Project Report: ICMP Pinger Lab (Python)

1. Title

ICMP Pinger Lab (Python) - Windows-Friendly Client Version

2. Abstract

This lab implements a Windows-friendly ICMP-based pinger in Python to explore low-level network programming, raw sockets, and ICMP protocol behavior. The program sends ICMP Echo Requests to a specified host, measures round-trip time (RTT), and computes packet loss. A GUI interface using Tkinter is provided for easier interaction. The implementation is designed for educational clarity and is Windows-compatible.

3. Introduction

The purpose of this lab is to understand ICMP, raw sockets, timing, and practical network troubleshooting tools such as ping. Additionally, the lab introduces a graphical interface to visualize ping operations and makes the program more user-friendly, especially on Windows systems where raw socket server functionality is limited.

4. Tools & Environment

- Python 3.8+
- Windows 10/11 (Administrator privileges required for raw sockets)
- Terminal/console and Tkinter GUI

5. Design and Implementation

- `icmp_pinger.py` uses raw sockets to send and receive ICMP packets.
- Packet structure follows ICMP echo request/reply format (type, code, checksum, id, sequence, payload).
- Timestamp is embedded in the payload to compute RTT.
- Replies are matched by process id and sequence.
- GUI built using Tkinter allows host/IP input and runs ping in a separate thread to keep interface responsive.
- ICMP server functionality is omitted due to Windows raw socket restrictions.

6. Features

- Configurable ping count, timeout, interval, and payload size.
- RTT statistics (min/avg/max/stddev) and packet loss percentage.
- Optional logging of ping results.
- GUI for user-friendly host entry and ping execution.
- Multithreading to avoid blocking GUI during ping operations.

7. Testing

- Tested against `google.com` and local IPs.
- Verified normal ping operations and RTT measurements.
- Verified timeout handling by pinging non-routable IPs.
- GUI tested to ensure responsiveness during ping sessions.

8. Results

- Sample console output:

PS F:\Git\ICMP-Pinger-Lab> python icmp_pinger.py -host google.com

PING google.com (142.251.223.238): Reply from 142.251.223.238: seq=1 time=54.37 ms Reply from 142.251.223.238: seq=2 time=54.36 ms Reply from 142.251.223.238: seq=3 time=54.63 ms Reply from 142.251.223.238: seq=4 time=54.54 ms --- google.com ping statistics --- 4 packets transmitted, 4 received, 0.0% packet loss rtt min/avg/max/std = 54.36/54.47/54.63/0.11 ms

- GUI allows host/IP entry and starts ping in a separate thread, output shown in console.

9. Limitations

- Requires Administrator privileges on Windows.
- Raw socket behavior differs from Linux/macOS.
- ICMP server functionality is not implemented.
- Not a full replacement for system `ping` — for educational purposes.

10. Conclusion

This lab reinforces networking concepts and practical use of ICMP. The Windows-friendly client with GUI provides a user-friendly, interactive way to measure network

latency, making the project suitable for cross-platform educational use.

11. References

- Kurose & Ross — Programming exercises: https://gaia.cs.umass.edu/kurose_ross/programming.php
- ICMP RFC (Request for Comments)
- Python official documentation and Tkinter tutorials