Name: ________________________

Project 3 – ICMP Traceroute

Introduction

In this lab you will learn how to implement a traceroute application using ICMP request and reply messages. The checksum code is provided. The header making is covered below.

Traceroute is a computer networking diagnostic tool which allows a user to trace the route from a host running the traceroute program to any other host in the world. Traceroute is implemented with ICMP messages. It works by sending ICMP echo (ICMP type '8') messages to the same destination with increasing value of the time-to-live (TTL) field. The routers along the traceroute path return ICMP Time Exceeded (ICMP type '11') when the TTL field become zero. The final destination sends an ICMP reply (ICMP type '0') messages on receiving the ICMP echo request. The IP addresses of the routers which send replies can be extracted from the received packets. The round-trip time between the sending host and a router is determined by setting a timer at the sending host.

Your task is to develop your own Traceroute application in python using ICMP. Your application will use ICMP but, in order to keep it simple, will not exactly follow the official specification in RFC 1739..

Code

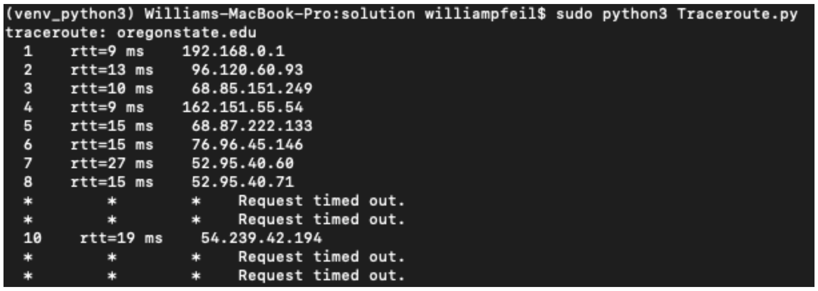
Below you will find the skeleton code for the client. You are to complete the skeleton code. The places where you need to fill in code are marked with `#Fill in start` and `#Fill in end`. Each place may require one or more lines of code.

Additional Notes

1. You do not need to be concerned about the checksum, as it is already given in the code.
2. This lab requires the use of raw sockets. In some operating systems, you may need administrator/root privileges to be able to run your Traceroute program.
3. See below for more information on ICMP header.
4. This will not work for websites that block ICMP traffic.
5. You will have to turn your firewall or antivirus software off to allow the messages to be sent and received.
6. After an initial run of 10 or so replies, you may start to get timeouts. Let the program run for a minute or so and then terminate with ctrl-c.
7. You may notice that the skeleton code allows the user to specify the size of the datagram payload sent. This will become very useful for Lab 4 ☺

What to turn in

1. In the Word doc:
   a. Include instructions on how to run your programs. Are they python3?
   b. Include screenshots of your running code for four different hosts with at least two on different continents.
   c. Include comments / questions (optional)
   d. Indicate if you worked the extra credit, and if so which part?
2. In your code listings:
   a. Include sources you used (web pages, tutorials, books, etc)
   b. Comment your code



```
(venv_python3) Williams-MacBook-Pro:solution williampfeil$ sudo python3 Traceroute.py
traceroute: oregonstate.edu
  1    rtt=9 ms    192.168.0.1
  2    rtt=13 ms    96.120.60.93
  3    rtt=10 ms    68.85.151.249
  4    rtt=9 ms    162.151.55.54
  5    rtt=15 ms    68.87.222.133
  6    rtt=15 ms    76.96.45.146
  7    rtt=27 ms    52.95.40.60
  8    rtt=15 ms    52.95.40.71
  *        *        *    Request timed out.
  *        *        *    Request timed out.
  10    rtt=19 ms    54.239.42.194
  *        *        *    Request timed out.
  *        *        *    Request timed out.
```

Extra Credit:

1. Currently, the program calculates the round-trip time for each packet and prints it out individually. Modify this to correspond to the way the standard ping program works. You will need to report the minimum, maximum, and average RTTs at the end of all pings from the client. In addition, calculate the packet loss rate (in percentage).
2. Your program can only detect timeouts in receiving ICMP echo responses. Modify the Pinger program to parse the ICMP response error codes and display the corresponding error results to the user. Examples of ICMP response error codes are 0: Destination Network Unreachable, 1: Destination Host Unreachable.

ICMP Header

The ICMP header starts after bit 160 of the IP header (unless IP options are used).

Bits	160-167	168-175	176-183	184-191
160	Type	Code	Checksum	
192	ID		Sequence	

- **Type** - ICMP type.
- **Code** - Subtype to the given ICMP type.
- **Checksum** - Error checking data calculated from the ICMP header + data, with value 0 for this field.
- **ID** - An ID value, should be returned in the case of echo reply.
- **Sequence** - A sequence value, should be returned in the case of echo reply.

Echo Request

The echo request is an ICMP message whose data is expected to be received back in an echo reply ("pong"). The host must respond to all echo requests with an echo reply containing the exact data received in the request message.

- Type must be set to 8.
- Code must be set to 0.
- The Identifier and Sequence Number can be used by the client to match the reply with the request that caused the reply. In practice, most Linux systems use a unique identifier for every ping process, and sequence number is an increasing number within that process. Windows uses a fixed identifier, which varies between Windows versions, and a sequence number that is only reset at boot time.
- The data received by the echo request must be entirely included in the echo reply.

Echo Reply

The echo reply is an ICMP message generated in response to an echo request, and is mandatory for all hosts and routers.

- Type and code must be set to 0.
- The identifier and sequence number can be used by the client to determine which echo requests are associated with the echo replies.
- The data received in the echo request must be entirely included in the echo reply.