

EXPERIMENT – 12

AIM: - a) Implement echo client server using TCP/UDP sockets.

CODE: -

```
import socket

def start_server(host='127.0.0.1', port=12345):
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        s.bind((host, port))
        print(f"UDP Server running on {host}:{port}")

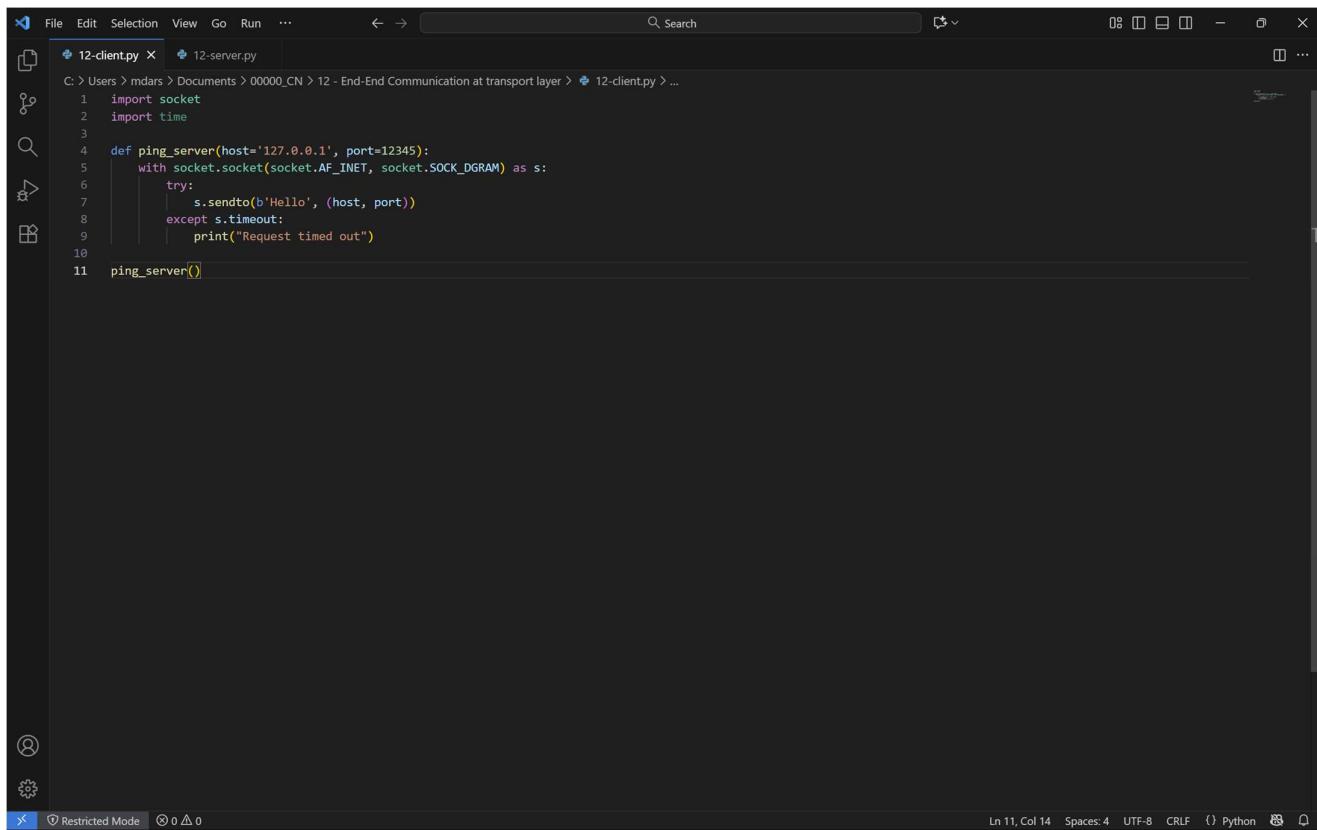
    while True:
        data, addr = s.recvfrom(1024)
        print(f"Received message from {addr}: {data.decode()}")

import socket
import time

def ping_server(host='127.0.0.1', port=12345):
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        try:
            s.sendto(b'Hello', (host, port))
        except socket.timeout:
            print("Request timed out")

ping_server()
```

OUTPUT: -



The screenshot shows a code editor window with a dark theme. At the top, there's a menu bar with File, Edit, Selection, View, Go, Run, and other options. Below the menu is a search bar labeled 'Search'. The main area contains two tabs: '12-client.py' and '12-server.py'. The '12-client.py' tab is active and displays the following Python code:

```
C:\> Users > mdars > Documents > 00000_CN > 12 - End-End Communication at transport layer > 12-client.py > ...
1 import socket
2 import time
3
4 def ping_server(host='127.0.0.1', port=12345):
5     with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
6         try:
7             s.sendto(b'Hello', (host, port))
8         except s.timeout:
9             print("Request timed out")
10
11 ping_server()
```

The code defines a function 'ping_server' that sends a UDP message to a specified host and port. It includes error handling for timeouts. The code editor also shows status information at the bottom: 'Ln 11, Col 14' and file statistics: 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', and icons for file operations like save and close.

RESULT: -

Implement echo client server using TCP/UDP sockets.

AIM: - b) Implement chat client server using TCP/UDP sockets.

CODE: -

```
import socket

def start_server(host='127.0.0.1', port=12345):
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        s.bind((host, port))
        print(f"UDP Server running on {host}:{port}")

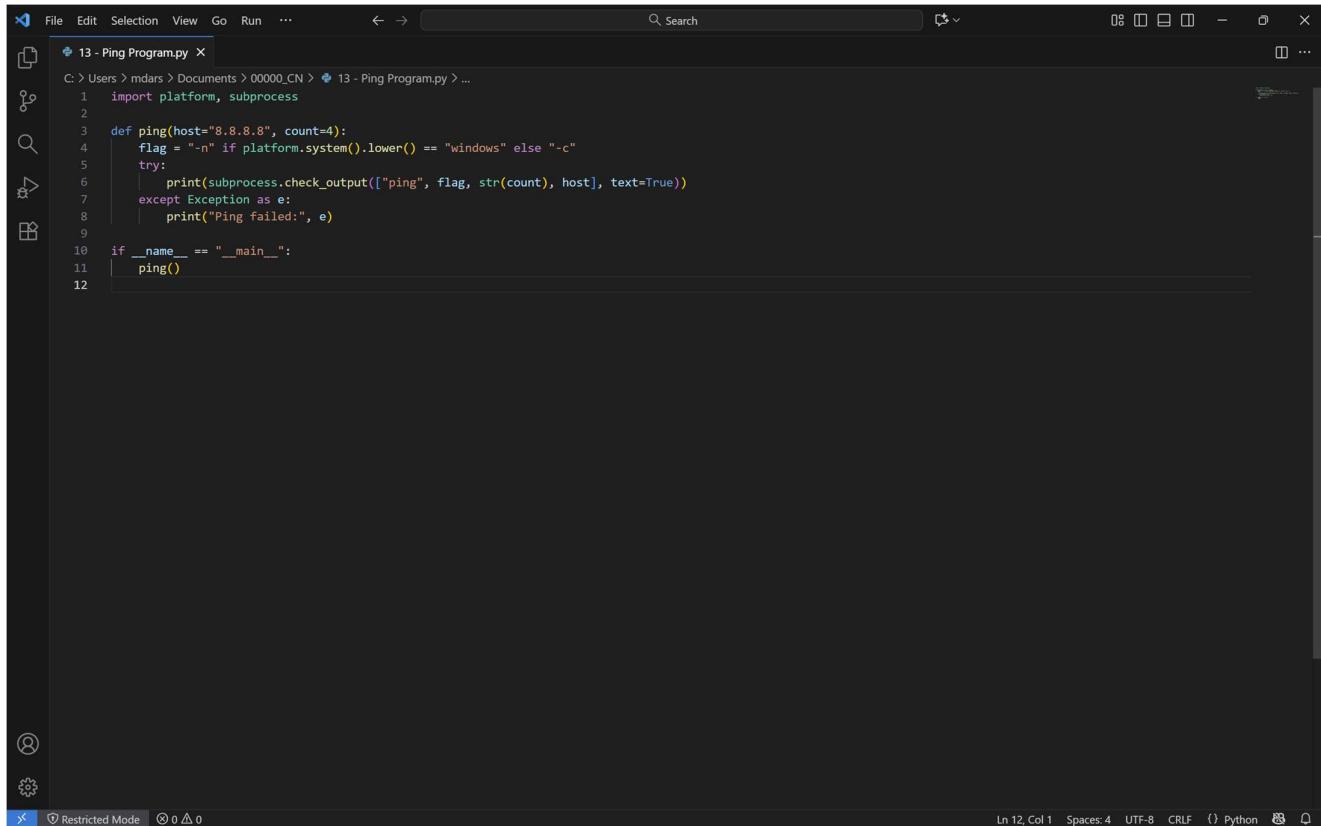
    while True:
        data, addr = s.recvfrom(1024)
        print(f"Received message from {addr}: {data.decode()}")

import socket
import time

def ping_server(host='127.0.0.1', port=12345):
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        try:
            s.sendto(b'Hello', (host, port))
        except socket.timeout:
            print("Request timed out")

ping_server()
```

OUTPUT: -



```
13 - Ping Program.py
C:\Users\mdars\Documents\00000.CN> 13 - Ping Program.py > ...
1  import platform, subprocess
2
3  def ping(host="8.8.8.8", count=4):
4      flag = "-n" if platform.system().lower() == "windows" else "-c"
5      try:
6          print(subprocess.check_output(["ping", flag, str(count), host], text=True))
7      except Exception as e:
8          print("Ping failed:", e)
9
10 if __name__ == "__main__":
11     ping()
12
```

The screenshot shows a code editor window with a dark theme. The title bar says "13 - Ping Program.py". The file path is "C:\Users\mdars\Documents\00000.CN> 13 - Ping Program.py > ...". The code editor displays the following Python script:

```
13 - Ping Program.py
C:\Users\mdars\Documents\00000.CN> 13 - Ping Program.py > ...
1  import platform, subprocess
2
3  def ping(host="8.8.8.8", count=4):
4      flag = "-n" if platform.system().lower() == "windows" else "-c"
5      try:
6          print(subprocess.check_output(["ping", flag, str(count), host], text=True))
7      except Exception as e:
8          print("Ping failed:", e)
9
10 if __name__ == "__main__":
11     ping()
```

The status bar at the bottom shows "Ln 12, Col 1" and "Spaces: 4" and "UTF-8" and "CRLF". There are also icons for Restricted Mode, Help, and other settings.

RESULT: -

Implement chat client server using TCP/UDP sockets.

