

NAME: RESHMA AL

DATE:25.08.25

ROLL NO.:241901089

EXERCISE 3

UDP CLIENT-SERVER COMMUNICATION USING SOCKET PROGRAMMING IN PYTHON

AIM:

To implement UDP client-server communication using socket programming in Python.

ALGORITHM:

SERVER:

1. Create a socket using `socket.socket()`.
2. Bind the socket to IP and port using `bind()`.
3. Receive message using `recvfrom ()`.
4. Send response using `sendto ()`.
5. Close connection.

CLIENT:

1. Create a socket using `socket.socket()`.
2. Send a message to the server using `sendto ()`.
3. Receive response using `recvform ()`.
4. Close connection.

CODE:SERVER:

```
import socket

sockfd = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

print("UDP Socket Created")

sockfd.bind(('localhost', 55555))

print("Waiting for messages on localhost:55555")

while True:

    data, addr = sockfd.recvfrom(1024)

    message = data.decode('utf-8')

    print(f'Received from {addr}: {message}')


    sockfd.sendto(data, addr)

    print(f'Echoed back to {addr}')

    choice = input("Continue receiving? (y/n): ")

    if choice.lower() == 'n':

        break

sockfd.close()

print("Server shutdown")
```

CLIENT:

```
import socket

clientfd = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_addr = ('localhost', 55555)

msg = input("Enter your message: ")

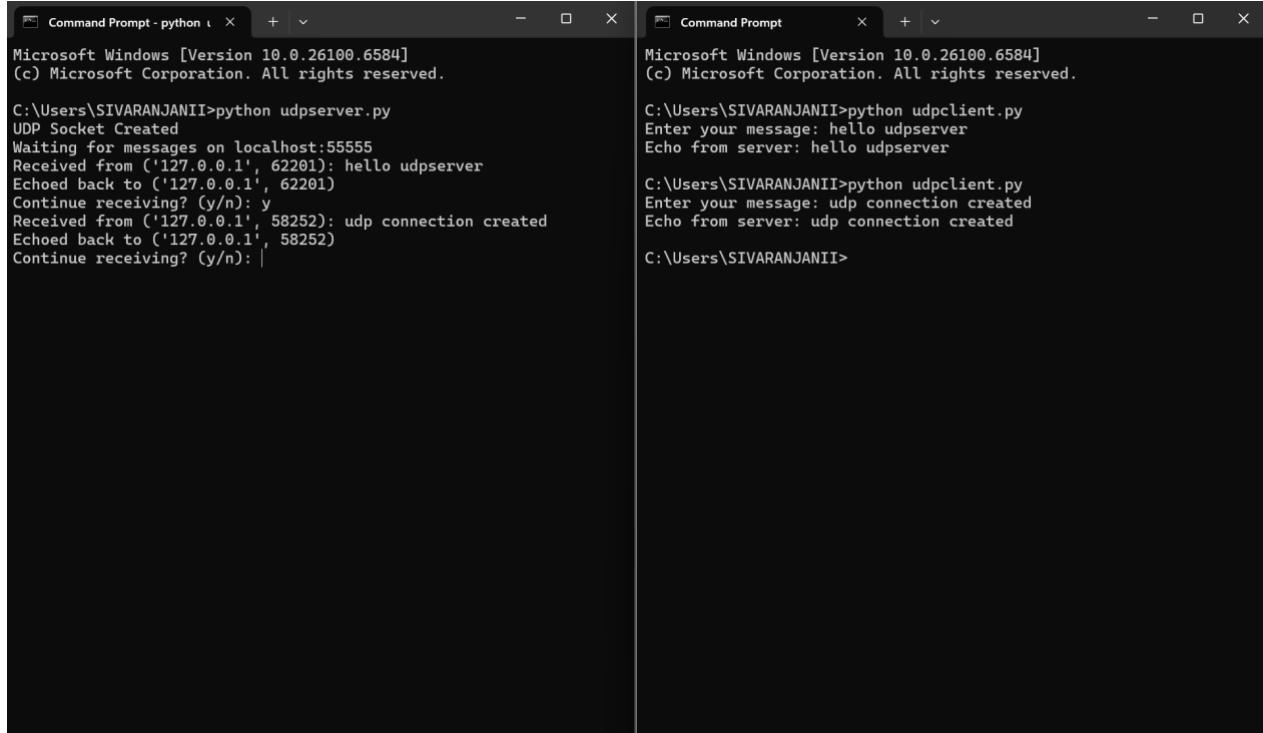
clientfd.sendto(msg.encode('utf-8'), server_addr)

data, _ = clientfd.recvfrom(1024)
```

```
print("Echo from server:", data.decode('utf-8'))
```

```
clientfd.close()
```

OUTPUT:



The image shows two Microsoft Windows Command Prompt windows side-by-side. The left window, titled 'Command Prompt - python', displays the output of a UDP server application. It starts with system information, then shows the command 'python udpserver.py' being run. The server creates a UDP socket and listens on port 55555. It receives a message from the client at ('127.0.0.1', 62201) containing 'hello udpserver'. It then echoes the message back to the client at ('127.0.0.1', 62201). The user is prompted to continue receiving messages. The right window, also titled 'Command Prompt', shows the output of a UDP client application. It starts with system information, then shows the command 'python udpclient.py' being run. The user is prompted to enter a message. The user types 'hello udpserver'. The client sends this message to the server. The server's response is received by the client, which then displays 'Echo from server: hello udpserver'. The client then exits, indicated by the command prompt 'C:\Users\SIVARANJANII>'.

```
Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SIVARANJANII>python udpserver.py
UDP Socket Created
Waiting for messages on localhost:55555
Received from ('127.0.0.1', 62201): hello udpserver
Echoed back to ('127.0.0.1', 62201)
Continue receiving? (y/n): y
Received from ('127.0.0.1', 58252): udp connection created
Echoed back to ('127.0.0.1', 58252)
Continue receiving? (y/n): |
```

```
Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SIVARANJANII>python udpclient.py
Enter your message: hello udpserver
Echo from server: hello udpserver

C:\Users\SIVARANJANII>python udpclient.py
Enter your message: udp connection created
Echo from server: udp connection created

C:\Users\SIVARANJANII>
```

RESULT:

Thus, UDP client-server communication was successfully implemented using Python.