

Computer Networks

Assignment – 9

Name: Anik Barury

Roll: CSE22017

Reg: 871

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.

Server.py

```
import socket
import pickle
```

```
# Simulated ARP Table
```

```
arp_table = {
    "192.168.1.1": "00:1A:2B:3C:4D:5E",
    "192.168.1.2": "00:1A:2B:3C:4D:5F"
}
```

```
def arp_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(("127.0.0.1", 9999))
    server_socket.listen(5)
    print("ARP Server is running and waiting for connections...")
```

```
while True:
    conn, addr = server_socket.accept()
    print("Connection from", addr)
```

```
# Receive ARP Request
```

```
data = conn.recv(1024)
```

```
if not data:
    conn.close()
    continue
```

```
try:
    request = pickle.loads(data)
    ip_address = request.get("ip")
```

```

        # Send ARP Reply
        mac_address = arp_table.get(ip_address, "Not Found")
        response = {"ip": ip_address, "mac": mac_address}
        conn.send(pickle.dumps(response))

        print(f"ARP Request for {ip_address}, Reply: {mac_address}")

    except pickle.UnpicklingError:
        print("Error: Invalid data received")

    conn.close()

if __name__ == "__main__":
    arp_server()

```

Client.py:

```

import socket
import pickle

def arp_client(ip_address):
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(("127.0.0.1", 9999))

    # Send ARP Request
    request = {"ip": ip_address}
    client_socket.send(pickle.dumps(request))

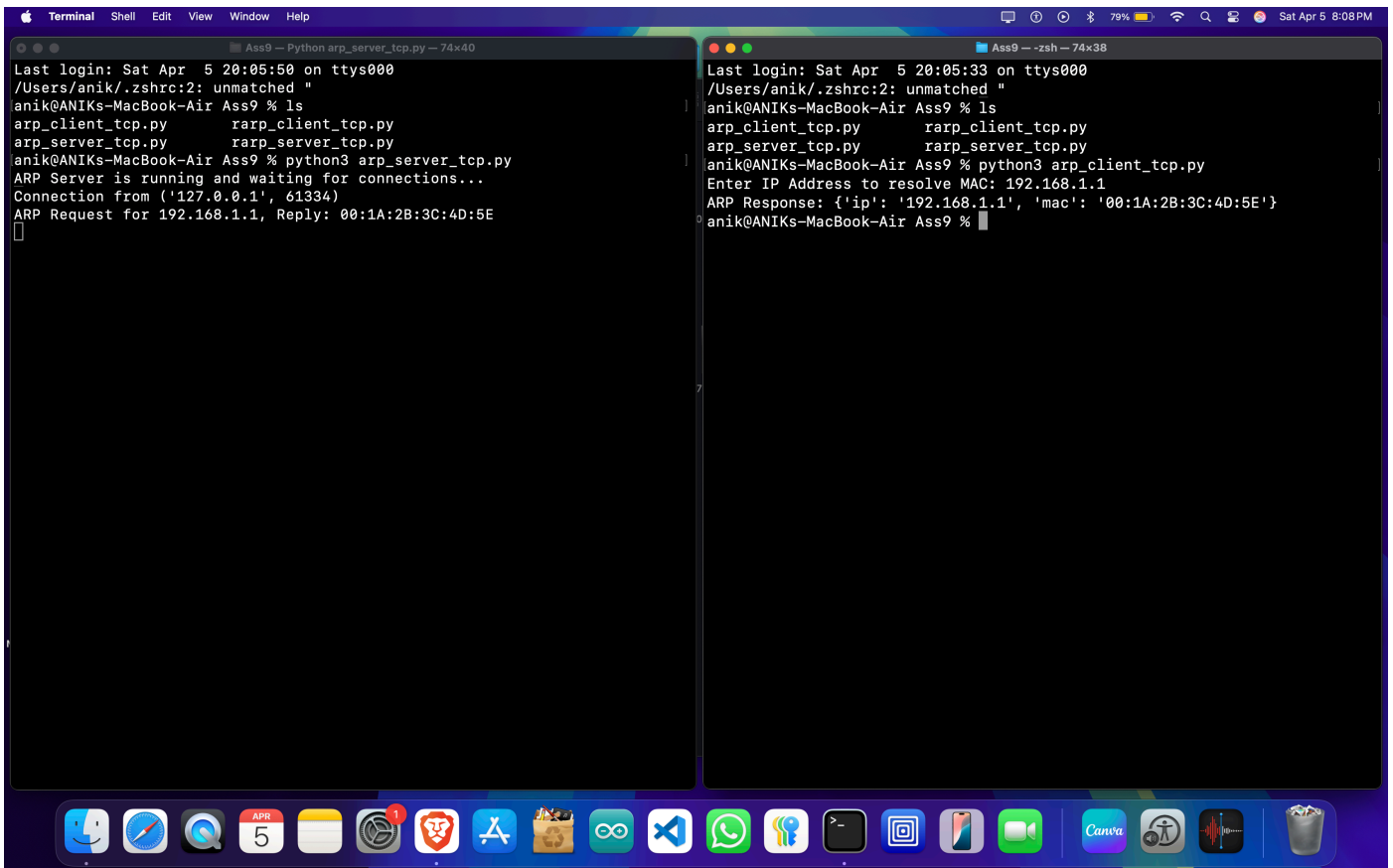
    # Receive ARP Reply
    response = pickle.loads(client_socket.recv(1024))
    print("ARP Response:", response)

    client_socket.close()

if __name__ == "__main__":
    ip = input("Enter IP Address to resolve MAC: ") # Use input() instead of raw_input()
    arp_client(ip)

```

OUTPUT:



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

Server.py

```
import socket
```

```
# RARP Table (MAC → IP)
```

```
rarp_table = {  
    "AA:BB:CC:DD:EE:01": "192.168.1.101",  
    "AA:BB:CC:DD:EE:02": "192.168.1.102",  
    "AA:BB:CC:DD:EE:03": "192.168.1.103"  
}
```

```
# Create UDP socket
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```

s.bind(("0.0.0.0", 12345)) # Bind to all available interfaces on port 12345

print("[SERVER] RARP Server is running and waiting for requests...")

while True:
    try:
        # Receive RARP request
        data, addr = s.recvfrom(1024)
        mac = data.decode().strip() # Decode MAC address request
        print(f"[RARP REQUEST] Received request for MAC: {mac} from {addr}")

        # Look up MAC in the RARP table
        ip = rarp_table.get(mac, "MAC address not found")

        # Send RARP reply
        s.sendto(ip.encode(), addr)
        print(f"[RARP REPLY] Sent IP: {ip} for MAC: {mac}")

    except Exception as e:
        print(f"[ERROR] {e}")

```

Client.py

```

import socket

def rarp_client():
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_address = ("127.0.0.1", 12345) # Use localhost (same as server)

    while True:
        try:
            mac = input("Enter MAC address (or 'exit' to quit): ").strip()
            if mac.lower() == "exit":
                print("Exiting RARP Client.")
                break

            # Send RARP request
            s.sendto(mac.encode(), server_address)

            # Receive RARP reply
            data, _ = s.recvfrom(1024)
            print("Received RARP reply:", data.decode())

        except ConnectionError:

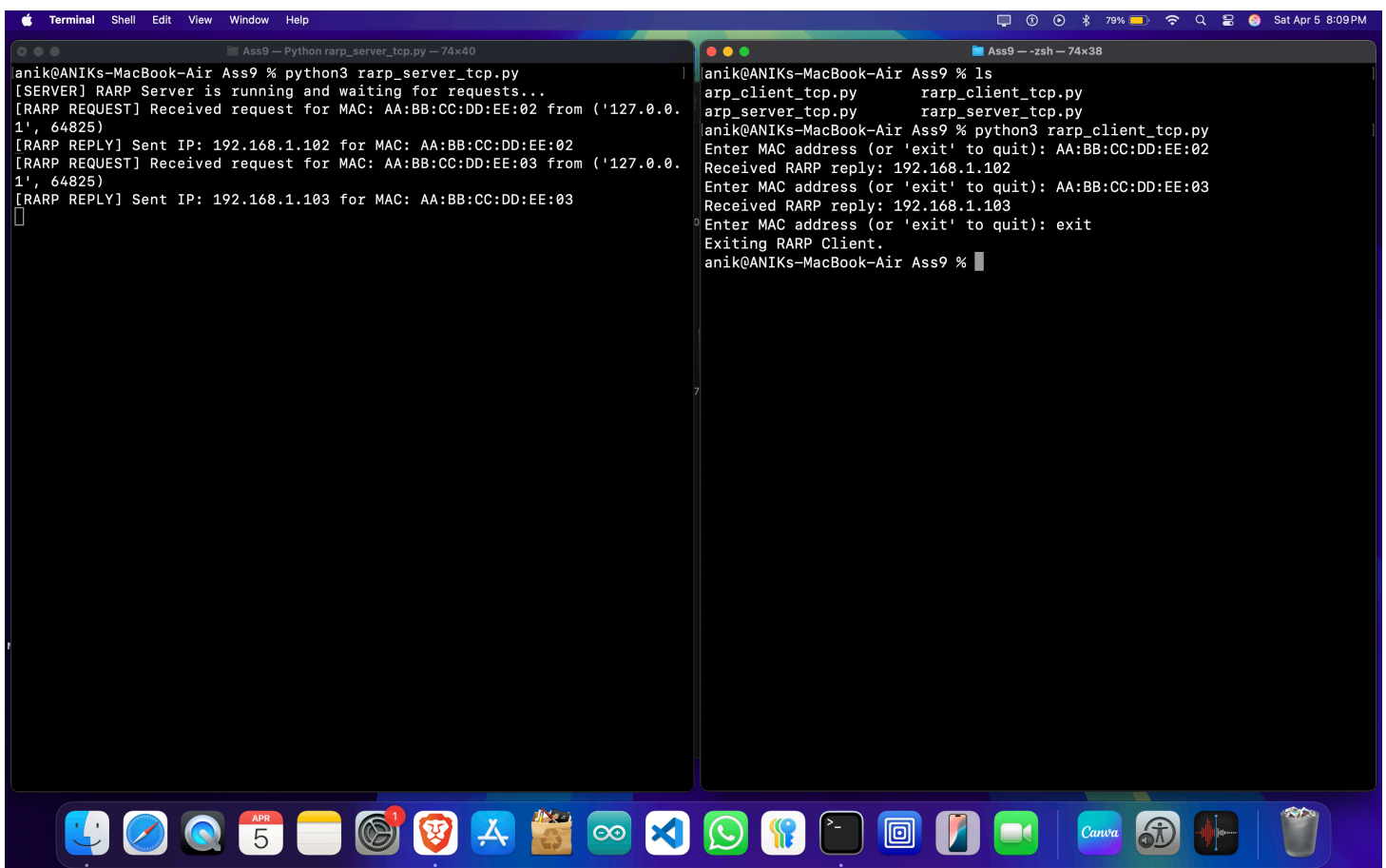
```

```
        print("[ERROR] Server is unavailable. Please check if it is running.")
        break
    except Exception as e:
        print(f"[ERROR] {e}")
        break
```

```
s.close() # Close socket when done
```

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

Output



The image shows two terminal windows on a macOS system. The left window, titled 'Ass9 - Python rarp_server_tcp.py - 74x40', shows the execution of the RARP server. It starts with the command 'python3 rarp_server_tcp.py', followed by a message '[SERVER] RARP Server is running and waiting for requests...'. It then receives two requests from '127.0.0.1' for MAC addresses 'AA:BB:CC:DD:EE:02' and 'AA:BB:CC:DD:EE:03', and responds with IP addresses '192.168.1.102' and '192.168.1.103' respectively. The right window, titled 'Ass9 - zsh - 74x38', shows the execution of the RARP client. It starts with the command 'ls', followed by 'python3 rarp_client_tcp.py'. It prompts for a MAC address, enters 'AA:BB:CC:DD:EE:02', receives the IP '192.168.1.102', then enters 'AA:BB:CC:DD:EE:03' and receives '192.168.1.103'. Finally, it enters 'exit' and displays 'Exiting RARP Client.'.

```
anik@ANIKs-MacBook-Air Ass9 % python3 rarp_server_tcp.py
[SERVER] RARP Server is running and waiting for requests...
[RARP REQUEST] Received request for MAC: AA:BB:CC:DD:EE:02 from ('127.0.0.1', 64825)
[RARP REPLY] Sent IP: 192.168.1.102 for MAC: AA:BB:CC:DD:EE:02
[RARP REQUEST] Received request for MAC: AA:BB:CC:DD:EE:03 from ('127.0.0.1', 64825)
[RARP REPLY] Sent IP: 192.168.1.103 for MAC: AA:BB:CC:DD:EE:03
^

anik@ANIKs-MacBook-Air Ass9 % ls
arp_client_tcp.py      rarp_client_tcp.py
arp_server_tcp.py      rarp_server_tcp.py
anik@ANIKs-MacBook-Air Ass9 % python3 rarp_client_tcp.py
Enter MAC address (or 'exit' to quit): AA:BB:CC:DD:EE:02
Received RARP reply: 192.168.1.102
Enter MAC address (or 'exit' to quit): AA:BB:CC:DD:EE:03
Received RARP reply: 192.168.1.103
Enter MAC address (or 'exit' to quit): exit
Exiting RARP Client.
anik@ANIKs-MacBook-Air Ass9 %
```