V.SWARNAMALA **241901116**

EXP 3: Develop a customized ping command to test the server connectivity Date:25/8/25

Aim

To develop a customized ping program in Python that tests the connectivity of a server by connecting to a specified host and port, measuring the Round-Trip Time (RTT) for the connection, and reporting the minimum, maximum, and average RTT over multiple attempts.

Introduction

The ping command is a network utility used to test the reachability of a host on an IP network and measure the round-trip time for messages sent from the source to the destination computer. Traditional ping uses ICMP packets, but this customized version uses TCP socket connections to a specific port (e.g., HTTP port 80) to test server availability and responsiveness. This approach is useful in environments where ICMP packets may be blocked but TCP connections are allowed. Measuring RTT helps diagnose network latency issues and server responsiveness as part of network troubleshooting.

ALGORITHM

1. **Setup:**
    - Import the required modules: socket for networking and time for measuring response time.
    - Define the target host (e.g., "google.com"), port (e.g., 80 for HTTP), and the number of ping attempts.
2. **Ping Logic:**
    - Loop for the defined number of attempts.
    - For each attempt, create a new socket object.
    - Record the start time using time.time().
    - Connect the socket to the target host and port.
    - Record the end time after the connection is established.
    - Close the socket.
    - Calculate the RTT in milliseconds as the difference between the end and start times.
    - Append the RTT to a list for later analysis.
    - Handle exceptions for connection failures, printing "Request timed out".
3. **Result Reporting:**
    - After all attempts, calculate and print the minimum, maximum, and average RTT from the recorded times.
    - Display each ping response time immediately after it is measured.

This experiment demonstrates basic network performance testing through Python socket programming without relying on ICMP, making it practical for restricted network environments.

Programs:

PRO1:

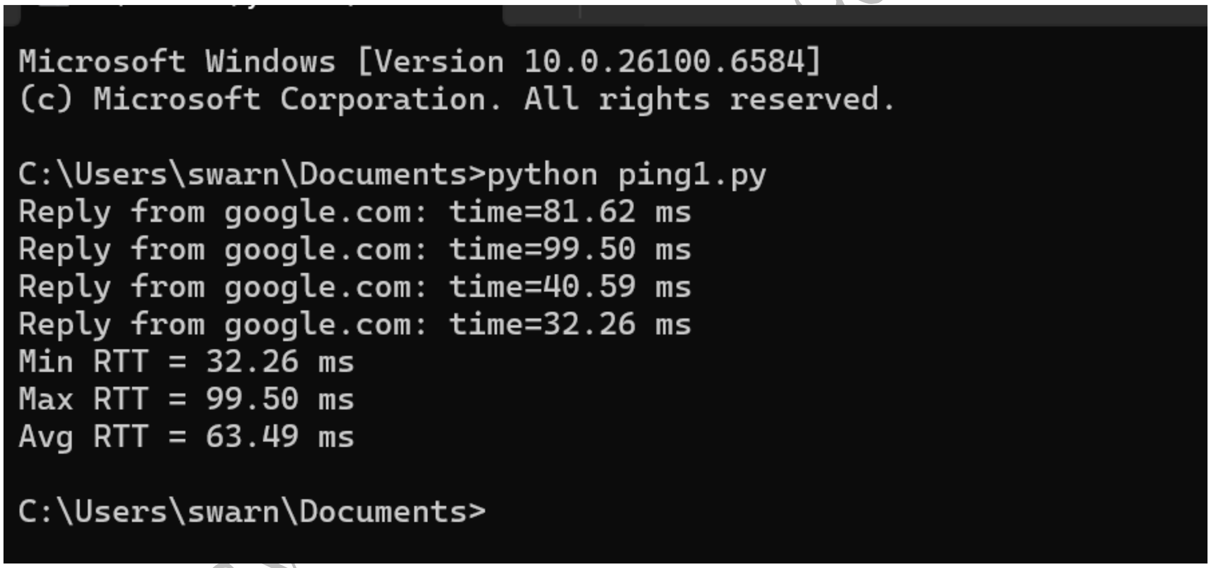
```
import socket

import time

host = "google.com"

port = 80

count = 4

for i in range(count):

    try:

        s = socket.socket()

        start = time.time()

        s.connect((host, port))

        end = time.time()

        s.close()

        print(f"Reply from {host}: time={(end-start)*1000:.2f} ms")

    except Exception:

        print("Request timed out")
```

Customized Ping Program to Measure Min, Max, and Average RTT

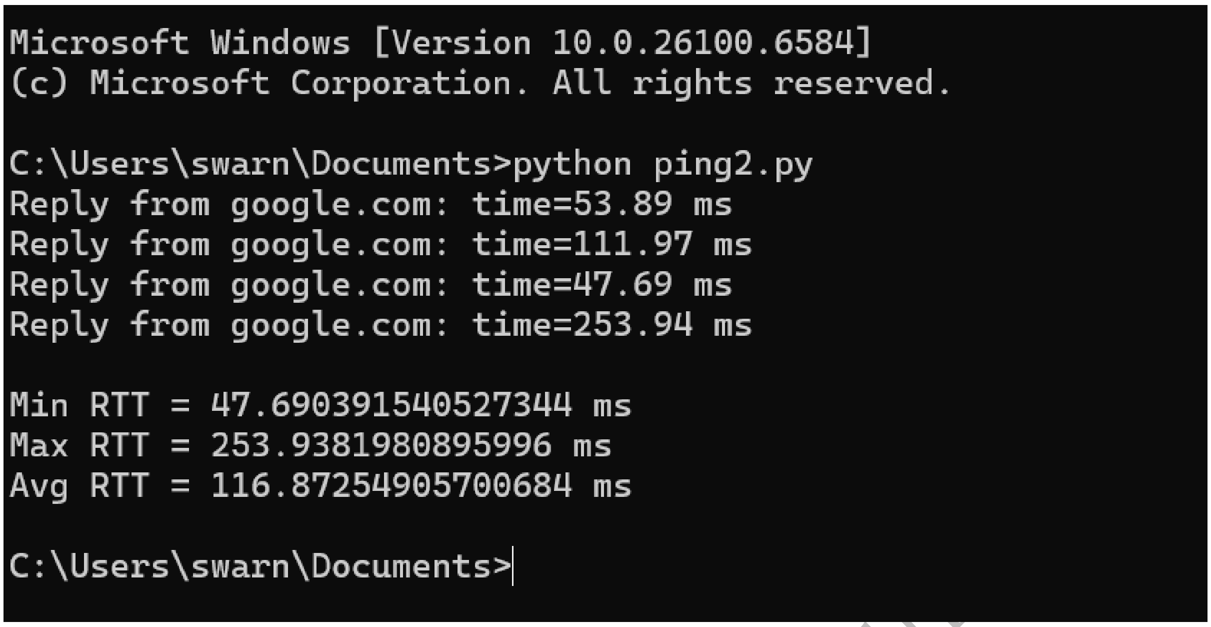
PRO2:

```
import socket, time

host = "google.com"

port = 80

count = 4

times = []

for i in range(count):

    try:

        s = socket.socket()

        start = time.time()

        s.connect((host, port))
        end = time.time()

        s.close()

        rtt = (end - start) * 1000

        times.append(rtt)

        print(f"Reply from {host}: time={rtt:.2f} ms")

    except:

        print("Request timed out")

if times:

    print("\nMin RTT =", min(times), "ms")

    print("Max RTT =", max(times), "ms")

    print("Avg RTT =", sum(times)/len(times), "ms")
```

OUTPUT:



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

C:\Users\swarn\Documents>python ping1.py
Reply from google.com: time=81.62 ms
Reply from google.com: time=99.50 ms
Reply from google.com: time=40.59 ms
Reply from google.com: time=32.26 ms
Min RTT = 32.26 ms
Max RTT = 99.50 ms
Avg RTT = 63.49 ms

C:\Users\swarn\Documents>
```



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

C:\Users\swarn\Documents>python ping2.py
Reply from google.com: time=53.89 ms
Reply from google.com: time=111.97 ms
Reply from google.com: time=47.69 ms
Reply from google.com: time=253.94 ms

Min RTT = 47.690391540527344 ms
Max RTT = 253.9381980895996 ms
Avg RTT = 116.87254905700684 ms

C:\Users\swarn\Documents>
```

Result:

Thus the we successfully implemented ping command.