**Name: Archana Purushothama**

**Id: N15100298**

**Computer Networking Assignment**

**SOCKET PROGRAMMING : ICMP Traceroute**

1. Python Code for the ICMP Traceroute

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

sum = 0

count\_to = (len(str) / 2) \* 2

for count in xrange(0, count\_to, 2):

this = (ord(str[count + 1]) << 8) + ord(str[count]) # <<8 = \*256

sum = sum + this

sum = sum & 0xffffffff

#Odd bit

if count\_to < len(str):

sum = sum + ord(str[len(str) - 1])

sum = sum & 0xffffffff

#Handling the carry bit

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

sum = sum + (sum >> 16)

#complement

chksum = ~sum & 0xffff

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

return chksum

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.

myChksum=0

myId = os.getpid()

#Make a dummy header with checksum 0.

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

bytes = struct.calcsize("d")

data = (64- bytes) \* "Q"

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

#CAlculate the checksum on the dummy header and data

myChksum = checksum(header + data)

#Append the calculated checksum into the header

header = struct.pack("bbHHh", ICMP\_ECHO\_REQUEST, 0, htons(myChksum), myId, 1)

packet = header + data

return packet

def get\_route(hostname):

timeLeft = TIMEOUT

for ttl in xrange(1,MAX\_HOPS):

for tries in xrange(TRIES):

destAddr = gethostbyname(hostname)

# Make a raw socket named mySocket

mySocket = socket(AF\_INET, SOCK\_RAW, IPPROTO\_ICMP)

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 " \* \* \* Request timed out."

recvPacket, addr = mySocket.recvfrom(1024)

timeReceived = time.time()

timeLeft = timeLeft - howLongInSelect

if timeLeft <= 0:

print " \* \* \* Request timed out."

except timeout:

continue

else:

# Fetch the icmp type from the IP packet

icmpHeader = recvPacket[20:28]

type, code, checksum, packet\_id, sequence = struct.unpack("bbHHh", icmpHeader)

#Name of each intermediate router

try:

name = gethostbyaddr(addr[0])

except herror:

name = " "

if type == 11:

bytes = struct.calcsize("d")

timeSent = struct.unpack("d", recvPacket[28:28 +

bytes])[0]

print " %d rtt=%.0f ms %s %s" %(ttl,

(timeReceived -t)\*1000, addr[0], name[0])

elif type == 3:

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

print " %d rtt=%.0f ms %s %s" %(ttl,

(timeReceived-t)\*1000, addr[0], name[0])

elif type == 0:

bytes = struct.calcsize("d")

timeSent = struct.unpack("d", recvPacket[28:28 +

bytes])[0]

print " %d rtt=%.0f ms %s %s" %(ttl,(timeReceived - timeSent)\*1000, addr[0], name[0])

return

else:

print "error"

break

finally:

mySocket.close()

print "Number of Tries - " + str(TRIES)

print "Maximum hop count - " + str(MAX\_HOPS)

print "Timeout value - " + str(TIMEOUT) + "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

print "For target host1: google.com"

get\_route("google.com")

print "\nFor target host2: twc.com"

get\_route("twc.com")

print "\nFor target host3: ubuntuforums.com"

get\_route("ubuntuforums.com")

print "\nFor target host4: www.gmail.com"

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

1. OUTPUT

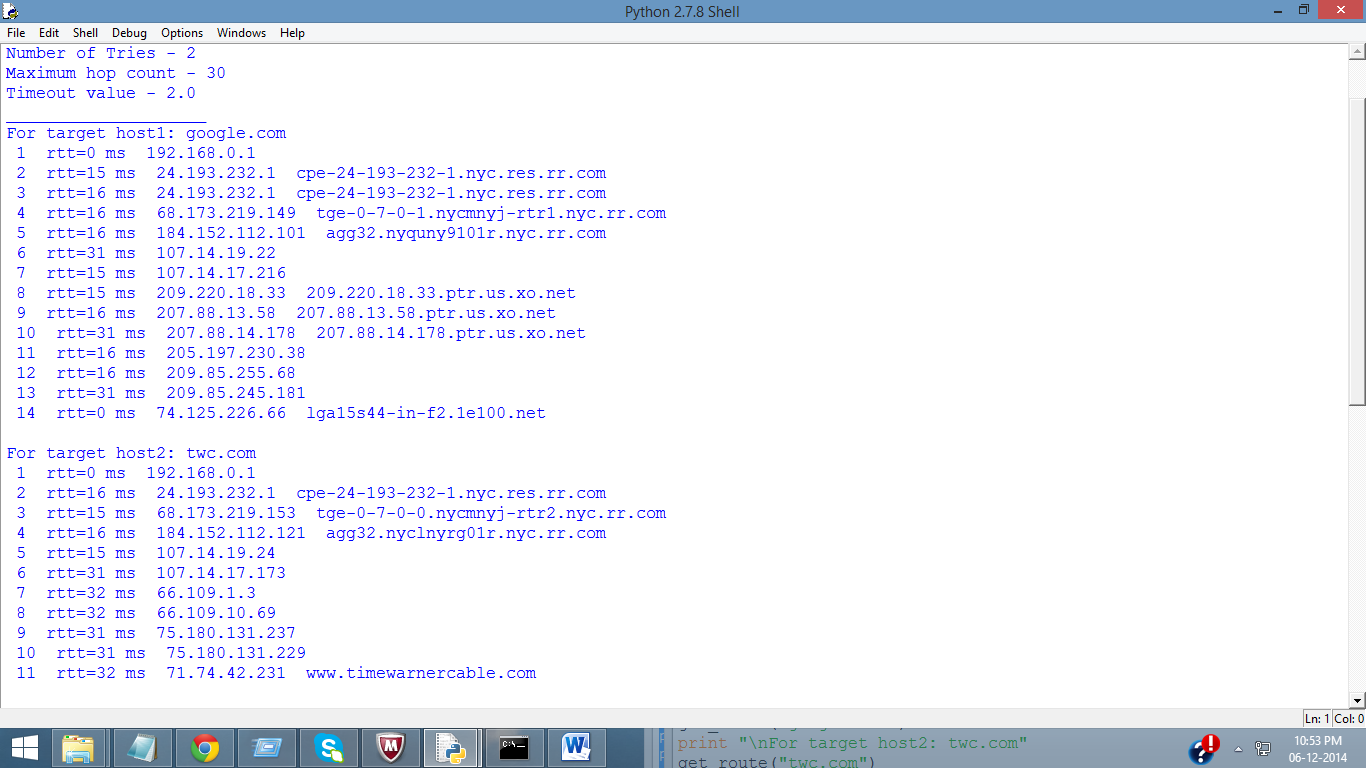


Figure 1: OutPut Screen of ICMP\_Tracroute on target hosts 'google.com' and 'twc.com'

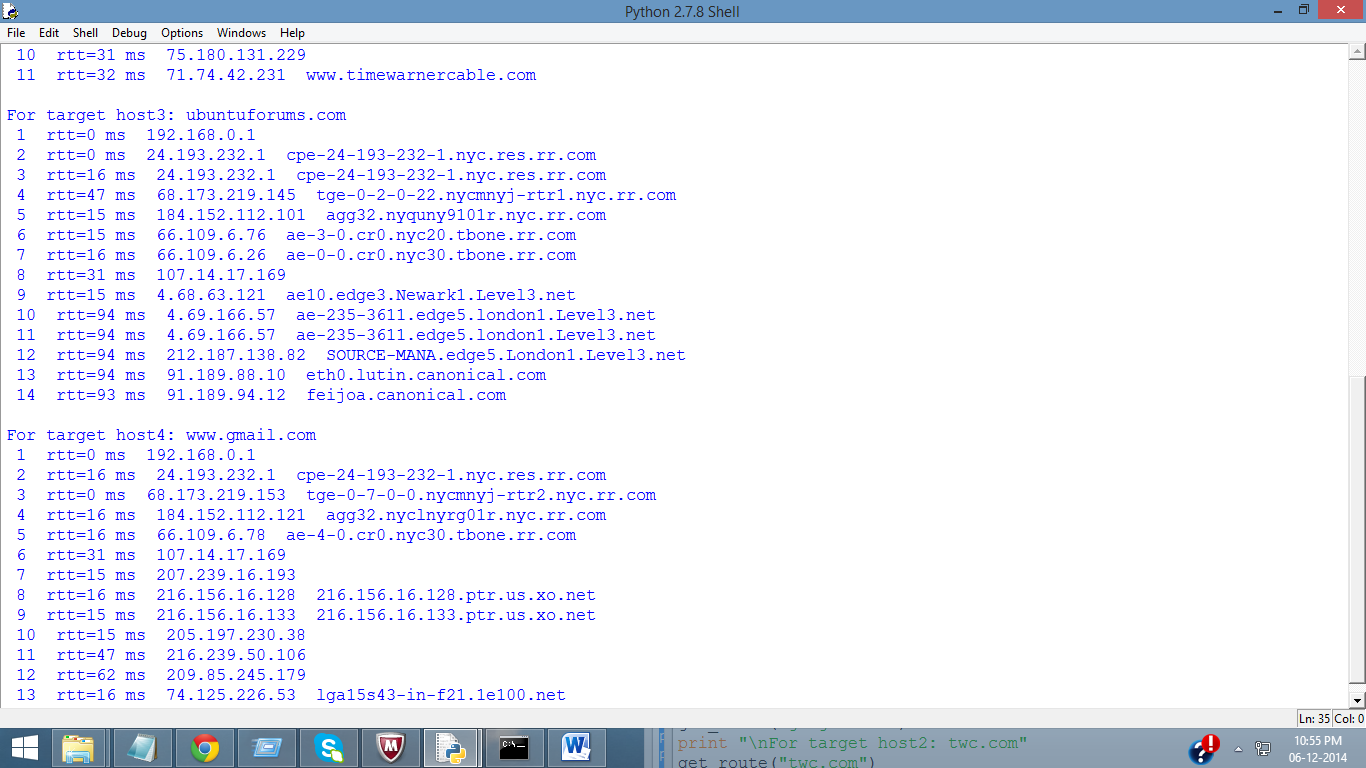


Figure 2: Output window when ICMP\_Traceroute is executed for target hosts 'ubuntuforums.com' and 'www.gmail.com'.

1. Optional Exercises

Currently the application only prints out a list of ip addresses of all the routers along the path from source to the destination. Try using the gethostbyname method to print out the names of each intermediate route along the route.

Implemented in the above solution.