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.

Ans: The ping utility is a network administration tool used to test the reachability of a host on an Internet Protocol (IP) network. It measures the round-trip time for messages sent from the originating host to a destination computer and back.

Purpose:

- **Check Connectivity:** ping is primarily used to verify if a host is reachable and available on the network.
- **Measure Latency:** It helps in measuring the time it takes for a packet to travel to the destination and return (round-trip time).
- **Diagnose Network Issues:** By using ping, you can identify issues like network congestion, packet loss, or whether a network device is down.

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

Ans: Run this command in windows terminal [ping google.com], then it sends ICMP Echo Request packets to google.com and waits for a reply, showing the round-trip time for each packet. And for localhost Run [ping 127.0.0.1].

2. Ping Output Analysis

- Run the command ping (google.com) and capture the output.

Ans: Pinging google.com [2404:6800:4009:813::200e] with 32 bytes of data:

Reply from 2404:6800:4009:813::200e: time=231ms

Reply from 2404:6800:4009:813::200e: time=235ms

Reply from 2404:6800:4009:813::200e: time=135ms

Reply from 2404:6800:4009:813::200e: time=330ms

Ping statistics for 2404:6800:4009:813::200e:

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

Approximate round trip times in milli-seconds:

Minimum = 135ms, Maximum = 330ms, Average = 232ms

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

Ans: Initial Line:→

Pinging google.com: Indicates the domain name being pinged.

[2404:6800:4009:813::200e]: This is the IPv6 address of google.com, resolving the domain name to its actual IP address.

32 bytes of data: Refers to the size of the ICMP Echo Request packets being sent to the target host.

Reply Lines:→

Reply from 2404:6800:4009:813::200e: This confirms that a response was received from the IP address 2404:6800:4009:813::200e.

time=231ms: Indicates the round-trip time for the packet, which is the time it took for the packet to reach the target and for the response to come back. The values (231ms, 235ms, 135ms, and 330ms) show the time in milliseconds for each of the four packets.

Ping Statistics:→

Packets: Sent = 4: Indicates that four ICMP Echo Request packets were sent to the target.

Received = 4: Shows that all four packets received a response.

Lost = 0 (0% loss): Indicates that no packets were lost during transmission, which is ideal as packet loss can suggest network issues.

Round-Trip Time Statistics:→

Minimum = 135ms: The fastest round-trip time recorded among the four packets.

Maximum = 330ms: The slowest round-trip time recorded.

Average = 232ms: The average round-trip time for the four packets.

Significance:

- **Packet Loss:** A 0% packet loss indicates a stable connection without any interruptions.
- **Round-Trip Time:** The times (135ms, 231ms, 235ms, 330ms) provide insight into the latency of the network. Higher round-trip times or large variations might indicate network congestion or routing issues.
- **Minimum, Maximum, and Average:** These values help assess the consistency of the connection. A wide range between minimum and maximum times could suggest variability

in network performance. An average time of 232ms is reasonable, though it could be higher or lower depending on the specific network conditions and distance to the server.

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

Ans: 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

Significance:

- **Testing Local Network Stack:** This ping test to 127.0.0.1 is a quick way to verify that the local machine's networking software and hardware are functioning correctly.
- **Zero Latency:** The time readings being less than 1ms or 0ms across the board are expected since the packets don't actually travel over an external network.
- **No Packet Loss:** The absence of packet loss confirms that the local network interface is functioning without issues.

3. Ping Options

- Research and explain the following ping options:

Ans: - -c (count): Specifies the number of packets to send. For example, -c 4 sends four packets.

- -s (size): Specifies the number of data bytes to be sent. For example, -s 64 sends a 64-byte packet.

- -t (ttl): Sets the Time To Live (TTL) for the packet, which determines how many hops the packet can take before being discarded.

- -W (deadline): The -W option specifies a timeout in seconds. The ping command will wait for a reply from the target for the specified number of seconds before stopping, regardless of how many packets have been sent or received. This is useful if you want to limit the time ping runs rather than the number of packets.

- Provide examples of how to use each option.

Ans: --c : Command- [ping -c 5 google.com] result: 5 packets transmitted, 5 received, 0% packet loss, time 4006ms

rtt min/avg/max/mdev = 43.069/54.825/72.401/14.155 ms.

--s : command- [ping -s 100 google.com] result:

--t : ping -t 10 google.com

--w : ping -W 2 google.com

4. Troubleshooting with Ping

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

Ans: *Scenario Overview***:**

Employees at a company are experiencing slow network speeds when accessing a web application hosted on a remote server. Other internet activities seem to be functioning normally, but the specific web application is sluggish, leading to complaints and productivity issues.

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

Ans: Step 1: *Test Basic Connectivity to the Server* : ping -c 4 appserver.example.com

Expected Output:

- Replies with consistent and reasonable round-trip times. If you see timeouts or no responses, it indicates that the server might be down or unreachable from your network.

Step 2: *Check for Network Congestion or Packet Loss* : ping -c 20 appserver.example.com

Expected Output:

- Ideally, all 20 pings should be returned with no packet loss. If there is packet loss (e.g., 10% loss), it could indicate network issues either on the local network or somewhere along the route to the server.
- Look for variations in round-trip times. Significant variation (e.g., some responses taking much longer than others) might suggest network congestion or an overloaded server.

Step 3: *Measure Latency Over Time:* ping -c 100 -i 1 appserver.example.com (The -c 100 option sends 100 pings, and -i 1 sets the interval between each ping to 1 second)

Expected Output:

- Consistent round-trip times. Spikes in latency (e.g., a sudden increase from 50ms to 500ms) could point to network congestion or issues with the server's performance.

Step 4*: Test Packet Size Handling* : ping -s 1000 appserver.example.com

Expected Output:

- Successful replies with reasonable round-trip times. If the larger packets are delayed, lost, or fragmented, it could indicate issues such as MTU (Maximum Transmission Unit) problems or network equipment that's struggling with larger packets.

Step 5: *Set a Deadline for the Ping Test* : ping -W 5 appserver.example.com

Expected Output:

- The ping command should complete within the set time limit, providing useful data without excessive waiting. If the server is slow to respond and takes more than 5 seconds, it could be indicative of performance issues on the server itself or severe network congestion.

Interpreting the Results:

Consistent High Latency: If you see consistently high latency across all pings, the issue might be related to the network path between the client and the server, possibly due to congestion or inefficient routing.

Intermittent High Latency or Packet Loss: If some pings show high latency or are lost while others are normal, it suggests intermittent network issues, possibly related to network equipment or transient congestion.

No Response: If the pings do not receive a reply, it could indicate that the server is down, the firewall is blocking ICMP traffic, or there's a network configuration issue.

Conclusion:

Using ping in this scenario allows you to systematically diagnose whether the slow network speeds are due to connectivity issues, network congestion, or problems with packet handling. Based on the results, you can decide whether to escalate the issue to your network team, check server performance, or look into routing and firewall configurations.

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.

Ans:

```
from scapy.all import *
import time
import statistics

def ping(dest_ip, count=4, ttl=64, packet_size=64, timeout=2):
    try:
        # Validate the destination IP
        socket.inet_aton(dest_ip)
    except socket.error:
        print(f"Invalid destination IP address: {dest_ip}")
        return

    # Initialize variables for tracking statistics
    rtts = []
    sent_packets = 0
    received_packets = 0

    for i in range(count):
        # Create an ICMP packet with the given TTL and packet size
        packet = IP(dst=dest_ip, ttl=ttl) / ICMP() / (b'X' * packet_size)
        sent_time = time.time()
```

```
        try:
            # Send the packet and wait for a response
            response = sr1(packet, timeout=timeout, verbose=0)
            sent_packets += 1

            if response:
                received_time = time.time()
                rtt = (received_time - sent_time) * 1000  # RTT in milliseconds
                rtts.append(rtt)
                print(f"Reply from {response.src}: bytes={packet_size} time={rtt:.2f}ms TTL={response.ttl}")
                received_packets += 1
            else:
                print(f"Request timed out.")

        except Exception as e:
            print(f"Error sending packet: {e}")
            continue

    # Calculate statistics
    if rtts:
        packet_loss = ((sent_packets - received_packets) / sent_packets) * 100
        avg_rtt = statistics.mean(rtts)
        max_rtt = max(rtts)
        min_rtt = min(rtts)
        print(f"\n--- {dest_ip} ping statistics ---")
        print(f"{sent_packets} packets transmitted, {received_packets} received, {packet_loss:.1f}%
packet loss")
        print(f"rtt min/avg/max = {min_rtt:.2f}/{avg_rtt:.2f}/{max_rtt:.2f} ms")
    else:
        print(f"\nNo response received from {dest_ip}.")
```

```
if __name__ == "__main__":
    # Example usage:
    ping(dest_ip="8.8.8.8", count=5, ttl=64, packet_size=100, timeout=2)
```

Explanation:

1. **Basic Functionality:**
    - The ping function sends ICMP Echo Requests to the specified destination IP.
    - The script counts sent and received packets and calculates RTT for each response.
2. **Additional Features:**
    - ttl: Controls the Time-To-Live of the packets.
    - packet_size: Controls the size of the ICMP payload.
    - timeout: Controls the time to wait for a reply.
3. **Error Handling:**
    - The script handles invalid IP addresses and timeouts.
    - Try-except blocks are used to catch socket errors and other exceptions during packet sending.
4. **Output Formatting:**
    - The output includes detailed information about each ping attempt, including RTT and TTL.
    - After all pings, it calculates and displays packet loss percentage, and the minimum, average, and maximum RTT.

Testing the Script:

- Test with different IPs, such as 8.8.8.8 (Google's public DNS) or a local IP.
- Change the count, ttl, packet_size, and timeout parameters to observe the behavior.

This script should work well for basic ping functionality with the added customization and error handling you requested.

Submission:

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)