**Task 1:** Develop a program that capture 10 Ethernet frames from the network and display the source and destination MAC addresses along with the data in each of the frames.

**Program:**

import socket, sys

from struct import \*

#Convert a string of 6 characters of ethernet address into a dash separated hex string

def eth\_addr (a) :

b = "%.2x:%.2x:%.2x:%.2x:%.2x:%.2x" % (ord(a[0]) , ord(a[1]) , ord(a[2]), ord(a[3]), ord(a[4]) , ord(a[5]))

return b

#create a AF\_PACKET type raw socket (thats basically packet level)

#define ETH\_P\_ALL 0x0003 /\* Every packet (be careful!!!) \*/

try:

s = socket.socket( socket.AF\_PACKET , socket.SOCK\_RAW , socket.ntohs(0x0003))

except socket.error , msg:

print 'Socket could not be created. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]

sys.exit()

# receive a packet

for i in range(0, 10):

print

print '--PACKET '+str(i+1)+'--'

print

packet = s.recvfrom(65565)

#packet string from tuple

packet = packet[0]

#parse ethernet header

eth\_length = 14

eth\_header = packet[:eth\_length]

eth = unpack('!6s6sH' , eth\_header)

eth\_protocol = socket.ntohs(eth[2])

print 'Destination MAC : ' + eth\_addr(packet[0:6]) + ' Source MAC : ' + eth\_addr(packet[6:12]) + ' Protocol : ' + str(eth\_protocol)

print '--ETHERNET HEADER--'

#Parse IP header

#take first 20 characters for the ip header

ip\_header = packet[eth\_length:20+eth\_length]

#now unpack them

iph = unpack('!BBHHHBBH4s4s' , ip\_header)

version\_ihl = iph[0]

version = version\_ihl >> 4

ihl = version\_ihl & 0xF

iph\_length = ihl \* 4

ttl = iph[5]

protocol = iph[6]

s\_addr = socket.inet\_ntoa(iph[8]);

d\_addr = socket.inet\_ntoa(iph[9]);

print 'Version : ' + str(version) + ' IP Header Length : ' + str(ihl) + ' TTL : ' + str(ttl) + ' Protocol : ' + str(protocol) + ' Source Address : ' + str(s\_addr) + ' Destination Address : ' + str(d\_addr)

#TCP protocol

if protocol == 6 :

print '--TCP--'

t = iph\_length + eth\_length

tcp\_header = packet[t:t+20]

#now unpack them :)

tcph = unpack('!HHLLBBHHH' , tcp\_header)

source\_port = tcph[0]

dest\_port = tcph[1]

sequence = tcph[2]

acknowledgement = tcph[3]

doff\_reserved = tcph[4]

tcph\_length = doff\_reserved >> 4

print 'Source Port : ' + str(source\_port) + ' Dest Port : ' + str(dest\_port) + ' Sequence Number : ' + str(sequence) + ' Acknowledgement : ' + str(acknowledgement) + ' TCP header length : ' + str(tcph\_length)

h\_size = eth\_length + iph\_length + tcph\_length \* 4

data\_size = len(packet) - h\_size

#get data from the packet

data = packet[h\_size:]

print 'Data : ' + data

#ICMP Packets

elif protocol == 1 :

print '--ICMP--'

u = iph\_length + eth\_length

icmph\_length = 4

icmp\_header = packet[u:u+4]

#now unpack them :)

icmph = unpack('!BBH' , icmp\_header)

icmp\_type = icmph[0]

code = icmph[1]

checksum = icmph[2]

print 'Type : ' + str(icmp\_type) + ' Code : ' + str(code) + ' Checksum : ' + str(checksum)

h\_size = eth\_length + iph\_length + icmph\_length

data\_size = len(packet) - h\_size

#get data from the packet

data = packet[h\_size:]

print 'Data : ' + data

#UDP packets

elif protocol == 17 :

print '--UDP--'

u = iph\_length + eth\_length

udph\_length = 8

udp\_header = packet[u:u+8]

#now unpack them :)

udph = unpack('!HHHH' , udp\_header)

source\_port = udph[0]

dest\_port = udph[1]

length = udph[2]

checksum = udph[3]

print 'Source Port : ' + str(source\_port) + ' Dest Port : ' + str(dest\_port) + ' Length : ' + str(length) + ' Checksum : ' + str(checksum)

h\_size = eth\_length + iph\_length + udph\_length

data\_size = len(packet) - h\_size

#get data from the packet

data = packet[h\_size:]

print 'Data : ' + data

#IGMP packets

elif protocol == 88 :

print '--IGMP--'

u = iph\_length + eth\_length

igmph\_length = 8

igmp\_header = packet[u:u+8]

#now unpack them :)

igmph = unpack('!HHHH' , igmp\_header)

source\_port = igmph[0]

dest\_port = igmph[1]

length = igmph[2]

checksum = igmph[3]

print 'Source Port : ' + str(source\_port) + ' Dest Port : ' + str(dest\_port) + ' Length : ' + str(length) + ' Checksum : ' + str(checksum)

h\_size = eth\_length + iph\_length + igmph\_length

data\_size = len(packet) - h\_size

#get data from the packet

data = packet[h\_size:]

print 'Data : ' + data

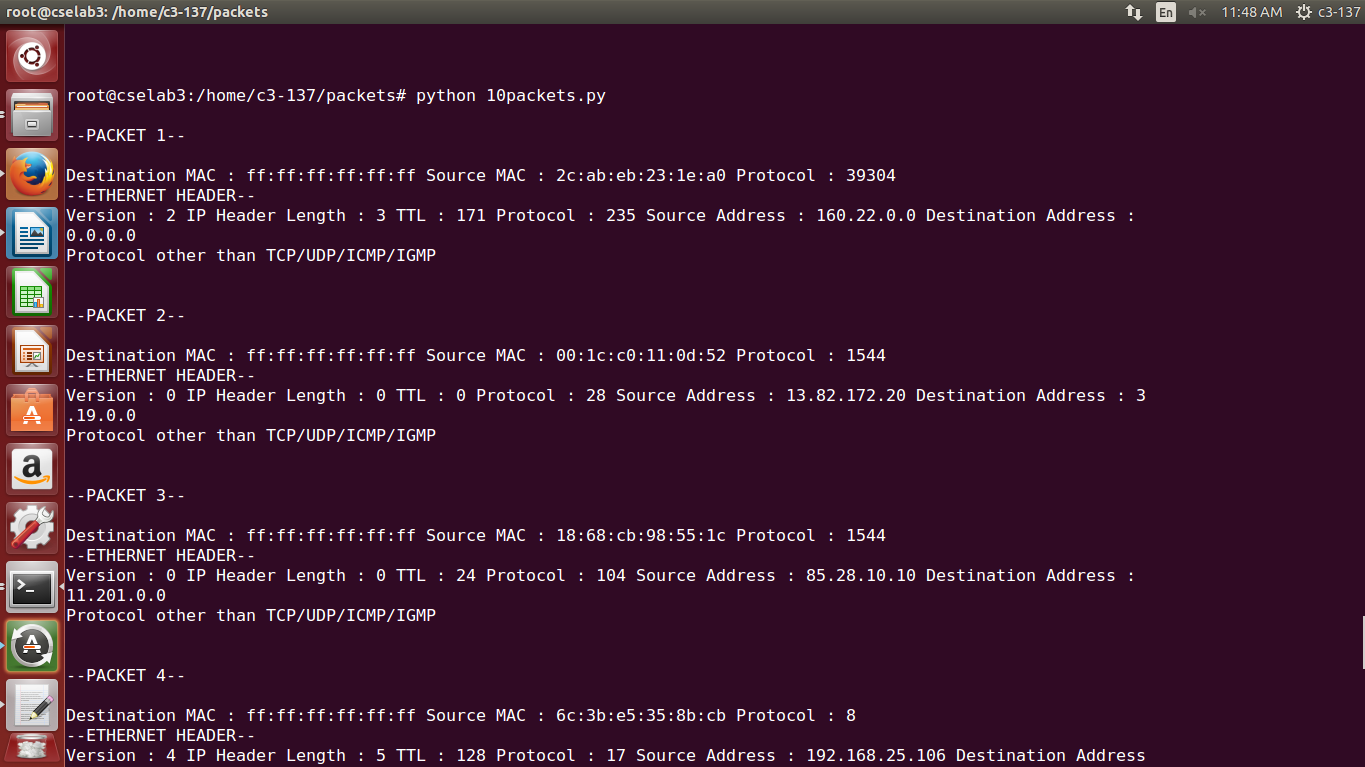
#some other IP packet

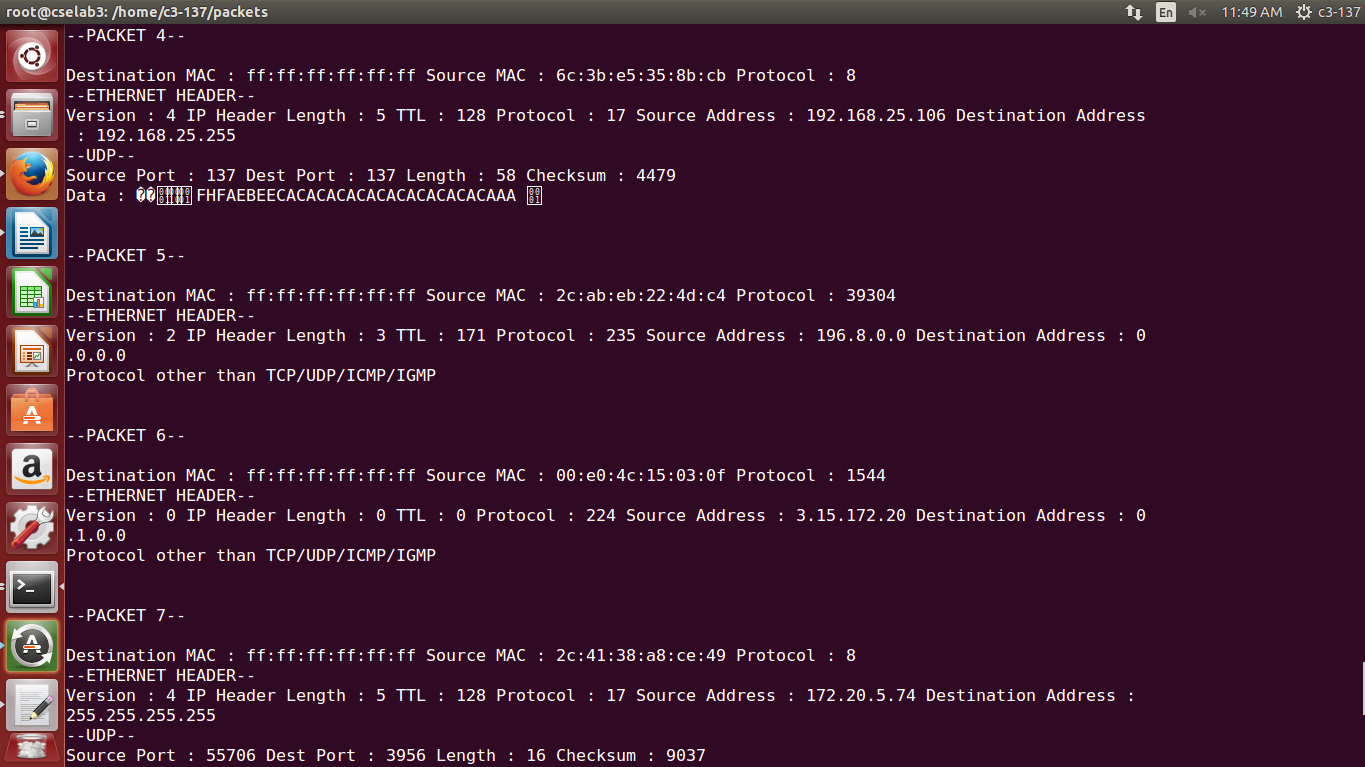
else :

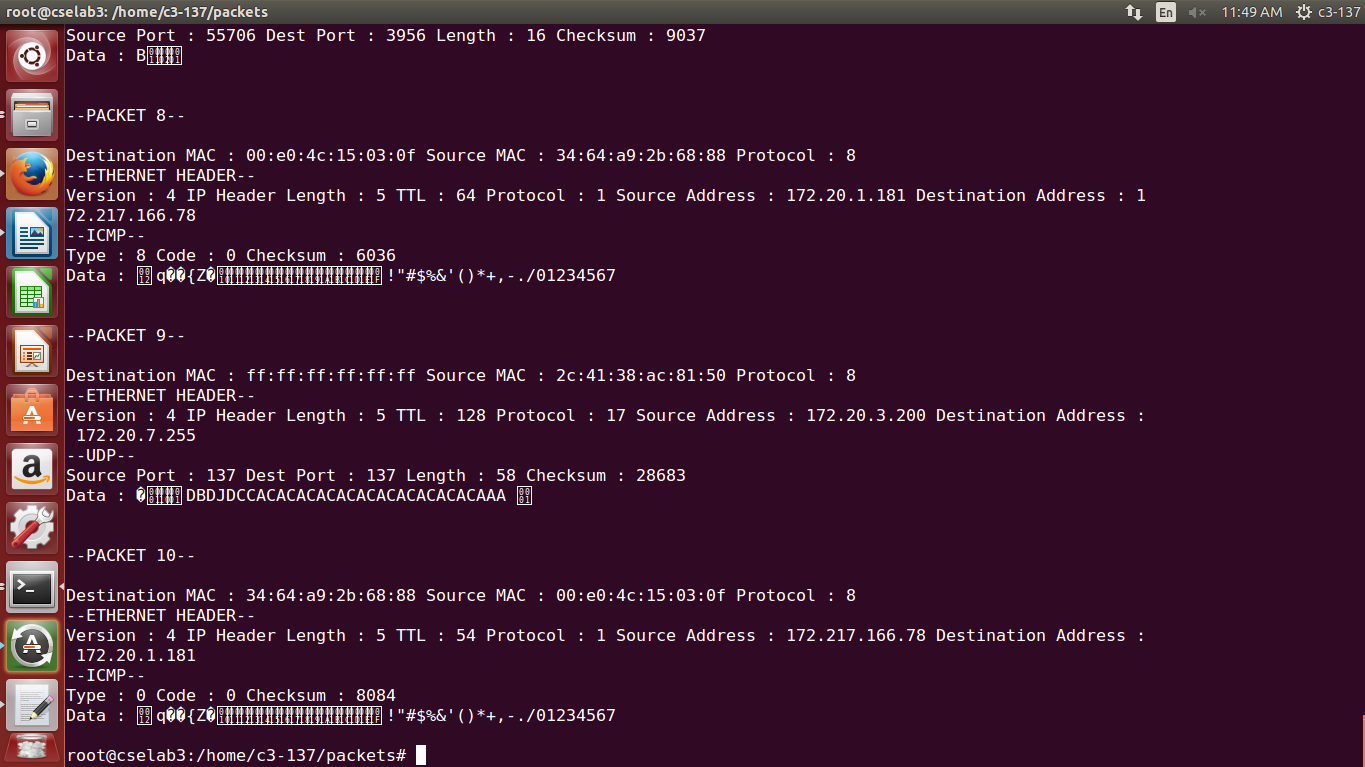
print 'Protocol other than TCP/UDP/ICMP/IGMP'

print

**Output**

****

****

****

**Task 2:** Develop a program that capture n packets and displays the complete packet details including the headers.

**Program:**

import socket, sys

from struct import \*

#Convert a string of 6 characters of ethernet address into a dash separated hex string

def eth\_addr (a) :

b = "%.2x:%.2x:%.2x:%.2x:%.2x:%.2x" % (ord(a[0]) , ord(a[1]) , ord(a[2]), ord(a[3]), ord(a[4]) , ord(a[5]))

return b

#create a AF\_PACKET type raw socket (thats basically packet level)

#define ETH\_P\_ALL 0x0003 /\* Every packet (be careful!!!) \*/

try:

s = socket.socket( socket.AF\_PACKET , socket.SOCK\_RAW , socket.ntohs(0x0003))

except socket.error , msg:

print 'Socket could not be created. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]

sys.exit()

n = input('Enter number of packets you want: ')

# receive a packet

for i in range(0, n):

print

print '--PACKET '+str(i+1)+'--'

print

packet = s.recvfrom(65565)

#packet string from tuple

packet = packet[0]

#parse ethernet header

eth\_length = 14

eth\_header = packet[:eth\_length]

eth = unpack('!6s6sH' , eth\_header)

eth\_protocol = socket.ntohs(eth[2])

print 'Destination MAC : ' + eth\_addr(packet[0:6]) + ' Source MAC : ' + eth\_addr(packet[6:12]) + ' Protocol : ' + str(eth\_protocol)

print '--ETHERNET HEADER--'

#Parse IP header

#take first 20 characters for the ip header

ip\_header = packet[eth\_length:20+eth\_length]

#now unpack them

iph = unpack('!BBHHHBBH4s4s' , ip\_header)

version\_ihl = iph[0]

version = version\_ihl >> 4

ihl = version\_ihl & 0xF

iph\_length = ihl \* 4

ttl = iph[5]

protocol = iph[6]

s\_addr = socket.inet\_ntoa(iph[8]);

d\_addr = socket.inet\_ntoa(iph[9]);

print 'Version : ' + str(version) + ' IP Header Length : ' + str(ihl) + ' TTL : ' + str(ttl) + ' Protocol : ' + str(protocol) + ' Source Address : ' + str(s\_addr) + ' Destination Address : ' + str(d\_addr)

#TCP protocol

if protocol == 6 :

print '--TCP--'

t = iph\_length + eth\_length

tcp\_header = packet[t:t+20]

#now unpack them :)

tcph = unpack('!HHLLBBHHH' , tcp\_header)

source\_port = tcph[0]

dest\_port = tcph[1]

sequence = tcph[2]

acknowledgement = tcph[3]

doff\_reserved = tcph[4]

tcph\_length = doff\_reserved >> 4

print 'Source Port : ' + str(source\_port) + ' Dest Port : ' + str(dest\_port) + ' Sequence Number : ' + str(sequence) + ' Acknowledgement : ' + str(acknowledgement) + ' TCP header length : ' + str(tcph\_length)

h\_size = eth\_length + iph\_length + tcph\_length \* 4

data\_size = len(packet) - h\_size

#get data from the packet

data = packet[h\_size:]

print 'Data : ' + data

#ICMP Packets

elif protocol == 1 :

print '--ICMP--'

u = iph\_length + eth\_length

icmph\_length = 4

icmp\_header = packet[u:u+4]

#now unpack them :)

icmph = unpack('!BBH' , icmp\_header)

icmp\_type = icmph[0]

code = icmph[1]

checksum = icmph[2]

print 'Type : ' + str(icmp\_type) + ' Code : ' + str(code) + ' Checksum : ' + str(checksum)

h\_size = eth\_length + iph\_length + icmph\_length

data\_size = len(packet) - h\_size

#get data from the packet

data = packet[h\_size:]

print 'Data : ' + data

#UDP packets

elif protocol == 17 :

print '--UDP--'

u = iph\_length + eth\_length

udph\_length = 8

udp\_header = packet[u:u+8]

#now unpack them :)

udph = unpack('!HHHH' , udp\_header)

source\_port = udph[0]

dest\_port = udph[1]

length = udph[2]

checksum = udph[3]

print 'Source Port : ' + str(source\_port) + ' Dest Port : ' + str(dest\_port) + ' Length : ' + str(length) + ' Checksum : ' + str(checksum)

h\_size = eth\_length + iph\_length + udph\_length

data\_size = len(packet) - h\_size

#get data from the packet

data = packet[h\_size:]

print 'Data : ' + data

#IGMP packets

elif protocol == 88 :

print '--IGMP--'

u = iph\_length + eth\_length

igmph\_length = 8

igmp\_header = packet[u:u+8]

#now unpack them :)

igmph = unpack('!HHHH' , igmp\_header)

source\_port = igmph[0]

dest\_port = igmph[1]

length = igmph[2]

checksum = igmph[3]

print 'Source Port : ' + str(source\_port) + ' Dest Port : ' + str(dest\_port) + ' Length : ' + str(length) + ' Checksum : ' + str(checksum)

h\_size = eth\_length + iph\_length + igmph\_length

data\_size = len(packet) - h\_size

#get data from the packet

data = packet[h\_size:]

print 'Data : ' + data

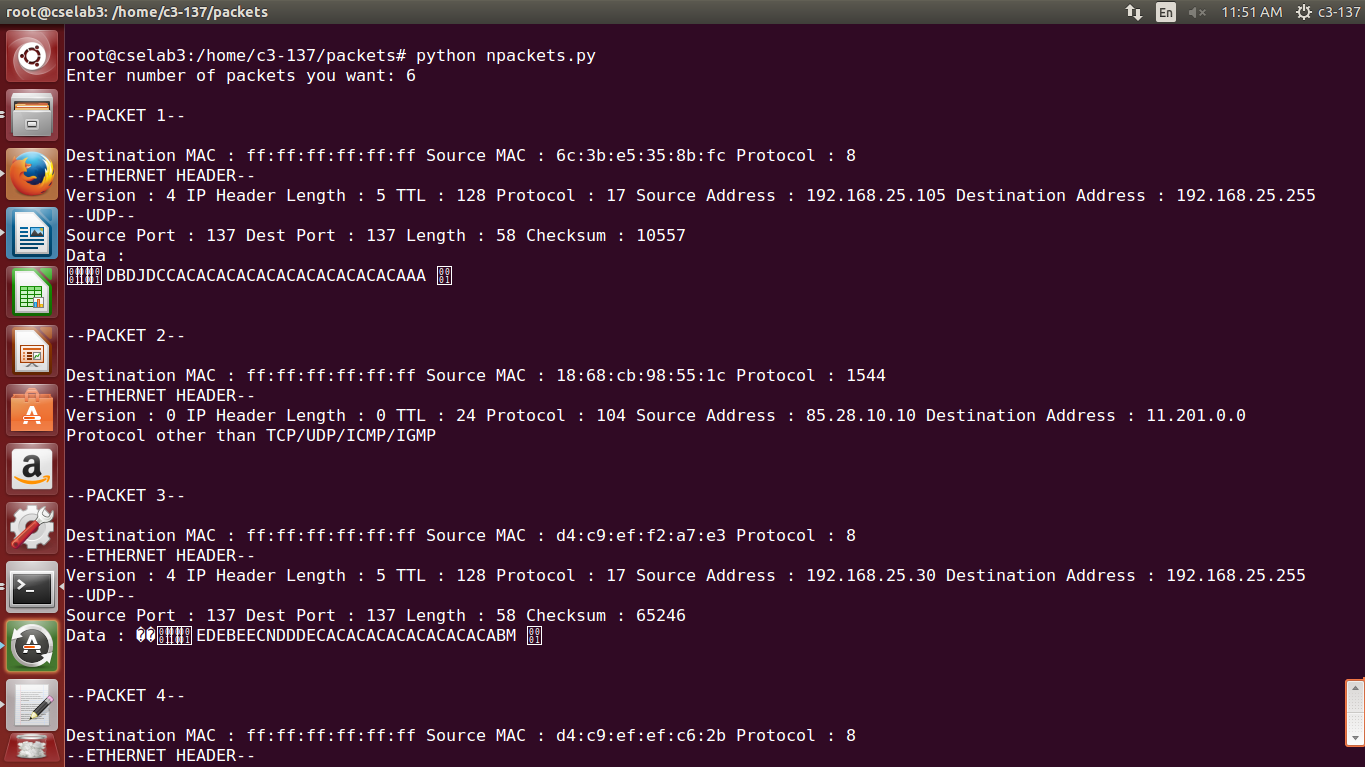
#some other IP packet

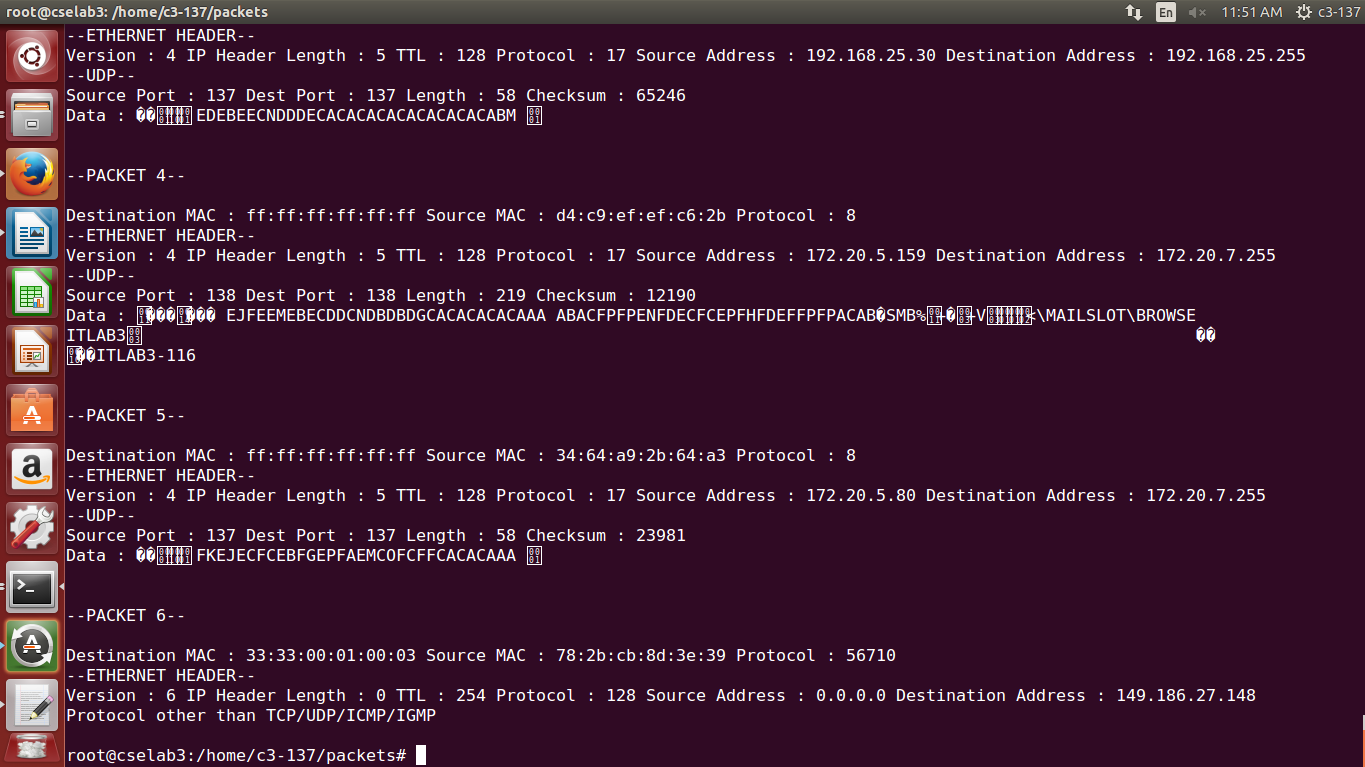
else :

print 'Protocol other than TCP/UDP/ICMP/IGMP'

print

**Output:**

****

****

**Task 3:** Write a program to extract and print the TCP segment information including the header.

**Program:**

import socket, sys

from struct import \*

#Convert a string of 6 characters of ethernet address into a dash separated hex string

def eth\_addr (a) :

b = "%.2x:%.2x:%.2x:%.2x:%.2x:%.2x" % (ord(a[0]) , ord(a[1]) , ord(a[2]), ord(a[3]), ord(a[4]) , ord(a[5]))

return b

#create a AF\_PACKET type raw socket (thats basically packet level)

#define ETH\_P\_ALL 0x0003 /\* Every packet (be careful!!!) \*/

try:

s = socket.socket( socket.AF\_PACKET , socket.SOCK\_RAW , socket.ntohs(0x0003))

except socket.error , msg:

print 'Socket could not be created. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]

sys.exit()

n = input('Enter number of packets you want: ')

i = 0

# receive a packet

while(i<n):

packet = s.recvfrom(65565)

#packet string from tuple

packet = packet[0]

#parse ethernet header

eth\_length = 14

eth\_header = packet[:eth\_length]

eth = unpack('!6s6sH' , eth\_header)

eth\_protocol = socket.ntohs(eth[2])

#Parse IP header

#take first 20 characters for the ip header

ip\_header = packet[eth\_length:20+eth\_length]

#now unpack them

iph = unpack('!BBHHHBBH4s4s' , ip\_header)

version\_ihl = iph[0]

version = version\_ihl >> 4

ihl = version\_ihl & 0xF

iph\_length = ihl \* 4

ttl = iph[5]

protocol = iph[6]

s\_addr = socket.inet\_ntoa(iph[8]);

d\_addr = socket.inet\_ntoa(iph[9]);

#TCP protocol

if protocol == 6 :

i+=1

print

print

print '--PACKET '+str(i)+'--'

print 'Destination MAC : ' + eth\_addr(packet[0:6]) + ' Source MAC : ' + eth\_addr(packet[6:12]) + ' Protocol : ' + str(eth\_protocol)

print '--ETHERNET HEADER--'

print 'Version : ' + str(version) + ' IP Header Length : ' + str(ihl) + ' TTL : ' + str(ttl) + ' Protocol : ' + str(protocol) + ' Source Address : ' + str(s\_addr) + ' Destination Address : ' + str(d\_addr)

t = iph\_length + eth\_length

tcp\_header = packet[t:t+20]

#now unpack them :)

tcph = unpack('!HHLLBBHHH' , tcp\_header)

source\_port = tcph[0]

dest\_port = tcph[1]

sequence = tcph[2]

acknowledgement = tcph[3]

doff\_reserved = tcph[4]

tcph\_length = doff\_reserved >> 4

print '--TCP--'

print 'Source Port : ' + str(source\_port) + ' Dest Port : ' + str(dest\_port) + ' Sequence Number : ' + str(sequence) + ' Acknowledgement : ' + str(acknowledgement) + ' TCP header length : ' + str(tcph\_length)

h\_size = eth\_length + iph\_length + tcph\_length \* 4

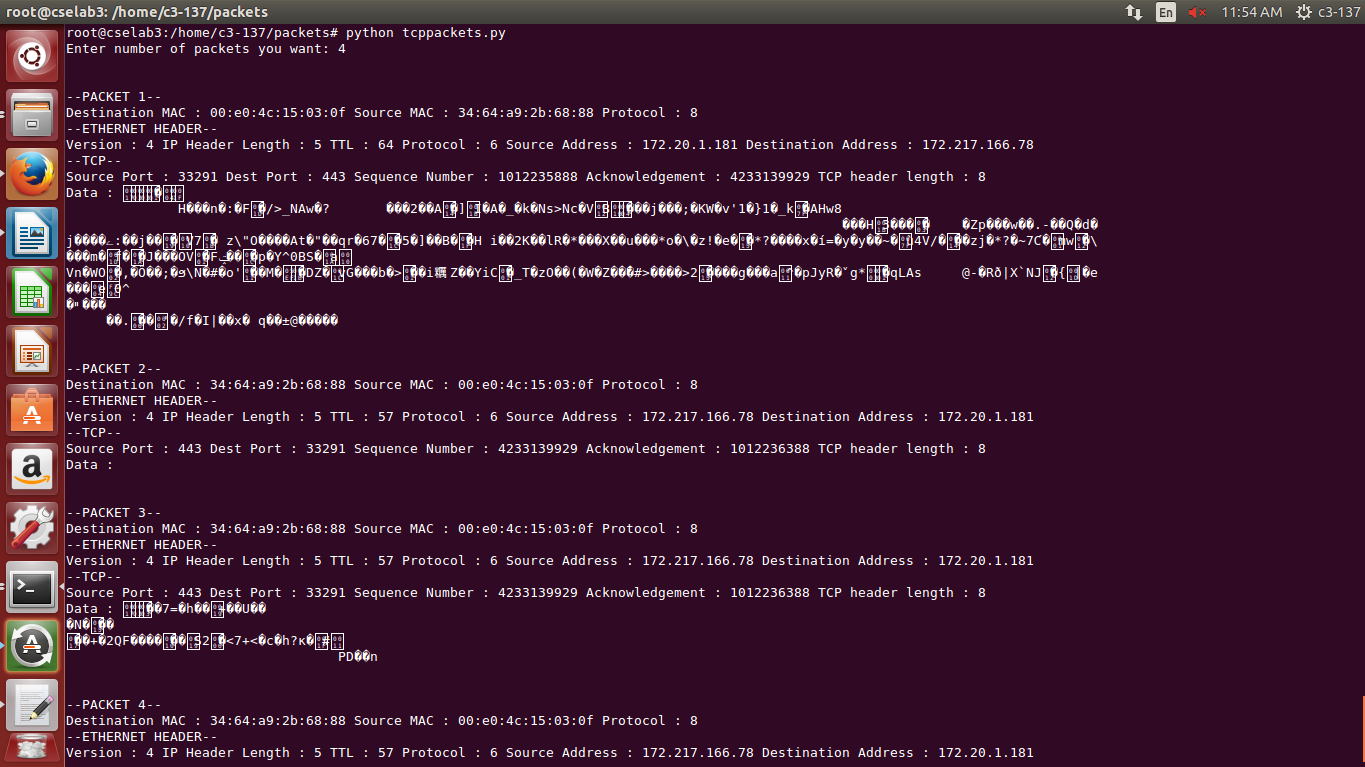
data\_size = len(packet) - h\_size

#get data from the packet

data = packet[h\_size:]

print 'Data : ' + data

**Output:**

****

**Task 4:** Write a program to extract and print the UDP segment information.

**Program:**

import socket, sys

from struct import \*

#Convert a string of 6 characters of ethernet address into a dash separated hex string

def eth\_addr (a) :

b = "%.2x:%.2x:%.2x:%.2x:%.2x:%.2x" % (ord(a[0]) , ord(a[1]) , ord(a[2]), ord(a[3]), ord(a[4]) , ord(a[5]))

return b

#create a AF\_PACKET type raw socket

try:

s = socket.socket( socket.AF\_PACKET , socket.SOCK\_RAW , socket.ntohs(0x0003))

except socket.error , msg:

print 'Socket could not be created. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]

sys.exit()

n = input('Enter number of packets you want: ')

i = 0

# receive a packet

while(i<n):

packet = s.recvfrom(65565)

#packet string from tuple

packet = packet[0]

#parse ethernet header

eth\_length = 14

eth\_header = packet[:eth\_length]

eth = unpack('!6s6sH' , eth\_header)

eth\_protocol = socket.ntohs(eth[2])

#Parse IP header

#take first 20 characters for the ip header

ip\_header = packet[eth\_length:20+eth\_length]

#now unpack them

iph = unpack('!BBHHHBBH4s4s' , ip\_header)

version\_ihl = iph[0]

version = version\_ihl >> 4

ihl = version\_ihl & 0xF

iph\_length = ihl \* 4

ttl = iph[5]

protocol = iph[6]

s\_addr = socket.inet\_ntoa(iph[8]);

d\_addr = socket.inet\_ntoa(iph[9]);

if protocol == 17 :

i+=1

print

print

print '--PACKET '+str(i)+'--'

print 'Destination MAC : ' + eth\_addr(packet[0:6]) + ' Source MAC : ' + eth\_addr(packet[6:12]) + ' Protocol : ' + str(eth\_protocol)

print '--UDP--'

u = iph\_length + eth\_length

udph\_length = 8

udp\_header = packet[u:u+8]

#now unpack them :)

udph = unpack('!HHHH' , udp\_header)

source\_port = udph[0]

dest\_port = udph[1]

length = udph[2]

checksum = udph[3]

print 'Source Port : ' + str(source\_port) + ' Dest Port : ' + str(dest\_port) + ' Length : ' + str(length) + ' Checksum : ' + str(checksum)

h\_size = eth\_length + iph\_length + udph\_length

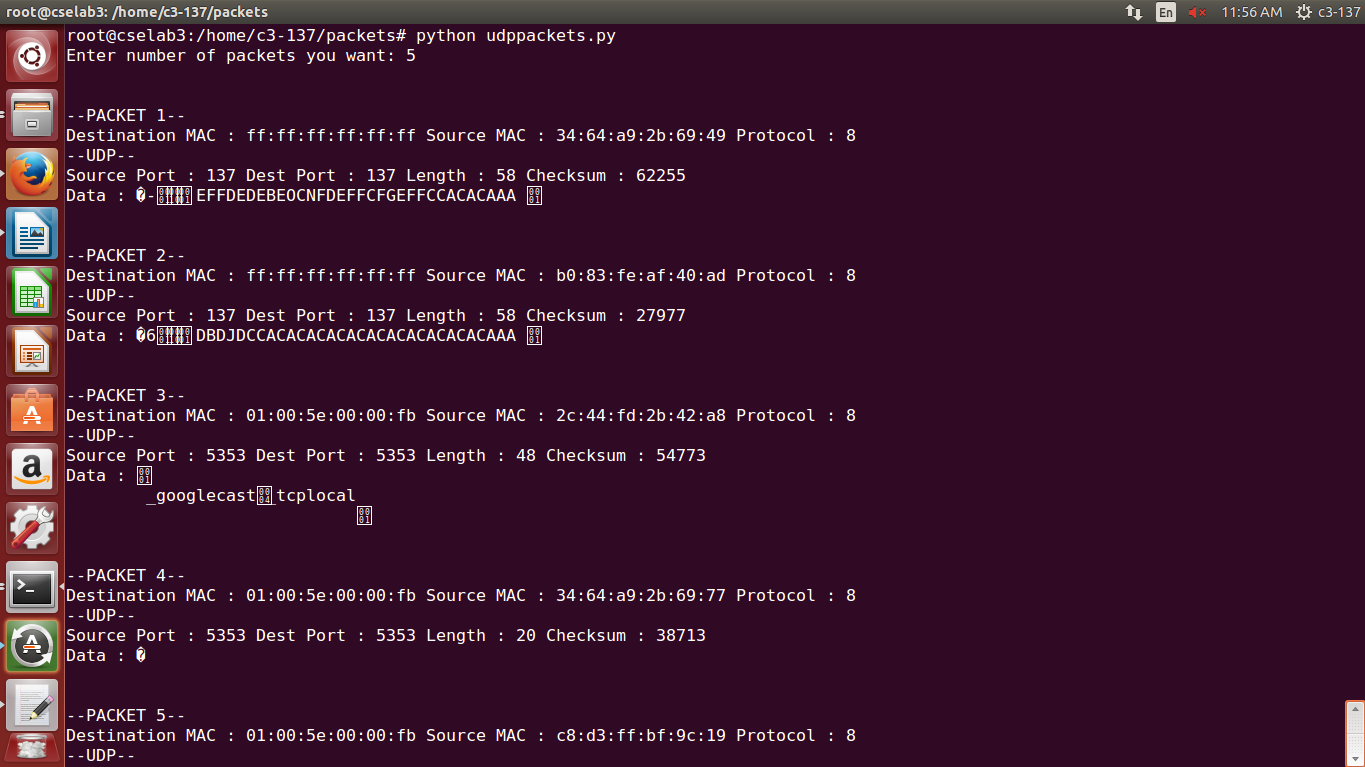
data\_size = len(packet) - h\_size

#get data from the packet

data = packet[h\_size:]

print 'Data : ' + data

**Output:**

****

**Task 5:** Write a program that captures 100 packets and display the count of ICMP, IGMP, TCP, UDP, and IP packets.

**Program:**

import socket, sys

from struct import \*

#Convert a string of 6 characters of ethernet address into a dash separated hex string

def eth\_addr (a) :

b = "%.2x:%.2x:%.2x:%.2x:%.2x:%.2x" % (ord(a[0]) , ord(a[1]) , ord(a[2]), ord(a[3]), ord(a[4]) , ord(a[5]))

return b

#create a AF\_PACKET type raw socket

try:

s = socket.socket( socket.AF\_PACKET , socket.SOCK\_RAW , socket.ntohs(0x0003))

except socket.error , msg:

print 'Socket could not be created. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]

sys.exit()

# receive a packet

icmp = 0

tcp = 0

udp = 0

igmp = 0

for i in range(0, 1000):

packet = s.recvfrom(65565)

#packet string from tuple

packet = packet[0]

eth\_length = 14

#Parse IP header

#take first 20 characters for the ip header

ip\_header = packet[eth\_length:20+eth\_length]

#now unpack them

iph = unpack('!BBHHHBBH4s4s' , ip\_header)

ttl = iph[5]

protocol = iph[6]

#TCP protocol

if protocol == 6 :

tcp+=1

#ICMP Packets

elif protocol == 1 :

icmp+=1

#UDP packets

elif protocol == 17 :

udp+=1

#IGMP Packets

elif protocol == 88 :

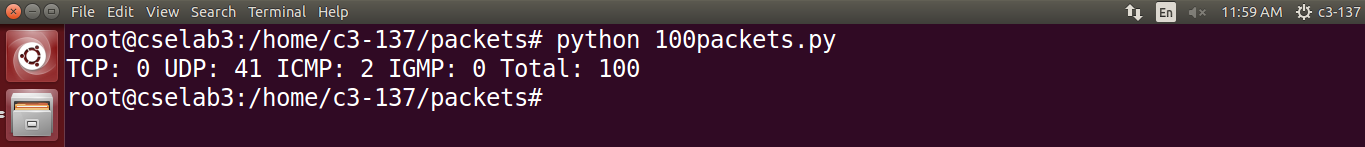
igmp+=1

sys.stdout.write("\rTCP: %d UDP: %d ICMP: %d IGMP: %d Total: %d" %(tcp, udp, icmp, igmp, i+1))

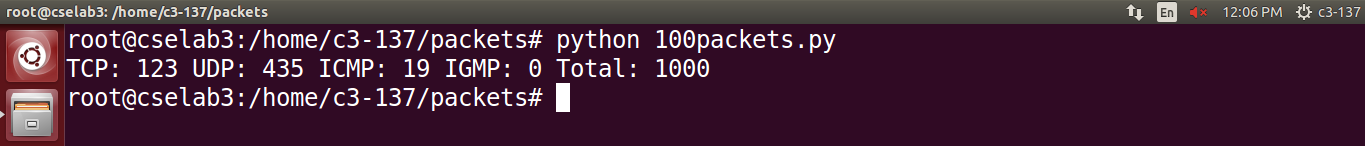
sys.stdout.flush()

print

**Output:**

****

For 1000 packets,



For 10,000 packets,

