Christopher Desir

cd2902@nyu.edu

ICMP Traceroute

```
from socket import *
import socket
import os
import sys
import struct
import time
import select
import binascii
# use mac or linux terminal to run code using sudo Python testing.py
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):
    csum = 0
    countTo = (len(str) / 2) * 2
    count = 0

    while count < countTo:
        thisVal = ord(str[count+1]) * 256 + ord(str[count])
```

```
        csum = csum + thisVal
        csum = csum & 0xffffffffL
        count = count + 2

    if countTo < len(str):
        csum = csum + ord(str[len(str) - 1])
        csum = csum & 0xffffffffL

    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.

    myChecksum = 0
    myID = os.getpid() & 0xFFFF
    header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, myChecksum, myID, 1)
    data = struct.pack("d", time.time())
    myChecksum = checksum(header + data)
    if sys.platform == 'darwin':
        myChecksum = socket.htons(myChecksum) & 0xffff
    else:
        myChecksum = htons(myChecksum)

    header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, myChecksum, 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 = socket.gethostbyname(hostname)

            #Fill in start
            # Make a raw socket named mySocket
            icmp = socket.getprotobyname("icmp")
            mySocket = socket.socket(socket.AF_INET, socket.SOCK_RAW, icmp)
            #Fill in end

            mySocket.setsockopt(socket.IPPROTO_IP, socket.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)
                print addr
                timeReceived = time.time()
                timeLeft = timeLeft - howLongInSelect
                if timeLeft <= 0:
                    print "*    *    * Request timed out."
```

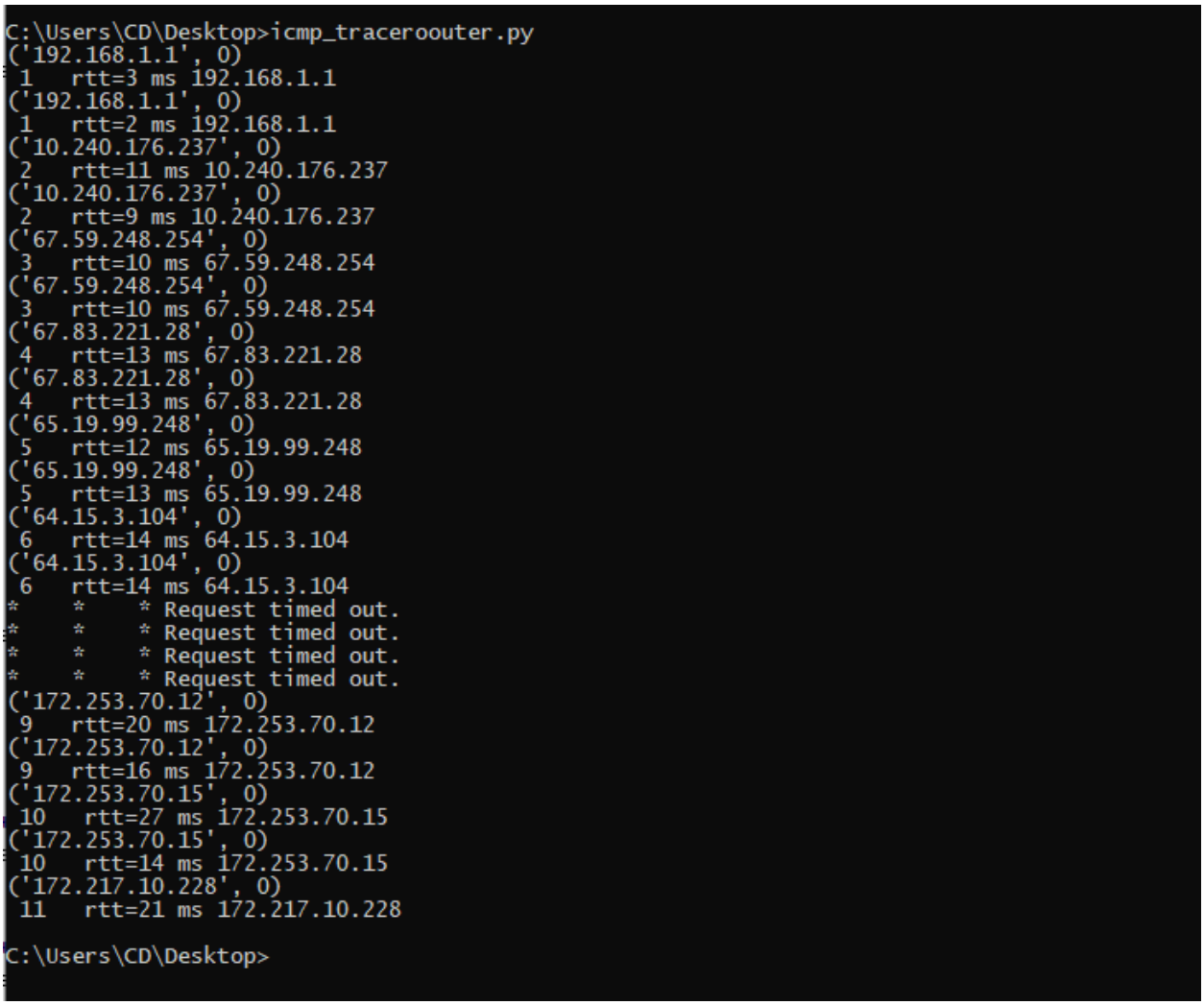
```
        except socket.timeout:
            continue
        else:

            #Fill in start
            # Fetch the icmp type from the IP packet
            icmpHeader = recvPacket[20:28]
            request_type, code, checksum, packetID, sequence = struct.unpack("bbHHh", icmpHeader)

            #Fill in end

            if request_type == 11:
                bytes = struct.calcsize("d")
                timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
                print " %d   rtt=%.0f ms %s" % (ttl,(timeReceived -t)*1000, addr[0])
            elif request_type == 3:
                bytes = struct.calcsize("d")
                timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
                print " %d   rtt=%.0f ms %s" % (ttl,(timeReceived -t)*1000, addr[0])
            elif request_type == 0:
                bytes = struct.calcsize("d")
                timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
                print " %d   rtt=%.0f ms %s" % (ttl,(timeReceived -timeSent)*1000, addr[0])
                return
            else:
                print "error"
                break
        finally:
            mySocket.close()


get_route("www.google.com")
```



ICMP Pinger Assignment

```
import socket
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."

    timeReceived = time.time()
    recPacket, addr = mySocket.recvfrom(1024)

    #Fill in start
    icmphead = recPacket[20:28]
    type, code, checksum, packetID, sequence = struct.unpack("bbHHh", icmphead)
    if packetID == ID:
      doubleBytes = struct.calcsize("d")
      timeSent = struct.unpack("d", recPacket[28:28 + doubleBytes])[0]
      return timeReceived - timeSent
    # Fill in end

    timeLeft = timeLeft - howLongInSelect
    if timeLeft <= 0:
      return "Request timed out."


def sendOnePing(mySocket, destAddr, ID):

  myChecksum = 0
  header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, myChecksum, ID, 1)
  data = struct.pack("d", time.time())
  myChecksum = checksum(header + data)
  if sys.platform == 'darwin':
    myChecksum = socket.htons(myChecksum) & 0xffff
  else:
    myChecksum = socket.htons(myChecksum)
```

```
  header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, myChecksum, ID, 1)
  packet = header + data
  mySocket.sendto(packet, (destAddr, 1))

def doOnePing(destAddr, timeout):
    icmp = socket.getprotobyname("icmp")
    # SOCK_RAW is a powerful socket type. For more details: http://sockraw.org/papers/sock_raw
    mySocket = socket.socket(socket.AF_INET, socket.SOCK_RAW, icmp)
    #prit "2"
    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 = socket.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)
    return delay


if __name__ == '__main__':
```

ping("www.google.com")

```
C:\Users\CD\Desktop>icmp_pinger.py
Pinging 172.217.10.228 using Python:

0.0160000324249
0.0139999389648
0.0120000839233
0.0119998455048
0.0120000839233
0.0119998455048
0.0139999389648
0.0120000839233
0.0139999389648
0.0160000324249
0.0119998455048
0.0120000839233
0.0210001468658
0.0130000114441
0.0120000839233
0.0110001564026
0.0130000114441
0.0179998874664
0.0130000114441
0.0120000839233
0.0199999809265
0.0139999389648
0.0119998455048
```