Socket Progrmming Assignment 4: ICMP

[INFO]

Student Name: Can Xu

Net id: cx461

[GENERL REVIEWS]

ICMP Socket Programming introduces the basics of socket programming in PING by ICMP request and reply.

We develop the client in Python 3.5.1.

There are many differences between Python 2.x and 3.x.

I have pointed out all the modification by red color in source code.

[SOCKET BASED RAW CODES]

```
from socket import *

import os

import sys

import struct

import time

import select

import binascii



ICMP_ECHO_REQUEST = 8



def checksum(str):

    csum = 0

    #countTo = (len(str) / 2) * 2

    countTo = (len(str) // 2) * 2      # // means div for integer in python 3.5



    count = 0

    while count < countTo:

        #thisVal = ord(str[count+1]) * 256 + ord(str[count])

        # in python 3.5, bytes[i] is an integer,

        # no need to use ord() to get the value of the specific char

        thisVal = str[count+1] * 256 + str[count]
        csum = csum + thisVal
        csum = csum & 0xffffffff
        count = count + 2

    if countTo < len(str):
        #csum = csum + ord(str[len(str) - 1])
        #python 3.5    below:
        csum = csum + str[len(str) - 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."



        timeReceived = time.time()

        recPacket, addr = mySocket.recvfrom(1024)



        #Fill in start

        #Fetch the ICMP header from the IP packet

        # fetch TTL

        ttl = recPacket[8]

        # fetch ICMP info

        pongType, pongCode, pongChecksum, pongID, pongSequence =
struct.unpack("bbHHh", recPacket[20:28])

        # display RTT in ms

        RTT = (timeReceived - struct.unpack("d", recPacket[28:36])[0]) * 1000



        result = "TTL: " + str(ttl) + "\n"

        result = result + "Type: " + str(pongType) + "\tCode: " + str(pongCode) +
"\tChecksum: " + str(pongChecksum) + "\tID: " + str(pongID) + "\tSequence: " +
str(pongSequence) + "\n"

        result = result + "RTT: %.2fms\n" % RTT         # to print RTT with 2 digit
        return result
        #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(header + data)
    # Get the right checksum, and put in the header
    if sys.platform == 'darwin':
        myChecksum = htons(myChecksum) & 0xffff
        #Convert 16-bit integers from host to network byte order.
    else:
        myChecksum = htons(myChecksum)


    header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, myChecksum, ID,
1)

    packet = header + data


    mySocket.sendto(packet, (destAddr, 1)) # AF_INET address must be tuple, not
str

    # 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 see: http://sock-
raw.org/papers/sock_raw


    #Fill in start

    #Create Socket here

    try:

        mySocket = socket(AF_INET, SOCK_RAW, icmp)

    except error as msg:

        print("Socket create error:", msg)
    #Fill in end

    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("127.0.0.1")
```

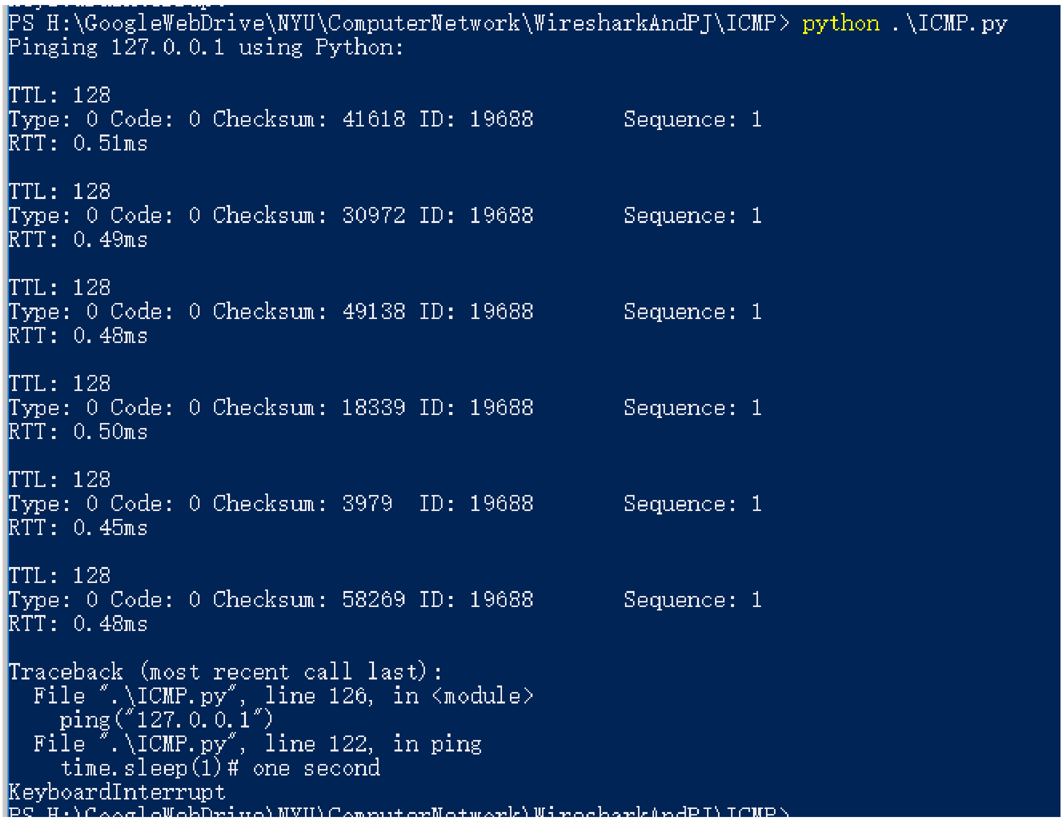
ping("www.google.com")

ping("www.poly.edu")

[KEY POINTS AND PROCESS]

We ping localhost to test the basic function

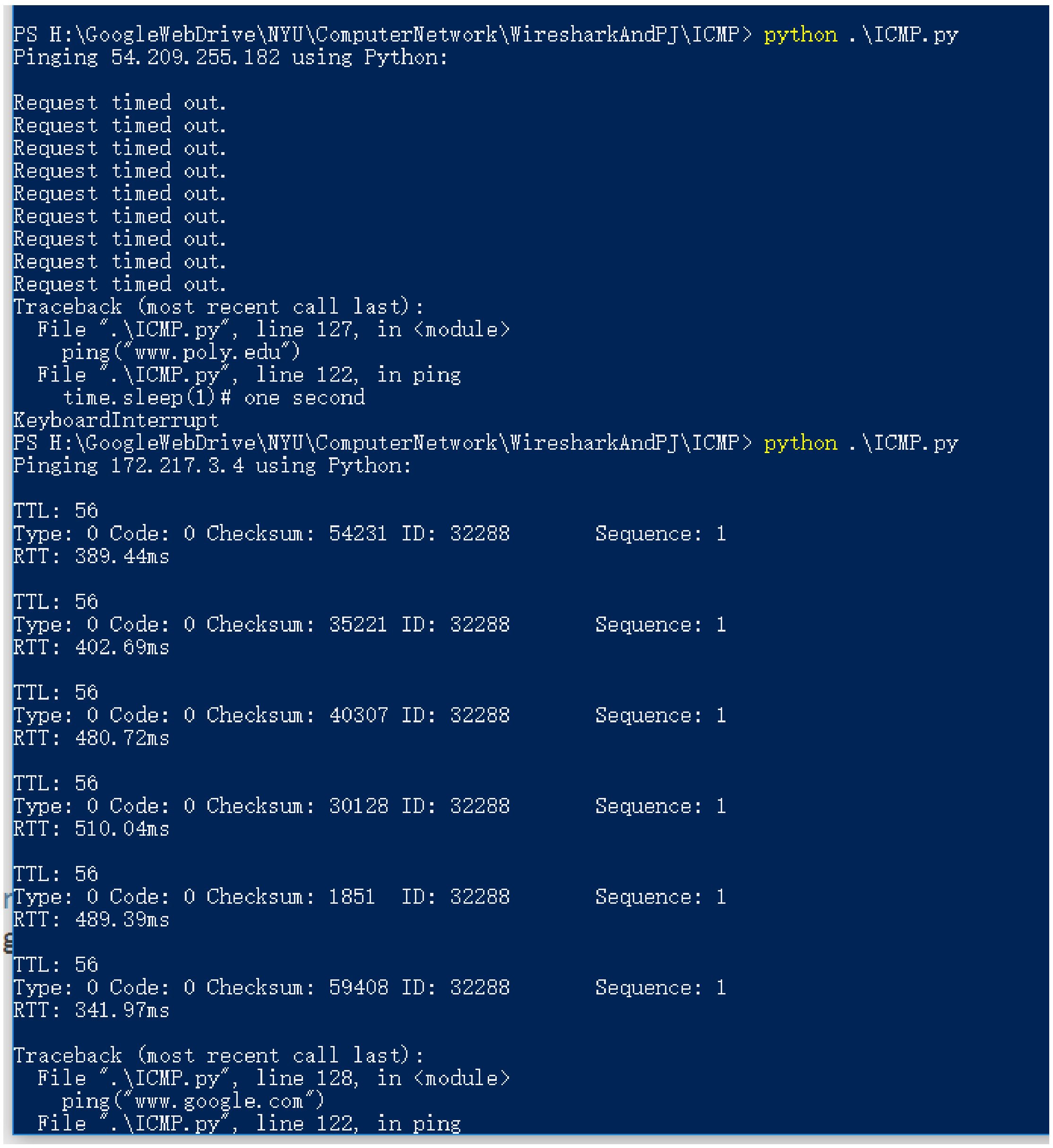
The content is below:



```
PS H:\GoogleWebDrive\NYU\ComputerNetwork\WiresharkAndPJ\ICMP> python .\ICMP.py
Pinging 127.0.0.1 using Python:

TTL: 128
Type: 0 Code: 0 Checksum: 41618 ID: 19688       Sequence: 1
RTT: 0.51ms

TTL: 128
Type: 0 Code: 0 Checksum: 30972 ID: 19688       Sequence: 1
RTT: 0.49ms

TTL: 128
Type: 0 Code: 0 Checksum: 49138 ID: 19688       Sequence: 1
RTT: 0.48ms

TTL: 128
Type: 0 Code: 0 Checksum: 18339 ID: 19688       Sequence: 1
RTT: 0.50ms

TTL: 128
Type: 0 Code: 0 Checksum: 3979  ID: 19688       Sequence: 1
RTT: 0.45ms

TTL: 128
Type: 0 Code: 0 Checksum: 58269 ID: 19688       Sequence: 1
RTT: 0.48ms

Traceback (most recent call last):
  File ".\ICMP.py", line 126, in <module>
    ping("127.0.0.1")
  File ".\ICMP.py", line 122, in ping
    time.sleep(1)# one second
KeyboardInterrupt
```

Then I ping the "www.google.com" and www.poly.edu .

I found the poly.edu is unreachable because the ping service is not ready in that server.



```
PS H:\GoogleWebDrive\NYU\ComputerNetwork\WiresharkAndPJ\ICMP> python .\ICMP.py
Pinging 54.209.255.182 using Python:

Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Traceback (most recent call last):
  File ".\ICMP.py", line 127, in <module>
    ping("www.poly.edu")
  File ".\ICMP.py", line 122, in ping
    time.sleep(1)# one second
KeyboardInterrupt
PS H:\GoogleWebDrive\NYU\ComputerNetwork\WiresharkAndPJ\ICMP> python .\ICMP.py
Pinging 172.217.3.4 using Python:

TTL: 56
Type: 0 Code: 0 Checksum: 54231 ID: 32288       Sequence: 1
RTT: 389.44ms

TTL: 56
Type: 0 Code: 0 Checksum: 35221 ID: 32288       Sequence: 1
RTT: 402.69ms

TTL: 56
Type: 0 Code: 0 Checksum: 40307 ID: 32288       Sequence: 1
RTT: 480.72ms

TTL: 56
Type: 0 Code: 0 Checksum: 30128 ID: 32288       Sequence: 1
RTT: 510.04ms

TTL: 56
Type: 0 Code: 0 Checksum: 1851  ID: 32288       Sequence: 1
RTT: 489.39ms

TTL: 56
Type: 0 Code: 0 Checksum: 59408 ID: 32288       Sequence: 1
RTT: 341.97ms

Traceback (most recent call last):
  File ".\ICMP.py", line 128, in <module>
    ping("www.google.com")
  File ".\ICMP.py", line 122, in ping
```

Socket Progrmming Assignment 5: Traceroute

[GENERL REVIEWS]

Traceroute Socket Programming introduces the basics of socket programming in Traceroute by ICMP request and reply.

We will learn how to implement a traceroute application using ICMP request and reply messages.

We develop the client in Python 3.5.1.

There are many differences between Python 2.x and 3.x.

I have pointed out all the modification by red color in source code.

[SOCKET BASED RAW CODES]

```
from socket import *

import os

import sys

import struct

import time

import select

import binascii


ICMP_ECHO_REQUEST = 8

MAX_HOPS = 30
```

```
TIMEOUT = 2.0

TRIES = 2



# The packet that we shall send to each router along the path is the ICMP echo

# request packet, which is exactly what we had used in the ICMP ping exercise.

# We shall use the same packet that we built in the Ping exercise



def checksum(str):

# In this function we make the checksum of our packet

# hint: see icmpPing lab

    csum = 0

    countTo = (len(str) // 2) * 2    #python 3.x

    count = 0

    while count < countTo:

        thisVal = str[count+1] * 256 + str[count]

        csum = csum + thisVal

        csum = csum & 0xffffffff

        count = count + 2



    if countTo < len(str):

        csum = csum + str[len(str) - 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 build_packet():

# In the sendOnePing() method of the ICMP Ping exercise, firstly the header of our

# packet to be sent was made, secondly the checksum was appended to the header and

# then finally the complete packet was sent to the destination.

# Make the header in a similar way to the ping exercise.

# Append checksum to the header.

# Don't send the packet yet , just return the final packet in this function.

# So the function ending should look like this packet = header + data return packet


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

    # 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(header + data)

    # Get the right checksum, and put in the header

    if sys.platform == 'darwin':

        myChecksum = htons(myChecksum) & 0xffff

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

    else:

        myChecksum = htons(myChecksum)



    header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, myChecksum, ID,
1)

    packet = header + data



    return packet



def get_route(hostname):

    #timeLeft = TIMEOUT          # Is this line in the wrong place? I changed it to
three lines below......
```

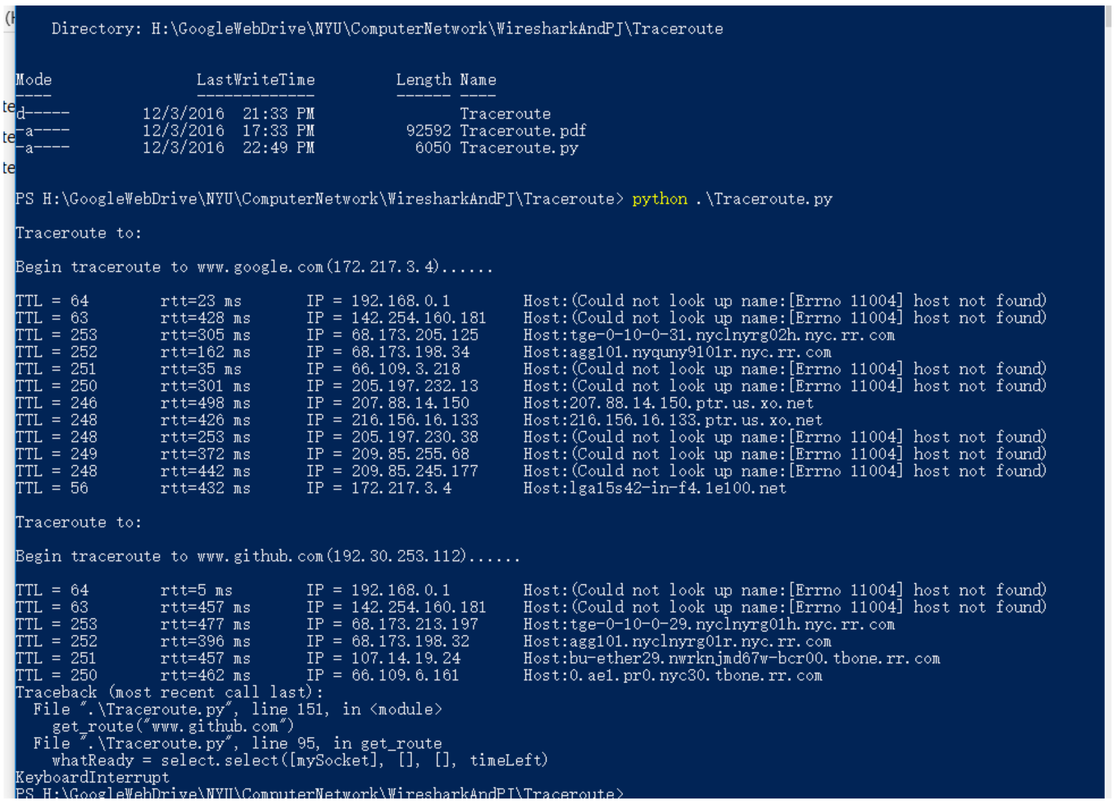
```
    print("Begin traceroute to " + hostname + "(" + gethostbyname(hostname) +
")......\n")

    for ttl in range(1,MAX_HOPS):
        for tries in range(TRIES):
            timeLeft = TIMEOUT
            destAddr = gethostbyname(hostname)
            #Fill in start
            # Make a raw socket named mySocket
            icmp = getprotobyname("icmp")
            try:
                mySocket = socket(AF_INET, SOCK_RAW, icmp)
            except error as msg:
                print("Socket create error:", msg)
            #Fill in end
            mySocket.setsockopt(IPPROTO_IP, IP_TTL, struct.pack('I', ttl))
            mySocket.settimeout(TIMEOUT)
            try:
                d = build_packet()
                mySocket.sendto(d, (hostname, 0))
                t = time.time()
                startedSelect = time.time()
                whatReady = select.select([mySocket], [], [], timeLeft)
                howLongInSelect = (time.time() - startedSelect)
                if whatReady[0] == []: # Timeout
                    print("\t*\t\t*\t\t*\t\tRequest timed out.")
                recvPacket, addr = mySocket.recvfrom(1024)
                timeReceived = time.time()



                timeLeft = timeLeft - howLongInSelect
                if timeLeft <= 0:
                    print("\t*\t*\t*\Request timed out.")
            except timeout:
                continue
            else:
                #Fill in start
                # Fetch the icmp type from the IP packet


                # fetch TTL
                ttl = recvPacket[8]
                # fetch ICMP info
                type, pongCode, pongChecksum, pongID, pongSequence =
struct.unpack("bbHHh", recvPacket[20:28])
```

```
                # display RTT in ms
                RTT = (timeReceived - struct.unpack("d", recvPacket[28:36])[0])
* 1000

                # try to get hostname of each router in the path
                try:
                    routerHostname = gethostbyaddr(addr[0])[0]
                except herror as emsg:
                    routerHostname = "(Could not look up name:" + str(emsg)
+")"

                #Fill in end
                if type == 11:
                    bytes = struct.calcsize("d")
                    timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
                    print("TTL = %d\trtt=%.0f ms\tIP = %s\tHost:%s" %(ttl,
(timeReceived -t)*1000, addr[0], routerHostname))
                elif type == 3:
                    bytes = struct.calcsize("d")
                    timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
                    print("TTL = %d\trtt=%.0f ms\tIP = %s\tHost:%s" %(ttl,
(timeReceived-t)*1000, addr[0], routerHostname))
                elif type == 0:
                    bytes = struct.calcsize("d")

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

                    print("TTL = %d\trtt=%.0f ms\tIP = %s\tHost:%s" %(ttl,
(timeReceived - timeSent)*1000, addr[0], routerHostname))

                    return

                else:

                    print("error")

                break

            finally:

                mySocket.close()




# traceroute to different host

print("\nTraceroute to: \n")

get_route("www.google.com")



print("\nTraceroute to: \n")

get_route("www.github.com")



print("\nTraceroute to: \n")

get_route("www.poly.edu")
```

[Results and Analysis]

We connect to Google.com and github.com

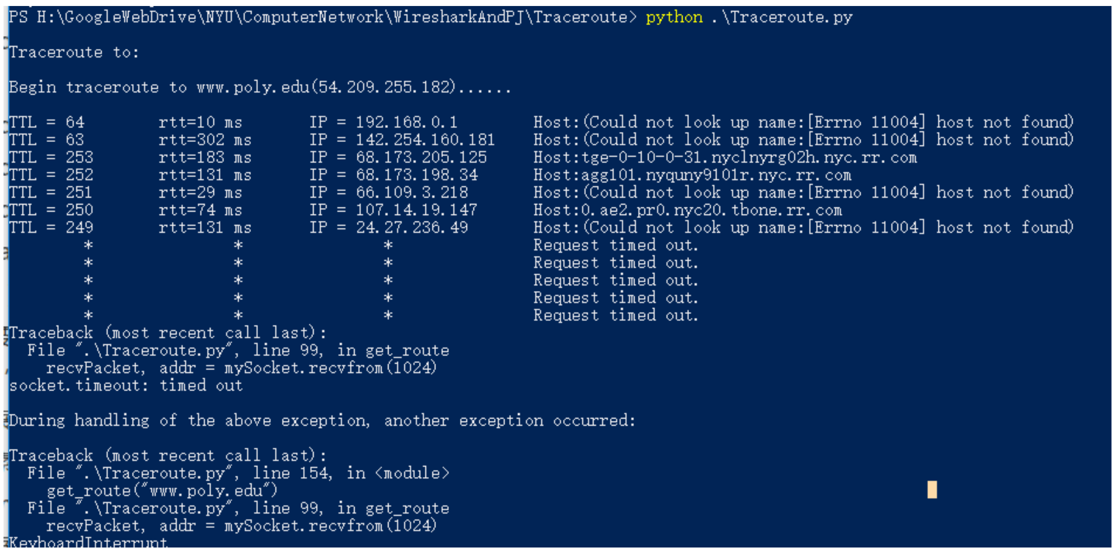
And it works with details of each hop.



```
    Directory: H:\GoogleWebDrive\NYU\ComputerNetwork\WiresharkAndPJ\Traceroute


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        12/3/2016  21:33 PM                Traceroute
-a----        12/3/2016  17:33 PM          92592 Traceroute.pdf
-a----        12/3/2016  22:49 PM           6050 Traceroute.py


PS H:\GoogleWebDrive\NYU\ComputerNetwork\WiresharkAndPJ\Traceroute> python .\Traceroute.py

Traceroute to:

Begin traceroute to www.google.com(172.217.3.4)......

TTL = 64        rtt=23 ms       IP = 192.168.0.1        Host:(Could not look up name:[Errno 11004] host not found)
TTL = 63        rtt=428 ms      IP = 142.254.160.181    Host:(Could not look up name:[Errno 11004] host not found)
TTL = 253       rtt=305 ms      IP = 68.173.205.125     Host:tge-0-10-0-31.nyclnyrg02h.nyc.rr.com
TTL = 252       rtt=162 ms      IP = 68.173.198.34      Host:agg101.nyquny9101r.nyc.rr.com
TTL = 251       rtt=35 ms       IP = 66.109.3.218       Host:(Could not look up name:[Errno 11004] host not found)
TTL = 250       rtt=301 ms      IP = 205.197.232.13     Host:(Could not look up name:[Errno 11004] host not found)
TTL = 246       rtt=498 ms      IP = 207.88.14.150      Host:207.88.14.150.ptr.us.xo.net
TTL = 248       rtt=426 ms      IP = 216.156.16.133     Host:216.156.16.133.ptr.us.xo.net
TTL = 248       rtt=253 ms      IP = 205.197.230.38     Host:(Could not look up name:[Errno 11004] host not found)
TTL = 249       rtt=372 ms      IP = 209.85.255.68      Host:(Could not look up name:[Errno 11004] host not found)
TTL = 248       rtt=442 ms      IP = 209.85.245.177     Host:(Could not look up name:[Errno 11004] host not found)
TTL = 56        rtt=432 ms      IP = 172.217.3.4        Host:lga15s42-in-f4.1e100.net

Traceroute to:

Begin traceroute to www.github.com(192.30.253.112)......

TTL = 64        rtt=5 ms        IP = 192.168.0.1        Host:(Could not look up name:[Errno 11004] host not found)
TTL = 63        rtt=457 ms      IP = 142.254.160.181    Host:(Could not look up name:[Errno 11004] host not found)
TTL = 253       rtt=477 ms      IP = 68.173.213.197     Host:tge-0-10-0-29.nyclnyrg01h.nyc.rr.com
TTL = 252       rtt=396 ms      IP = 68.173.198.32      Host:agg101.nyclnyrg01r.nyc.rr.com
TTL = 251       rtt=457 ms      IP = 107.14.19.24       Host:bu-ether29.nwrknjmd67w-bcr00.tbone.rr.com
TTL = 250       rtt=462 ms      IP = 66.109.6.161       Host:0.ae1.pr0.nyc30.tbone.rr.com
Traceback (most recent call last):
  File ".\Traceroute.py", line 151, in <module>
    get_route("www.github.com")
  File ".\Traceroute.py", line 95, in get_route
    whatReady = select.select([mySocket], [], [], timeLeft)
KeyboardInterrupt
PS H:\GoogleWebDrive\NYU\ComputerNetwork\WiresharkAndPJ\Traceroute>
```

The reason for LAB4 ping request timeout is clear.

The host 54.209.255.182 didn’t reply any ICMP reply from us.



```
PS H:\GoogleWebDrive\NYU\ComputerNetwork\WiresharkAndPJ\Traceroute> python .\Traceroute.py

Traceroute to:

Begin traceroute to www.poly.edu(54.209.255.182)......

TTL = 64        rtt=10 ms       IP = 192.168.0.1        Host:(Could not look up name:[Errno 11004] host not found)
TTL = 63        rtt=302 ms      IP = 142.254.160.181    Host:(Could not look up name:[Errno 11004] host not found)
TTL = 253       rtt=183 ms      IP = 68.173.205.125     Host:tge-0-10-0-31.nyclnyrg02h.nyc.rr.com
TTL = 252       rtt=131 ms      IP = 68.173.198.34      Host:agg101.nyquny9101r.nyc.rr.com
TTL = 251       rtt=29 ms       IP = 66.109.3.218       Host:(Could not look up name:[Errno 11004] host not found)
TTL = 250       rtt=74 ms       IP = 107.14.19.147      Host:0.ae2.pr0.nyc20.tbone.rr.com
TTL = 249       rtt=131 ms      IP = 24.27.236.49       Host:(Could not look up name:[Errno 11004] host not found)
        *               *               *               Request timed out.
        *               *               *               Request timed out.
        *               *               *               Request timed out.
        *               *               *               Request timed out.
        *               *               *               Request timed out.
Traceback (most recent call last):
  File ".\Traceroute.py", line 99, in get_route
    recvPacket, addr = mySocket.recvfrom(1024)
socket.timeout: timed out

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File ".\Traceroute.py", line 154, in <module>
    get_route("www.poly.edu")
  File ".\Traceroute.py", line 99, in get_route
    recvPacket, addr = mySocket.recvfrom(1024)
KeyboardInterrupt
```