Assignment 9

Implementation of TCP/UDP Socket Programming

NAME: Shirish Manoj Bobde

Reg. No.: 812

Roll No.: ECE/21152

Problem Statement 1

Write a TCP/UDP socket program (in C/C++/Java/Python) to establish a connection between client and server. The server should act as a network device maintaining an ARP table. Implement ARP request and reply functionality.
Display appropriate messages indicating the ARP request and response. Test your program with multiple clients requesting ARP resolution for different IP addresses.

Codes:

Server:

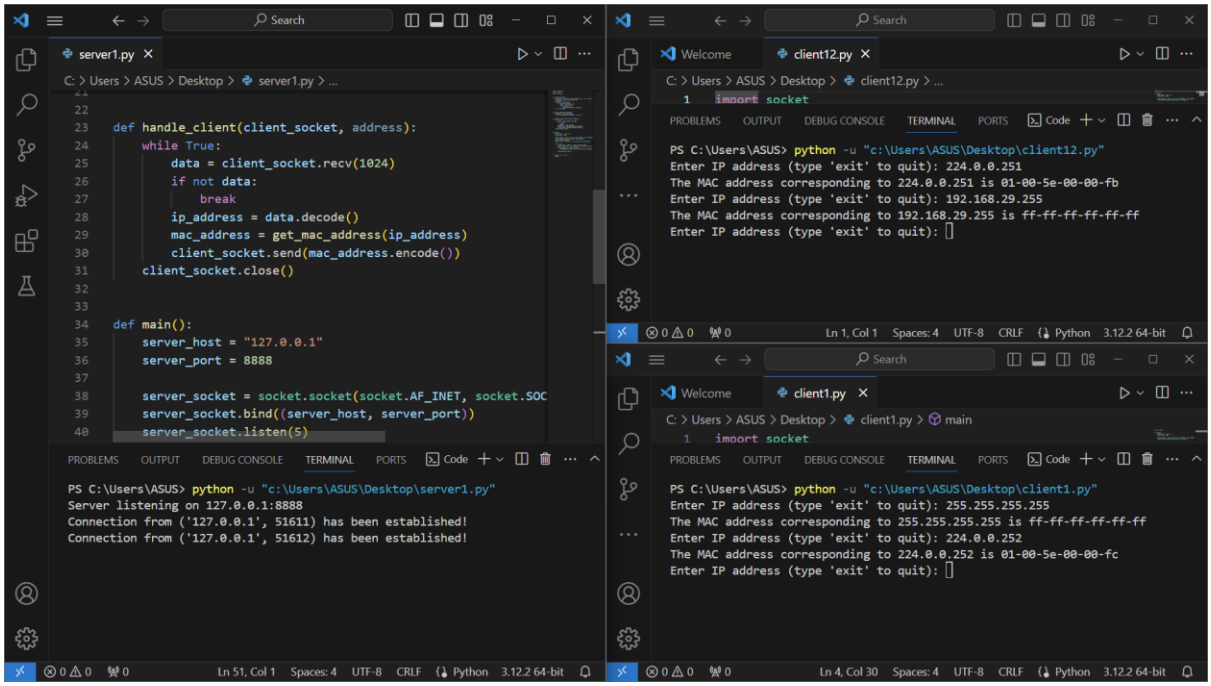
```python
import socket

SERVER_IP = '127.0.0.1'
SERVER_PORT = 12345

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

server_socket.bind((SERVER_IP, SERVER_PORT))

server_socket.listen(5)
print(f"Server listening on {SERVER_IP}:{SERVER_PORT}")

arp_table = {}

def handle_arp_request(ip_address):
    if ip_address in arp_table:
        return arp_table[ip_address]
    else:
        return "IP address not found in ARP table"


def handle_client_connection(client_socket):
    while True:
        data = client_socket.recv(1024).decode()
        if not data:
            break
```

```
        if data.startswith("ARP_REQUEST"):
            ip_address = data.split()[1]
            mac_address = handle_arp_request(ip_address)
            response = f"ARP_RESPONSE {mac_address}"
            client_socket.send(response.encode())

    client_socket.close()

while True:

    client_socket, client_address = server_socket.accept()
    print(f"Connection from {client_address[0]}:{client_address[1]}")

    client_handler = threading.Thread(target=handle_client_connection,
args=(client_socket,))
    client_handler.start()
```

Client:

```
import socket

def main():
    server_host = "127.0.0.1"
    server_port = 8888

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_host, server_port))

    while True:
        ip_address = input("Enter IP address (type 'exit' to quit): ")
        if ip_address.lower() == 'exit':
            break
        client_socket.send(ip_address.encode())
        mac_address = client_socket.recv(1024).decode()
        print(f"The MAC address corresponding to {ip_address} is
{mac_address}")

    client_socket.close()

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

Output:



Problem Statement 2

Write a TCP/UDP socket program (in C/C++/Java/Python) to establish a connection between client and server. The server should act as a network device maintaining a RARP table mapping MAC addresses to IP addresses. Implement RARP request and reply functionality. Display appropriate messages indicating the RARP request and response. Test your program with multiple clients requesting RARP resolution for different MAC addresses.

Code:

Server

```
import subprocess
import socket
import threading


def fetch_arp_table():
    arp_output = subprocess.check_output(["arp", "-a"], text=True)
    arp_entries = arp_output.split("\n")
    arp_table = {}
    for entry in arp_entries:
        if len(entry.strip()) > 0:
            parts = entry.split()
            if len(parts) == 3:
                arp_table[parts[1]] = parts[0]
```

```
    return arp_table


def get_ip_address(mac_address):
    arp_table = fetch_arp_table()
    return arp_table.get(mac_address, "Unknown")


def handle_client(client_socket, address):
    while True:
        data = client_socket.recv(1024)
        if not data:
            break
        mac_address = data.decode()
        ip_address = get_ip_address(mac_address)
        client_socket.send(ip_address.encode())
    client_socket.close()

def main():
    server_host = "127.0.0.1"
    server_port = 8889

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((server_host, server_port))
    server_socket.listen(5)
    print(f"Server listening on {server_host}:{server_port}")

    while True:
        client_socket, address = server_socket.accept()
        print(f"Connection from {address} has been established!")
        client_handler = threading.Thread(
            target=handle_client, args=(client_socket, address)
        )
        client_handler.start()


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

Client

```
import socket


def main():
    server_host = "127.0.0.1"
```

```
    server_port = 8889

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_host, server_port))

    while True:
        mac_address = input("Enter MAC address (type 'exit' to quit): ")
        if mac_address.lower() == "exit":
            break
        client_socket.send(mac_address.encode())
        ip_address = client_socket.recv(1024).decode()
        print(f"The IP address corresponding to {mac_address} is
{ip_address}")

    client_socket.close()


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

Output: