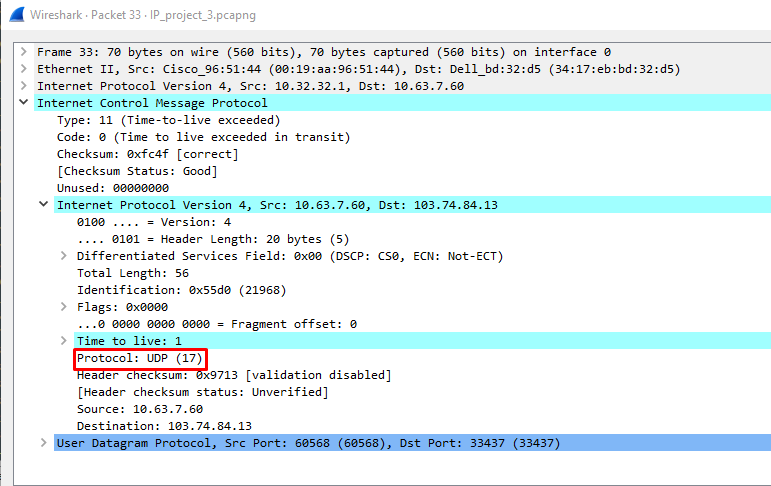
# CS 6043 Computer Networking

# PROJECT 3

By Mitra Sasanka Darbha, M#13523468

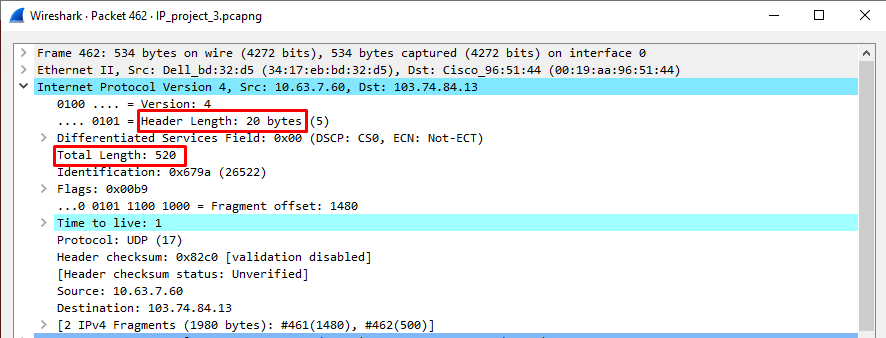
# Part 1: IP

## 1.1

Upper layer protocol is UDP (17)

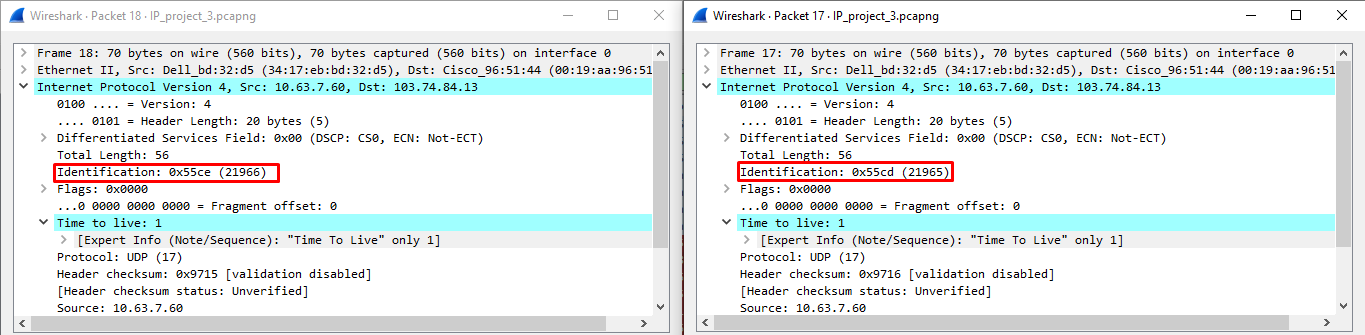
## 1.2

There are 20 bytes in IP header and total length is 520 bytes. Therefore, payload = (520 – 20) = 500 bytes.



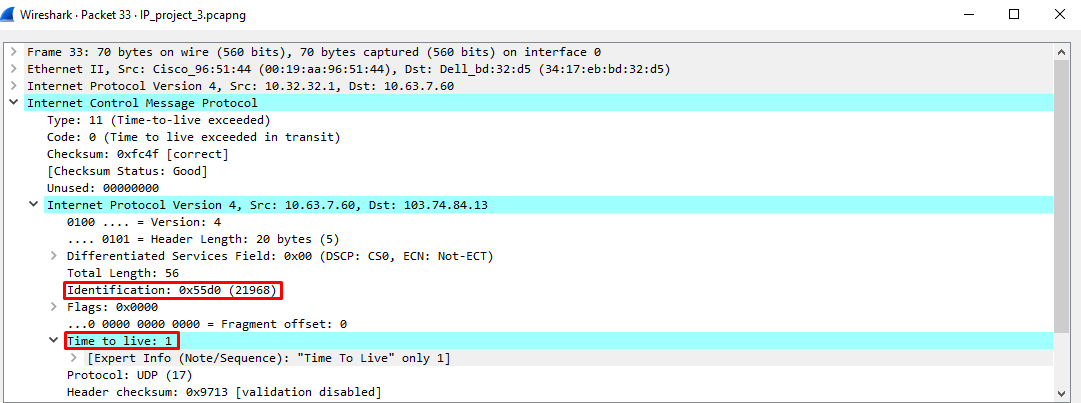
## 1.3

Identification field is incrementing for every frame. Time to Live is changing but does not always change. (TTL for frames 17, 18 and 19 is same)



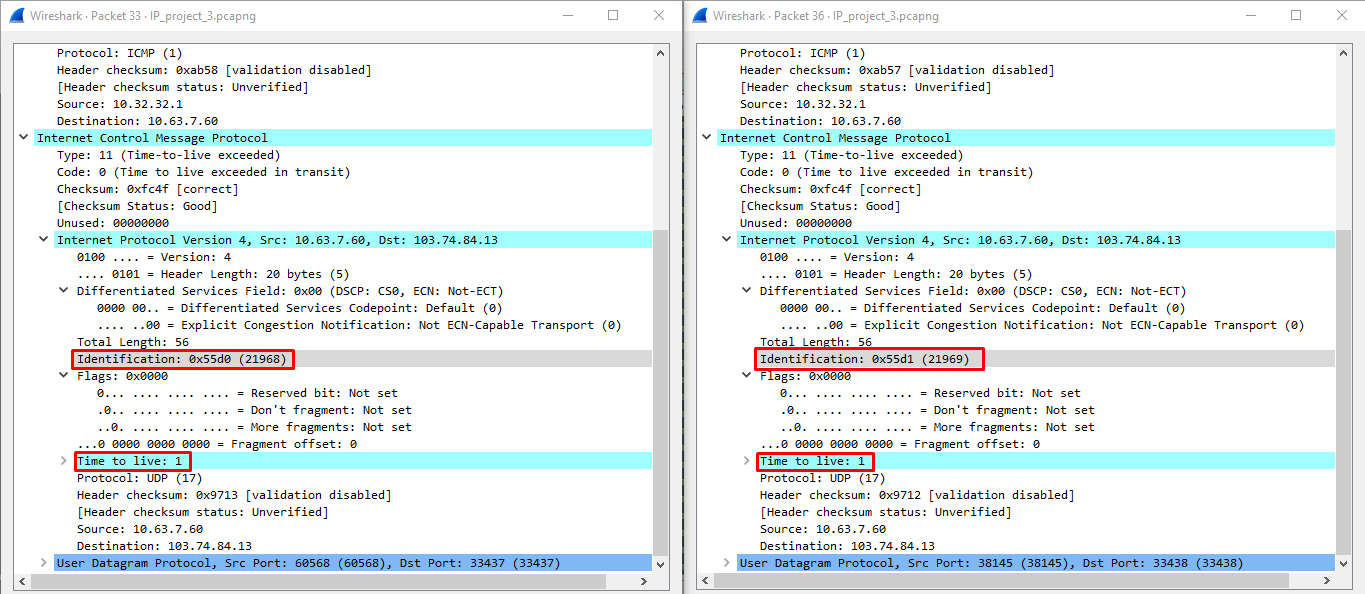
## 1.4

Identification: 0x55d0 (21968). Time to Live: 1



## 1.5

The values of identification field changes for all the ICMP TTL-exceeded replies since the identification value is a unique value. If two or more datagrams have the same identification value, that means these IP datagrams are fragments of a single large IP datagram.

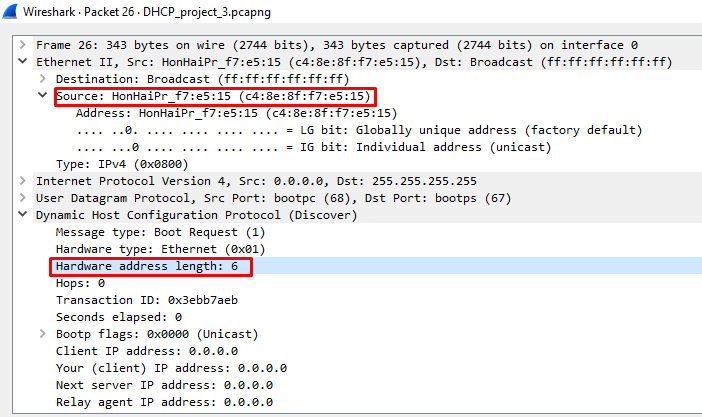
The TTL field was unchanged since the TTL for the nearest router is always the same.

# Part 2: DHCP

## 2.1

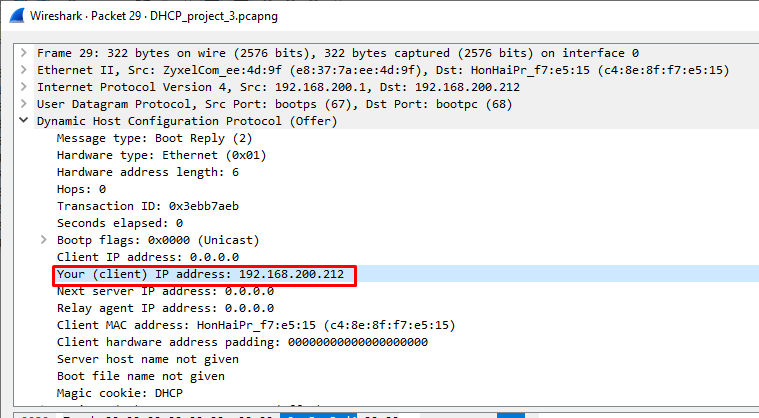
Link layer address = c4:8e:8f:f7:e5:15

Hardware address length = 6



## 2.2

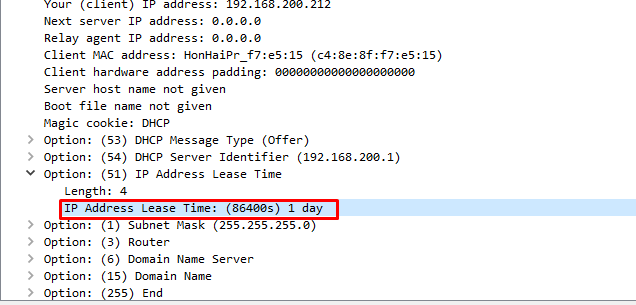
IP Address offered is 192.168.200.212



## 2.3

During the lease time, the DHCP server cannot assign that IP address to any other clients. The purpose of a lease is to limit the length of time that a client can use an IP address.

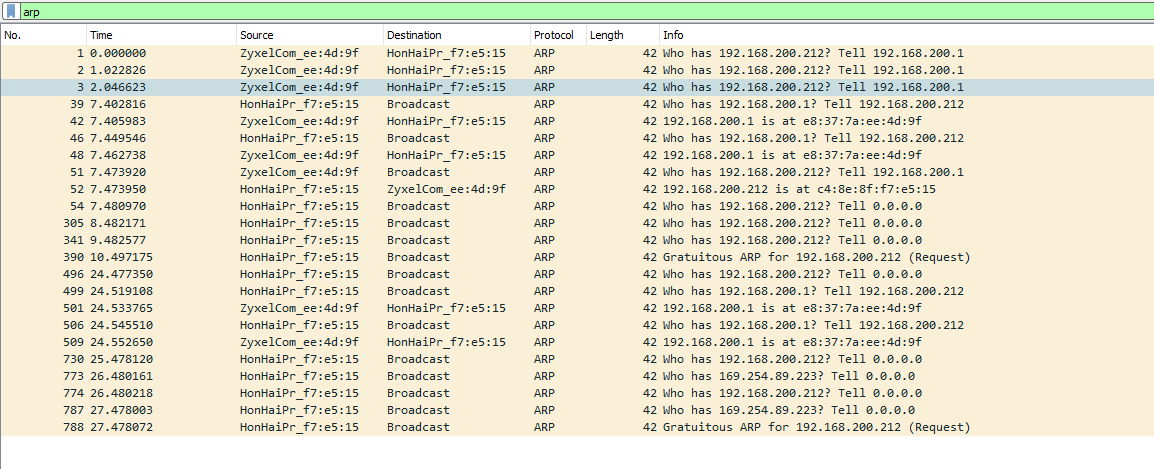
IP address lease time given is 1 day.



## 2.4

Yes, there are ARP packets in the trace.

They appear to be broadcasts sent out by network to map IP address to MAC address to build up the known IP address by the clients.



# Part 3: Socket Programming

**ICMP Pinger code:**

from socket import \*

import os

import sys

import struct

import time

import select

import binascii

import socket

ICMP\_ECHO\_REQUEST = 8

def checksum(str\_):

# In this function we make the checksum of our packet

str\_ = bytearray(str\_)

csum = 0

countTo = (len(str\_) // 2) \* 2

for count in range(0, countTo, 2):

thisVal = str\_[count+1] \* 256 + str\_[count]

csum = csum + thisVal

csum = csum & 0xffffffff

if countTo < len(str\_):

csum = csum + str\_[-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)

icmpHeader = recPacket[20:28]

icmpType, code, mychecksum, packetID, sequence = struct.unpack("bbHHh", icmpHeader)

if type != 8 and packetID == ID:

bytesInDouble = struct.calcsize("d")

timeSent = struct.unpack("d", recPacket[28:28 + bytesInDouble])[0]

return timeReceived - timeSent

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 = htons(myChecksum) & 0xffff

#Convert 16-bit integers from host to network byte order.

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")

#Create Socket here

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):

dest = gethostbyname(host)

print ("Pinging google.com:" + dest + " using Python:")

print ("")

#Send ping requests to a server separated by approximately one second

while 1 :

delay = doOnePing(dest, timeout)

print (delay)

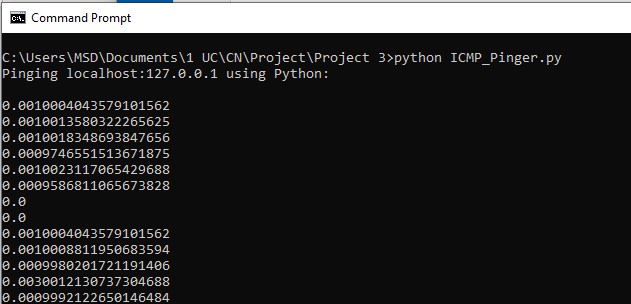
time.sleep(1)# one second

return delay

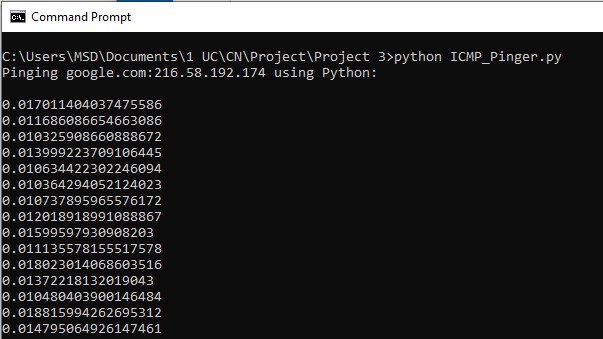
ping("google.com")

**Outputs:**

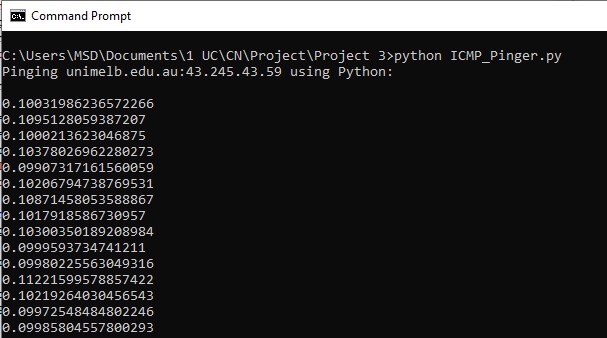
Pinging Localhost:



Pinging google.com:



Pinging University of Melbourne(Australia):



Pining Fudan University(China):

