Github link:

[ICMPPing/ICMP.py at master · lhkhoi95/ICMPPing (github.com)](https://github.com/lhkhoi95/ICMPPing/blob/master/ICMP.py)

from socket import \*

import os

import sys

import struct

import time

import select

import binascii

ICMP\_ECHO\_REQUEST = 8

def checksum(*string*):

    csum = 0

    countTo = (len(*string*) // 2) \* 2

    count = 0

    while count < countTo:

        thisVal = ord(*string*[count+1]) \* 256 + ord(*string*[count])

        csum = csum + thisVal

        csum = csum & 0xffffffff

        count = count + 2

    if countTo < len(*string*):

        csum = csum + ord(*string*[len(*string*) - 1])

        csum = csum & 0xffffffff

    csum = (csum >> 16) + (csum & 0xffff)

    csum = csum + (csum >> 16)

    answer = ~csum

    answer = answer & 0xffff

    answer = answer >> 8 | (answer << 8 & 0xff00)

    return answer

def receiveOnePing(*mySocket*, *ID*, *timeout*, *destAddr*):

    timeLeft = *timeout*

    while 1:

        startedSelect = time.time()

        whatReady = select.select([*mySocket*], [], [], timeLeft)

        howLongInSelect = (time.time() - startedSelect)

        if whatReady[0] == []:  # Timeout

            return "Request timed out."

        print("What ready is: ", whatReady)

        timeReceived = time.time()

        recPacket, addr = *mySocket*.recvfrom(1024)

        print(addr)

        # Fill in start

        # Fetch the ICMP header from the IP packet

        icmpHeader = recPacket[20:28]

        icmpType, code, myChecksum, packetID, sequence = struct.unpack(

            "bbHHh", icmpHeader)

        # verify the ID of packet

        if icmpType != 8 and packetID == *ID*:

            bytesInDouble = struct.calcsize("d")

            timeSent = struct.unpack("d", recPacket[28:28 + bytesInDouble])[0]

            return timeReceived - timeSent

        timeLeft = timeLeft - howLongInSelect

        # Fill in end

        timeLeft = timeLeft - howLongInSelect

        if timeLeft <= 0:

            return "Request timed out."

def sendOnePing(*mySocket*, *destAddr*, *ID*):

    # Header is type (8), code (8), checksum (16), id (16), sequence (16)

    myChecksum = 0

    # Make a dummy header with a 0 checksum

    # struct -- Interpret strings as packed binary data

    header = struct.pack("bbHHh", ICMP\_ECHO\_REQUEST, 0, myChecksum, *ID*, 1)

    data = struct.pack("d", time.time())

    # Calculate the checksum on the data and the dummy header.

    myChecksum = checksum(str(header + data))

    # Get the right checksum, and put in the header

    if sys.platform == 'darwin':

        # Convert 16-bit integers from host to network  byte order

        myChecksum = htons(myChecksum) & 0xffff

    else:

        myChecksum = htons(myChecksum)

    header = struct.pack("bbHHh", ICMP\_ECHO\_REQUEST, 0, myChecksum, *ID*, 1)

    packet = header + data

    print(*mySocket*)

    # AF\_INET address must be tuple, not str

*mySocket*.sendto(packet, (*destAddr*, 1))

    print(*mySocket*)

    # Both LISTS and TUPLES consist of a number of objects

    # which can be referenced by their position number within the object.

def doOnePing(*destAddr*, *timeout*):

    icmp = getprotobyname("icmp")

    # SOCK\_RAW is a powerful socket type. For more details:   http://sock-raw.org/papers/sock\_raw

    mySocket = socket(AF\_INET, SOCK\_RAW, icmp)

    myID = os.getpid() & 0xFFFF  # Return the current process i

    sendOnePing(mySocket, *destAddr*, myID)

    delay = receiveOnePing(mySocket, myID, *timeout*, *destAddr*)

    mySocket.close()

    return delay

def ping(*host*, *timeout*=1):

    # timeout=1 means: If one second goes by without a reply from the server,

    # the client assumes that either the client's ping or the server's pong is lost

    dest = gethostbyname(*host*)

    print("Pinging " + dest + " using Python:")

    print("")

    # Send ping requests to a server separated by approximately one second

    while 1:

        delay = doOnePing(dest, *timeout*)

        print(delay)

        time.sleep(1)  # one second

        return delay

ping("google.com")

Screenshot:

Text

Description automatically generated