"])

    # Allow the server to initialize
    time.sleep(3)

    # Run the client
    print("Starting client.py...")
    client_stdout, client_stderr = run_script("client.py", [requested_file])

    # Display client output
    print("\nClient Output:")
    print(client_stdout)
    if client_stderr:
        print("\nClient Errors:")
        print(client_stderr)

finally:
    # Terminate the server process
    server_process.terminate()
    print("\nServer process terminated.")
```

```
Starting server.py...
Starting client.py...

Client Output:
From Server: This is the file

Server process terminated.
```


AIM: Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

2. Using UDP Sockets, write a client-server program to make client sending the filename and the server to send back the contents of the requested file if present.

clientUDP.py

```
from socket import *
ServerName = "127.0.0.1"
ServerPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
Sentence = input("Enter file name")
clientSocket.sendto(bytes(Sentence, "utf-8"),
                    (ServerName, ServerPort))
fileContents, serverAddress = clientSocket.recvfrom(2048)
print("From Server:", fileContents)
clientSocket.close()
```

ServerUDP.py

```
from socket import *
ServerPort = 12000
ServerSocket = socket(AF_INET, SOCK_DGRAM)
ServerSocket.bind(("127.0.0.1", ServerPort))
print("The server is ready to receive")
while 1:
    Sentence, ClientAddress = ServerSocket.recvfrom(2048)
    file = open(Sentence, "r")
    I = file.read(2048)
    ServerSocket.sendto(bytes(I, "utf-8"), ClientAddress)
    print("Sent back to client", I)
    file.close()
```

Output:

The server is ready to receive
Enter file name = example.txt
This is the message sent

Sr
Sudhakar

Code:

ClientUDP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("Enter file name: ")
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048)
print ("Reply from Server:")
print (filecontents.decode("utf-8"))
# for i in filecontents:
# print(str(i), end = " ")
clientSocket.close()
clientSocket.close()
```

ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print ("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file=open(sentence,"r")
    con=file.read(2048)
    serverSocket.sendto(bytes(con,"utf-8"),clientAddress)
    print ("Sent contents of ", end = " ")
    print (sentence)
    # for i in sentence:
    # print (str(i), end = " ")
    file.close()
```

Output

```
import socket
import threading

# Server Code
def start_server():
    serverPort = 12000
    serverSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    serverSocket.bind(("127.0.0.1", serverPort))
    print("The server is ready to receive")

    while True:
        sentence, clientAddress = serverSocket.recvfrom(2048)
        filename = sentence.decode("utf-8")

        try:
            # Try to open the file and send its contents
            with open(filename, "r") as file:
                file_contents = file.read(2048)
                serverSocket.sendto(file_contents.encode("utf-8"), clientAddress)
                print(f"Sent back to client: {file_contents}")
        except FileNotFoundError:
            # If file not found, notify the client
            serverSocket.sendto("File not found.".encode("utf-8"), clientAddress)
            print(f"File '{filename}' not found.")
        except Exception as e:
            serverSocket.sendto(f"Error: {str(e)}.encode("utf-8"), clientAddress)
            print(f"Error processing file '{filename}': {str(e)}")

# Client Code
def start_client(filename):
    serverName = "127.0.0.1"
    serverPort = 12000
    clientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # Send the file name to the server
    clientSocket.sendto(filename.encode("utf-8"), (serverName, serverPort))

    # Receive the file contents or error message from the server
    filecontents, serverAddress = clientSocket.recvfrom(2048)
    print(f"From Server: {filecontents.decode('utf-8')}")
    clientSocket.close()

# Run the server in a separate thread
server_thread = threading.Thread(target=start_server)
server_thread.daemon = True # This makes the server thread stop when the main program exits
server_thread.start()

# Simulate the client-side file request
requested_file = "example.txt" # Ensure this file exists in your working directory for testing
start_client(requested_file)
```

```
The server is ready to receive
From Server: This is the file
Sent back to client: This is the file
```