

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('!HLLBBHHH' , 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 :)

```

```

    icmp_h = unpack('!BBH' , icmp_header)

    icmp_type = icmp_h[0]
    code = icmp_h[1]
    checksum = icmp_h[2]

    print 'Type : ' + str(icmp_type) + ' Code : ' + str(code)
+ ' Checksum : ' + str(checksum)

    h_size = eth_length + iph_length + icmp_h_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
    igmp_h_length = 8
    igmp_header = packet[u:u+8]

    #now unpack them :)
    igmp_h = 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:]

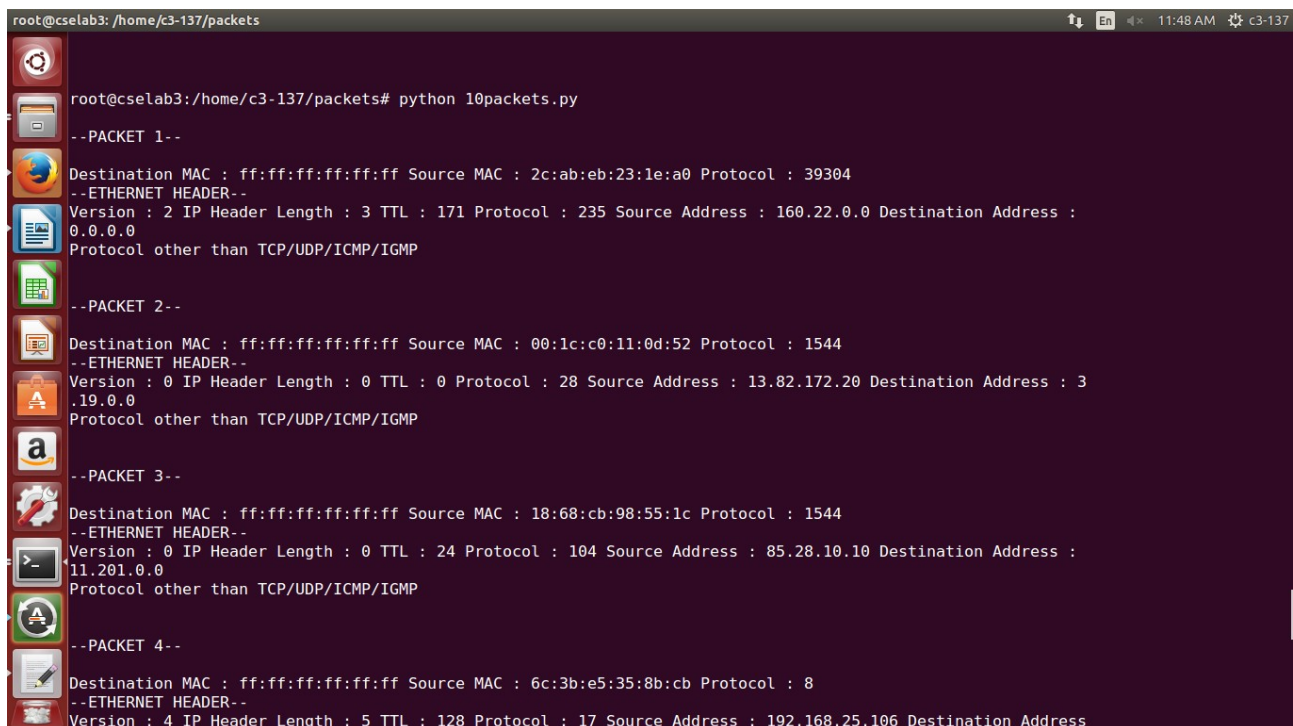
        print 'Data : ' + data

    #some other IP packet
    else :
        print 'Protocol other than TCP/UDP/ICMP/IGMP'

print

```

Output



```

root@c3-137: /home/c3-137/packets
root@c3-137:/home/c3-137/packets# python 10packets.py
--PACKET 1--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 2c:ab:eb:23:1e:a0 Protocol : 39304
--ETHERNET HEADER--
Version : 2 IP Header Length : 3 TTL : 171 Protocol : 235 Source Address : 160.22.0.0 Destination Address : 0.0.0.0
Protocol other than TCP/UDP/ICMP/IGMP
--PACKET 2--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 00:1c:c0:11:0d:52 Protocol : 1544
--ETHERNET HEADER--
Version : 0 IP Header Length : 0 TTL : 0 Protocol : 28 Source Address : 13.82.172.20 Destination Address : 3.19.0.0
Protocol other than TCP/UDP/ICMP/IGMP
--PACKET 3--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 18:68:cb:98:55:1c Protocol : 1544
--ETHERNET HEADER--
Version : 0 IP Header Length : 0 TTL : 24 Protocol : 104 Source Address : 85.28.10.10 Destination Address : 11.201.0.0
Protocol other than TCP/UDP/ICMP/IGMP
--PACKET 4--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 6c:3b:e5:35:8b:cb Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 128 Protocol : 17 Source Address : 192.168.25.106 Destination Address :

```

```
root@cslab3: /home/c3-137/packets
--PACKET 4--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 6c:3b:e5:35:8b:cb Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 128 Protocol : 17 Source Address : 192.168.25.106 Destination Address : 192.168.25.255
--UDP--
Source Port : 137 Dest Port : 137 Length : 58 Checksum : 4479
Data : 00000000FHFAEBEECACACACACACACACACACAA00

--PACKET 5--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 2c:ab:eb:22:4d:c4 Protocol : 39304
--ETHERNET HEADER--
Version : 2 IP Header Length : 3 TTL : 171 Protocol : 235 Source Address : 196.8.0.0 Destination Address : 0.0.0.0
Protocol other than TCP/UDP/ICMP/IGMP

--PACKET 6--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 00:e0:4c:15:03:0f Protocol : 1544
--ETHERNET HEADER--
Version : 0 IP Header Length : 0 TTL : 0 Protocol : 224 Source Address : 3.15.172.20 Destination Address : 0.1.0.0
Protocol other than TCP/UDP/ICMP/IGMP

--PACKET 7--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 2c:41:38:a8:ce:49 Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 128 Protocol : 17 Source Address : 172.20.5.74 Destination Address : 255.255.255.255
--UDP--
Source Port : 55706 Dest Port : 3956 Length : 16 Checksum : 9037
```

```
root@cslab3: /home/c3-137/packets
Source Port : 55706 Dest Port : 3956 Length : 16 Checksum : 9037
Data : 00000000000000000000000000000000

--PACKET 8--
Destination MAC : 00:e0:4c:15:03:0f Source MAC : 34:64:a9:2b:68:88 Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 64 Protocol : 1 Source Address : 172.20.1.181 Destination Address : 72.217.166.78
--ICMP--
Type : 8 Code : 0 Checksum : 6036
Data : 00000000000000000000000000000000!"#$%&'()*+,-./01234567

--PACKET 9--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 2c:41:38:a8:ce:49 Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 128 Protocol : 17 Source Address : 172.20.3.200 Destination Address : 172.20.7.255
--UDP--
Source Port : 137 Dest Port : 137 Length : 58 Checksum : 28683
Data : 00000000DBDJDCACACACACACACACACACACAA00

--PACKET 10--
Destination MAC : 34:64:a9:2b:68:88 Source MAC : 00:e0:4c:15:03:0f Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 54 Protocol : 1 Source Address : 172.217.166.78 Destination Address : 172.20.1.181
--ICMP--
Type : 0 Code : 0 Checksum : 8084
Data : 00000000000000000000000000000000!"#$%&'()*+,-./01234567
root@cslab3: /home/c3-137/packets#
```

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('!BBHHBHH4s4s' , 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('!HLLBBHHH' , 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
    icmp_length = 4
    icmp_header = packet[u:u+4]

```

```

    #now unpack them :)
    icmp_ph = unpack('!BBH' , icmp_header)

    icmp_type = icmp_ph[0]
    code = icmp_ph[1]
    checksum = icmp_ph[2]

    print 'Type : ' + str(icmp_type) + ' Code : ' + str(code)
+ ' Checksum : ' + str(checksum)

    h_size = eth_length + iph_length + icmp_ph_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
    igmp_ph_length = 8
    igmp_header = packet[u:u+8]

```



```

#now unpack them :)
igmp_header = unpack('!HHHH' , igmp_header)

source_port = igmp_header[0]
dest_port = igmp_header[1]
length = igmp_header[2]
checksum = igmp_header[3]

print 'Source Port : ' + str(source_port) + ' Dest Port : ' + str(dest_port) + ' Length : ' + str(length) + ' Checksum : ' + str(checksum)

h_size = eth_length + iph_length + igmp_header_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:

```

root@cse3lab3: /home/c3-137/packets
root@cse3lab3:/home/c3-137/packets# python npackets.py
Enter number of packets you want: 6

--PACKET 1--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 6c:3b:e5:35:8b:fc Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 128 Protocol : 17 Source Address : 192.168.25.105 Destination Address : 192.168.25.255
--UDP--
Source Port : 137 Dest Port : 137 Length : 58 Checksum : 10557
Data :
DBDJCCACACACACACACACACACACAA

--PACKET 2--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 18:68:cb:98:55:1c Protocol : 1544
--ETHERNET HEADER--
Version : 0 IP Header Length : 0 TTL : 24 Protocol : 104 Source Address : 85.28.10.10 Destination Address : 11.201.0.0
Protocol other than TCP/UDP/ICMP/IGMP

--PACKET 3--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : d4:c9:ef:f2:a7:e3 Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 128 Protocol : 17 Source Address : 192.168.25.30 Destination Address : 192.168.25.255
--UDP--
Source Port : 137 Dest Port : 137 Length : 58 Checksum : 65246
Data : 00000000EDEBEECNDDEACACACACACACACABM

--PACKET 4--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : d4:c9:ef:ef:c6:2b Protocol : 8
--ETHERNET HEADER--

```

```

root@c3-137: /home/c3-137/packets
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 128 Protocol : 17 Source Address : 192.168.25.30 Destination Address : 192.168.25.255
--UDP--
Source Port : 137 Dest Port : 137 Length : 58 Checksum : 65246
Data : 00000000EDEBEECNDDEECACACACACACACACABM 00

--PACKET 4--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : d4:c9:ef:ef:c6:2b Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 128 Protocol : 17 Source Address : 172.20.5.159 Destination Address : 172.20.7.255
--UDP--
Source Port : 138 Dest Port : 138 Length : 219 Checksum : 12190
Data : 00000000EJFEEMEBECDDCNDDBDGCACACACACAAA ABACFPFPENFDECFCFHFDEFPPACAB0SMB%000V00000000MAILSLOT\BROWSE
ITLAB303
00ITLAB3-116

--PACKET 5--
Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 34:64:a9:2b:64:a3 Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 128 Protocol : 17 Source Address : 172.20.5.80 Destination Address : 172.20.7.255
--UDP--
Source Port : 137 Dest Port : 137 Length : 58 Checksum : 23981
Data : 00000000FKEJECFCBFBGEPFAEMCOFCFFCACACAAA 00

--PACKET 6--
Destination MAC : 33:33:00:01:00:03 Source MAC : 78:2b:cb:8d:3e:39 Protocol : 56710
--ETHERNET HEADER--
Version : 6 IP Header Length : 0 TTL : 254 Protocol : 128 Source Address : 0.0.0.0 Destination Address : 149.186.27.148
Protocol other than TCP/UDP/ICMP/IGMP

root@c3-137: /home/c3-137/packets#

```

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('!HLLBBHHH' , tcp_header)

    source_port = tcph[0]
    dest_port = tcph[1]
    sequence = tcph[2]
    acknowledgement = tcph[3]

```

Output:

The image shows a Kali Linux desktop environment with a terminal window open. The terminal displays the output of a tcpdump command, showing details for four captured packets. The interface includes a sidebar with various application icons (like a web browser, file manager, and terminal) and a top status bar indicating the time as 11:54 AM and the user as root. The terminal output is as follows:

root@cslab3:/home/c3-137/packets# python tcpdump.py
Enter number of packets you want: 4

--PACKET 1--
Destination MAC : 00:e0:4c:15:03:0f Source MAC : 34:64:a9:2b:68:88 Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 64 Protocol : 6 Source Address : 172.20.1.181 Destination Address : 172.217.166.78
--TCP--
Source Port : 33291 Dest Port : 443 Sequence Number : 1012235888 Acknowledgement : 4233139929 TCP header length : 8
Data : [REDACTED]

--PACKET 2--
Destination MAC : 34:64:a9:2b:68:88 Source MAC : 00:e0:4c:15:03:0f Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 57 Protocol : 6 Source Address : 172.217.166.78 Destination Address : 172.20.1.181
--TCP--
Source Port : 443 Dest Port : 33291 Sequence Number : 4233139929 Acknowledgement : 1012236388 TCP header length : 8
Data : [REDACTED]

--PACKET 3--
Destination MAC : 34:64:a9:2b:68:88 Source MAC : 00:e0:4c:15:03:0f Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 57 Protocol : 6 Source Address : 172.217.166.78 Destination Address : 172.20.1.181
--TCP--
Source Port : 443 Dest Port : 33291 Sequence Number : 4233139929 Acknowledgement : 1012236388 TCP header length : 8
Data : [REDACTED]

--PACKET 4--
Destination MAC : 34:64:a9:2b:68:88 Source MAC : 00:e0:4c:15:03:0f Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 57 Protocol : 6 Source Address : 172.217.166.78 Destination Address : 172.20.1.181

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('!BBHHBHH4s4s' , 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)+'--'

```


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

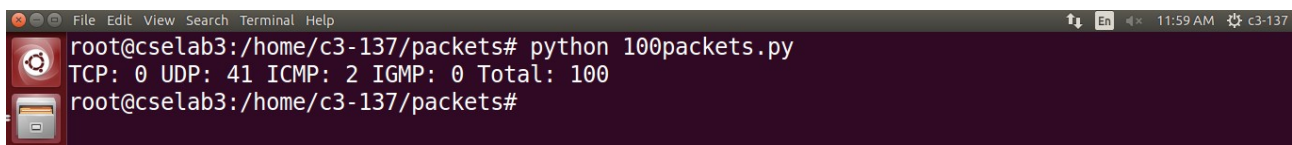
    #ICMP Packets
    elif protocol == 1 :
        icmp+=1

    #UDP packets
```



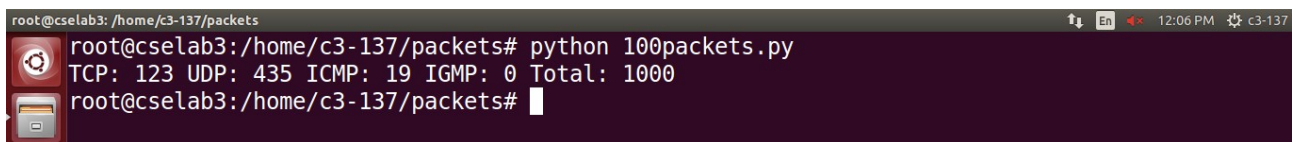
```
elif protocol == 17 :  
    udp+=1  
  
#IGMP Packets  
elif protocol == 88 :  
    igmp+=1  
    sys.stdout.write("\rTCP: %d UDP: %d ICMP: %d IGMP: %d Total:  
%d" %(tcp, udp, icmp, igmp, i+1))  
    sys.stdout.flush()  
print
```

Output:

A terminal window titled 'root@c3-137' with a menu bar (File, Edit, View, Search, Terminal, Help) and a status bar (11:59 AM, c3-137). The prompt is 'root@c3-137:/home/c3-137/packets#'. The command 'python 100packets.py' has been executed, resulting in the output 'TCP: 0 UDP: 41 ICMP: 2 IGMP: 0 Total: 100'.

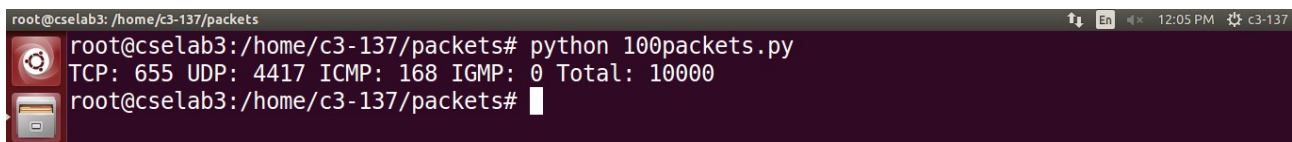
```
root@c3-137:/home/c3-137/packets# python 100packets.py  
TCP: 0 UDP: 41 ICMP: 2 IGMP: 0 Total: 100  
root@c3-137:/home/c3-137/packets#
```

For 1000 packets,

A terminal window titled 'root@c3-137' with a menu bar (File, Edit, View, Search, Terminal, Help) and a status bar (12:06 PM, c3-137). The prompt is 'root@c3-137:/home/c3-137/packets#'. The command 'python 100packets.py' has been executed, resulting in the output 'TCP: 123 UDP: 435 ICMP: 19 IGMP: 0 Total: 1000'.

```
root@c3-137:/home/c3-137/packets# python 100packets.py  
TCP: 123 UDP: 435 ICMP: 19 IGMP: 0 Total: 1000  
root@c3-137:/home/c3-137/packets#
```

For 10,000 packets,

A terminal window titled 'root@c3-137' with a menu bar (File, Edit, View, Search, Terminal, Help) and a status bar (12:05 PM, c3-137). The prompt is 'root@c3-137:/home/c3-137/packets#'. The command 'python 100packets.py' has been executed, resulting in the output 'TCP: 655 UDP: 4417 ICMP: 168 IGMP: 0 Total: 10000'.

```
root@c3-137:/home/c3-137/packets# python 100packets.py  
TCP: 655 UDP: 4417 ICMP: 168 IGMP: 0 Total: 10000  
root@c3-137:/home/c3-137/packets#
```