Lab 04: Assignment: Ping Utility Analysis

Objective:

In this assignment, you will analyze the ping utility, a fundamental network diagnostic tool. You will explore its functionality, usage, and output, and demonstrate your understanding through a series of tasks.

Tasks:

1. Ping Basics

- Explain the purpose of the ping utility and its basic syntax.

Ans : A ping (Packet Internet or Inter-Network Groper) is a basic Internet program that allows a user to test and verify if a particular destination IP address exists and can accept requests in computer network administration. It is also used diagnostically to ensure that the host computer the user is trying to reach is operating.

Syntax : ping <website>

- **Provide examples of how to use ping to test connectivity to a website and a local host.**

Ans :
ping <website>

ping codeforces
for localhost : ping 127.0.0.1

2. Ping Output Analysis

- **Run the command ping (link unavailable) and capture the output.**

```
C:\Users\sakha>ping codeforces.com

Pinging codeforces.com [104.26.6.164] with 32 bytes of data:
Reply from 104.26.6.164: bytes=32 time=49ms TTL=50
Reply from 104.26.6.164: bytes=32 time=46ms TTL=53
Reply from 104.26.6.164: bytes=32 time=57ms TTL=50
Reply from 104.26.6.164: bytes=32 time=55ms TTL=50

Ping statistics for 104.26.6.164:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 46ms, Maximum = 57ms, Average = 51ms
```

- **Analyze the output, explaining each line and its significance (e.g., packet loss, round-trip time, etc.).**

- ➔ Here 32 bytes of data packet is being received and sent. (to 104.26.6.164 (codeforces.com)
- ➔ Round trip time for first packet to go to the server and back is 49ms, then 46ms, then 57 ms, and 55 ms
- ➔ As we can see here 4 packets were sent and 4 were received so 0 packets were lost and hence 0% loss.
- ➔ TTL : time to live for every packet tells us how many hops is the packet allowed to make (the number of routers in between)

- **Repeat the process for a local host (e.g., ping 127.0.0.1).**

```
C:\Users\sakha>ping 127.0.0.1

Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Analysis :

- ➔ Here 32 bytes of data packet is being received and sent. (to 127.0.0.1(localhost)
- ➔ Round trip time for all the packages is <1ms (almost instantaneous because its localhost running on the same machine)
- ➔ As we can see here 4 packets were sent and 4 were received so 0 packets were lost and hence 0% loss.
- ➔ TTL : time to live for every packet tells us how many hops is the packet allowed to make (the number of routers in between)

3. Ping Options

- **Research and explain the following ping options:**

- -c (count) (-n in windows)

syntax : ping -c <count> <destination>

Use : specifies the number of ping requests to send.

ping -c 4 codeforces.com , will only send 4 packets

```
sakha@Akhands-VB14 MINGW64 ~ (master)
$ ping -n 4 codeforces.com

Pinging codeforces.com [104.26.7.164] with 32 bytes of data:
Reply from 104.26.7.164: bytes=32 time=36ms TTL=54
Reply from 104.26.7.164: bytes=32 time=64ms TTL=52
Reply from 104.26.7.164: bytes=32 time=58ms TTL=52
Reply from 104.26.7.164: bytes=32 time=51ms TTL=54

Ping statistics for 104.26.7.164:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 36ms, Maximum = 64ms, Average = 52ms
```

- -s (size) (-l in windows)

Syntax : ping -s <size> <destination>

Use : sets the size of the ICMP (internet control message protocol) packet's payload in bytes.

```
sakha@Akhands-VB14 MINGW64 ~ (master)
$ ping -l 32 codeforces.com

Pinging codeforces.com [172.67.68.254] with 32 bytes of data:
Reply from 172.67.68.254: bytes=32 time=49ms TTL=51
Reply from 172.67.68.254: bytes=32 time=35ms TTL=55
Reply from 172.67.68.254: bytes=32 time=37ms TTL=55
Reply from 172.67.68.254: bytes=32 time=45ms TTL=51

Ping statistics for 172.67.68.254:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 35ms, Maximum = 49ms, Average = 41ms
```

- -t (ttl) (-i in windows)

Syntax : ping -t <ttl> <destination>

Use: sets the Time To Live (TTL) value for the outgoing packets

```
sakha@Akhands-VB14 MINGW64 ~ (master)
$ ping -i 50 codeforces.com

Pinging codeforces.com [172.67.68.254] with 32 bytes of data:
Reply from 172.67.68.254: bytes=32 time=46ms TTL=55
Reply from 172.67.68.254: bytes=32 time=38ms TTL=55
Reply from 172.67.68.254: bytes=32 time=46ms TTL=51
Reply from 172.67.68.254: bytes=32 time=35ms TTL=55

Ping statistics for 172.67.68.254:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 35ms, Maximum = 46ms, Average = 41ms
```

- -W (deadline)

Syntax : ping -W <deadline> <destination>

Use : sets a deadline (in seconds) for the ping command to run

```
sakha@Akhands-VB14 MINGW64 ~ (master)
$ ping -w 2 codeforces.com

Pinging codeforces.com [172.67.68.254] with 32 bytes of data:
Reply from 172.67.68.254: bytes=32 time=47ms TTL=51
Reply from 172.67.68.254: bytes=32 time=44ms TTL=55
Reply from 172.67.68.254: bytes=32 time=46ms TTL=55
Reply from 172.67.68.254: bytes=32 time=48ms TTL=51

Ping statistics for 172.67.68.254:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 44ms, Maximum = 48ms, Average = 46ms
```

- Provide examples of how to use each option.

Examples :

-c : ping -c 4 codeforces.com , will only send 4 packets

-s : ping -s 100 codeforces.com, will send packets of size 100 bytes

-t : ping -t 10 codeforces.com, will send packets with ttl 10

-W : ping -W 10 codeforces.com, will send packets with with a deadline of 10 second

4. Troubleshooting with Ping

- **Describe a scenario where ping would be used for network troubleshooting (e.g., connectivity issues, slow network speeds).**

- **Explain how to use ping to diagnose the issue, including which options to use and why.**

Scenario: A user is experiencing **slow internet speeds and intermittent connectivity issues** while trying to access a company's web application.

Let's assume the website to be iitp.ac.in

To diagnose this issue using ping:

1) Start with a basic ping to the web application's domain:

   ping iitp.ac.in

   This will help determine if there's basic connectivity and reveal any packet loss.

2) Use the -c option to send a specific number of pings:

   ping -c 100 iitp.ac.in

   This sends 100 pings, providing a larger sample size to detect intermittent issues.

3) Employ the -s option to test with larger packet sizes:

   ping -s 1000 iitp.ac.in

   Larger packets can help identify Maximum Transmission Unit or fragmentation issues.

4) Utilize the -i option to set a shorter interval between pings:

   ping -i 0.2 iitp.ac.in

   This sends pings more frequently, potentially catching brief outages.

5) Combine options for comprehensive testing:

   ping -c 1000 -i 0.2 -s 1000 iitp.ac.in

   This sends 1000 large packets at short intervals, providing detailed data on connectivity.

By analysing the results, you can:

-> Identify packet loss (indicating connection issues)

-> Assess latency (high times suggest network congestion)

-> Detect variations in response times (pointing to unstable connections)

-> Compare results to known good connections to isolate the problem

This approach helps determine if the issue is with the local network, ISP, or the remote server, guiding further troubleshooting steps.

5. **Develop a ping type utility using Scapy. It should have the following points.**

- 1. Basic Functionality
- Ensure the provided code works correctly.
- Test with different destination IPs and counts.

2. Additional Features

- Implement the following features:
  - Option to specify TTL (Time-To-Live)
  - Option to specify packet size
  - Option to specify timeout
- Use Scapy's built-in functions to implement these features.

3. Error Handling

- Add error handling for cases like:
  - Invalid destination IP
  - Invalid count or TTL values
  - Timeout errors
- Use try-except blocks to catch and handle exceptions.

4. Output Formatting

  - Improve the output formatting to include:
  - Packet loss percentage
  - Average RTT (Round-Trip Time)
  - Maximum and minimum RTT values
- Use Python's built-in formatting options to create a clean output.

Code:

```python
from scapy.all import *
import sys

def ping(destination, count=4, packet_size=64, ttl=64, timeout=2):
    # Validate and prepare the destination IP
    try:
        ip = IP(dst=destination, ttl=ttl)
    except Exception as e:
        print(f"Error: Could not resolve destination IP '{destination}'. Please check the address and try again.")
        return

    # Basic input validation for count, packet size, and TTL
    if not isinstance(count, int) or count <= 0:
        print("Error: 'count' should be a positive integer.")
        return
    if not isinstance(packet_size, int) or packet_size <= 0:
        print("Error: 'packet_size' should be a positive integer.")
        return
    if not isinstance(ttl, int) or ttl <= 0:
        print("Error: 'ttl' should be a positive integer.")
        return

    # Loop to send the specified number of ping requests
    for i in range(count):
        # Build the ICMP packet with the desired payload size
        packet = ip / ICMP() / (b'Q' * packet_size)
        try:
            # Send the packet and wait for a reply (timeout set by user)
            reply = sr1(packet, timeout=timeout, verbose=0)
            if reply:
                # Calculate and display round-trip time (in milliseconds)
                rtt = (reply.time - packet.sent_time) * 1000
                print(f"Reply from {reply.src}: bytes={packet_size} time={rtt:.2f}ms TTL={reply.ttl}")
            else:
                print("Request timed out.")
        except Exception as e:
            # Handle any exceptions that occur during packet send/receive
            print(f"An error occurred: {e}")
            break
```

```python
if __name__ == "__main__":
    # Ensure at least the destination is provided
    if len(sys.argv) < 2:
        print("Usage: python ping.py <destination> [count] [packet_size] [ttl] [timeout]")
    else:
        # Parse command-line arguments
        destination = sys.argv[1]
        count = int(sys.argv[2]) if len(sys.argv) > 2 else 4
        packet_size = int(sys.argv[3]) if len(sys.argv) > 3 else 64
        ttl = int(sys.argv[4]) if len(sys.argv) > 4 else 64
        timeout = int(sys.argv[5]) if len(sys.argv) > 5 else 2

        # Run the ping function with provided arguments
        ping(destination, count, packet_size, ttl, timeout)
```

Submission:

```
sakha@Akhands-VB14 MINGW64 /d/Project Durgakund/Networks/Lab
$ python Lab3_2201cs11.py google.com 5 128 64 3
Reply from 142.250.206.142: bytes=128 time=74.52ms TTL=114
Reply from 142.250.206.142: bytes=128 time=71.30ms TTL=114
Reply from 142.250.206.142: bytes=128 time=54.97ms TTL=114
Reply from 142.250.206.142: bytes=128 time=56.05ms TTL=114
Reply from 142.250.206.142: bytes=128 time=53.40ms TTL=114
```

Please submit a written report (PDF or Word document) containing your answers to the tasks above. Include screenshots or output captures where relevant.

For programming you need to create a github page (instruction to be given later)