PA3 – Python Ping Using Raw Sockets[1]

Overview

For this project, you will write a Python script that implements the well-known Ping program.

Ping is a computer network application used to test whether a particular host is reachable across an IP network. It is also used to self-test the network interface card of the computer or as a latency test. It works by sending ICMP “echo request” packets to the target host and listening for ICMP “echo reply” replies. The "echo reply" is sometimes called a pong. Ping measures the round-trip time, records packet loss, and prints a statistical summary of the echo reply packets received (the minimum, maximum, and the mean of the round-trip times and in some versions the standard deviation of the mean).

Your task is to develop your own Ping application in Python. Your application will use ICMP but, in order to keep it simple, will not exactly follow the official specification in RFC 1739. Note that you will only need to write the client side of the program, as the functionality needed on the server side is built into almost all operating systems.

You should complete the Ping application so that it sends ping requests to a specified host separated by approximately one second. Each message contains a payload of data that includes a timestamp. After sending each packet, the application waits up to one second to receive a reply. If one second goes by without a reply from the server, then the client assumes that either the ping packet or the pong packet was lost in the network (or that the server is down).

A code template is provided on the course website. It includes some helpful comments to guide you. DO NOT CHANGE ANY OF THE CODE THAT IS IN THE TEMPLATE, INCLUDING THE COMMENTS! Your main task is to fill in the pieces of missing code between the asterisks in the receiveOnePing() method:

```
print("[*] Received packet of length:", len(recPacket))

# ********************************************
#Fetch the IP header data you need

#Print IP data

#Fetch the ICMP header data you need

#Print ICMP data

#Return total time elapsed in ms

# ********************************************

timeLeft = timeLeft - howLongInSelect
if timeLeft <= 0:
    return "Request timed out."
```

[1] * This write-up is an adaptation of the ICMP Pinger assignment that accompanies the textbook

What to Hand In

1. A screen shot of the console output of your program (see example below)

Obtaining a Proof of Concept with the Real Ping

From a command prompt using your computer (Linux, Windows, and Mac OSs all work the same way) you can use the real Ping program to make sure that the machine you are attempting to reach is accessible. The program you write should work where the real Ping program works, and fail where the real Ping program fails. Regardless of your system's security settings, you should be able to get a successful result by testing with localhost as follows:

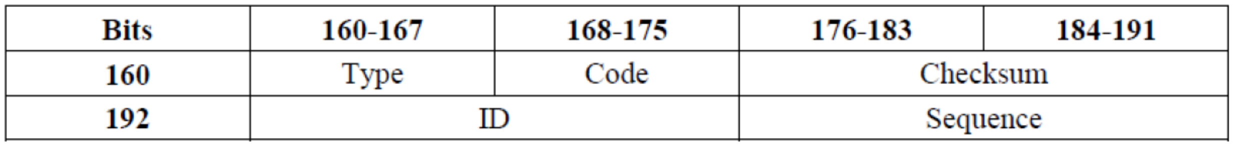
```
john%landon>ping localhost
```

Additional Details

1. In the receiveOnePing() method, you need to receive the structure ICMP_ECHO_REPLY and fetch the information you need, such as checksum, sequence number, time to live (TTL), etc. Study the sendOnePing() method before trying to complete the receiveOnePing() method.
2. You do not need to be concerned about how to compute the checksum—use the provided function.
3. This lab requires the use of raw sockets. In some operating systems, **you may need administrator/root privileges** to be able to run your Ping program. If you are using an IDE (e.g., IDLE), this means you need to run your IDE as Administrator on a Windows machine:

Helpful Links
https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol
https://en.wikipedia.org/wiki/IPv4#Header
https://en.wikipedia.org/wiki/List_of_IP_protocol_numbers
https://tools.ietf.org/html/rfc792
https://docs.python.org/3/library/struct.html
https://docs.python.org/3/library/socket.html

Internet Control Message Protocol (ICMP)
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.
- 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.

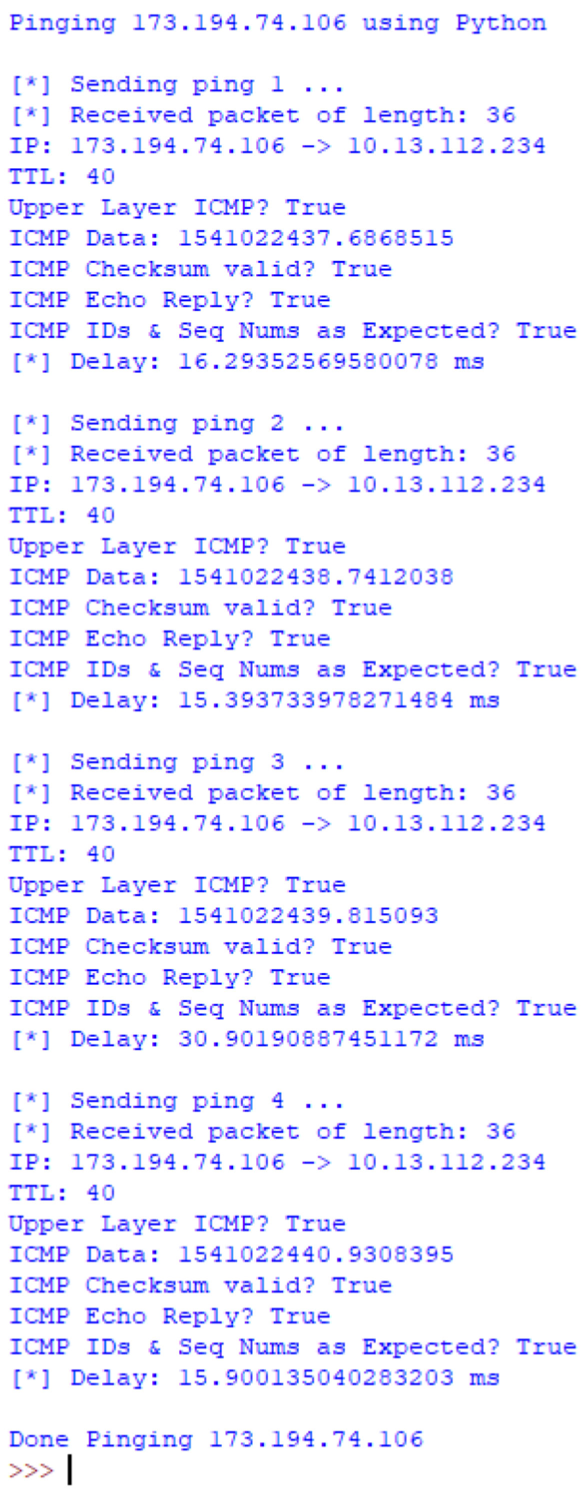
Echo Request
The echo request is an ICMP message whose data is expected to be received back in an echo reply ("pong"). The host must respond to all echo requests with an echo reply containing the exact data received in the request message.

- 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.

Screenshot of Successful Ping Console Output



```
Pinging 173.194.74.106 using Python

[*] Sending ping 1 ...
[*] Received packet of length: 36
IP: 173.194.74.106 -> 10.13.112.234
TTL: 40
Upper Layer ICMP? True
ICMP Data: 1541022437.6868515
ICMP Checksum valid? True
ICMP Echo Reply? True
ICMP IDs & Seq Nums as Expected? True
[*] Delay: 16.29352569580078 ms

[*] Sending ping 2 ...
[*] Received packet of length: 36
IP: 173.194.74.106 -> 10.13.112.234
TTL: 40
Upper Layer ICMP? True
ICMP Data: 1541022438.7412038
ICMP Checksum valid? True
ICMP Echo Reply? True
ICMP IDs & Seq Nums as Expected? True
[*] Delay: 15.393733978271484 ms

[*] Sending ping 3 ...
[*] Received packet of length: 36
IP: 173.194.74.106 -> 10.13.112.234
TTL: 40
Upper Layer ICMP? True
ICMP Data: 1541022439.815093
ICMP Checksum valid? True
ICMP Echo Reply? True
ICMP IDs & Seq Nums as Expected? True
[*] Delay: 30.90190887451172 ms

[*] Sending ping 4 ...
[*] Received packet of length: 36
IP: 173.194.74.106 -> 10.13.112.234
TTL: 40
Upper Layer ICMP? True
ICMP Data: 1541022440.9308395
ICMP Checksum valid? True
ICMP Echo Reply? True
ICMP IDs & Seq Nums as Expected? True
[*] Delay: 15.900135040283203 ms

Done Pinging 173.194.74.106
>>> 
```