

CS542
COMPUTER NETWORKS 1
PROJECT REPORT

TEAM MEMBERS:

Rahul Maddula-A20488730

Akanksha Raghvesh - A20483075

Kuladeep Bushan Mantri - A20498452

ABSTRACT:

We basically implement the ping command in Python using the raw socket technique in socket programming with the ICMP echo request with 256 sizes of data and echo reply between local client and local server in this project. We wrote functions for sending requests, receiving requests, checking errors in packets using checksum, and pinging the local server and local client for implementation.

In this project, the local client sends 2 ICMP requests to a local server, and again the local server sends replies to the local client. For the replies and requests, we are using the time difference between the request and replies which is actually the time delay of the packet between the local client and local server. As result, we are using file concepts to save the results to the local client file system.

BACKGROUND:

When network difficulties hinder IP packet delivery, ICMP (Internet Control Message Protocol) is an error-reporting protocol that network devices like routers utilize to generate error messages to the originating IP address. ICMP creates and delivers messages to the source IP address. ICMP messages can be sent, received, and processed by an IP network device.

While ICMP isn't commonly utilized in end-user apps, network administrators use it to troubleshoot internet connections with diagnostic tools like ping and traceroute. In ICMP, Ping is a simple traceroute tool. It sends out pings, which are also known as echo request messages, and then timing how long it takes for the message to reach its destination and return to the source. These are known as echo reply messages. Pings are important for determining the latency of a specific device.

ICMP Packet Structure:

Type: The type specifies a concise description of the message's purpose, allowing the receiving network device to understand why it is receiving the message and how to handle it.

Type 0- Echo Reply

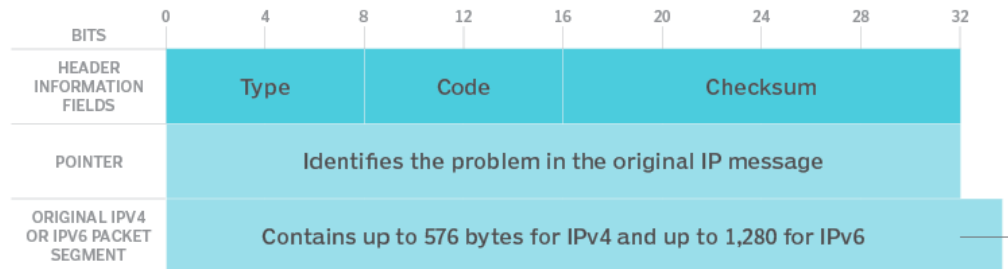
Type 8- Echo Request

Code: The message type code, which offers further information about the error kind, is represented by the following 8 bits.

Checksum: The last 16 bits are used to check for message integrity. The checksum displays the total amount of bits in the message and allows the ICMP tool to verify that the entire message was delivered by comparing it to the ICMP message header.

The structure of an ICMP packet

Packets have a header and pointer that are 32 bits each and a copy of the error-containing IP message with a variable number of bytes.



IMPLEMENTATION

Step 1: Importing required packages and assigning the icmp echo type as the 8(Echo Request) and 0(Echo Reply).

```
from socket import *
import os
import sys
import array
import struct
import time
import select
import signal
import binascii
from ICMPChecksum import ICMPchecksum
icmpchotype= 8
imcpechotype1=0
```

Step 2:

Now, we are creating a function which is known as the sPing which is used to send the request ping and define the header and payload for the ICMP packet. In the function, we are defining the Socket, destination address, and identifier(ID). And we will calculate the checksum at the sender side and receiver side, so Initially, we are assigning the initial checksum value as zero, In the next line, we are creating a packet header with the variables type, code, checksum, packet, sequence number. Next, we are saving time on requests sent, and next line we are creating a payload with 256 bytes, and then we are calling the checksum function with the packet. Next, we are calculating the checksum and combine the packet with the header and a data portion and sent it to a destination address which is a local server. And sent the ICMP packet with Echo request type 8.

```
def sPing(s, destinationaddress, Identifier):
    Initialchecksumvalue = 0
    Packetheader = struct.pack("bbHHh", icmpechotype, 0, Initialchecksumvalue, Identifier, 1)
    dsend='Local system'
    timetakenforpacket=time.time()
    payloadportion = struct.pack("d248s", timetakenforpacket,dsend.encode())
    Initialchecksumvalue = ICMPchecksum(Packetheader + payloadportion)
    if sys.platform == 'darwin':
        Initialchecksumvalue = htons(Initialchecksumvalue) & 0xffff
    Packetheader = struct.pack("bbHHh", icmpechotype, 0, Initialchecksumvalue, Identifier, 1)
    totalICMPpacket = Packetheader + payloadportion
    print(destinationaddress)
    s.sendto(totalICMPpacket, (destinationaddress, 0))
    print("Client sent requests to Server Successfully !!!")
```

Step 3:

Now, we are writing the function to receive the requests, which is known as rPing which consists of Socket, identifier, and time taken for a packet which is nothing but the sent time which is defined in sPing and unpacking the sent packet header in which we get values variables, type, code, checksum, packetID, sequence number. And here we can define the time taken, and at last, we calculate the delay by subtracting the request sent time and request received time. This function basically reads the ICMP packets and returns the time taken to reach the client from the server with Echo reply 0.

```
def rPing(s, number, timetakenforpacket):
    strb = 248
    dsend='HI LOCAL SERVER'
    b = struct.calcsize("d")
    Servertimetaken = time.time()
    while True:
        packetrecieved, addr = s.recvfrom(1024)
        tr = time.time()
        packetheader = packetrecieved[20:28]
        typeoficmp, codeonicmpheader, icmpchecksum, identifier, sequencenumber = struct.unpack("bbHHh", packetheader)
        if identifier == number and typeoficmp != 8:
            senttimerequest = struct.unpack("d", packetrecieved[28:36])[0]
            ICMPpacketpayload = struct.unpack("d", packetrecieved[36:36 + b])[0]
```

Step 4:

Now, we are creating a function, for ping, which create a socket with socket function and we use this function to call other two functions sPing and rPing.

```
def ICMPPINGING():  
    icmp = getprotobyname("icmp")  
    s = socket(AF_INET, SOCK_RAW, icmp)  
    ICMPIdentifier = os.getpid() & 0xFFFF
```

```
sPing(s, destinationaddress, ICMPIdentifier)  
rPing(s, ICMPIdentifier, timeout)
```

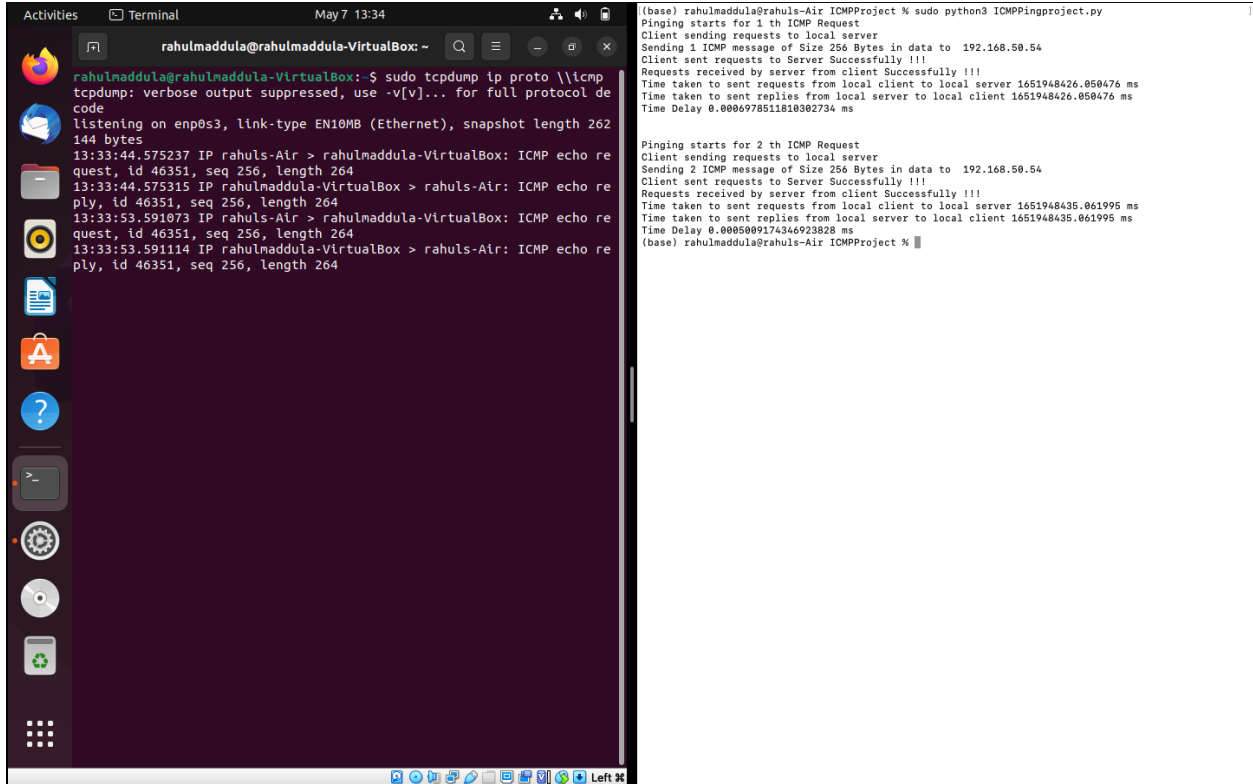
Step 5:

Now, we are a separate python code for calculating the checksum, When the ICMP packet is transmitted, the checksum is computed. When the data packet is received, the checksum is again computed and verified against the checksum field. If the two checksums do not match then an error has occurred.

```
def ICMPchecksum(ICMPPacket):  
    if (len(ICMPPacket) % 2):  
        ICMPPacket += "\x00"  
    Transform= array.array("H", ICMPPacket)  
    if sys.byteorder == "big":  
        Transform.byteswap()  
    evaluatev = sum(Transform)  
    evaluatev &= 0xffffffff  
    while (evaluatev >> 16):  
        evaluatev = (evaluatev & 0xFFFF) + (evaluatev >> 16)  
    resultval = ~evaluatev & 0xffff  
    resultval = resultval >> 8 | (resultval << 8 & 0xff00)  
    return resultval
```

OUTCOMES

For the local server, we are using Virtual machine in which we get IP address and we can ping that IP address as below:



The screenshot shows a terminal window with a dark background. On the left, there is a sidebar with various application icons. The terminal output is as follows:

```
rahulmaddula@rahulmaddula-VirtualBox: ~  
$ sudo tcpdump ip proto icmp  
tcpdump: verbose output suppressed, use -v[v]... for full protocol de  
code  
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262  
144 bytes  
13:33:44.575237 IP rahuls-Air > rahulmaddula-VirtualBox: ICMP echo re  
quest, id 46351, seq 256, length 264  
13:33:44.575315 IP rahulmaddula-VirtualBox > rahuls-Air: ICMP echo re  
ply, id 46351, seq 256, length 264  
13:33:53.591073 IP rahuls-Air > rahulmaddula-VirtualBox: ICMP echo re  
quest, id 46351, seq 256, length 264  
13:33:53.591114 IP rahulmaddula-VirtualBox > rahuls-Air: ICMP echo re  
ply, id 46351, seq 256, length 264
```


On the right side of the terminal window, there is a separate text area showing the output of a Python script:

```
(base) rahulmaddula@rahuls-Air ICMPProject % sudo python3 ICMPpingproject.py  
Pinging starts for 1 th ICMP Request  
Client sending requests to local server  
Sending 1 ICMP message of Size 256 Bytes in data to 192.168.50.54  
Client sent requests to Server Successfully !!!  
Requests received by server from client Successfully !!!  
Time taken to sent requests from local client to local server 1651948426.050476 ms  
Time taken to sent replies from local server to local client 1651948426.050476 ms  
Time Delay 0.0006978511810302734 ms  
  
Pinging starts for 2 th ICMP Request  
Client sending requests to local server  
Sending 2 ICMP message of Size 256 Bytes in data to 192.168.50.54  
Client sent requests to Server Successfully !!!  
Requests received by server from client Successfully !!!  
Time taken to sent requests from local client to local server 1651948435.061995 ms  
Time taken to sent replies from local server to local client 1651948435.061995 ms  
Time Delay 0.0005009174346923828 ms  
(base) rahulmaddula@rahuls-Air ICMPProject %
```

And Now, we had try to use our system as local Server rather than VM machine, we ping our own local system.

```
(base) rahulmaddula@rahuls-Air ICMPProject % sudo python3 ICMPpingproject.py  
Pinging starts for 1 th ICMP Request  
Client sending requests to local server  
Sending 1 ICMP message of Size 256 Bytes in data to 127.0.0.1  
Client sent requests to Server Successfully !!!  
Requests received by server from client Successfully !!!  
Time taken to sent requests from local client to local server 1651946465.308834 ms  
Time taken to sent replies from local server to local client 1651946465.308834 ms  
Time Delay 0.0002689361572265625 ms  
  
Pinging starts for 2 th ICMP Request  
Client sending requests to local server  
Sending 2 ICMP message of Size 256 Bytes in data to 127.0.0.1  
Client sent requests to Server Successfully !!!  
Requests received by server from client Successfully !!!  
Time taken to sent requests from local client to local server 1651946474.323891 ms  
Time taken to sent replies from local server to local client 1651946474.323891 ms  
Time Delay 0.00024088750915527344 ms  
(base) rahulmaddula@rahuls-Air ICMPProject %
```

We save the requests and replies in Text file which is shown below:

A screenshot of a text editor window titled "ICMPPINGOUTPUT.txt — Locked". The window contains two sections of ICMP ping output, separated by lines of asterisks. The first section shows the results for the 1st ICMP request to the local server 127.0.0.1, with a round-trip time of 1651962281.469252 ms and a message "HI LOCAL SERVER". The second section shows the results for the 2nd ICMP request, with a round-trip time of 1651962290.4819672 ms and the same message. The text is as follows:

```
*****
Pinging starts for 1 th ICMP Request
The Local Server 127.0.0.1
Time Taken to sent requests from local client to local server in ms 1651962281.469252
Time Taken to sent replies from local server to local client in ms 1651962281.469513
The ICMP MESSAGE TIME DELAY is: 0.0002608299255371094
The Message HI LOCAL SERVER
*****
Pinging starts for 2 th ICMP Request
The Local Server 127.0.0.1
Time Taken to sent requests from local client to local server in ms 1651962290.4819672
Time Taken to sent replies from local server to local client in ms 1651962290.482187
The ICMP MESSAGE TIME DELAY is: 0.00021982192993164062
The Message HI LOCAL SERVER
```

OBSERAVTIONS:

- Firstly, I and my teammates, feel excited to do this project as we are doing the practical project with our technical knowledge of computer networks.
- Actually, It takes us some valuable time to understand how sockets and other functions work in the code.
- With that understanding, we tried to code the Sending request but got a few errors such as packing the header and the payload and sending the request to a particular server.
- After reading some blogs and papers, we understand where went wrong and changed the code.
- And the local server we tried for a Virtual machine, we had to use a command to listen to the requests as the server.
- And Finally, we did with code as per the requirements given to us.

CONCLUSION

Hence, we implemented the ping command using the raw socket techniques in socket programming, In which the local client will send the 2 requests with a data size of 256 bytes to the local server, and with the local server replies to the requests.

REFREENCES

<https://www.techtarget.com/searchnetworking/definition/ICMP>

<https://stackoverflow.com/questions/9660254/how-to-send-icmp-packet-through-sockets>