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(string):

# In this function we make the checksum of our packet

# hint: see icmpPing lab

csum = 0

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

count = 0

while count < countTo:

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

csum = csum + thisVal

csum = csum & 0xffffffff

count = count + 2

if countTo < len(string):

csum = csum + ord(str[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 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

# get TTL

ttl = recvPacket[8]

# get ICMP info

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

# get 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 four different host

get\_route("www.baidu.com")

print("Traceroute Finished!\n\n\n\n\n\n")

get\_route("www.google.com")

print("Traceroute Finished!\n\n\n\n\n\n")

get\_route("www.tsinghua.edu.cn")

print("Traceroute Finished!\n\n\n\n\n\n")

get\_route("www.github.com")

print("Traceroute Finished!\n\n\n\n\n\n")





