EXERCISE 3

Simulation of PING and TRACEROUTE commands using Twisted Python

Deepitha P
3122225002028

Objectives:

- Understanding Network Diagnostic Tools
- Hands-on Experience with Twisted Python
- Developing Simulation Skills
- Analyzing Network Connectivity
- Problem-solving and Troubleshooting

Possible Test Cases :

Accuracy of PING Responses:

```
from twisted.internet import reactor, protocol

class PingPort(protocol.Protocol):
    def connectionMade(self):
        print("A connection has been made")

    def connectionLost(self, reason):
        print("Connection has been lost.")

    def dataReceived(self, data: bytes) -> None:
        print(f"Received Data: {data.decode()}")
        self.transport.write(b"Data received and acknowledged by server...")

class PingPortFactory(protocol.Factory):
    def buildProtocol(self, addr):
        return PingPort()

reactor.listenTCP(8000, PingPortFactory())
print("Server is Running...")
reactor.run()
```

Handling of Unreachable Hosts

```
from twisted.internet import reactor, protocol
import sys

class PingProtocol(protocol.Protocol):
    def__init_(self, verbose=False, v4=True, v6=True):
        self.verbose = verbose
        self.v4 = v4
        self.v6 = v6
```

```
    def connectionMade(self):
        if self.v4 and self.v6:
            self.transport.write(b'PING using both!')
        elif self.v4:
            self.transport.write(b'PING using IPv4!')
        elif self.v6:
            self.transport.write(b'PING using IPv6!')

    def dataReceived(self, data):
        if self.verbose:
            print("Received:", data.decode())
        else:
            print("Host is reachable.")


class PingFactory(protocol.ClientFactory):
    def__init_(self, verbose=False, v4=True, v6=True):
        self.verbose = verbose
        self.v4 = v4
        self.v6 = v6

    def buildProtocol(self, addr):
        return PingProtocol(self.verbose, self.v4, self.v6)

    def clientConnectionFailed(self, connector, reason):
        print("Connection failed.")

    def clientConnectionLost(self, connector, reason):
        print("Connection lost.")

  def main():
    host = 'localhost'
    port = 8000
    verbose = False

    # Check for command-line arguments
    v4 = v6 = True
    if "-v" in sys.argv or "--verbose" in sys.argv:
        verbose = True
    if "-4" in sys.argv:
        v4 = True
        v6 = False
    if "-6" in sys.argv:
        v4 = False
        v6 = True

    factory = PingFactory(verbose, v4, v6)
    reactor.connectTCP(host, port, factory)
    reactor.run()

  if__name__== '_main_':
    try:
        main()
    except Exception as e:
        print("Connection Terminated")
```

TRACEROUTE Path Discovery

```
from twisted.internet.protocol import DatagramProtocol
from twisted.internet import reactor

class UDPServer(DatagramProtocol):
    # Lets say these ips are connected to the UDP server Now!

    connected_ips = [
        "10.0.1.1",
        "192.168.2.3",
        "271.8.9.2",
        "8.8.8.8",  # Google's Public DNS
        "4.2.2.2",  # Level 3 Public DNS
        "151.101.193.69", # IP of www.example.com
        "185.199.108.153", # IP of GitHub
        "13.107.21.200", # IP of Microsoft.com
        "104.16.249.5"  # IP of OpenAI.com
    ]

    def datagramReceived(self, datagram, address):
        print(f"Received datagram from {address}: {datagram.decode()}")
        if datagram.decode() in UDPServer.connected_ips:
            print("Hello")
            self.transport.write(b"IP Present", address)

def main():
    reactor.listenUDP(8000, UDPServer())
    reactor.run()

if__name__== "_main_":
    main()
```

TTL Incrementation in TRACEROUTE

```
from twisted.internet import defer, reactor
from twisted.internet.protocol import DatagramProtocol

class TracerouteProtocol(DatagramProtocol):
    def__init_(self, destination, max_hops=30):
        self.destination = destination

    def startProtocol(self):
        self.transport.write(b"Hello Server", ("127.0.0.1", 8000))
        # Say we send the ip address of the required machine to check!
        self.transport.write(bytes(f"{self.destination}", 'utf-8'),  ("127.0.0.1", 8000))

    def datagramReceived(self, data, addr):
        print(f"Received from {addr}")
        if data.decode() == "IP Present":
            print("Reached destination!")
            print(f"Destination is in address : {addr}")
            reactor.stop()
        else:
            print("Unable to find the host!")
```

```
            reactor.stop()

def run_traceroute(destination):
    protocol = TracerouteProtocol(destination)
    reactor.listenUDP(0, protocol)
    reactor.run()

if__name__== "_main_":
    import sys
    if len(sys.argv) != 2:
        print("Usage: python traceroute.py <destination>")
        sys.exit(1)
    destination = sys.argv[1]
    run_traceroute(destination)
```