Programming Assignment 3: ICMP Ping and ARP Packet Capture and Analysis
CSE 310, Spring 2022
Instructor: Aruna Balasubramanian
Due date: April 19 2022, 9.00pm

The goal of this assignment is to implement a Ping application using ICMP request and reply messages and learn how to perform ARP packet capture and analysis on Wireshark.

Part A ICMP Ping Tool (50 points)

In this assignment, 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 request” packets to the target host and listening for ICMP “echo reply” replies.

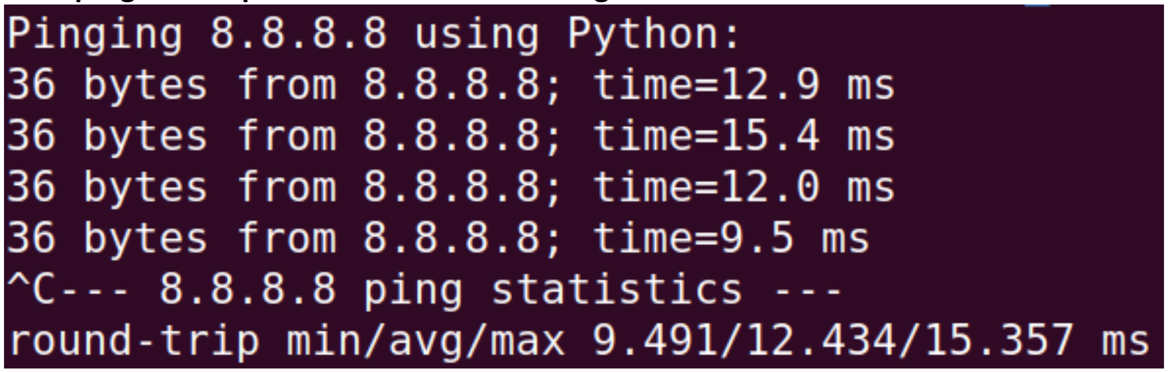
The "echo reply" is sometimes called a ping. Ping measures the round-trip time, records packet loss, and prints a statistical summary of the echo reply packets received (the minimum, maximum, and the mean of the round-trip times and in some versions the standard deviation of the mean). Your task is to 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 until you exit the program using a Control C command. 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).

You need to calculate the round-trip time for each packet and print it out individually. You will need to report the minimum, maximum, and average RTTs at the end of all pings from the client (i.e when the program is closed (control c)).

In the resources section on Piazza, 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 #TODO and #TODO END. Each place may require one or more lines of code.

Your program output should be like following:



```
Pinging 8.8.8.8 using Python:
36 bytes from 8.8.8.8; time=12.9 ms
36 bytes from 8.8.8.8; time=15.4 ms
36 bytes from 8.8.8.8; time=12.0 ms
36 bytes from 8.8.8.8; time=9.5 ms
^C--- 8.8.8.8 ping statistics ---
round-trip min/avg/max 9.491/12.434/15.357 ms
```

Testing the Pinger

Test your Pinger for the following scenario and include screenshots for each.

- Test your client by sending packets to localhost, that is, 127.0.0.1.
- Test your client by sending packets to stonybrook.edu or cs.stonybrook.edu.
- Select and ping 4 root servers outside the U.S.
- Explain the differences in minimum round trip time to each of these servers in parts A, B, and C.

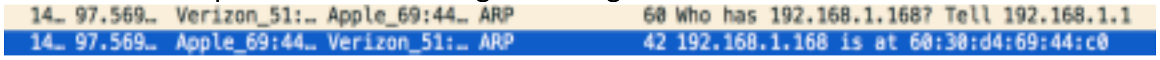
<u>Your submission should include your ping implementation along with explanations for the pinger testing scenarios, above.</u>

Part B ARP Packet Capture and Analysis (50 points)

The goal of this part of the assignment is to learn how to perform packet capture and analysis on Wireshark. We are specifically going to look at ARP packets.

Your first task is to capture an ARP exchange from your computer. To do this, open Wireshark and start recording. It will be easier if you filter for the ``arp” packets. You should wait for some time, browse the Web etc and you will see an ARP message exchange. The message exchange includes an ARP request and an ARP response. Once you see the exchange, stop the capture and store the packet as ``assignment4_my_arp.pcap”. Read a Wireshark tutorial on the Web if you are unsure how to capture messages and store them.

Here is an example of an ARP message exchange



```
14… 97.569… Verizon_51:… Apple_69:44… ARP        60 Who has 192.168.1.168? Tell 192.168.1.1
14… 97.569… Apple_69:44… Verizon_51:… ARP        42 192.168.1.168 is at 60:30:d4:69:44:c0
```

Submit the assignment4_my_arp.pcap and the screenshot of your ARP message exchange (similar to the example above).

Your second task is to write a program ``analysis_pcap_arp” that analyzes the pcap trace for the ARP packet. This is similar to your previous assignment, but this time you are not allowed to use any structure. Perform a byte-level programming to read each byte and convert it to the ARP header element--- the sender MAC address, target MAC address, protocol type, etc. Refer to the ARP message structure in your book to determine the elements of the ARP message. You can use existing pcap libraries to get each packet in *bytes*.

Your program should process the ARP packets. For each packet, you should determine if it is an ARP packet, and if it is an ARP packet then process it further. Based on your analysis, answer the following questions:
(i) Print the entire ARP request and response for **one** ARP packet exchange (preferably the one you show in the screenshot).
(ii) Based on the ARP messages, tell us the IP address and MAC address of your router. Explain how you determined this. You do not need to write code for this.

<u>Submit your well formatted program, answers to (i) and (ii) and a README that explains how to run your program including details of your program logic for Part B.</u>

Submission instruction

As before, you should write your program in Python or C/C++. If you want to write in any other language, please talk to me.

You need to submit your homework in a single zip file as follows:

• The zip file and (the root folder inside) should be named using your last name, first name, and the assignment number, all separated by a dash (‘-‘) e.g. lastname-firstname assignment3.zip
• The zip file should contain all submissions for parts A and B.

Additional Notes for Part A

1. 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 the “sendOnePing” method before trying to complete the “receiveOnePing” method.
2. You do not need to be concerned about the checksum, as it is already given in the code.
3. This assignment requires the use of raw sockets. In some operating systems, <u>you may need administrator/root privileges</u> to be able to run your Pinger program.
4. The program execution <u>should be stopped using Ctrl+C</u>, like the ping command.

4. See the end of this programming assignment for more information on ICMP.

Internet Control Message Protocol (ICMP)

<u>ICMP Header</u>

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.

<u>Echo Request</u>

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.