Anthony Chavez

Professor Sun

Socket Programming 4

Source Code:

```
from socket import *
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)

		#Feth the ICMP header from the IP packet
		type, code, revChecksum, id, revSequence  = struct.unpack('bbHHh',
```

```
                if ID == id:
                        timeData, = struct.unpack('d', recPacket[28:])
                        RTT = (timeReceived - timeData) * 1000  # Calculate RTT and convert to ms
                        ipHeader = struct.unpack('!BBHHHBBH4s4s' , recPacket[:20])
                        TTL = ipHeader[5]          # Get TTL from packet header
                        srcAddr = inet_ntop(AF_INET, ipHeader[8])        # Get src address from
packet header
                        length = len(recPacket)
                        return 'Reply from {}: bytes={} time={:.3f}ms TTL={}'.format(srcAddr, length,
RTT, TTL)
                else:
                        return "ID does not match"

                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(str(header + data))

        # Get the right checksum, and put in the header
        if sys.platform == 'darwin':
                # Convert 16-bit integers from host to network byte order
                myChecksum = htons(myChecksum) & 0xffff
        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:
http://sockraw.org/papers/sock_raw
        mySocket = socket(AF_INET, SOCK_RAW, icmp)
        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("google.com")
```

Here we have the source code. I developed my own Ping application in Python using the Skeleton Code provided from the course book.

Expected output:

```
D:\Workspace\csc138\icmpping>python icmpping.py
Pinging 172.217.11.78 using Python:

Reply from 172.217.11.78: bytes=36 time=15.0001055ms TTL=51
Reply from 172.217.11.78: bytes=36 time=13.0000115ms TTL=51
Reply from 172.217.11.78: bytes=36 time=13.0000115ms TTL=51
Reply from 172.217.11.78: bytes=36 time=13.0000115ms TTL=51
Reply from 172.217.11.78: bytes=36 time=13.9999395ms TTL=51
Reply from 172.217.11.78: bytes=36 time=13.0000115ms TTL=51
Reply from 172.217.11.78: bytes=36 time=13.0000115ms TTL=51
Reply from 172.217.11.78: bytes=36 time=19.9999815ms TTL=51
```

My Outputs:

Run 1 (Google.com):

```
wolf@Markus:/mnt/f/Users/Mushooshu/Desktop/school/3 junior year/CPE 138/Labs/Lab6$ sudo python2.7 ICMPpinger.py
[sudo] password for wolf:
Pinging 216.58.195.78 using Python:

Reply from 216.58.195.78: bytes=36 time=20.608ms TTL=55
Reply from 216.58.195.78: bytes=36 time=19.200ms TTL=55
Reply from 216.58.195.78: bytes=36 time=17.573ms TTL=55
Reply from 216.58.195.78: bytes=36 time=26.471ms TTL=55
Reply from 216.58.195.78: bytes=36 time=22.785ms TTL=55
Reply from 216.58.195.78: bytes=36 time=28.692ms TTL=55
```

Run 2 (local host):

```
wolf@Markus:/mnt/f/Users/Mushooshu/Desktop/school/3 junior year/CPE 138/Labs/Lab6$ sudo python2.7 ICMPpinger.py
Pinging 127.0.0.1 using Python:

Reply from 127.0.0.1: bytes=36 time=0.173ms TTL=128
Reply from 127.0.0.1: bytes=36 time=0.353ms TTL=128
Reply from 127.0.0.1: bytes=36 time=0.345ms TTL=128
Reply from 127.0.0.1: bytes=36 time=0.355ms TTL=128
Reply from 127.0.0.1: bytes=36 time=0.363ms TTL=128
Reply from 127.0.0.1: bytes=36 time=0.344ms TTL=128
```

Run 3 (Google.com):

```
wolf@Markus:/mnt/f/Users/Mushooshu/Desktop/school/3 junior year/CPE 138/Labs/Lab6$ sudo python2.7 ICMPpinger.py
Pinging 142.250.72.206 using Python:

Reply from 142.250.72.206: bytes=36 time=24.455ms TTL=55
Reply from 142.250.72.206: bytes=36 time=17.823ms TTL=55
Reply from 142.250.72.206: bytes=36 time=17.298ms TTL=55
Reply from 142.250.72.206: bytes=36 time=25.875ms TTL=55
Reply from 142.250.72.206: bytes=36 time=23.340ms TTL=55
Reply from 142.250.72.206: bytes=36 time=22.864ms TTL=55
```

Google.com was pinged twice to ensure IP was being displayed correctly. After TTL value is met, the IP of Google will change. As for the local host, the IP will stay the same.