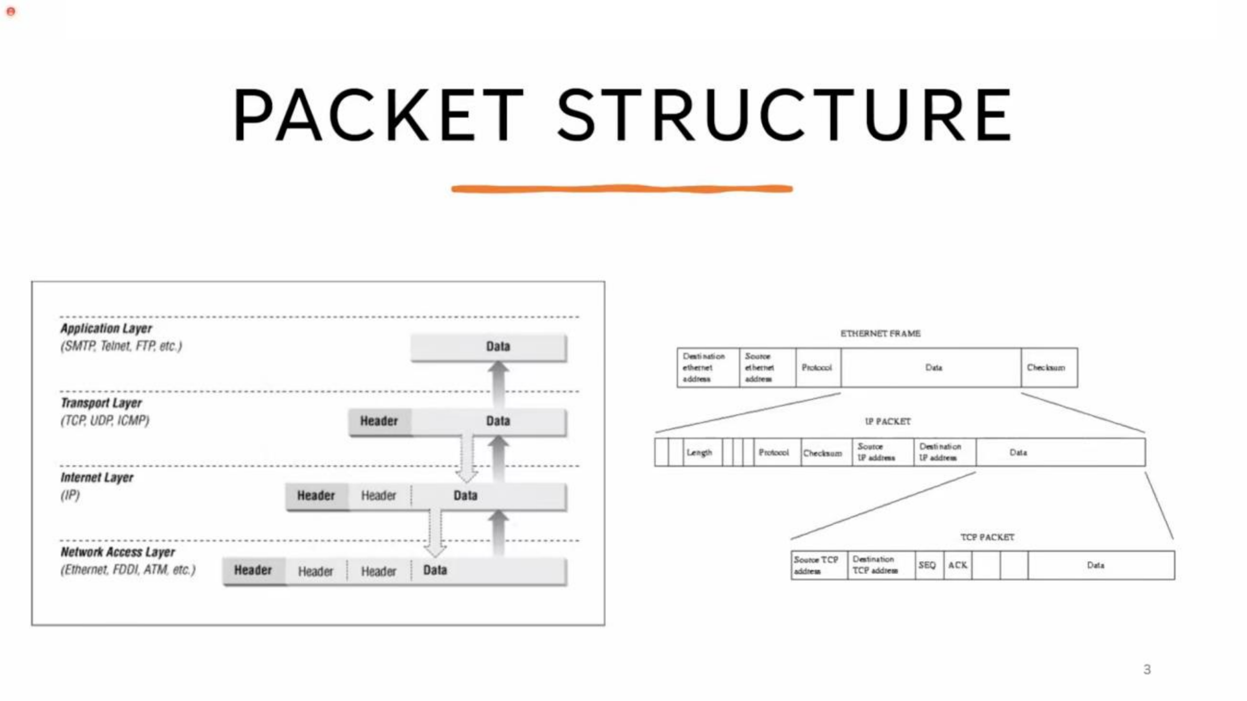
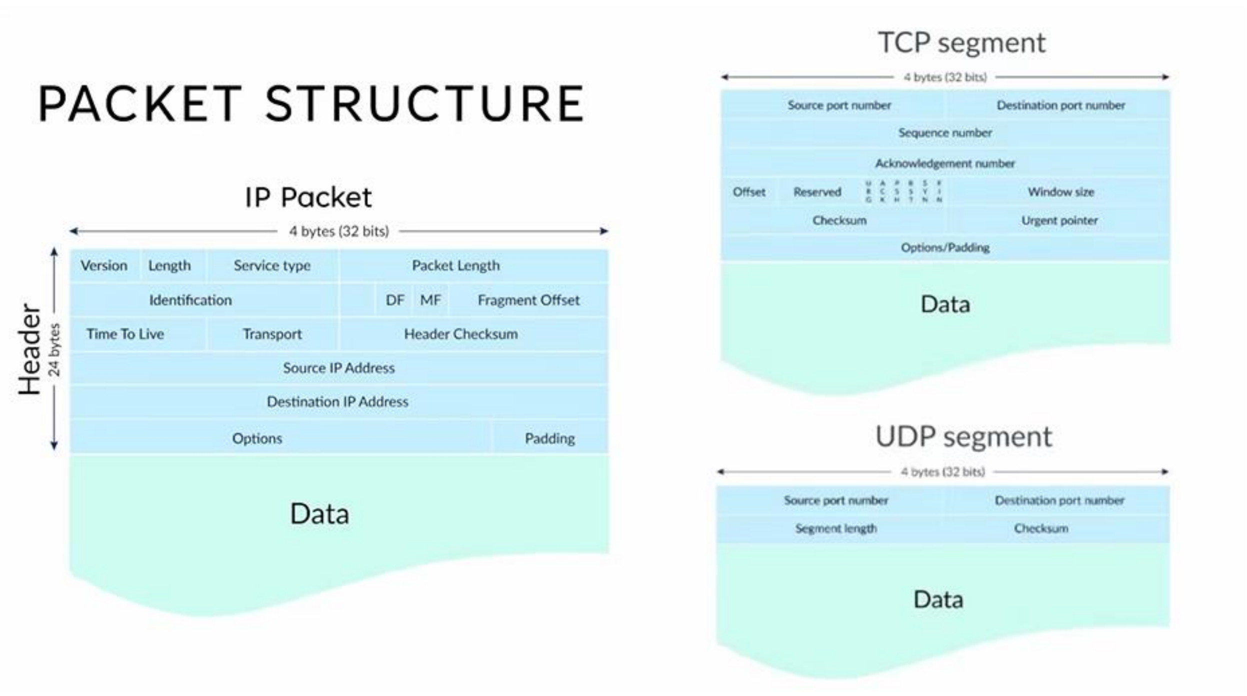
Packet Dissection Using Python.



PACKET STRUCTURE



PACKET STRUCTURE

IP Packet

4 bytes (32 bits)

Header 24 bytes

Version	Length	Service type	Packet Length		
Identification			DF	MF	Fragment Offset
Time To Live		Transport	Header Checksum		
Source IP Address					
Destination IP Address					
Options					Padding

Data

TCP segment

4 bytes (32 bits)

Source port number			Destination port number
Sequence number			
Acknowledgement number			
Offset	Reserved	URG ACK PSH RST SYN FIN	Window size
Checksum			Urgent pointer
Options/Padding			

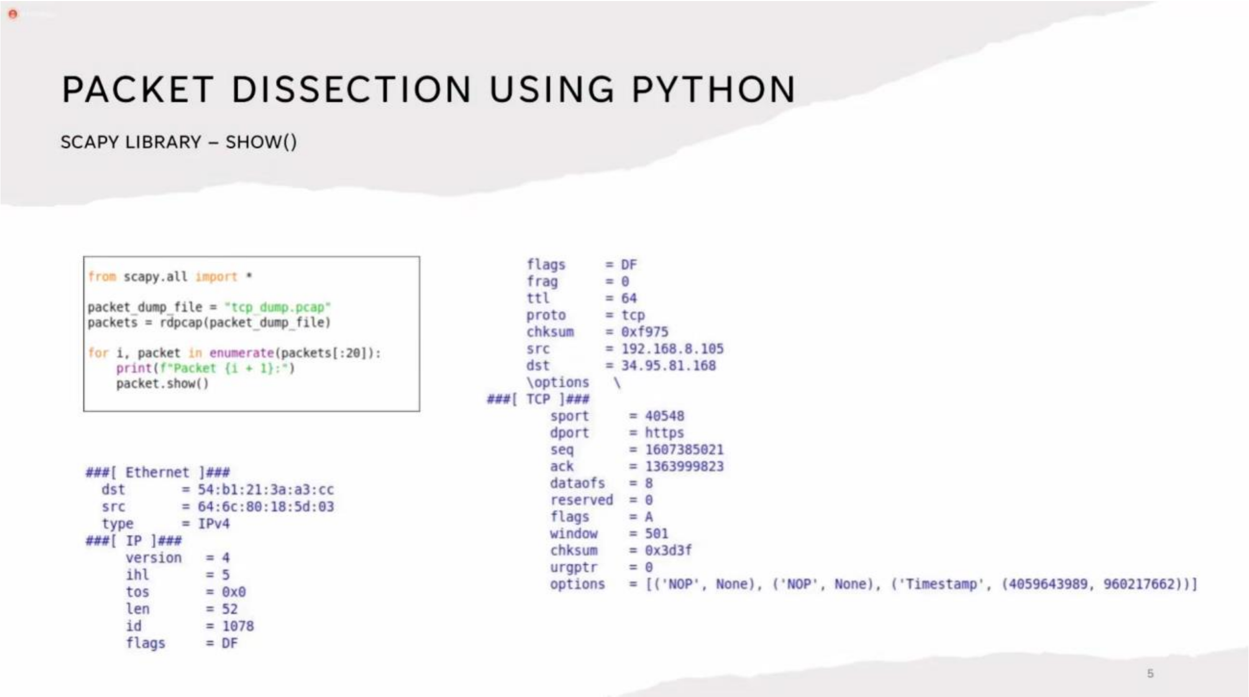
Data

UDP segment

4 bytes (32 bits)

Source port number	Destination port number
Segment length	Checksum

Data

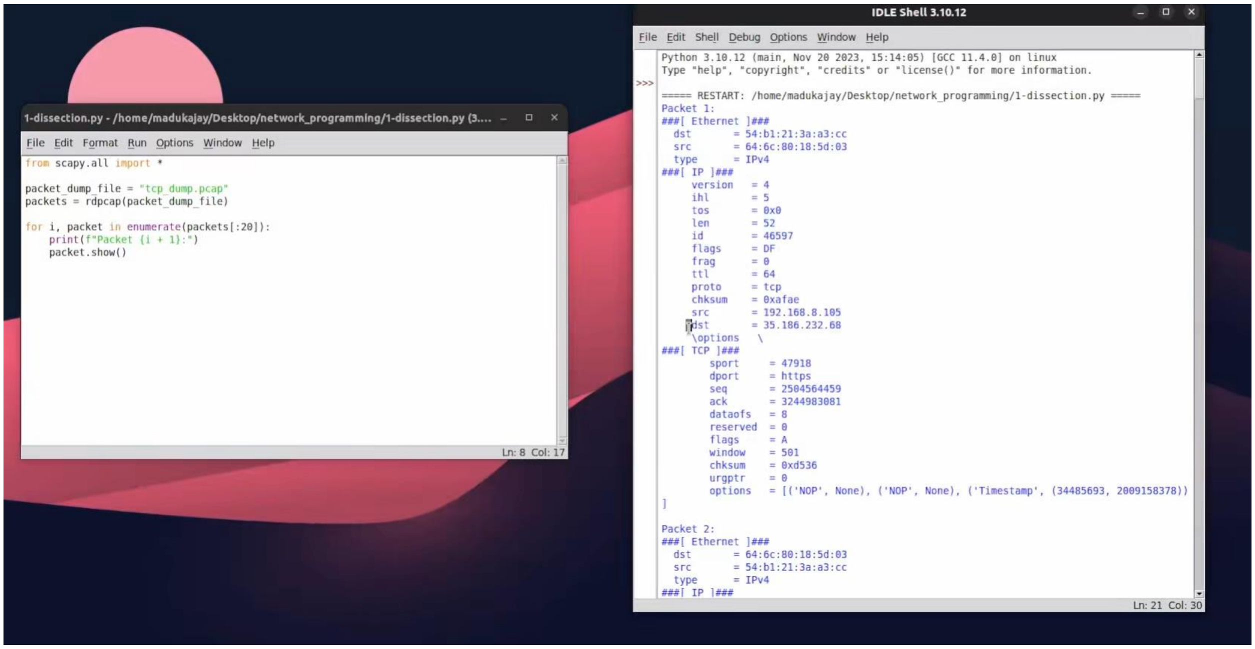


PACKET DISSECTION USING PYTHON

SCAPY LIBRARY – SHOW()

```python
from scapy.all import *

packet_dump_file = "tcp_dump.pcap"
packets = rdpcap(packet_dump_file)

for i, packet in enumerate(packets[:20]):
    print(f"Packet {i + 1}:")
    packet.show()
```

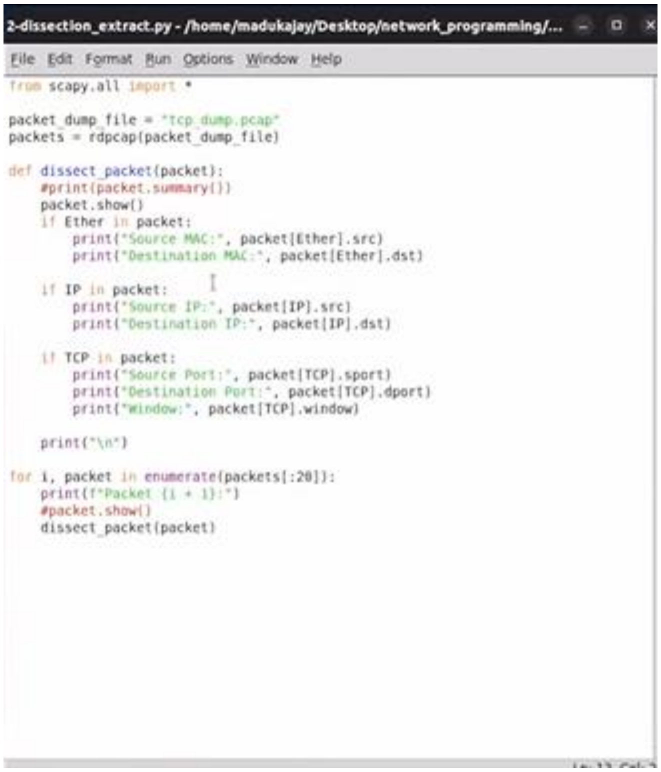
```
###[ Ethernet ]###
  dst       = 54:b1:21:3a:a3:cc
  src       = 64:6c:80:18:5d:03
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 52
     id        = 1078
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xf975
     src       = 192.168.8.105
     dst       = 34.95.81.168
     \options   \
###[ TCP ]###
        sport     = 40548
        dport     = https
        seq       = 1607385021
        ack       = 1363999823
        dataofs   = 8
        reserved  = 0
        flags     = A
        window    = 501
        chksum    = 0x3d3f
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (4059643989, 960217662))]
```



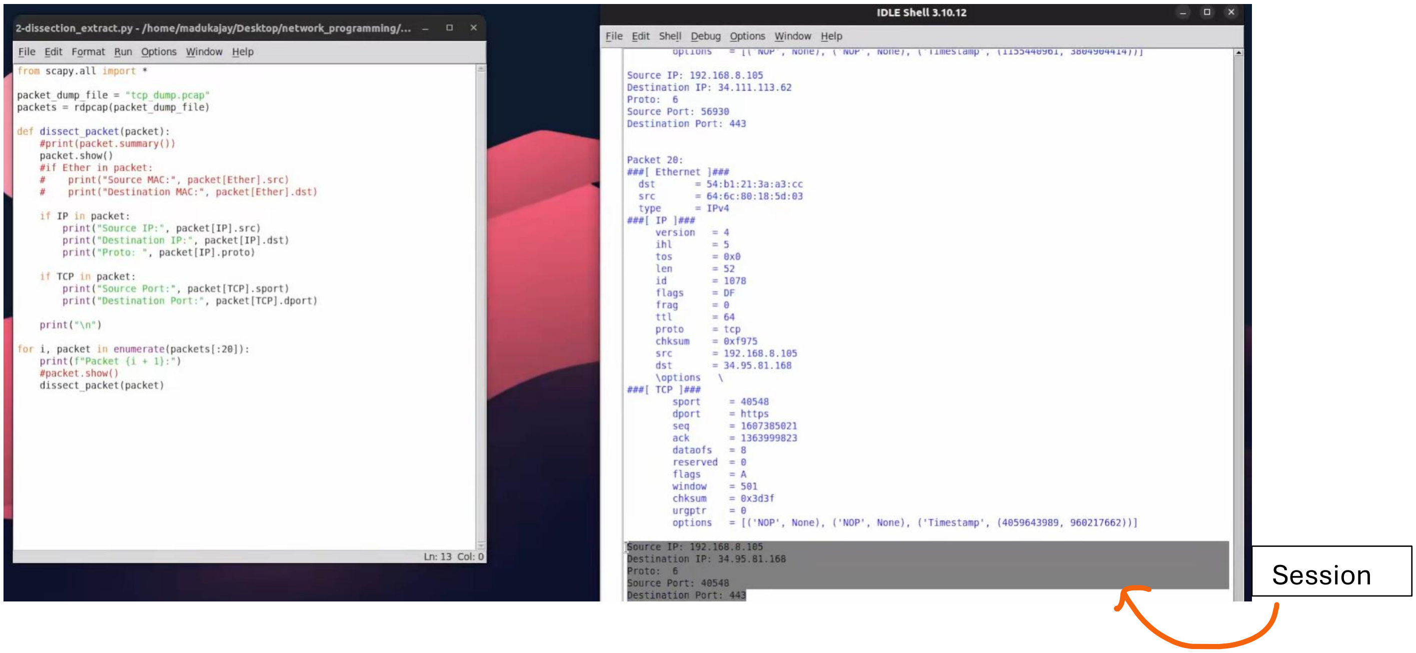
1-dissection.py - /home/madukajay/Desktop/network_programming/1-dissection.py (3....
File Edit Format Run Options Window Help
Ln: 8 Col: 17
IDLE Shell 3.10.12
File Edit Shell Debug Options Window Help
Ln: 21 Col: 30

```python
from scapy.all import *

packet_dump_file = "tcp_dump.pcap"
packets = rdpcap(packet_dump_file)

for i, packet in enumerate(packets[:20]):
    print(f"Packet {i + 1}:")
    packet.show()
```

```
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/madukajay/Desktop/network_programming/1-dissection.py =====
Packet 1:
###[ Ethernet ]###
  dst       = 54:b1:21:3a:a3:cc
  src       = 64:6c:80:18:5d:03
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 52
     id        = 46597
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xafae
     src       = 192.168.8.105
     dst       = 35.186.232.68
     \options   \
###[ TCP ]###
        sport     = 47918
        dport     = https
        seq       = 2504564459
        ack       = 3244983081
        dataofs   = 8
        reserved  = 0
        flags     = A
        window    = 501
        chksum    = 0xd536
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (34485693, 2009158378))
]

Packet 2:
###[ Ethernet ]###
  dst       = 64:6c:80:18:5d:03
  src       = 54:b1:21:3a:a3:cc
  type      = IPv4
###[ IP ]###
```

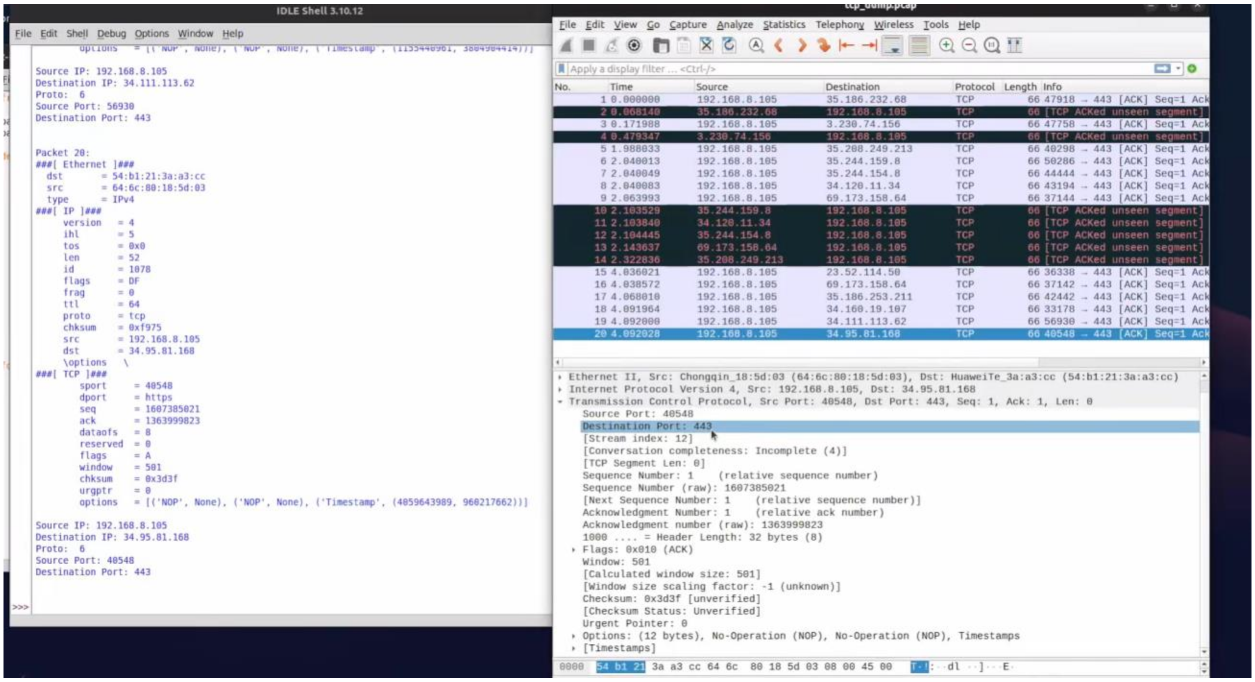

2-dissection_extract.py - /home/madukajay/Desktop/network_programming/...
File Edit Format Run Options Window Help

```python
from scapy.all import *

packet_dump_file = "tcp_dump.pcap"
packets = rdpcap(packet_dump_file)

def dissect_packet(packet):
    #print(packet.summary())
    packet.show()
    if Ether in packet:
        print("Source MAC:", packet[Ether].src)
        print("Destination MAC:", packet[Ether].dst)

    if IP in packet:
        print("Source IP:", packet[IP].src)
        print("Destination IP:", packet[IP].dst)

    if TCP in packet:
        print("Source Port:", packet[TCP].sport)
        print("Destination Port:", packet[TCP].dport)
        print("Window:", packet[TCP].window)

    print("\n")

for i, packet in enumerate(packets[:20]):
    print(f"Packet {i + 1}:")
    #packet.show()
    dissect_packet(packet)
```


2-dissection_extract.py - /home/madukajay/Desktop/network_programming/...
File Edit Format Run Options Window Help
Ln: 13 Col: 0
IDLE Shell 3.10.12
File Edit Shell Debug Options Window Help
Session

```python
from scapy.all import *

packet_dump_file = "tcp_dump.pcap"
packets = rdpcap(packet_dump_file)

def dissect_packet(packet):
    #print(packet.summary())
    packet.show()
    #if Ether in packet:
    #    print("Source MAC:", packet[Ether].src)
    #    print("Destination MAC:", packet[Ether].dst)

    if IP in packet:
        print("Source IP:", packet[IP].src)
        print("Destination IP:", packet[IP].dst)
        print("Proto: ", packet[IP].proto)

    if TCP in packet:
        print("Source Port:", packet[TCP].sport)
        print("Destination Port:", packet[TCP].dport)

    print("\n")

for i, packet in enumerate(packets[:20]):
    print(f"Packet {i + 1}:")
    #packet.show()
    dissect_packet(packet)
```

```
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (1155440961, 3804904414))]

Source IP: 192.168.8.105
Destination IP: 34.111.113.62
Proto:  6
Source Port: 56930
Destination Port: 443


Packet 20:
###[ Ethernet ]###
  dst       = 54:b1:21:3a:a3:cc
  src       = 64:6c:80:18:5d:03
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 52
     id        = 1078
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xf975
     src       = 192.168.8.105
     dst       = 34.95.81.168
     \options   \
###[ TCP ]###
        sport     = 40548
        dport     = https
        seq       = 1607385021
        ack       = 1363999823
        dataofs   = 8
        reserved  = 0
        flags     = A
        window    = 501
        chksum    = 0x3d3f
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (4059643989, 960217662))]

Source IP: 192.168.8.105
Destination IP: 34.95.81.168
Proto:  6
Source Port: 40548
Destination Port: 443
```

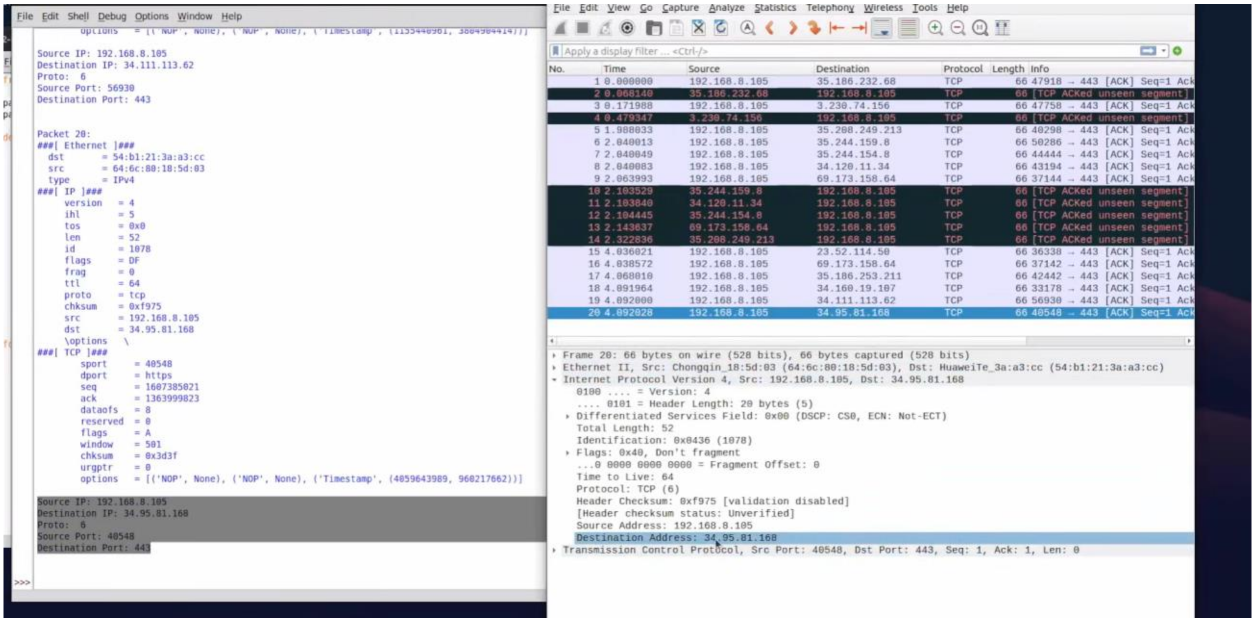


```
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (1135440961, 3804904414))]

Source IP: 192.168.8.105
Destination IP: 34.111.113.62
Proto:  6
Source Port: 56930
Destination Port: 443


Packet 20:
###[ Ethernet ]###
  dst       = 54:b1:21:3a:a3:cc
  src       = 64:6c:80:18:5d:03
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 52
     id        = 1078
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xf975
     src       = 192.168.8.105
     dst       = 34.95.81.168
     \options   \
###[ TCP ]###
        sport     = 40548
        dport     = https
        seq       = 1607385021
        ack       = 1363999823
        dataofs   = 8
        reserved  = 0
        flags     = A
        window    = 501
        chksum    = 0x3d3f
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (4059643989, 960217662))]

Source IP: 192.168.8.105
Destination IP: 34.95.81.168
Proto:  6
Source Port: 40548
Destination Port: 443

>>>
```

tcp_dump.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time        Source           Destination      Protocol Length Info
   1 0.000000    192.168.8.105    35.186.232.68    TCP      66 47918 → 443 [ACK] Seq=1 Ack
   2 0.068140    35.186.232.68    192.168.8.105    TCP      66 [TCP ACKed unseen segment]
   3 0.171988    192.168.8.105    3.230.74.156     TCP      66 47758 → 443 [ACK] Seq=1 Ack
   4 0.479347    3.230.74.156     192.168.8.105    TCP      66 [TCP ACKed unseen segment]
   5 1.988033    192.168.8.105    35.208.249.213   TCP      66 40298 → 443 [ACK] Seq=1 Ack
   6 2.040013    192.168.8.105    35.244.159.8     TCP      66 50286 → 443 [ACK] Seq=1 Ack
   7 2.040049    192.168.8.105    35.244.154.8     TCP      66 44444 → 443 [ACK] Seq=1 Ack
   8 2.040083    192.168.8.105    34.120.11.34     TCP      66 43194 → 443 [ACK] Seq=1 Ack
   9 2.063993    192.168.8.105    69.173.158.64    TCP      66 37144 → 443 [ACK] Seq=1 Ack
  10 2.103529    35.244.159.8     192.168.8.105    TCP      66 [TCP ACKed unseen segment]
  11 2.103840    34.120.11.34     192.168.8.105    TCP      66 [TCP ACKed unseen segment]
  12 2.104445    35.244.154.8     192.168.8.105    TCP      66 [TCP ACKed unseen segment]
  13 2.143637    69.173.158.64    192.168.8.105    TCP      66 [TCP ACKed unseen segment]
  14 2.322836    35.208.249.213   192.168.8.105    TCP      66 [TCP ACKed unseen segment]
  15 4.036021    192.168.8.105    23.52.114.50     TCP      66 36338 → 443 [ACK] Seq=1 Ack
  16 4.038572    192.168.8.105    69.173.158.64    TCP      66 37142 → 443 [ACK] Seq=1 Ack
  17 4.068010    192.168.8.105    35.186.253.211   TCP      66 42442 → 443 [ACK] Seq=1 Ack
  18 4.091964    192.168.8.105    34.160.19.107    TCP      66 33178 → 443 [ACK] Seq=1 Ack
  19 4.092000    192.168.8.105    34.111.113.62    TCP      66 56930 → 443 [ACK] Seq=1 Ack
  20 4.092028    192.168.8.105    34.95.81.168     TCP      66 40548 → 443 [ACK] Seq=1 Ack
```

```
▸ Ethernet II, Src: Chongqin_18:5d:03 (64:6c:80:18:5d:03), Dst: HuaweiTe_3a:a3:cc (54:b1:21:3a:a3:cc)
▸ Internet Protocol Version 4, Src: 192.168.8.105, Dst: 34.95.81.168
▾ Transmission Control Protocol, Src Port: 40548, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
    Source Port: 40548
    Destination Port: 443
    [Stream index: 12]
    [Conversation completeness: Incomplete (4)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 1607385021
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1363999823
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window: 501
    [Calculated window size: 501]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x3d3f [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [Timestamps]
0000  54 b1 21 3a a3 cc 64 6c  80 18 5d 03 08 00 45 00   T·!:··dl ··]···E·
```

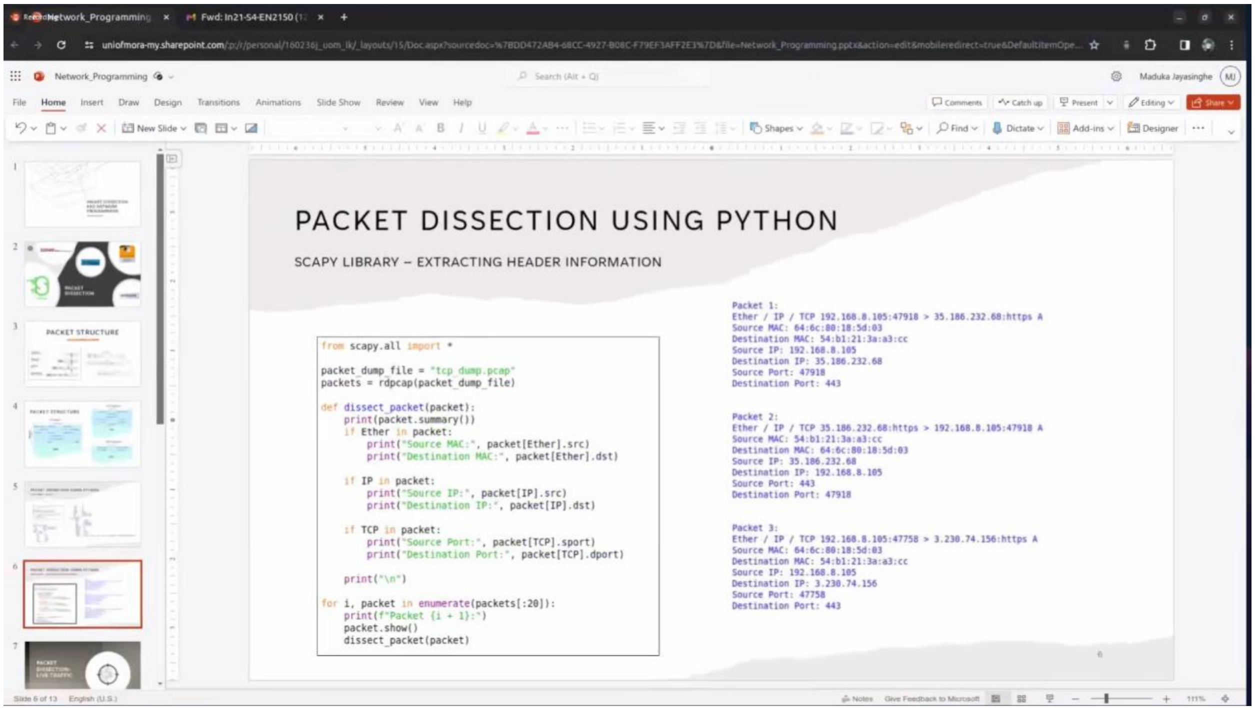


```
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (1135440961, 3804904414))]

Source IP: 192.168.8.105
Destination IP: 34.111.113.62
Proto:  6
Source Port: 56930
Destination Port: 443


Packet 20:
###[ Ethernet ]###
  dst       = 54:b1:21:3a:a3:cc
  src       = 64:6c:80:18:5d:03
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 52
     id        = 1078
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xf975
     src       = 192.168.8.105
     dst       = 34.95.81.168
     \options   \
###[ TCP ]###
        sport     = 40548
        dport     = https
        seq       = 1607385021
        ack       = 1363999823
        dataofs   = 8
        reserved  = 0
        flags     = A
        window    = 501
        chksum    = 0x3d3f
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (4059643989, 960217662))]

Source IP: 192.168.8.105
Destination IP: 34.95.81.168
Proto:  6
Source Port: 40548
Destination Port: 443

>>>
```

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time        Source           Destination      Protocol Length Info
   1 0.000000    192.168.8.105    35.186.232.68    TCP      66 47918 → 443 [ACK] Seq=1 Ack
   2 0.068140    35.186.232.68    192.168.8.105    TCP      66 [TCP ACKed unseen segment]
   3 0.171988    192.168.8.105    3.230.74.156     TCP      66 47758 → 443 [ACK] Seq=1 Ack
   4 0.479347    3.230.74.156     192.168.8.105    TCP      66 [TCP ACKed unseen segment]
   5 1.988033    192.168.8.105    35.208.249.213   TCP      66 40298 → 443 [ACK] Seq=1 Ack
   6 2.040013    192.168.8.105    35.244.159.8     TCP      66 50286 → 443 [ACK] Seq=1 Ack
   7 2.040049    192.168.8.105    35.244.154.8     TCP      66 44444 → 443 [ACK] Seq=1 Ack
   8 2.040083    192.168.8.105    34.120.11.34     TCP      66 43194 → 443 [ACK] Seq=1 Ack
   9 2.063993    192.168.8.105    69.173.158.64    TCP      66 37144 → 443 [ACK] Seq=1 Ack
  10 2.103529    35.244.159.8     192.168.8.105    TCP      66 [TCP ACKed unseen segment]
  11 2.103840    34.120.11.34     192.168.8.105    TCP      66 [TCP ACKed unseen segment]
  12 2.104445    35.244.154.8     192.168.8.105    TCP      66 [TCP ACKed unseen segment]
  13 2.143637    69.173.158.64    192.168.8.105    TCP      66 [TCP ACKed unseen segment]
  14 2.322836    35.208.249.213   192.168.8.105    TCP      66 [TCP ACKed unseen segment]
  15 4.036021    192.168.8.105    23.52.114.50     TCP      66 36338 → 443 [ACK] Seq=1 Ack
  16 4.038572    192.168.8.105    69.173.158.64    TCP      66 37142 → 443 [ACK] Seq=1 Ack
  17 4.068010    192.168.8.105    35.186.253.211   TCP      66 42442 → 443 [ACK] Seq=1 Ack
  18 4.091964    192.168.8.105    34.160.19.107    TCP      66 33178 → 443 [ACK] Seq=1 Ack
  19 4.092000    192.168.8.105    34.111.113.62    TCP      66 56930 → 443 [ACK] Seq=1 Ack
  20 4.092028    192.168.8.105    34.95.81.168     TCP      66 40548 → 443 [ACK] Seq=1 Ack
```

```
▸ Frame 20: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
▸ Ethernet II, Src: Chongqin_18:5d:03 (64:6c:80:18:5d:03), Dst: HuaweiTe_3a:a3:cc (54:b1:21:3a:a3:cc)
▾ Internet Protocol Version 4, Src: 192.168.8.105, Dst: 34.95.81.168
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0x0436 (1078)
  ▸ Flags: 0x40, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0xf975 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.8.105
    Destination Address: 34.95.81.168
▸ Transmission Control Protocol, Src Port: 40548, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
```



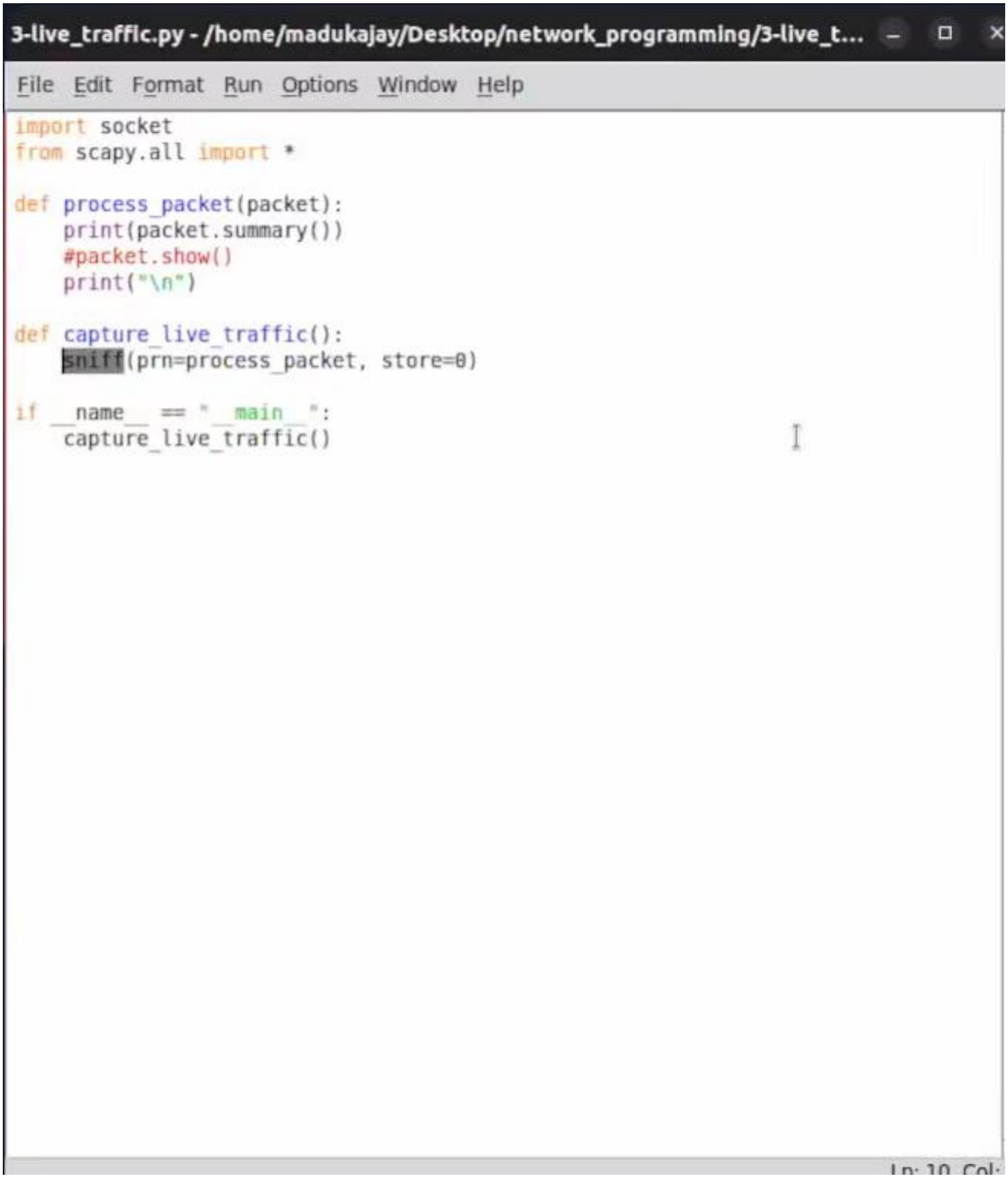
PACKET DISSECTION USING PYTHON

SCAPY LIBRARY – EXTRACTING HEADER INFORMATION

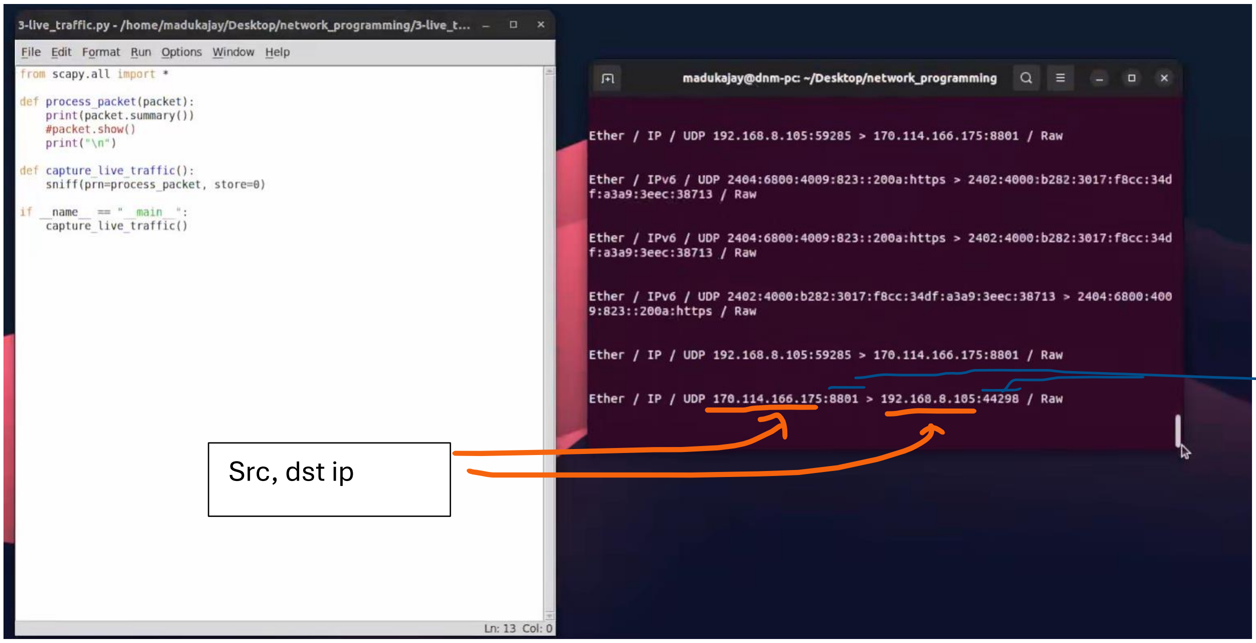
```python
from scapy.all import *

packet_dump_file = "tcp_dump.pcap"
packets = rdpcap(packet_dump_file)

def dissect_packet(packet):
    print(packet.summary())
    if Ether in packet:
        print("Source MAC:", packet[Ether].src)
        print("Destination MAC:", packet[Ether].dst)

    if IP in packet:
        print("Source IP:", packet[IP].src)
        print("Destination IP:", packet[IP].dst)

    if TCP in packet:
        print("Source Port:", packet[TCP].sport)
        print("Destination Port:", packet[TCP].dport)

    print("\n")

for i, packet in enumerate(packets[:20]):
    print(f"Packet {i + 1}:")
    packet.show()
    dissect_packet(packet)
```

```
Packet 1:
Ether / IP / TCP 192.168.8.105:47918 > 35.186.232.68:https A
Source MAC: 64:6c:80:18:5d:03
Destination MAC: 54:b1:21:3a:a3:cc
Source IP: 192.168.8.105
Destination IP: 35.186.232.68
Source Port: 47918
Destination Port: 443


Packet 2:
Ether / IP / TCP 35.186.232.68:https > 192.168.8.105:47918 A
Source MAC: 54:b1:21:3a:a3:cc
Destination MAC: 64:6c:80:18:5d:03
Source IP: 35.186.232.68
Destination IP: 192.168.8.105
Source Port: 443
Destination Port: 47918


Packet 3:
Ether / IP / TCP 192.168.8.105:47758 > 3.230.74.156:https A
Source MAC: 64:6c:80:18:5d:03
Destination MAC: 54:b1:21:3a:a3:cc
Source IP: 192.168.8.105
Destination IP: 3.230.74.156
Source Port: 47758
Destination Port: 443
```

PACKET DISSECTION- LIVE TRAFFIC


3-live_traffic.py - /home/madukajay/Desktop/network_programming/3-live_t...
File Edit Format Run Options Window Help
Ln: 10 Col:

```
import socket
from scapy.all import *

def process_packet(packet):
    print(packet.summary())
    #packet.show()
    print("\n")

def capture_live_traffic():
    sniff(prn=process_packet, store=0)

if __name__ == "__main__":
    capture_live_traffic()
```

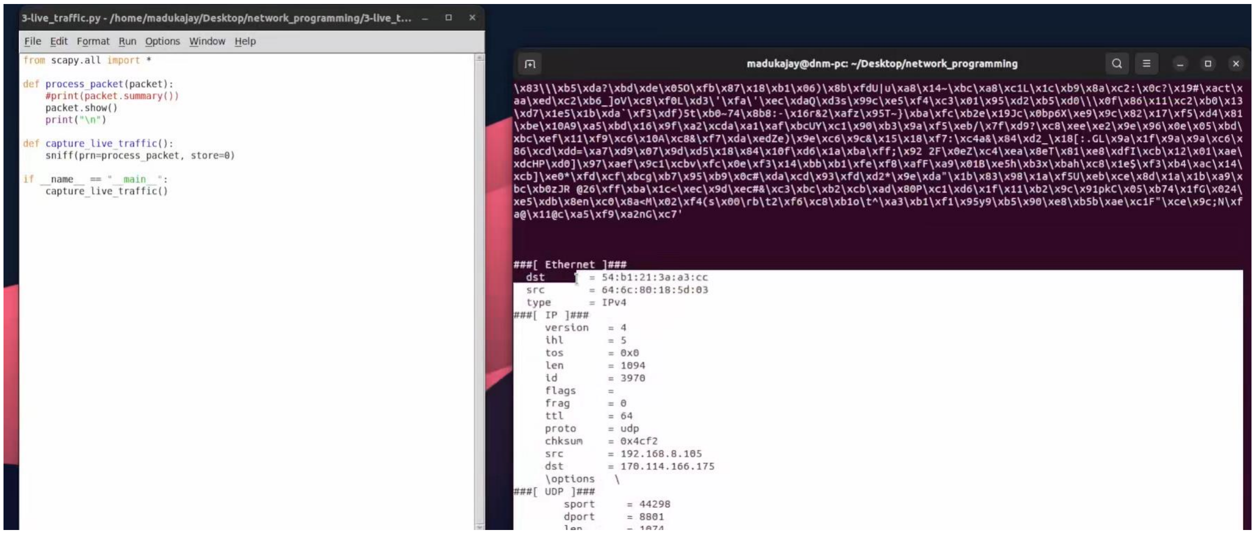

3-live_traffic.py - /home/madukajay/Desktop/network_programming/3-live_t...
File Edit Format Run Options Window Help
Ln: 13 Col: 0
madukajay@dnm-pc: ~/Desktop/network_programming

```python
from scapy.all import *

def process_packet(packet):
    print(packet.summary())
    #packet.show()
    print("\n")

def capture_live_traffic():
    sniff(prn=process_packet, store=0)

if __name__ == "__main__":
    capture_live_traffic()
```

```
Ether / IP / UDP 192.168.8.105:59285 > 170.114.166.175:8801 / Raw

Ether / IPv6 / UDP 2404:6800:4009:823::200a:https > 2402:4000:b282:3017:f8cc:34d
f:a3a9:3eec:38713 / Raw

Ether / IPv6 / UDP 2404:6800:4009:823::200a:https > 2402:4000:b282:3017:f8cc:34d
f:a3a9:3eec:38713 / Raw

Ether / IPv6 / UDP 2402:4000:b282:3017:f8cc:34df:a3a9:3eec:38713 > 2404:6800:400
9:823::200a:https / Raw

Ether / IP / UDP 192.168.8.105:59285 > 170.114.166.175:8801 / Raw

Ether / IP / UDP 170.114.166.175:8801 > 192.168.8.105:44298 / Raw
```

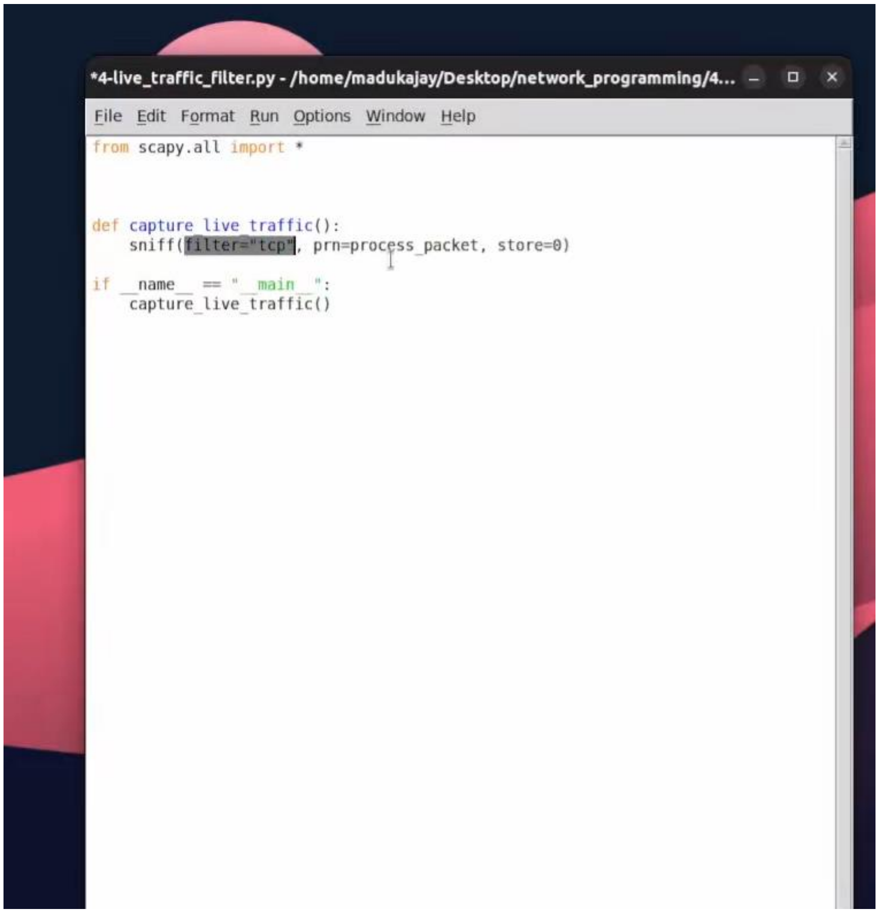
Src, dst ip

Src, dst port

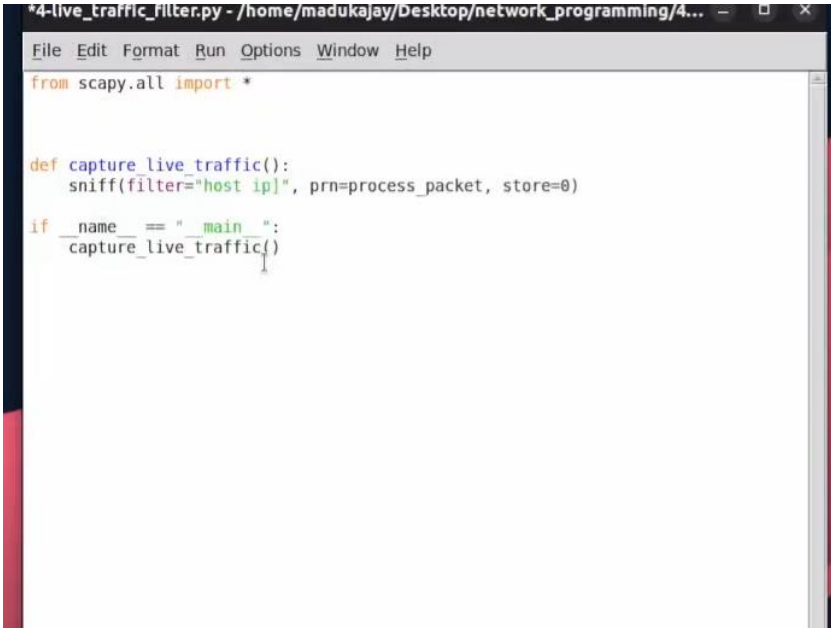

3-live_traffic.py - /home/madukajay/Desktop/network_programming/3-live_t...
File Edit Format Run Options Window Help
madukajay@dnm-pc: ~/Desktop/network_programming

```python
from scapy.all import *

def process_packet(packet):
    #print(packet.summary())
    packet.show()
    print("\n")

def capture_live_traffic():
    sniff(prn=process_packet, store=0)

if __name__ == "__main__":
    capture_live_traffic()
```

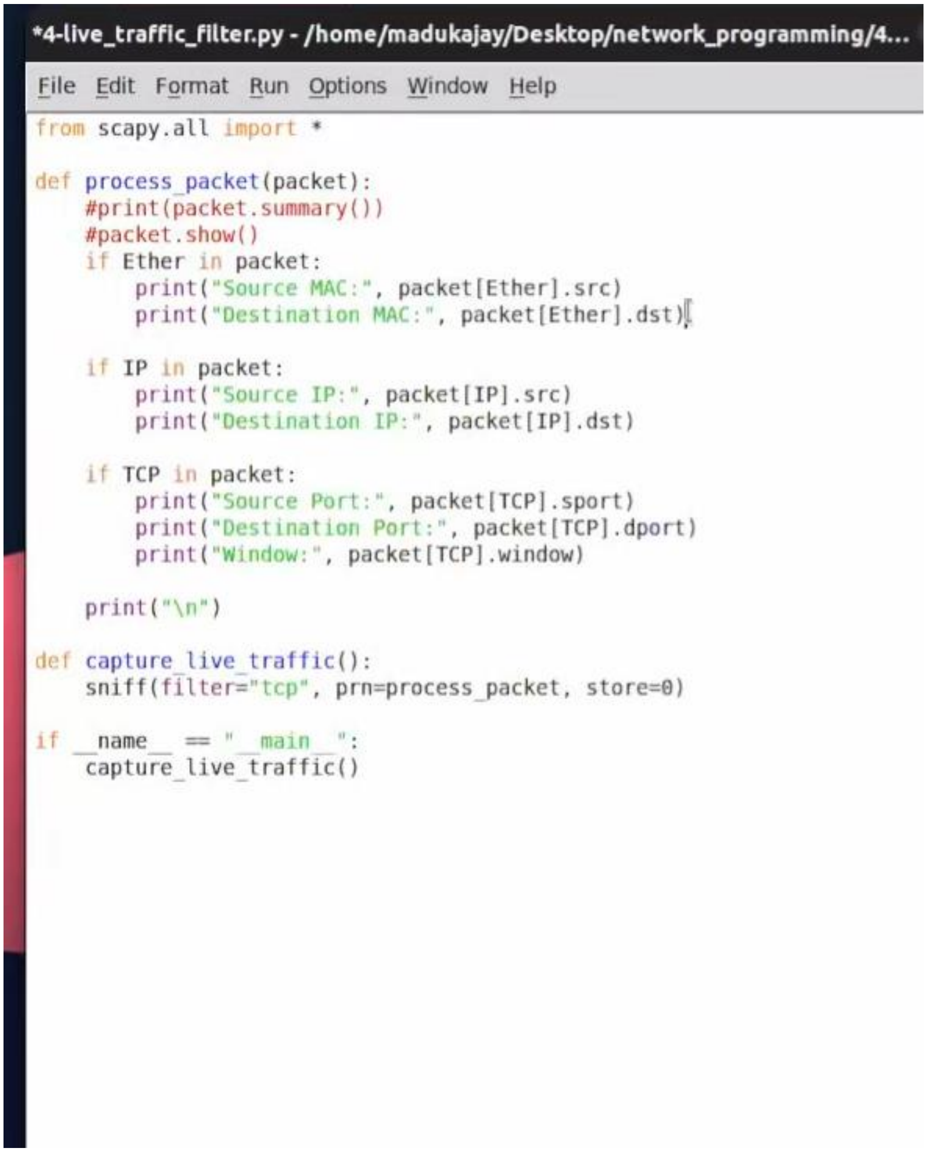
```
\x83\\\xb5\xda?\xbd\xde\x050\xfb\x87\x18\xb1\x06)\x8b\xfdU|u\xa8\x14~\xbc\xa8\xc1L\x1c\xb9\x8a\xc2:\x0c?\x19#\xact\x
aa\xed\xc2\xb6_]oV\xc8\xf0L\xd3\'\xfa\'\xec\xdaQ\xd3s\x99c\xe5\xf4\xc3\x01\x95\xd2\xb5\xd0\\\x0f\x86\x11\xc2\xb0\x13
\xd7\x1e5\x1b\xda`\xf3\xdf)5t\xb0~74\x8b8:-\x16r&2\xafz\x95T~}\xba\xfc\xb2e\x19Jc\x0bp6X\xe9\x9c\x82\x17\xf5\xd4\x81
\xbe\x10A9\xa5\xbd\x16\x9f\xa2\xcda\xa1\xaf\xbcUY\xc1\x90\xb3\x9a\xf5\xeb/\x7f\xd9?\xc8\xee\xe2\x9e\x96\x0e\x05\xbd\
xbc\xef\x11\xf9\xc6\x10A\xc8&\xf7\xda\xedZe)\x9e\xc6\x9c&\x15\x18\xf7:\xc4a&\x84\xd2_\x18[:.GL\x9a\x1f\x9a\x9a\xc6\x
86\xcd\xdd=\xa7\xd9\x07\x9d\xd5\x18\x84\x10f\xd6\x1a\xba\xff;\x92 2F\x0eZ\xc4\xea\x8eT\x81\xe8\xdfI\xcb\x12\x01\xae\
xdcHP\xd0]\x97\xaef\x9c1\xcbv\xfc\x0e\xf3\x14\xbb\xb1\xfe\xf8\xafF\xa9\x01B\xe5h\xb3x\xbah\xc8\x1e$\xf3\xb4\xac\x14\
xcb]\xe0*\xfd\xcf\xbcg\xb7\x95\xb9\x0c#\xda\xcd\x93\xfd\xd2*\x9e\xda"\x1b\x83\x98\x1a\xf5U\xeb\xce\x8d\x1a\x1b\xa9\x
bc\xb0zJR @26\xff\xba\x1c<\xec\x9d\xec#&\xc3\xbc\xb2\xcb\xad\x80P\xc1\xd6\x1f\x11\xb2\x9c\x91pkC\x05\xb74\x1fG\x024\
xe5\xdb\x8en\xc0\x8a<M\x02\xf4(s\x00\rb\t2\xf6\xc8\xb1o\t^\xa3\xb1\xf1\x95y9\xb5\x90\xe8\xb5b\xae\xc1F"\xce\x9c;N\xf
a@\x11@c\xa5\xf9\xa2nG\xc7'


###[ Ethernet ]###
  dst       = 54:b1:21:3a:a3:cc
  src       = 64:6c:80:18:5d:03
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 1094
     id        = 3970
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x4cf2
     src       = 192.168.8.105
     dst       = 170.114.166.175
     \options   \
###[ UDP ]###
        sport     = 44298
        dport     = 8801
        len       = 1074
```


*4-live_traffic_filter.py - /home/madukajay/Desktop/network_programming/4...
File Edit Format Run Options Window Help

```
from scapy.all import *


def capture_live_traffic():
    sniff(filter="tcp", prn=process_packet, store=0)

if __name__ == "__main__":
    capture_live_traffic()
```

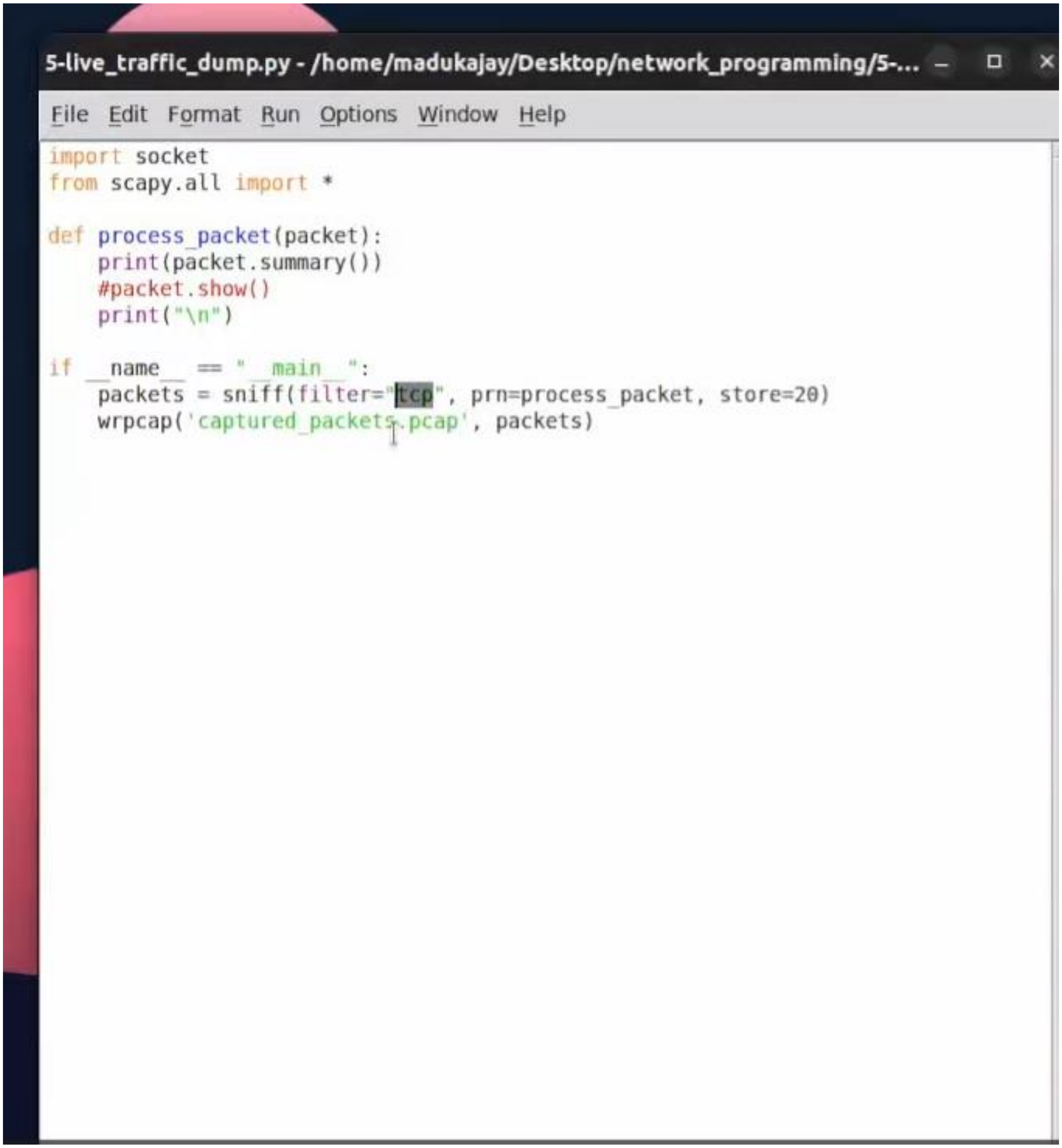

*4-live_traffic_filter.py - /home/madukajay/Desktop/network_programming/4...
File Edit Format Run Options Window Help

```
from scapy.all import *


def capture_live_traffic():
    sniff(filter="host ip]", prn=process_packet, store=0)

if __name__ == "__main__":
    capture_live_traffic()
```


*4-live_traffic_filter.py - /home/madukajay/Desktop/network_programming/4...
File Edit Format Run Options Window Help

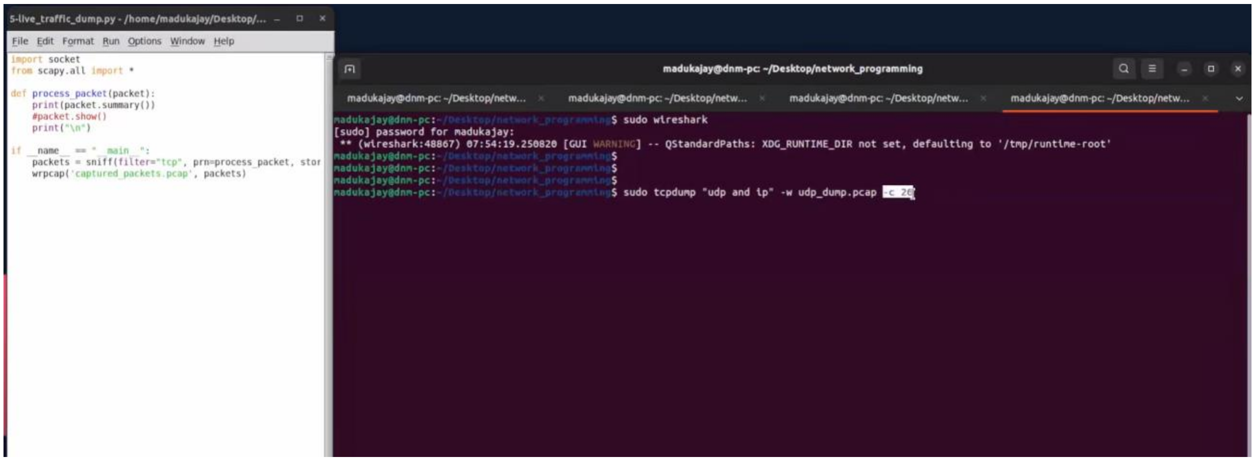
```
from scapy.all import *

def process_packet(packet):
    #print(packet.summary())
    #packet.show()
    if Ether in packet:
        print("Source MAC:", packet[Ether].src)
        print("Destination MAC:", packet[Ether].dst)

    if IP in packet:
        print("Source IP:", packet[IP].src)
        print("Destination IP:", packet[IP].dst)

    if TCP in packet:
        print("Source Port:", packet[TCP].sport)
        print("Destination Port:", packet[TCP].dport)
        print("Window:", packet[TCP].window)

    print("\n")

def capture_live_traffic():
    sniff(filter="tcp", prn=process_packet, store=0)

if __name__ == "__main__":
    capture_live_traffic()
```

- 6-TCP and 17-UDP



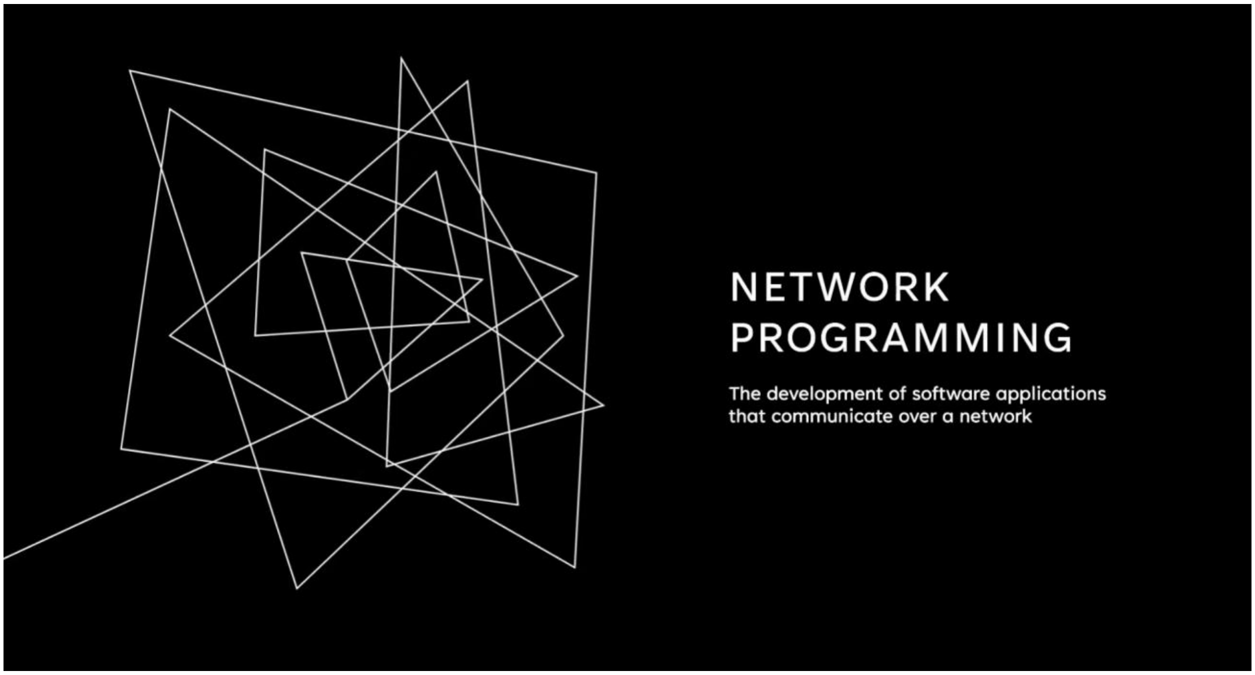
5-live_traffic_dump.py - /home/madukajay/Desktop/network_programming/5-...
File
Edit
Format
Run
Options
Window
Help

```
import socket
from scapy.all import *

def process_packet(packet):
    print(packet.summary())
    #packet.show()
    print("\n")

if __name__ == "__main__":
    packets = sniff(filter="tcp", prn=process_packet, store=20)
    wrpcap('captured_packets.pcap', packets)
```


5-live_traffic_dump.py - /home/madukajay/Desktop/...
File Edit Format Run Options Window Help
madukajay@dnm-pc: ~/Desktop/network_programming
madukajay@dnm-pc: ~/Desktop/netw...
madukajay@dnm-pc: ~/Desktop/netw...
madukajay@dnm-pc: ~/Desktop/netw...
madukajay@dnm-pc: ~/Desktop/netw...

```python
import socket
from scapy.all import *

def process_packet(packet):
    print(packet.summary())
    #packet.show()
    print("\n")

if __name__ == "__main__":
    packets = sniff(filter="tcp", prn=process_packet, stor
    wrpcap('captured_packets.pcap', packets)
```

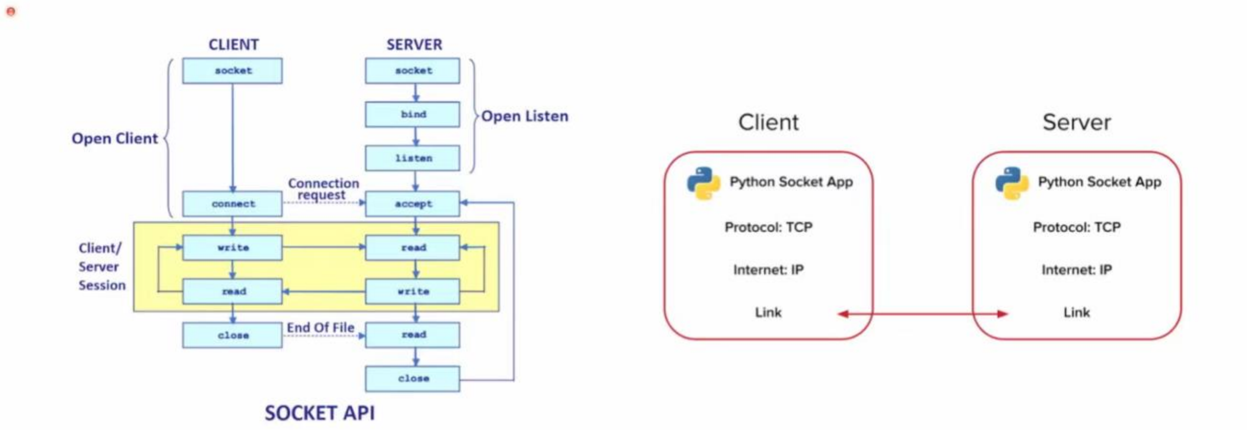
```
madukajay@dnm-pc:~/Desktop/network_programming$ sudo wireshark
[sudo] password for madukajay:
 ** (wireshark:48867) 07:54:19.250820 [GUI WARNING] -- QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
madukajay@dnm-pc:~/Desktop/network_programming$
madukajay@dnm-pc:~/Desktop/network_programming$
madukajay@dnm-pc:~/Desktop/network_programming$
madukajay@dnm-pc:~/Desktop/network_programming$ sudo tcpdump "udp and ip" -w udp_dump.pcap -c 20
```



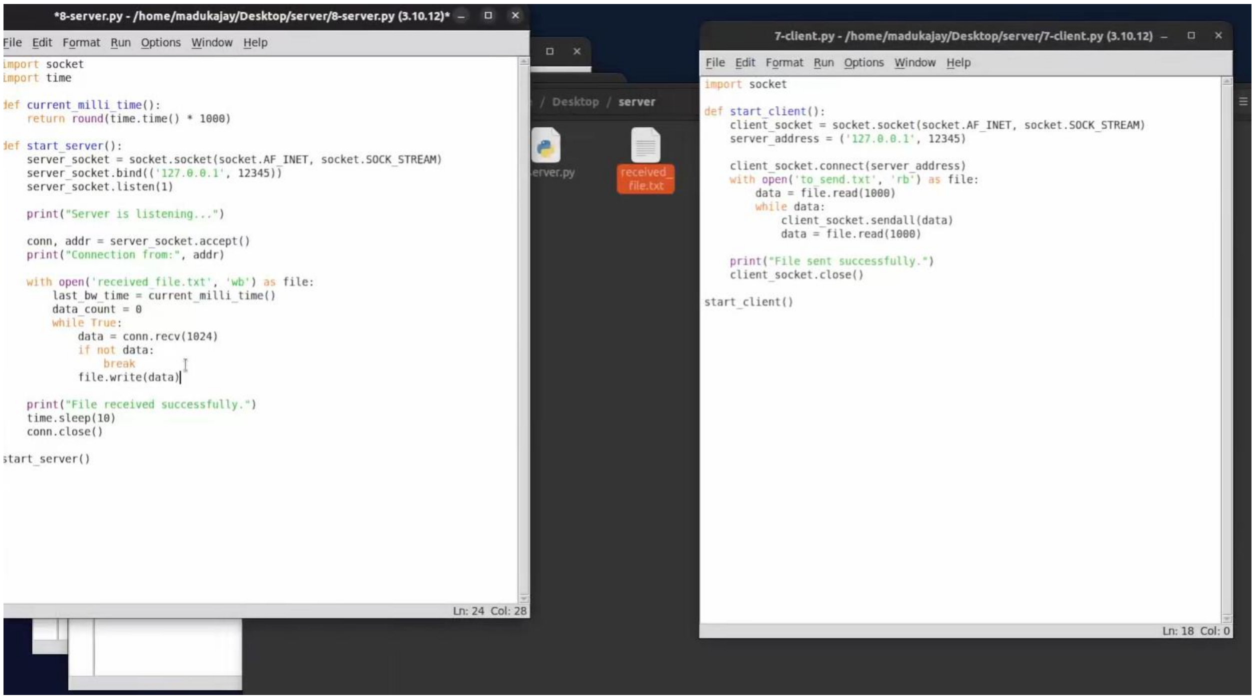
NETWORK
PROGRAMMING

The development of software applications
that communicate over a network

We will look at the socket programming part in the network programming…………………



SERVER AND CLIENT


8-server.py - /home/madukajay/Desktop/server/8-server.py (3.10.12)
File Edit Format Run Options Window Help
Desktop / server
erver.py
received_file.txt
7-client.py - /home/madukajay/Desktop/server/7-client.py (3.10.12)
File Edit Format Run Options Window Help
Ln: 24 Col: 28
Ln: 18 Col: 0

```
import socket
import time

def current_milli_time():
    return round(time.time() * 1000)

def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('127.0.0.1', 12345))
    server_socket.listen(1)

    print("Server is listening...")

    conn, addr = server_socket.accept()
    print("Connection from:", addr)

    with open('received_file.txt', 'wb') as file:
        last_bw_time = current_milli_time()
        data_count = 0
        while True:
            data = conn.recv(1024)
            if not data:
                break
            file.write(data)

    print("File received successfully.")
    time.sleep(10)
    conn.close()

start_server()
```

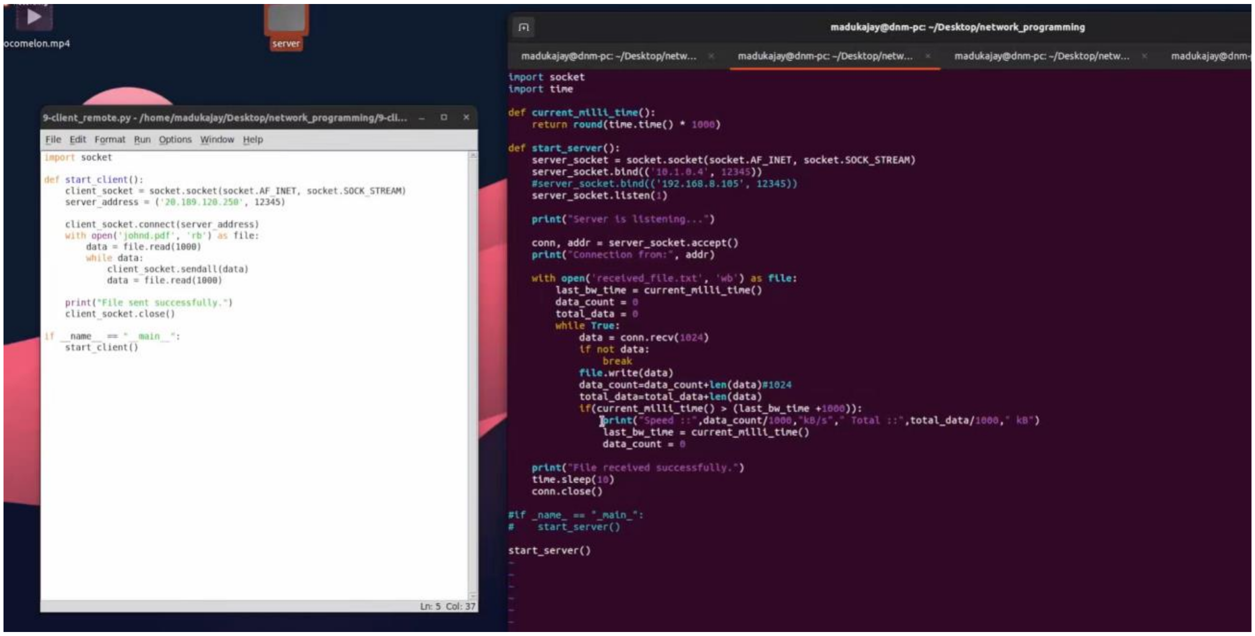
```
import socket

def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_address = ('127.0.0.1', 12345)

    client_socket.connect(server_address)
    with open('to_send.txt', 'rb') as file:
        data = file.read(1000)
        while data:
            client_socket.sendall(data)
            data = file.read(1000)

    print("File sent successfully.")
    client_socket.close()

start_client()
```

Run the client.py in cmd after running the server.py in the shell.

We can observe this in loopback traffic capture in wireshark.
Also filter by : ip.addr == 192.168.8.160 || tcp.port == 12345

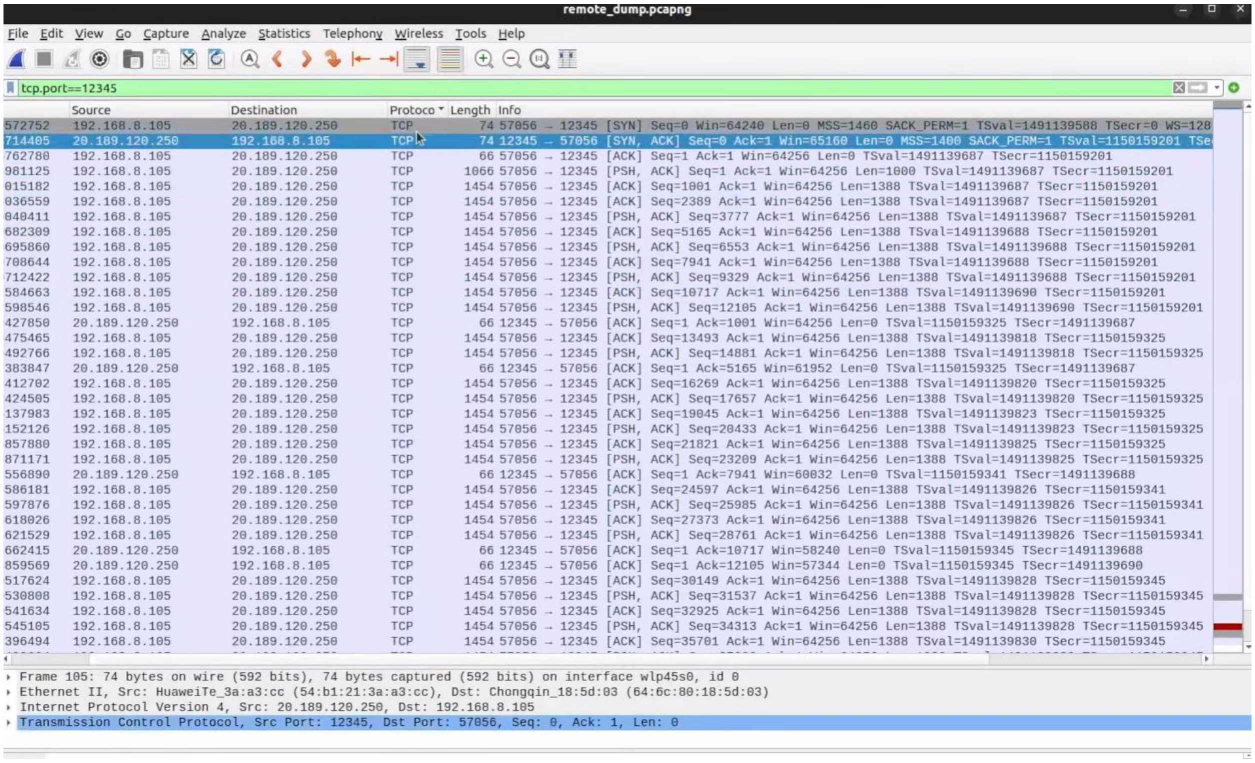
Let us see for remote server..........

We have to log into the server using our cmd and run the server code in that.


ocomelon.mp4
server
9-client_remote.py - /home/madukajay/Desktop/network_programming/9-cli...
File Edit Format Run Options Window Help
Ln: 5 Col: 37
madukajay@dnm-pc: ~/Desktop/network_programming
madukajay@dnm-pc: ~/Desktop/netw...
madukajay@dnm-pc: ~/Desktop/netw...
madukajay@dnm-pc: ~/Desktop/netw...
madukajay@dnm-

```
import socket

def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_address = ('20.189.120.250', 12345)

    client_socket.connect(server_address)
    with open('johnd.pdf', 'rb') as file:
        data = file.read(1000)
        while data:
            client_socket.sendall(data)
            data = file.read(1000)

    print("File sent successfully.")
    client_socket.close()

if __name__ == "__main__":
    start_client()
```

```
import socket
import time

def current_milli_time():
    return round(time.time() * 1000)

def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('10.1.0.4', 12345))
    #server_socket.bind(('192.168.8.105', 12345))
    server_socket.listen(1)

    print("Server is listening...")

    conn, addr = server_socket.accept()
    print("Connection from:", addr)

    with open('received_file.txt', 'wb') as file:
        last_bw_time = current_milli_time()
        data_count = 0
        total_data = 0
        while True:
            data = conn.recv(1024)
            if not data:
                break
            file.write(data)
            data_count=data_count+len(data)#1024
            total_data=total_data+len(data)
            if(current_milli_time() > (last_bw_time +1000)):
                print("Speed ::",data_count/1000,"kB/s"," Total ::",total_data/1000," kB")
                last_bw_time = current_milli_time()
                data_count = 0

    print("File received successfully.")
    time.sleep(10)
    conn.close()

#if _name_ == "_main_":
#    start_server()

start_server()
~
~
~
~
~
~
```

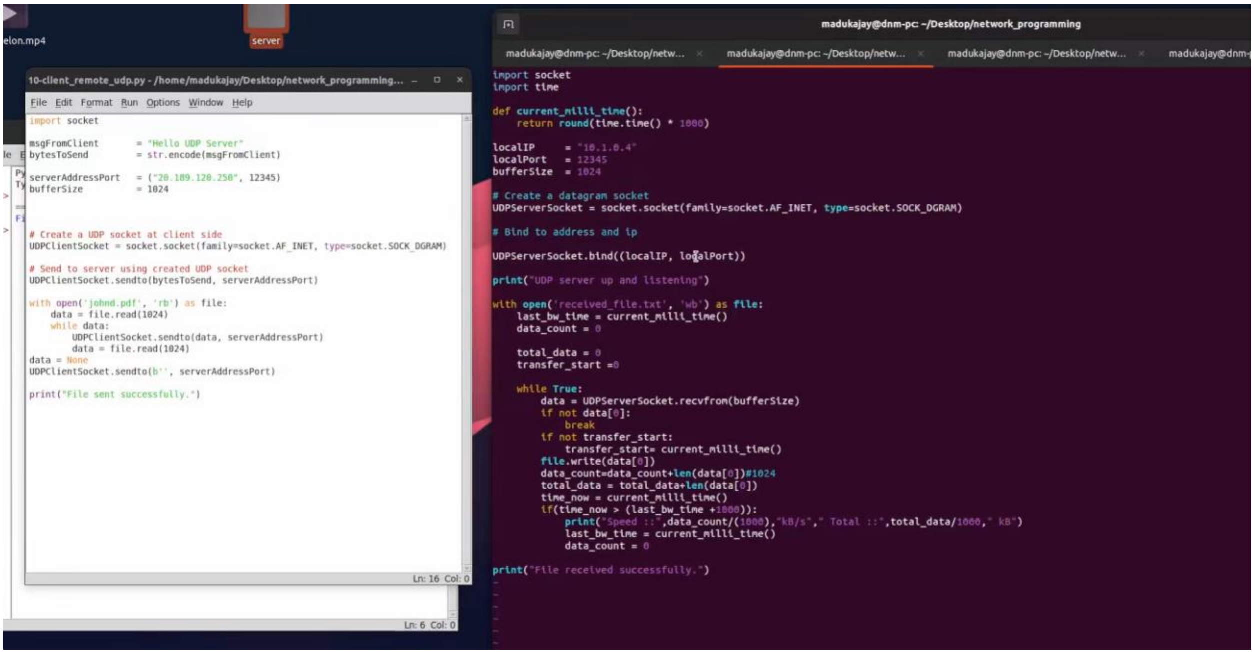

remote_dump.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
tcp.port==12345
```

```
        Source           Destination      Protoco  Length Info
572752  192.168.8.105    20.189.120.250   TCP        74 57056 → 12345 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1491139588 TSecr=0 WS=128
714405  20.189.120.250   192.168.8.105    TCP        74 12345 → 57056 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1400 SACK_PERM=1 TSval=1150159201 TSe
762780  192.168.8.105    20.189.120.250   TCP        66 57056 → 12345 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1491139687 TSecr=1150159201
981125  192.168.8.105    20.189.120.250   TCP      1066 57056 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=1000 TSval=1491139687 TSecr=1150159201
015182  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=1001 Ack=1 Win=64256 Len=1388 TSval=1491139687 TSecr=1150159201
036559  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=2389 Ack=1 Win=64256 Len=1388 TSval=1491139687 TSecr=1150159201
040411  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [PSH, ACK] Seq=3777 Ack=1 Win=64256 Len=1388 TSval=1491139687 TSecr=1150159201
682309  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=5165 Ack=1 Win=64256 Len=1388 TSval=1491139688 TSecr=1150159201
695860  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [PSH, ACK] Seq=6553 Ack=1 Win=64256 Len=1388 TSval=1491139688 TSecr=1150159201
708644  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=7941 Ack=1 Win=64256 Len=1388 TSval=1491139688 TSecr=1150159201
712422  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [PSH, ACK] Seq=9329 Ack=1 Win=64256 Len=1388 TSval=1491139688 TSecr=1150159201
584663  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=10717 Ack=1 Win=64256 Len=1388 TSval=1491139690 TSecr=1150159201
598546  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [PSH, ACK] Seq=12105 Ack=1 Win=64256 Len=1388 TSval=1491139690 TSecr=1150159201
427850  20.189.120.250   192.168.8.105    TCP        66 12345 → 57056 [ACK] Seq=1 Ack=1001 Win=64256 Len=0 TSval=1150159325 TSecr=1491139687
475465  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=13493 Ack=1 Win=64256 Len=1388 TSval=1491139818 TSecr=1150159325
492766  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [PSH, ACK] Seq=14881 Ack=1 Win=64256 Len=1388 TSval=1491139818 TSecr=1150159325
383847  20.189.120.250   192.168.8.105    TCP        66 12345 → 57056 [ACK] Seq=1 Ack=5165 Win=61952 Len=0 TSval=1150159325 TSecr=1491139687
412702  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=16269 Ack=1 Win=64256 Len=1388 TSval=1491139820 TSecr=1150159325
424505  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [PSH, ACK] Seq=17657 Ack=1 Win=64256 Len=1388 TSval=1491139820 TSecr=1150159325
137983  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=19045 Ack=1 Win=64256 Len=1388 TSval=1491139823 TSecr=1150159325
152126  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [PSH, ACK] Seq=20433 Ack=1 Win=64256 Len=1388 TSval=1491139823 TSecr=1150159325
857880  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=21821 Ack=1 Win=64256 Len=1388 TSval=1491139825 TSecr=1150159325
871171  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [PSH, ACK] Seq=23209 Ack=1 Win=64256 Len=1388 TSval=1491139825 TSecr=1150159325
556890  20.189.120.250   192.168.8.105    TCP        66 12345 → 57056 [ACK] Seq=1 Ack=7941 Win=60032 Len=0 TSval=1150159341 TSecr=1491139688
586181  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=24597 Ack=1 Win=64256 Len=1388 TSval=1491139826 TSecr=1150159341
597876  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [PSH, ACK] Seq=25985 Ack=1 Win=64256 Len=1388 TSval=1491139826 TSecr=1150159341
618026  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=27373 Ack=1 Win=64256 Len=1388 TSval=1491139826 TSecr=1150159341
621529  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [PSH, ACK] Seq=28761 Ack=1 Win=64256 Len=1388 TSval=1491139826 TSecr=1150159341
662415  20.189.120.250   192.168.8.105    TCP        66 12345 → 57056 [ACK] Seq=1 Ack=10717 Win=58240 Len=0 TSval=1150159345 TSecr=1491139688
859569  20.189.120.250   192.168.8.105    TCP        66 12345 → 57056 [ACK] Seq=1 Ack=12105 Win=57344 Len=0 TSval=1150159345 TSecr=1491139690
517624  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=30149 Ack=1 Win=64256 Len=1388 TSval=1491139828 TSecr=1150159345
530808  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [PSH, ACK] Seq=31537 Ack=1 Win=64256 Len=1388 TSval=1491139828 TSecr=1150159345
541634  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=32925 Ack=1 Win=64256 Len=1388 TSval=1491139828 TSecr=1150159345
545105  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [PSH, ACK] Seq=34313 Ack=1 Win=64256 Len=1388 TSval=1491139828 TSecr=1150159345
396494  192.168.8.105    20.189.120.250   TCP      1454 57056 → 12345 [ACK] Seq=35701 Ack=1 Win=64256 Len=1388 TSval=1491139830 TSecr=1150159345
```

```
› Frame 105: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlp45s0, id 0
› Ethernet II, Src: HuaweiTe_3a:a3:cc (54:b1:21:3a:a3:cc), Dst: Chongqin_18:5d:03 (64:6c:80:18:5d:03)
› Internet Protocol Version 4, Src: 20.189.120.250, Dst: 192.168.8.105
› Transmission Control Protocol, Src Port: 12345, Dst Port: 57056, Seq: 0, Ack: 1, Len: 0
```

Let us see for remote server using udp……….


elon.mp4
server
10-client_remote_udp.py - /home/madukajay/Desktop/network_programming...
File Edit Format Run Options Window Help
Ln: 16 Col: 0
Ln: 6 Col: 0
madukajay@dnm-pc: ~/Desktop/network_programming
madukajay@dnm-pc: ~/Desktop/netw...
madukajay@dnm-pc: ~/Desktop/netw...
madukajay@dnm-pc: ~/Desktop/netw...
madukajay@dnm-

```
import socket

msgFromClient       = "Hello UDP Server"
bytesToSend         = str.encode(msgFromClient)

serverAddressPort   = ("20.189.120.250", 12345)
bufferSize          = 1024



# Create a UDP socket at client side
UDPClientSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)

# Send to server using created UDP socket
UDPClientSocket.sendto(bytesToSend, serverAddressPort)

with open('johnd.pdf', 'rb') as file:
    data = file.read(1024)
    while data:
        UDPClientSocket.sendto(data, serverAddressPort)
        data = file.read(1024)
data = None
UDPClientSocket.sendto(b'', serverAddressPort)

print("File sent successfully.")
```

```
import socket
import time

def current_milli_time():
    return round(time.time() * 1000)

localIP     = "10.1.0.4"
localPort   = 12345
bufferSize  = 1024

# Create a datagram socket
UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)

# Bind to address and ip

UDPServerSocket.bind((localIP, localPort))

print("UDP server up and listening")

with open('received_file.txt', 'wb') as file:
    last_bw_time = current_milli_time()
    data_count = 0

    total_data = 0
    transfer_start =0

    while True:
        data = UDPServerSocket.recvfrom(bufferSize)
        if not data[0]:
            break
        if not transfer_start:
            transfer_start= current_milli_time()
        file.write(data[0])
        data_count=data_count+len(data[0])#1024
        total_data = total_data+len(data[0])
        time_now = current_milli_time()
        if(time_now > (last_bw_time +1000)):
            print("Speed ::",data_count/(1000),"kB/s"," Total ::",total_data/1000," kB")
            last_bw_time = current_milli_time()
            data_count = 0

print("File received successfully.")
~
~
~
~
~
~
```