

1. Lab Manual: Developing a TCP Client-Server Application using Python Socket Programming

Lab Setup:

1. Setting up the Environment:

- Ensure that you have a Linux environment with Python installed.
- Open a terminal.

2. Creating the Project Directory:

- Create a new directory for your project.

```
bash
```

```
2. mkdir tcp_message_lab  
cd tcp_message_lab
```

Part 1: Server Side

Step 1: Writing the Server Code

1. Create a file named `server.py` in the project directory.

```
python
```

```
1. # server.py  
import socket  
  
PORT = 8080  
MAX_BUFFER_SIZE = 1024  
  
def main():  
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
    server_socket.bind(('0.0.0.0', PORT))  
    server_socket.listen(3)  
  
    print("Server listening on port", PORT)  
  
    client_socket, client_address = server_socket.accept()  
    print("Accepted connection from", client_address)  
  
    data = client_socket.recv(MAX_BUFFER_SIZE).decode('utf-8')  
    print("Received message from client:", data)  
  
    client_socket.close()  
    server_socket.close()  
  
if __name__ == "__main__":  
    main()
```

Step 2: Running the Server Code

1. Run the server.

```
bash
```

1. `python server.py`

Part 2: Client Side

Step 1: Writing the Client Code

1. Create a file named `client.py` in the project directory.

```
python
```

1.

```
# client.py
import socket

PORT = 8080
MAX_BUFFER_SIZE = 1024

def main():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_address = ('127.0.0.1', PORT)

    try:
        client_socket.connect(server_address)
        print("Connected to server on port", PORT)

        message = input("Enter a message to send to the server: ")
        client_socket.sendall(message.encode('utf-8'))

    finally:
        client_socket.close()

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

Step 2: Running the Client Code

1. Run the client (in a separate terminal).

```
bash
```

```
python client.py
```

OUTPUT:

~/lab/1

```
jejo@thinkpad:~/lab/1$ python3 server.py
Server listening on port 8080
Accepted connection from ('127.0.0.1', 60316)
Received message from client: hello
jejo@thinkpad:~/lab/1$
```

~/lab/1

```
jejo@thinkpad:~/lab/1$ python3 client.py
Connected to server on port 8080
Enter a message to send to the server: hello
jejo@thinkpad:~/lab/1$
```

2.Lab Manual: Developing a UDP Client-Server Application using Python Socket Programming

Lab Setup:

1. Setting up the Environment:

- Ensure that you have a UNIX-like operating system.
- Open a terminal.

2. Creating the Project Directory:

```
bash
```

- ```
2. mkdir udp_message_lab
 cd udp_message_lab
```

### Part 1: Server Side

#### Step 1: Writing the Server Code

1. Create a file named `udp_server.py` in the project directory.

```
python
```

- ```
1. # udp_server.py
   import socket

   PORT = 8080
   MAX_BUFFER_SIZE = 1024

   def main():
       sockfd = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
       server_addr = ('0.0.0.0', PORT)
       sockfd.bind(server_addr)

       print(f"UDP Server is listening on port {PORT}...")

       while True:
           data, client_addr = sockfd.recvfrom(MAX_BUFFER_SIZE)
           message = data.decode('utf-8')
           print(f"Message from client: {message}")

       sockfd.close()

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

Step 2: Running the Server Code

1. Run the server.

```
bash
```

- ```
1. python udp_server.py
```

## Part 2: Client Side

### Step 1: Writing the Client Code

1. Create a file named `udp_client.py` in the project directory.

`python`

```
1. # udp_client.py
import socket

PORT = 8080
SERVER_IP = '127.0.0.1'
MAX_BUFFER_SIZE = 1024

def main():
 sockfd = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
 server_addr = (SERVER_IP, PORT)

 while True:
 message = input("Enter a message to send to the server: ")
 sockfd.sendto(message.encode('utf-8'), server_addr)

 sockfd.close()

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

### Step 2: Running the Client Code

1. Run the client (in a separate terminal).

`bash`

```
1. python udp_client.py
```

In both the server and client code, the `socket` module is used for creating and managing sockets. The logic is similar to the C code provided, demonstrating the use of UDP for communication between the client and the server. The `recvfrom` method is used to receive data from the client, and `sendto` is used to send data to the server.

## OUTPUT :

| python3 server_udp.py                                                                                                                                                                                         | ~/lab/2                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>jejo@thinkpad:~/lab/2\$ python3 server_udp.py UDP Server is listening on port 8080... Message from client: hello Message from client: msg 2 Message from client: msg 3 Message from client: exit()</pre> | <pre>jejo@thinkpad:~/lab/2\$ python3 client_udp.py Enter a message to send to the server: hello Enter a message to send to the server: msg 2 Enter a message to send to the server: msg 3 Enter a message to send to the server: exit() Enter a message to send to the server: ^CTraceback (most recent ca ll last):</pre> |

# 3. Network Tools Demonstration Manual

## 1. Ping:

**Purpose:** To check the reachability of a host on an Internet Protocol (IP) network.

**Command:**

```
bash
```

```
ping [hostname or IP address]
```

**Example:**

```
bash
```

```
ping www.example.com
```

**Explanation:**

- This command sends ICMP Echo Request messages to the specified host and prints the round-trip time for each packet.
  - It's a basic tool for checking network connectivity.
- 

## 2. TCPDump:

**Purpose:** To capture and analyze network packets.

**Command:**

```
bash
```

```
sudo tcpdump [options]
```

**Example:**

```
bash
```

```
sudo tcpdump -i eth0
```

**Explanation:**

- This command captures packets on the specified network interface (-i option).
  - You can apply various filters and options to narrow down the captured data.
- 

## 3. Traceroute:

**Purpose:** To trace the route that packets take to reach a destination.

**Command:**

```
bash
```

```
traceroute [hostname or IP address]
```

**Example:**

```
bash
```

```
tracert www.example.com
```

**Explanation:**

- This command shows the path packets take from your computer to the specified destination.
  - It displays the IP addresses and round-trip times for each hop along the route.
- 

## 4. Netstat:

**Purpose:** To display network connections, routing tables, interface statistics, masquerade connections, etc.

**Command:**

```
bash
```

```
netstat [options]
```

**Example:**

```
bash
```

```
netstat -an
```

**Explanation:**

- This command shows various network-related information, such as active connections, listening ports, and routing tables.
- The `-a` option displays all connections and listening ports, and the `-n` option shows numeric addresses.

## OUTPUT:

```
jejo@thinkpad: ~
jejo@thinkpad:~$ ping www.google.com
PING www.google.com(maa03s36-in-x04.1e100.net (2404:6800:4007:814::2004)) 56 dat
a bytes
64 bytes from maa03s36-in-x04.1e100.net (2404:6800:4007:814::2004): icmp_seq=1 t
tl=58 time=82.7 ms
64 bytes from maa03s36-in-x04.1e100.net (2404:6800:4007:814::2004): icmp_seq=2 t
tl=58 time=80.3 ms
64 bytes from maa03s36-in-x04.1e100.net (2404:6800:4007:814::2004): icmp_seq=3 t
tl=58 time=80.6 ms
64 bytes from maa03s36-in-x04.1e100.net (2404:6800:4007:814::2004): icmp_seq=4 t
tl=58 time=329 ms
^C
--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 80.340/143.069/328.657/107.152 ms
jejo@thinkpad:~$ █
```



## TCPDump:

```
(no such device exists)
jejo@thinkpad:~$ sudo tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on wlp3s0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
18:13:30.612019 IP6 thinkpad.59770 > maa05s22-in-x0a.1e100.net.https: UDP, length 29
18:13:30.664437 IP thinkpad.56237 > _gateway.domain: 23265+ PTR? a.0.0.2.0.0.0.0.0.0.0.0.0.0.d.1.8.0.7.0.0.4.0.0.8.6.4.0.4.2.ip6.arpa. (90)
18:13:30.973439 IP6 maa05s22-in-x0a.1e100.net.https > thinkpad.59770: UDP, length 28
18:13:31.383084 IP _gateway.domain > thinkpad.56237: 23265 1/0/0 PTR maa05s22-in-x0a.1e100.net. (129)
18:13:31.383427 IP thinkpad.57122 > _gateway.domain: 21143+ PTR? e.9.a.7.0.4.0.4.c.2.e.c.5.7.b.0.e.3.5.9.2.9.e.1.d.8.0.4.9.0.4.2.ip6.arpa. (90)
18:13:31.390760 IP _gateway.domain > thinkpad.57122: 21143 NXDomain 0/0/0 (90)
18:13:31.391708 IP thinkpad.47916 > _gateway.domain: 51307+ PTR? 34.107.168.192.in-addr.arpa. (45)
18:13:31.587928 IP _gateway.domain > thinkpad.47916: 51307 NXDomain* 0/1/0 (104)
18:13:31.588400 IP thinkpad.33574 > _gateway.domain: 14110+ PTR? 125.107.168.192.in-addr.arpa. (46)
18:13:31.595613 IP _gateway.domain > thinkpad.33574: 14110 NXDomain 0/0/0 (46)
18:13:33.028414 IP thinkpad.34425 > _gateway.domain: 60838+ A? network-test.debian.org. (41)
18:13:33.028434 IP thinkpad.34425 > _gateway.domain: 4514+ AAAA? network-test.debian.org. (41)
18:13:33.169563 IP _gateway.domain > thinkpad.34425: 4514 2/0/0 CNAME debian.map.fastlydns.net., AAAA 2a04:4e42:25::644 (107)
18:13:34.873457 IP _gateway.domain > thinkpad.34425: 60838 2/0/0 CNAME debian.map.fastlydns.net., A 151.101.158.132 (95)
18:13:34.874106 IP thinkpad.35850 > 151.101.158.132.http: Flags [S], seq 546386806, win 64240, options [mss 1460,sackOK,TS val 2925865683 ecr 0,nop,wscale 7], length 0
18:13:34.927871 IP thinkpad.49096 > _gateway.domain: 525+ PTR? 132.158.101.151.in-addr.arpa. (46)
18:13:35.023378 IP 151.101.158.132.http > thinkpad.35850: Flags [S.], seq 539355995, ack 546386807, win 65535, options [mss 1370,sackOK,TS val 3306311557 ecr 2925865683,nop,wscale 9], length 0
18:13:35.023432 IP thinkpad.35850 > 151.101.158.132.http: Flags [.], ack 1, win 502, options [nop,nop,TS val 2925865833 ecr 3306311557], length 0
18:13:35.023610 IP thinkpad.35850 > 151.101.158.132.http: Flags [P.], seq 1:84, ack 1, win 502, options [nop,nop,TS val 2925865833 ecr 3306311557], length 83: HTTP: GET /nm HTTP/1.1
18:13:35.176207 IP 151.101.158.132.http > thinkpad.35850: Flags [.], ack 84, win 283, options [nop,nop,TS val 3306311726 ecr 2925865833], length 0
18:13:35.176228 IP 151.101.158.132.http > thinkpad.35850: Flags [P.], seq 1:338, ack 84, win 283, options [nop,nop,TS val 3306311726 ecr 2925865833], length 337: HTTP: HTTP/1.1 200 OK
18:13:35.176245 IP thinkpad.35850 > 151.101.158.132.http: Flags [.], ack 338, win 501, options [nop,nop,TS val 2925865986 ecr 3306311726], length 0
18:13:35.176253 IP 151.101.158.132.http > thinkpad.35850: Flags [F.], seq 338, ack 84, win 283, options [nop,nop,TS val 3306311726 ecr 2925865833], length 0
18:13:35.176395 IP thinkpad.35850 > 151.101.158.132.http: Flags [F.], seq 84, ack 339, win 501, options [nop,nop,TS val 2925865986 ecr 3306311726], length 0
18:13:35.195922 IP _gateway.domain > thinkpad.49096: 525 NXDomain 0/1/0 (106)
18:13:35.327386 IP 151.101.158.132.http > thinkpad.35850: Flags [.], ack 85, win 283, options [nop,nop,TS val 3306311836 ecr 2925865986], length 0
```

## Traceroute:

```
jejo@thinkpad:~$ traceroute www.google.com
traceroute to www.google.com (142.250.76.36), 30 hops max, 60 byte packets
 1 _gateway (192.168.107.34) 3.283 ms 3.254 ms 7.282 ms
 2 * * *
 3 255.0.0.1 (255.0.0.1) 707.437 ms 722.338 ms 912.888 ms
 4 255.0.0.2 (255.0.0.2) 912.875 ms 912.860 ms 912.847 ms
 5 172.17.180.3 (172.17.180.3) 912.833 ms 172.17.180.2 (172.17.180.2) 1116.953 ms *
 6 * * *
 7 * * *
 8 * * *
 9 * * *
10 * 74.125.51.4 (74.125.51.4) 614.495 ms *
11 * * 72.14.217.252 (72.14.217.252) 819.174 ms
12 209.85.175.48 (209.85.175.48) 956.765 ms 72.14.217.252 (72.14.217.252) 1382.088 ms 209.8
5.175.48 (209.85.175.48) 1551.927 ms
13 142.250.235.107 (142.250.235.107) 1551.904 ms * *
14 * * *
15 maa03s36-in-f4.1e100.net (142.250.76.36) 1707.190 ms * 142.250.235.105 (142.250.235.105)
 1619.650 ms
jejo@thinkpad:~$
```

## NETSTAT:

```
jejo@thinkpad:~$ netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 127.0.0.1:631 0.0.0.0:* LISTEN
tcp 0 0 192.168.107.125:58122 34.107.243.93:443 ESTABLISHED
tcp6 0 0 :::1:631 :::* LISTEN
udp 0 0 0.0.0.0:5353 0.0.0.0:*
udp 0 0 0.0.0.0:52503 0.0.0.0:*
udp 0 0 192.168.107.125:68 192.168.107.34:67 ESTABLISHED
udp 0 0 0.0.0.0:631 0.0.0.0:*
udp6 0 0 :::54442 :::*
udp6 0 0 :::5353 :::*
udp6 0 0 :::59770 :::*
raw6 0 0 :::58 :::* 7

Active UNIX domain sockets (servers and established)
Proto RefCnt Flags Type State I-Node Path
unix 3 [] STREAM CONNECTED 42609 /run/dbus/system_bus_socket
unix 3 [] STREAM CONNECTED 33224
unix 3 [] STREAM CONNECTED 18733
unix 3 [] STREAM CONNECTED 29168
unix 3 [] STREAM CONNECTED 29154
unix 3 [] STREAM CONNECTED 31815 /run/user/1000/at-spi/bus
unix 3 [] STREAM CONNECTED 19066 /run/systemd/journal/stdout
unix 3 [] STREAM CONNECTED 18933 /run/systemd/journal/stdout
unix 3 [] STREAM CONNECTED 24120
unix 3 [] STREAM CONNECTED 30330
unix 2 [ACC] STREAM LISTENING 33511 /run/user/1000/app/io.github.mimbre
ro.WhatsAppDesktop/scoped_dir5rxeQS/SingletonSocket
unix 3 [] STREAM CONNECTED 17262 /run/systemd/journal/stdout
unix 3 [] STREAM CONNECTED 31832 /run/user/1000/bus
unix 3 [] STREAM CONNECTED 29151 /run/systemd/journal/stdout
unix 3 [] STREAM CONNECTED 29088 @/home/jejo/.cache/ibus/dbus-DjYIhn
EU
unix 2 [] DGRAM CONNECTED 20272
unix 2 [] DGRAM CONNECTED 16700
unix 2 [] STREAM CONNECTED 48151
unix 3 [] STREAM CONNECTED 31115 /run/user/1000/bus
unix 3 [] STREAM CONNECTED 30217 /run/systemd/journal/stdout
unix 3 [] STREAM CONNECTED 33716
unix 3 [] STREAM CONNECTED 36495 /run/dbus/system_bus_socket
unix 3 [] STREAM CONNECTED 26301
unix 3 [] STREAM CONNECTED 25710
unix 2 [ACC] STREAM LISTENING 15762 /run/acpid.socket
unix 3 [] STREAM CONNECTED 31779
unix 3 [] STREAM CONNECTED 17219 /run/systemd/journal/stdout
unix 3 [] STREAM CONNECTED 30090 /run/user/1000/bus
unix 3 [] STREAM CONNECTED 33233
unix 3 [] STREAM CONNECTED 19136
unix 3 [] STREAM CONNECTED 16321
unix 3 [] STREAM CONNECTED 26308 /run/user/1000/bus
unix 2 [ACC] STREAM LISTENING 15764 /run/avahi-daemon/socket
```

