CN Assignment1

Name: Tarala Trilokesh
Class: ECE B
Roll No.:- B190367EC

Procedure:

1. Create a raw socket for sending ICMP packet
2. Create an ICMP_ECHO_request packet (it should have type 8)
3. Wait for the ICMP_REPLY packet
4. Remove the TCP header from the packer which is of size 20 bytes
5. And the check the type of reply message
6. If it is zero proceed else print the corresponding error message
7. Print the packets number along with RTT(round trip time) and save rtt in a list
8. When ctrl +c is pressed, print the number of packets received and lost, also print min, max and average RTT time.

CODE:

```
import socket
import struct
import time
from cv2 import add


from numpy import average

#    Echo Request and Echo Reply messages                          RFC 792

#     0                   1                   2                   3
#     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
#    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    |     Type      |     Code      |           Checksum            |
```

```
#    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    |           Identifier          |        Sequence Number        |
#    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    |     Data ...
#    +-+-+-+-+-

#    ICMPv4 Error message                                   RFC 792
#
#     0                   1                   2                   3
#     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
#    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    |     Type      |     Code      |           Checksum            |
#    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    |                 Unused / Depends on the error                 |
#    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    |      Internet Header + 64 bits of Original Data Datagram      |
#    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


class IcmpSocket:
   def __init__(self, address=None):
       self.dummy_port = 0

       # get the ip address from DNS
       try:
           address = socket.gethostbyname(input("Enter hostname: "))
       except:
           print("address is wrong try again\n")
           address = socket.gethostbyname(input("Enter hostname: "))
       # create the raw socket for sending ICMP protocol
       self.icmp_socket = socket.socket(
           socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP)

       # set the timeout
       self.icmp_socket.settimeout(2)

       # set the time to live
       self.set_ttl(10)
```

```
        # create the ICMP packet for type echo_request
        packet = self.create_packet(1, 1, b'\x89')

        i = 0
        j = 0
        time_list = []

        # start sending and receiving
        while True:
            try:
                self.icmp_socket.sendto(packet, (address,
self.dummy_port))
                time_present = time.time()

                (time_future, out) = self.await_recieve()

                if(out):
                    time_list.append(time_future - time_present)
                    print(
                        f'recieved packet {i+1} from {address} with rtt
{(time_future - time_present)*1000}ms')

                    i = i+1
                else:
                    j = j+1

            except KeyboardInterrupt:
                print(
                    f"\ntotal packets recieved: {i} total packets lossed:
{j}")
                if(i == 0):
                    break
                print(
                    f'minimum rtt: {min(time_list)*1000}ms, average_time:
{sum(time_list)/(len(time_list) )*1000}ms, max time
rtt:{max(time_list)*1000}ms')
                break

        self.icmp_socket.close()
```

```
    def create_packet(self, id, seq, payload):
        # create echo request
        header = struct.pack("!BBHHH", 8, 0, 0, id, seq)
        checksum = self.checksum(header + payload)

        header = struct.pack("!BBHHH", 8, 0, checksum, id, seq)
        packet = header + payload
        self.checksum(packet)
        return packet

    def checksum(self, data):
        # calculate checksum
        data += b'\x00'
        sum = 0
        countTo = (len(data) // 2) * 2
        count = 0

        while count < countTo:
            sum += data[count] * 256 + data[count+1]
            sum = (sum & 0xffff) + (sum >> 16)
            count = count + 2
        return ~sum & 0xffff

    def set_ttl(self, ttl):
        # set ttl
        self.icmp_socket.setsockopt(socket.IPPROTO_IP, socket.IP_TTL,
ttl)

    def await_recieve(self):
        # wait for the receiving packet
        while True:
            out = False
            try:
                msg = self.icmp_socket.recvfrom(1024)
            except socket.timeout:
                print("time out")
                return (0, False)
            time_future = time.time()
```

```
            time.sleep(1)
            # bytes one to 20 are tcp header bytes and the remaining are
icmp echo reply packet bytes.
            msg = msg[0][20:]
            (type_, code_, checksum_, p_id_,
             sequence_) = struct.unpack('!2B3H', msg[0:8])
            if self.checksum(msg) == 0 and type_ == 0:
                out = True
            self.type_handler(type_)
            return (time_future, out)

    def type_handler(self, type_):
        switcher = {
            0: 'Echo reply',
            3: 'Destination unreachable',
            4: 'Source quench',
            5: 'Redirect',
            8: 'Echo',
            9: 'Router advertisement',
            10: 'Router selection',
            11: 'Time exceeded',
            12: 'Parameter problem',
            13: 'Timestamp',
            14: 'Timestamp reply',
            15: 'Information request',
            16: 'Information reply',
            17: 'Address mask request',
            18: 'Address mask reply',
            30: 'Traceroute'}
        if (type_ != 0):
            print(switcher.get(type_, "Invalid type"))

        return switcher.get(type_, "Invalid type")


if __name__ == '__main__':
    icmp_socket = IcmpSocket()
```

Output:

For the trilokesh.com icmp echo_reply message type is 11 which corresponds to the time exceeded error

```
greyhat@greyhat:~/CN$ sudo /bin/python3 /home/greyhat/CN/Ping/Ping2.py
Enter hostname: trilokesh.com
Time exceeded
Time exceeded
Time exceeded
Time exceeded
Time exceeded
Time exceeded
^C
total packets recieved: -6 total packets lossed: 6
```

For nitc.ac.in the icmp echo_reply packet is not reached within timeout i.e., 2 seconds

```
greyhat@greyhat:~/CN$ sudo /bin/python3 /home/greyhat/CN/Ping/Ping2.py
Enter hostname: nitc.ac.in
time out
time out
time out
time out
time out
^C
total packets recieved: 0 total packets lossed: 5
```

For google.com the icmp echo_reply package is successfully reached with no error

```
greyhat@greyhat:~/CN$ sudo /bin/python3 /home/greyhat/CN/Ping/Ping2.py
Enter hostname: google.com
recieved packet 1 from 142.250.193.142 with rtt 13.889074325561523ms
recieved packet 2 from 142.250.193.142 with rtt 13.91744613647461ms
recieved packet 3 from 142.250.193.142 with rtt 13.978719711303711ms
recieved packet 4 from 142.250.193.142 with rtt 13.976335525512695ms
recieved packet 5 from 142.250.193.142 with rtt 13.992547988891602ms
recieved packet 6 from 142.250.193.142 with rtt 13.975858688354492ms
recieved packet 7 from 142.250.193.142 with rtt 13.848304748535156ms
recieved packet 8 from 142.250.193.142 with rtt 13.980627059936523ms
^C
total packets recieved: 8 total packets lossed: 0
minimum rtt: 13.848304748535156ms, average_time: 13.944864273071289ms, max time rt
t:13.992547988891602ms
```