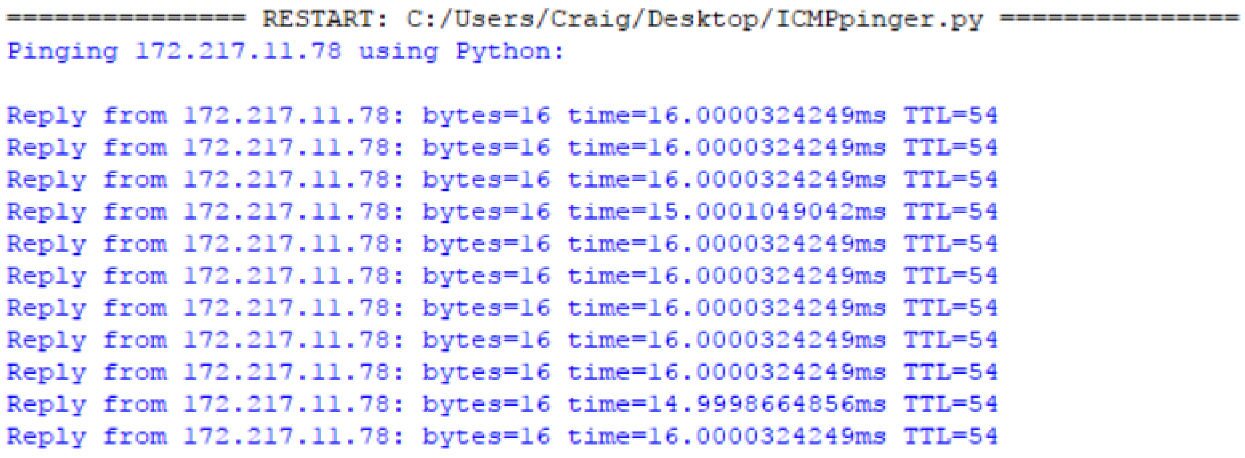
Socket Programming Assignment 4 – ICMPpinger

Source Code:

```
#Skeleton Code taken from textbook
#Code additions made by Craig Hulsebus 12/01/2017
#CSC138 – ICMPpinger
import os
import sys
import struct
import time
import select
import socket
import binascii

ICMP_ECHO_REQUEST = 8

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 receiveOnePing(mySocket, ID, timeout, destAddr):
   global rtt_min, rtt_max, rtt_sum, rtt_cnt
   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
        type, code, checksum, id, seq = struct.unpack('bbHHh', recPacket[20:28])
        if type != 0:
            return 'expected type=0, but got {}'.format(type)
        if code != 0:
            return 'expected code=0, but got {}'.format(code)
        if ID != id:
            return 'expected id={}, but got {}'.format(ID, id)
        send_time,  = struct.unpack('d', recPacket[28:])

        rtt = (timeReceived - send_time) * 1000
        rtt_cnt += 1
        rtt_sum += rtt
        rtt_min = min(rtt_min, rtt)
        rtt_max = max(rtt_max, rtt)

        ip_header = struct.unpack('!BBHHHBBH4s4s' , recPacket[:20])
        ttl = ip_header[5]
        saddr = socket.inet_ntoa(ip_header[8])
        length = len(recPacket) - 20

        return 'Reply from {}: bytes={} time={}ms TTL={}'.format(saddr, length, rtt, ttl)
        #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 = socket.htons(myChecksum) & 0xffff
        #Convert 16-bit integers from host to network byte order.
    else:
        myChecksum = socket.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 = socket.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
    mySocket = socket.socket(socket.AF_INET, socket.SOCK_RAW, icmp)
    #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):
    global rtt_min, rtt_max, rtt_sum, rtt_cnt
    rtt_min = float('+inf')
    rtt_max = float('-inf')
    rtt_sum = 0
    rtt_cnt = 0
    cnt = 0
    #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 True:
        cnt += 1
        print doOnePing(dest, timeout)
        time.sleep(1)
    return delay

ping("172.217.11.78")
```

Confirmation Response:



```
================ RESTART: C:/Users/Craig/Desktop/ICMPpinger.py ================
Pinging 172.217.11.78 using Python:

Reply from 172.217.11.78: bytes=16 time=16.0000324249ms TTL=54
Reply from 172.217.11.78: bytes=16 time=16.0000324249ms TTL=54
Reply from 172.217.11.78: bytes=16 time=16.0000324249ms TTL=54
Reply from 172.217.11.78: bytes=16 time=15.0001049042ms TTL=54
Reply from 172.217.11.78: bytes=16 time=16.0000324249ms TTL=54
Reply from 172.217.11.78: bytes=16 time=16.0000324249ms TTL=54
Reply from 172.217.11.78: bytes=16 time=16.0000324249ms TTL=54
Reply from 172.217.11.78: bytes=16 time=16.0000324249ms TTL=54
Reply from 172.217.11.78: bytes=16 time=16.0000324249ms TTL=54
Reply from 172.217.11.78: bytes=16 time=14.9998664856ms TTL=54
Reply from 172.217.11.78: bytes=16 time=16.0000324249ms TTL=54
```