Worcester Polytechnic Institute
Department of Computer Science
CS4516– Computer Networks

Project 2 – ICMP Pinger

Goal: This assignment is to help you understand the IP and ICMP packet structure, and be familiar with the ICMP protocol.

Instructions:

In this lab, you will gain a better understanding of Internet Control Message Protocol (ICMP). You will learn to implement a Ping application using ICMP request and reply messages.

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 reply" 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).

Please develop your own Ping application in Python. If you prefer C or Java implementation, that's OK, but the provided skeleton code uses Python. Please run your code on your own computer, as it needs administrator privilege. Your job is to complete the skeleton code (Please note that you will need to do extra for the additional functions, which are explained later) and test your client, as shown in the screenshot below:

Example Commands in Snapshots:

```
xiaoyan.sun@macbook-pro-52 Desktop % sudo python3 icmp.py
Pinging 142.251.40.142 using Python:

Reply from 142.251.40.142: bytes_data=36 time=21.7089655ms TTL=57
Reply from 142.251.40.142: bytes_data=36 time=21.8100555ms TTL=57
Reply from 142.251.40.142: bytes_data=36 time=29.2940145ms TTL=57
Reply from 142.251.40.142: bytes_data=36 time=23.6401565ms TTL=57
Reply from 142.251.40.142: bytes_data=36 time=21.5218075ms TTL=57
```

Snapshot of running the python code locally

About Internet Control Message Protocol (ICMP):

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

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.

Code
You will be provided 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
In “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 and complete the “sendOnePing” method before trying to complete the “receiveOnePing” method.

This lab requires the use of raw sockets. In some operating systems, you may need administrator/root privileges to be able to run your Pinger program.

Testing the Pinger
First, test your client by sending packets to localhost, that is, 127.0.0.1.

Then, you should see how your Pinger application communicates across the network by pinging servers in different continents.

Additional functions:

AF1. 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.

Please note the provided skeleton code didn't provide the ***#Fill in start*** and ***#Fill in end*** signs for the additional functions. Please change the code where appropriate to implement the additional functions.

Deliverables:

1. The client code without additional functions AF1;
2. The client code with additional functions AF1.
3. A README (txt) file that spells out exactly how to run the code.

Code plagiarism is absolutely **NOT** allowed. If needed, you may be asked for a **demonstration** of running your program in front of the instructor/SAs and answer their questions about your code. In this case, your grade will be based on both the report and your performance during demonstration.

Grading:

Grade breakup (%) assuming all documents are present:

1. Client code without additional functions AF1 = 75 points
2. Additional function AF1 = 20 points
3. Readme file = 5 points

The grade will be a **ZERO** if the code does not run or gives a run-time error.

To get the FULL CREDIT for the code:

1. The code should be able to ping localhost (127.0.0.1) successfully;
2. The code should be able to ping a remote server successfully;
3. The code should be able to display the bytes_data, time between packet sent and received, and TTL correctly;
4. The code should be able to report the minimum, maximum, and average RTTs at the end of all pings from the client (***AF1***).