Name: Ujjawal Sinha
Roll No: 2201CS73

Lab 03: 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.

Purpose of Ping Utility: The ping utility is a network diagnostic tool used to test the reachability of a host on an Internet Protocol (IP) network. It works by sending Internet Control Message Protocol (ICMP) Echo Request messages to the target host and waits for an Echo Reply. The main purpose is to measure the round-trip time (RTT) for messages sent from the originating host to a destination computer and back.

The basic syntax of the ping command is: **ping [options] destination**

Here, the destination can be an IP address or a domain name.

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

Testing connectivity to a website:

ping google.com

This command sends ICMP Echo Request messages to Google's servers and waits for a response.

Testing connectivity to a local host:

ping 127.0.0.1

This command tests the network stack of your own machine by pinging the loopback interface.

2. Ping Output Analysis

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

ping www.google.com

Pinging www.google.com [142.250.182.164] with 32 bytes of data:

Reply from 142.250.182.164: bytes=32 time=111ms TTL=56

Reply from 142.250.182.164: bytes=32 time=236ms TTL=52

Reply from 142.250.182.164: bytes=32 time=72ms TTL=52

Reply from 142.250.182.164: bytes=32 time=139ms TTL=52

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

Ping statistics for 142.250.182.164:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 72ms, Maximum = 236ms, Average = 139 ms

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

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

3. Ping Options

- Research and explain the following ping options:

- Provide examples of how to use each option.

- -c (count)

- **Purpose**: Specifies the number of ICMP Echo Requests to send.
- **Example**:

  ping -c 4 google.com

  This will send 4 ICMP Echo Requests to google.com.

- -s (size)

- **Purpose**: Specifies the size of the ICMP packet payload in bytes.

- **Example**:

  ping -s 100 google.com

  This sends packets with a payload of 100 bytes.

- -t (ttl)

  - **Purpose**: Specifies the Time-To-Live value for the packets, which limits the number of hops the packet can take.
  - **Example**:

    ping -t 64 google.com

    This sets the TTL to 64 hops.

- -W (deadline)

  - **Purpose**: Specifies a timeout in seconds before the ping command exits, regardless of how many packets have been sent or received.
  - **Example**:

    ping -W 5 google.com

    This will terminate the ping command after 5 seconds.

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: Intermittent Connectivity Issues

Imagine a scenario where a user reports intermittent connectivity issues to a specific website. The user occasionally experiences timeouts or very slow response times when trying to access the site.

Using Ping to Diagnose the Issue

1. **Basic Connectivity Test**:
    - **Command**: ping example.com
    - **Purpose**: The first step is to verify whether the website is reachable and to observe the basic round-trip times (RTT). This helps in identifying if there is a consistent connection to the server.
    - **Expected Outcome**: If the pings are successful with low RTT values, basic connectivity is not the issue. If there are timeouts or high RTTs, there may be a problem with the network path or server.
2. **Checking for Packet Loss**:
    - **Command**: ping -c 100 example.com
    - **Purpose**: Sending a larger number of pings (e.g., 100) helps identify packet loss over time. Packet loss can cause intermittent connectivity issues.
    - **Expected Outcome**: If there is a significant percentage of packet loss, this indicates an issue in the network path, possibly due to congestion or faulty hardware.
3. **Testing with Varying TTL Values**:
    - **Command**: ping -t 1 example.com, ping -t 2 example.com, and so on.
    - **Purpose**: Time-To-Live (TTL) values control how many hops (routers) a packet can pass through before being discarded. By incrementally increasing the TTL, you can identify at which hop the problem occurs.
    - **Expected Outcome**: If packets are being dropped at a particular hop, this could indicate an issue with a specific router or a segment of the network.
4. **Analyzing with Different Packet Sizes**:
    - **Command**: ping -s 1500 example.com

- **Purpose**: Larger packets are more susceptible to being dropped if there is a problem with the network. Testing with different packet sizes can help identify issues with MTU (Maximum Transmission Unit) settings or fragmentation.
- **Expected Outcome**: If larger packets are consistently dropped, it may indicate an MTU mismatch or other network configuration issue.

5. **Setting a Timeout**:
   - **Command**: ping -W 5 example.com
   - **Purpose**: This sets a deadline for how long the ping command should wait for a response. It helps in identifying slow responses.
   - **Expected Outcome**: If responses are consistently received just before the timeout, this may indicate a slow network path, possibly due to congestion or a remote server under heavy load.

Summary of Diagnosis

- **Consistent High RTT**: Indicates possible congestion or a distant server.
- **Packet Loss**: Suggests network instability, possibly due to hardware issues, interference, or overloaded routers.
- **Issues at a Specific TTL**: Points to a problem with a particular router or hop in the network.
- **Dropped Large Packets**: May indicate an MTU issue or network configuration problem.
- **Timeouts**: Can be a sign of severe network delays or server issues.

Using these ping options helps narrow down the root cause of connectivity issues, whether they are related to the local network, the ISP, or the destination server.

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.

```
from scapy.all import IP, ICMP, sr1
import time

def basic_ping(dest_ip, count=4):
        print(f"Pinging {dest_ip} with {count} packets...")
        for i in range(count):
        packet = IP(dst=dest_ip)/ICMP()
        start_time = time.time()
        response = sr1(packet, timeout=1, verbose=0)
        end_time = time.time()
        if response:
        rtt = (end_time - start_time) * 1000
         print(f"Reply from {dest_ip}: time={rtt:.2f} ms")
        else:
         print("Request timed out")
        print("Ping complete.")

# Testing basic functionality
basic_ping("8.8.8.8", count=4)
```

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.

```
def custom_ping(dest_ip, count=4, ttl=64, size=56, timeout=1):
    print(f"Pinging {dest_ip} with {count} packets, TTL={ttl}, Size={size} bytes, Timeout={timeout} seconds...")
    for i in range(count):
    packet = IP(dst=dest_ip, ttl=ttl)/ICMP()/("X" * size)
    start_time = time.time()
    response = sr1(packet, timeout=timeout, verbose=0)
    end_time = time.time()
    if response:
      rtt = (end_time - start_time) * 1000
      print(f"Reply from {dest_ip}: time={rtt:.2f} ms")
    else:
      print("Request timed out")
  print("Ping complete.")
# Testing additional features
custom_ping("8.8.8.8", count=5, ttl=128, size=64, timeout=2)
```

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.

```
def safe_ping(dest_ip, count=4, ttl=64, size=56, timeout=1):
    try:
        if not (1 <= ttl <= 255):
            raise ValueError("TTL must be between 1 and 255")
        if size < 0:
            raise ValueError("Size must be non-negative")
        if timeout <= 0:
            raise ValueError("Timeout must be positive")
        if count <= 0:
            raise ValueError("Count must be positive")

        print(f"Pinging {dest_ip} with {count} packets, TTL={ttl}, Size={size} bytes, Timeout={timeout} seconds...")
        for i in range(count):
            packet = IP(dst=dest_ip, ttl=ttl)/ICMP()/("X" * size)
            start_time = time.time()
```

```
            response = sr1(packet, timeout=timeout, verbose=0)
            end_time = time.time()
            if response:
                rtt = (end_time - start_time) * 1000
                print(f"Reply from {dest_ip}: time={rtt:.2f} ms")
            else:
                print("Request timed out")
        print("Ping complete.")
    except ValueError as e:
        print(f"Error: {e}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

# Testing error handling
safe_ping("8.8.8.8", count=-1, ttl=300, size=64, timeout=-2)
```

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.

```
def enhanced_ping(dest_ip, count=4, ttl=64, size=56, timeout=1):
    rtt_times = []
    packets_sent = 0
    packets_received = 0

    try:
        if not (1 <= ttl <= 255):
            raise ValueError("TTL must be between 1 and 255")
        if size < 0:
            raise ValueError("Size must be non-negative")
        if timeout <= 0:
            raise ValueError("Timeout must be positive")
        if count <= 0:
            raise ValueError("Count must be positive")

        print(f"Pinging {dest_ip} with {count} packets, TTL={ttl}, Size={size} bytes,
Timeout={timeout} seconds...")
        for i in range(count):
            packets_sent += 1
            packet = IP(dst=dest_ip, ttl=ttl)/ICMP()/("X" * size)
            start_time = time.time()
            response = sr1(packet, timeout=timeout, verbose=0)
            end_time = time.time()
            if response:
```

```
            packets_received += 1
            rtt = (end_time - start_time) * 1000
            rtt_times.append(rtt)
            print(f"Reply from {dest_ip}: time={rtt:.2f} ms")
        else:
            print("Request timed out")

    print("Ping complete.")
    packet_loss = ((packets_sent - packets_received) / packets_sent) * 100
    print(f"Packets: Sent = {packets_sent}, Received = {packets_received}, Lost
= {packets_sent - packets_received} ({packet_loss:.2f}% loss)")

    if rtt_times:
        min_rtt = min(rtt_times)
        max_rtt = max(rtt_times)
        avg_rtt = sum(rtt_times) / len(rtt_times)
        print(f"Approximate round trip times: Min = {min_rtt:.2f} ms, Max =
{max_rtt:.2f} ms, Avg = {avg_rtt:.2f} ms")

  except ValueError as e:
    print(f"Error: {e}")
  except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

```
# Testing enhanced output

enhanced_ping("8.8.8.8", count=4, ttl=128, size=64, timeout=2)
```