NOVA SCHOOL OF SCIENCE & TECHNOLOGY
DEPARTAMENTO DE INFORMÁTICA

ICMP Pinger[1]

(Lab07 – Redes de Computadores 2024/2025)

Summary

- ICMP protocol
- Exercise – implement a ping application using ICMP request and reply messages

1. Introduction

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).

2. Exercise – ICMP Pinger application

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 a 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).
For information regarding the ICMP packet structure consult file icmp-info.pdf (available on zip file)

[1] This assignment is a copy (with very small changes changes) from exercise *ICMP Pinger* available at site http://gaia.cs.umass.edu/kurose_ross/programming.php , authors' website for the textbook *Computer Networking: a Top Down Approach* (Pearson 8th edition)

Code

The skeleton code for the client is in a separated file (available on the zip file). You must complete the code. The place where you need to fill in code are marked with *#Fill in start* and *#Fill in end*.
To correctly complete the code, you need to know the format of ICMP frames; for that, consult the file icmp-info.pdf that was provided on zip file.

Additional Notes

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 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.
4. See in a PDF file also in annex more information on ICMP.

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.

Python – struct module[2]

This module converts between Python values and C structs represented as Python bytes objects. Compact format strings describe the intended conversions to/from Python values.

struct.**pack**(*format*, *v1*, *v2*, ...)

Return a bytes object containing the values *v1*, *v2*, … packed according to the format string *format*. The arguments must match the values required by the format exactly.

struct.**unpack**(*format*, *buffer*)

Unpack from the buffer *buffer* (presumably packed by pack(format, ...)) according to the format string *format*. The result is a tuple even if it contains exactly one item. The buffer's size in bytes must match the size required by the format, as reflected by calcsize().

format[3]

Format strings describe the data layout when packing and unpacking data. They are built up from format characters, which specify the type of data being packed/unpacked. In addition, special characters control the byte order, size and alignment. Each format string consists of an optional prefix character which describes the overall properties of the data and one or more format characters which describe the actual data values and padding.

[2] For more information see https://docs.python.org/3/library/struct.html#

[3] For more information see https://docs.python.org/3/library/struct.html#format-strings