Q1. Write a program in python to open a UDP socket between a client and a server, then use wireshark to monitor the packets sent. [Hint: Write a small program for the UDP client side and another for the server side. Use parts of the code in your book pages 159-164]. Include in the message your name, and the city of your birth place. Use port number that is the last 4 digits of your UFID. Explain the address and port numbers used.

Ans.1 =

Python Scripts for UDPServer & UDPclient.

UDPServer.py

```
1  from socket import *
2  serverPortNumber = 5826
3  serverSocketVar = socket(AF_INET, SOCK_DGRAM)
4  serverSocketVar.bind(('', serverPortNumber))
5  print ('The server is ready to receive')
6  while True:
7      message, clientAddress = serverSocketVar.recvfrom(2048)
8      changedMessage = message.decode().upper()
9      serverSocketVar.sendto(changedMessage.encode(), clientAddress)
```

UDPClient.py

```
1  from socket import *
2  serverNameIdentifier = '127.0.0.1'
3  serverPortNumber = 5826
4  clientSocketVar = socket(AF_INET,SOCK_DGRAM)
5  inputSentence = input('Input lowercase sentence:')
6  clientSocketVar.sendto(inputSentence.encode(),(serverNameIdentifier, serverPortNumber))
7  changedSentence, serverAddress = clientSocketVar.recvfrom(2048)
8  print ('Response From Server:', changedSentence.decode())
9  clientSocketVar.close()
```

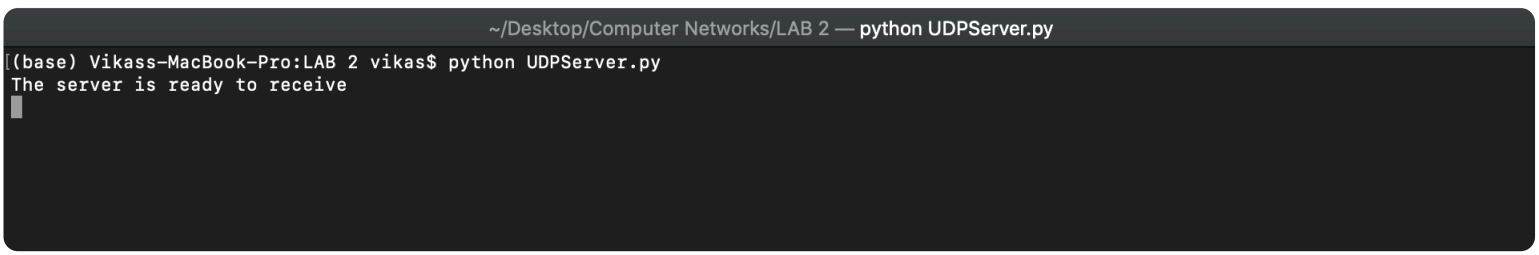
Running python scripts ⇒

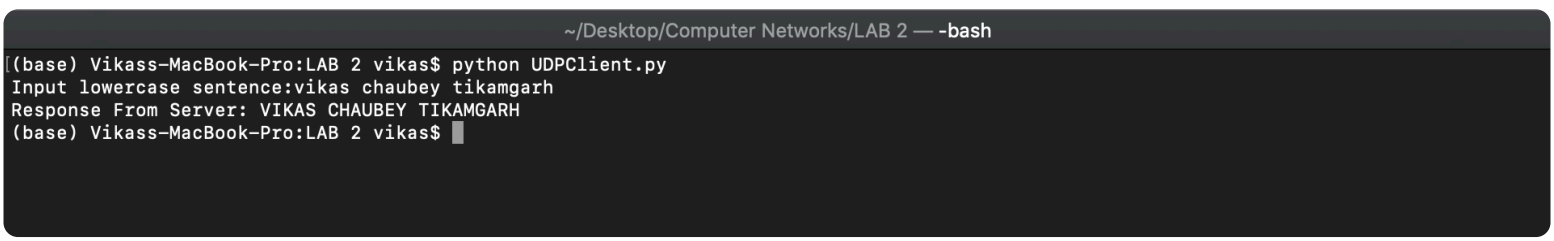
① = first UDPServer.py is run.

② = Secondly UDPClient.py is run.

③ = Input is given as lower cased Sentence (myname + Birth place)

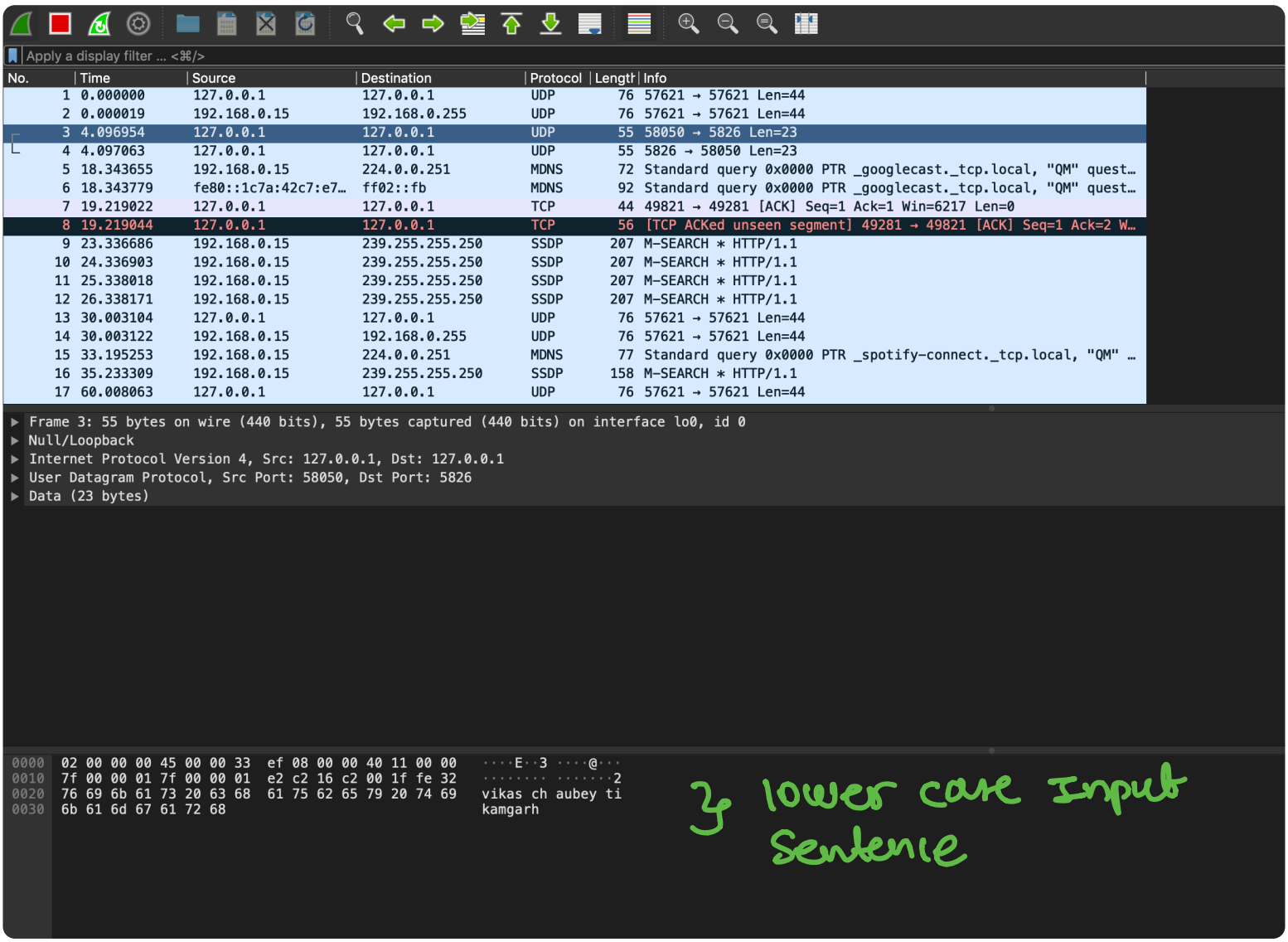
④ = Server Returns output as upper cased sentence.





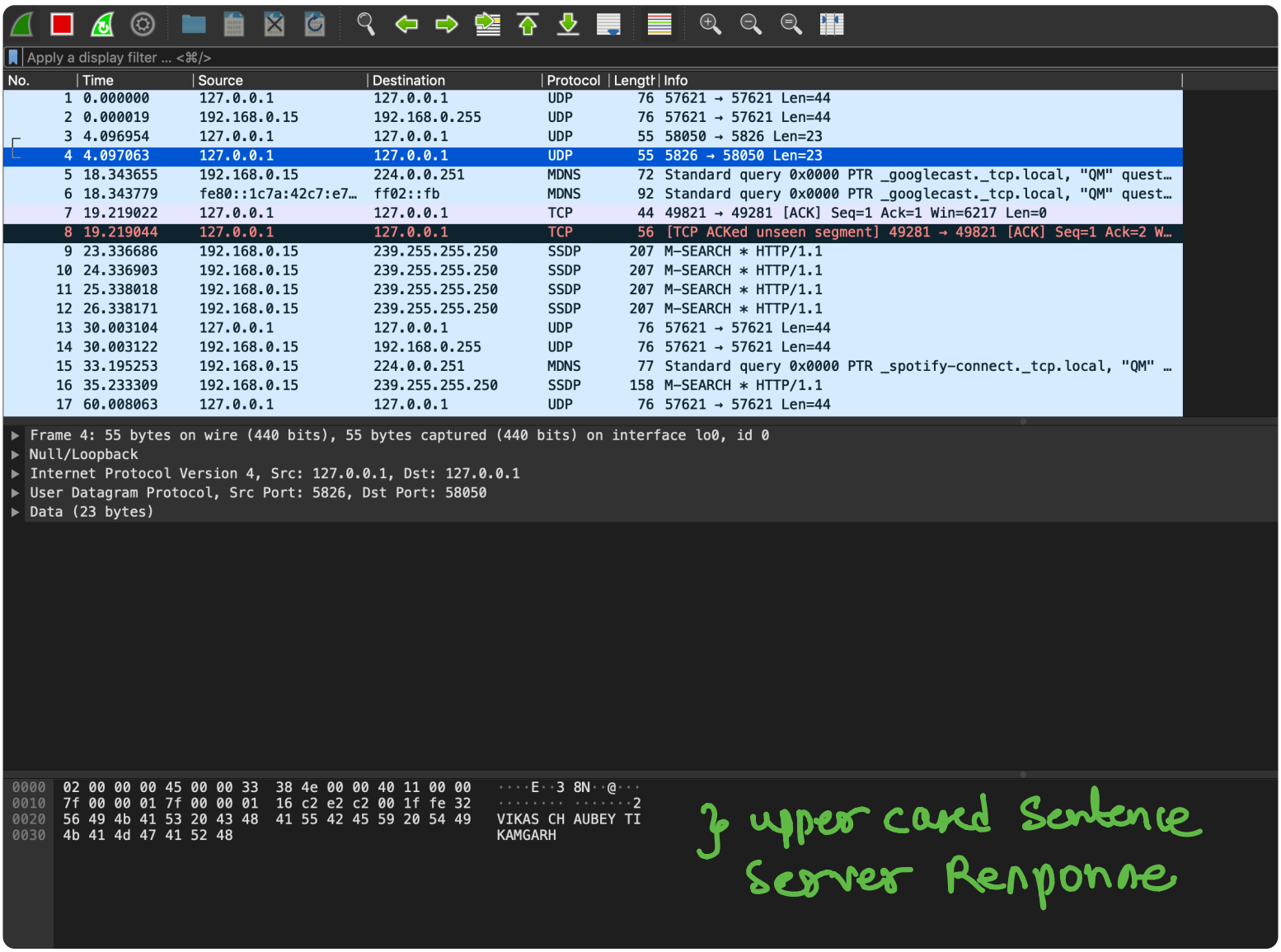
Wireshark tracer the activity .⇒

① = request is sent to Server.



② lower case Input Sentence

②= Response is obtained from the server.



} upper cased Sentence Server Response

Address & Server Port -

In a client-server model. In order to establish a connection with server client requires Ip address of the server machine and the port number.

Address in the python script is the Ip address of the server machine in the network. Since the server is hosted on my local computer Hence address is 127.0.0.1 . which is address for localhost.

Port number is an identifier for socket socket is bound with a specific port using bind() method on server.
Server port is defined as = 5826

(last four digits of UF Id)

Q2. Write a program in python to open a TCP socket between a client and a server, then user wireshark to monitor the packets sent. [Hint: Write a small program for the TCP client side and another for the TCP server side. Use parts of the code in your book pages 164-170]. Include in the message your name, and the city of your birth place. Use port number that is the last 4 digits of your UFID. Explain the address and port numbers used.

Ans:2 =

Python scripts for TCPServer.py and TCPClient.py.

TCPServer.py

```python
from socket import *
serverPortNumber = 5826
serverSocketVar = socket(AF_INET,SOCK_STREAM)
serverSocketVar.bind(('',serverPortNumber))
serverSocketVar.listen(1)
print ('The server is ready to receive')
while True:
    connectionSocketVar, address = serverSocketVar.accept()
    inputSentence = connectionSocketVar.recv(1024)
    upperCasedSentence = inputSentence.decode().upper()
    connectionSocketVar.send(upperCasedSentence.encode())
    connectionSocketVar.close()
```

TCPClient.py

```python
from socket import *
serverNameIdentifier = '127.0.0.1'
serverPortNumber = 5826
clientSocketVar = socket(AF_INET, SOCK_STREAM)
clientSocketVar.connect((serverNameIdentifier,serverPortNumber))
inputSentence = input('Input lowercase sentence:')
clientSocketVar.send(inputSentence.encode())
changedSentence = clientSocketVar.recv(1024)
print ('Response From Server:', changedSentence.decode())
clientSocketVar.close()
```

Running python Scripts ⇒

① = first TCP server.py is run.

② = secondly TCP client.py is run.

③ = Input is given as lower cased Sentence (Myname + birthplace)

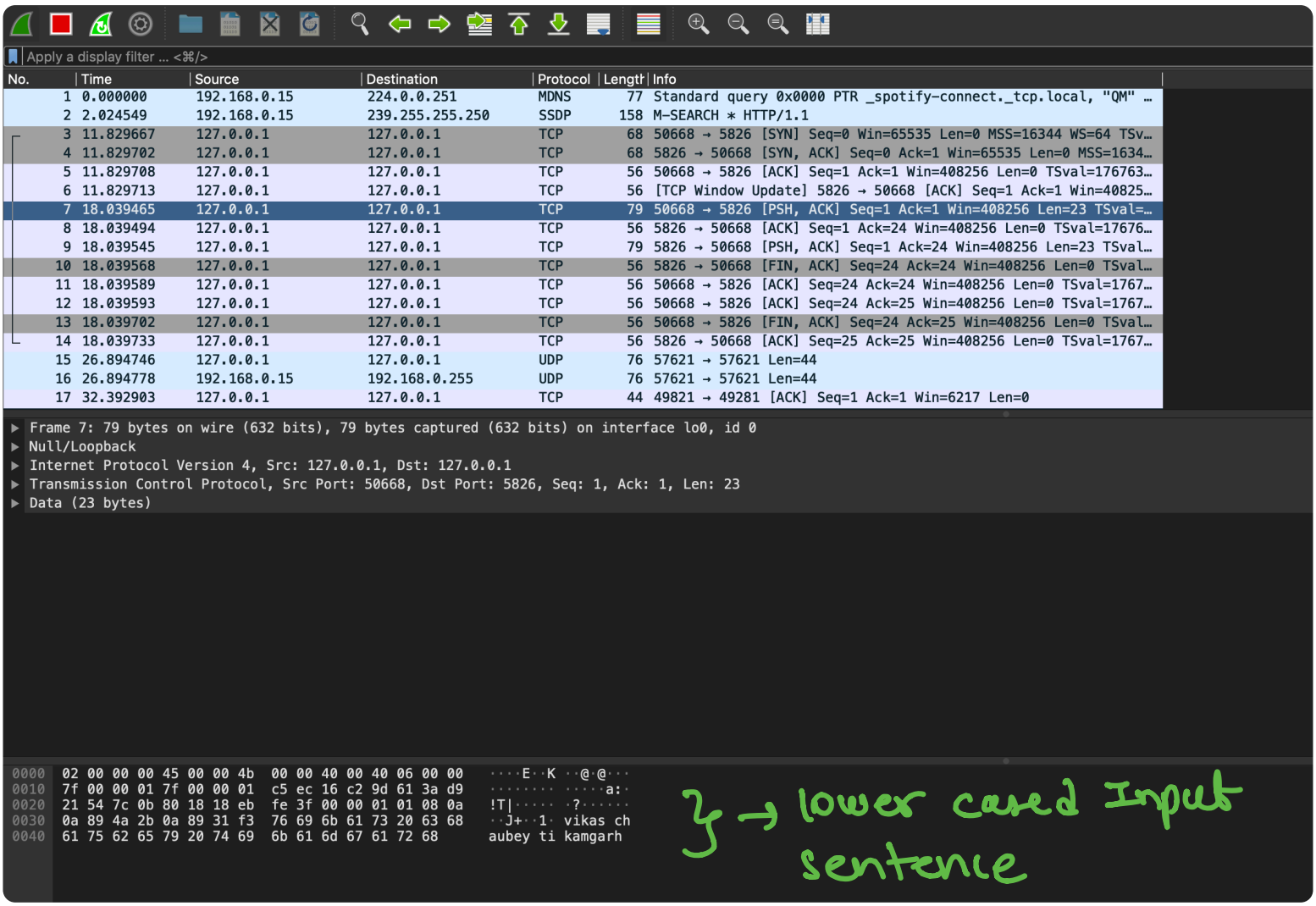
④ = Server Returns output as upper cased sentence.

```
~/Desktop/Computer Networks/LAB 2 — python TCPServer.py
(base) Vikass-MacBook-Pro:LAB 2 vikas$ python TCPServer.py
The server is ready to receive
```

```
~/Desktop/Computer Networks/LAB 2 — -bash
(base) Vikass-MacBook-Pro:LAB 2 vikas$ python TCPClient.py
Input lowercase sentence:vikas chaubey tikamgarh
Response From Server: VIKAS CHAUBEY TIKAMGARH
(base) Vikass-MacBook-Pro:LAB 2 vikas$
```

wireshark traces the activity.

① = wireshark screenshot when the request is sent to the Server.



Apply a display filter ... <⌘/>

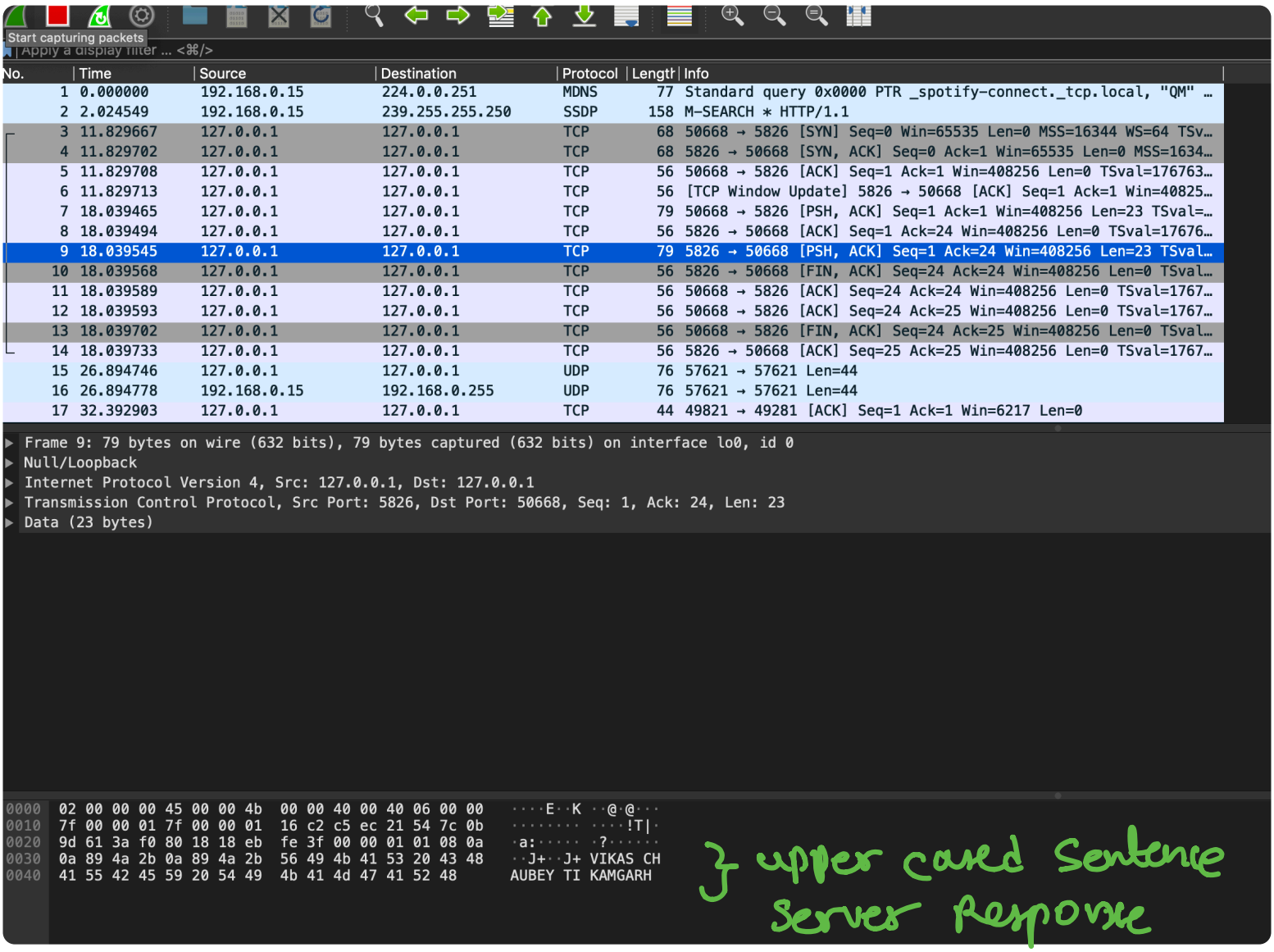
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.15	224.0.0.251	MDNS	77	Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" …
2	2.024549	192.168.0.15	239.255.255.250	SSDP	158	M-SEARCH * HTTP/1.1
3	11.829667	127.0.0.1	127.0.0.1	TCP	68	50668 → 5826 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSv…
4	11.829702	127.0.0.1	127.0.0.1	TCP	68	5826 → 50668 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1634…
5	11.829708	127.0.0.1	127.0.0.1	TCP	56	50668 → 5826 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=176763…
6	11.829713	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 5826 → 50668 [ACK] Seq=1 Ack=1 Win=40825…
7	18.039465	127.0.0.1	127.0.0.1	TCP	79	50668 → 5826 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=23 TSval=…
8	18.039494	127.0.0.1	127.0.0.1	TCP	56	5826 → 50668 [ACK] Seq=1 Ack=24 Win=408256 Len=0 TSval=17676…
9	18.039545	127.0.0.1	127.0.0.1	TCP	79	5826 → 50668 [PSH, ACK] Seq=1 Ack=24 Win=408256 Len=23 TSval…
10	18.039568	127.0.0.1	127.0.0.1	TCP	56	5826 → 50668 [FIN, ACK] Seq=24 Ack=24 Win=408256 Len=0 TSval…
11	18.039589	127.0.0.1	127.0.0.1	TCP	56	50668 → 5826 [ACK] Seq=24 Ack=24 Win=408256 Len=0 TSval=1767…
12	18.039593	127.0.0.1	127.0.0.1	TCP	56	50668 → 5826 [ACK] Seq=24 Ack=25 Win=408256 Len=0 TSval=1767…
13	18.039702	127.0.0.1	127.0.0.1	TCP	56	50668 → 5826 [FIN, ACK] Seq=24 Ack=25 Win=408256 Len=0 TSval…
14	18.039733	127.0.0.1	127.0.0.1	TCP	56	5826 → 50668 [ACK] Seq=25 Ack=25 Win=408256 Len=0 TSval=1767…
15	26.894746	127.0.0.1	127.0.0.1	UDP	76	57621 → 57621 Len=44
16	26.894778	192.168.0.15	192.168.0.255	UDP	76	57621 → 57621 Len=44
17	32.392903	127.0.0.1	127.0.0.1	TCP	44	49821 → 49281 [ACK] Seq=1 Ack=1 Win=6217 Len=0

▶ Frame 7: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface lo0, id 0
▶ Null/Loopback
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 50668, Dst Port: 5826, Seq: 1, Ack: 1, Len: 23
▶ Data (23 bytes)

```
0000  02 00 00 00 45 00 00 4b  00 00 40 00 40 06 00 00   ····E··K ··@·@···
0010  7f 00 00 01 7f 00 00 01  c5 ec 16 c2 9d 61 3a d9   ········ ·····a:·
0020  21 54 7c 0b 80 18 18 eb  fe 3f 00 00 01 01 08 0a   !T|····· ·?······
0030  0a 89 4a 2b 0a 89 31 f3  76 69 6b 61 73 20 63 68   ··J+··1· vikas ch
0040  61 75 62 65 79 20 74 69  6b 61 6d 67 61 72 68      aubey ti kamgarh
```

} → lower cased Input sentence

(2) = wireshark screenshot when the server sends response



3 upper cased Sentence Server Response

Addrerr & Server Port -

In a client-server model. In order to entablish a connection with server client requirer Ip addrerr of the server machine and the port number.

Addrerr in the python script is the Ip addrerr of the server machine in the network. Since the server is honted on my local computer Hence addrerr is 127.0.0.1 . which is addrerr for localhont.

Port number is an identifier for Socket Socket is bound with a specific port using bind() method on server.
Server port is defined as = 5826

(last four digitn of UF ID)

Q3. Show the ARP table of your machine and describe it.

Ans.3 =

Running command = "arp -a" on terminal to obtain the the arp table.

```
(base) Vikass-MacBook-Pro:LAB 2 vikas$ arp -a
? (169.254.21.195) at 64:5d:86:bf:2a:ed on en0 [ethernet]
? (192.168.0.1) at 8:36:c9:c7:c7:c3 on en0 ifscope [ethernet]
? (192.168.0.13) at 30:fd:38:a0:5c:a6 on en0 ifscope [ethernet]
? (192.168.0.14) at 64:5d:86:bf:2a:ed on en0 ifscope [ethernet]
? (192.168.0.15) at a4:83:e7:aa:b7:f2 on en0 ifscope permanent [ethernet]
? (192.168.0.255) at ff:ff:ff:ff:ff:ff on en0 ifscope [ethernet]
? (224.0.0.251) at 1:0:5e:0:0:fb on en0 ifscope permanent [ethernet]
? (239.255.255.250) at 1:0:5e:7f:ff:fa on en0 ifscope permanent [ethernet]
(base) Vikass-MacBook-Pro:LAB 2 vikas$
```

Arp table Explanation = "arp -a" command displays arp cache tables for all interfaces.

① = The first column of the arp table denotes the Hostname, if the hostname can't be resolved then a question mark (?) is Printed.

(2) = The second column gives the Ip address of the interfaces. given Ip address 192.168.0.1 represents the local interface.

(3) = The third column represents the physical MAC address of the interface.

(4) = The fourth column (Ex. eno) repre -sents the interface identifier, in this case eno is the only ether -net adapter which is started

(5) = The fifth column (Ex. ifscope) represents the "Route command" Ex:- Here the route command is ifscope. It is used to bind a route to a specific Interface. In this case ifscope identifies that Ip addresses are routed to eno adapter.

⑥ = The Sixth column (Ex. permanent) represents the "arp cache entry State". there are two types of cache entries states =

① = Static Arp cache Entries =

These Entries are manually added to the cache table and are kept in the cache on a permanent basis.

② = Dynamic Arp cache Entries =

These entries are created by the software itself as a result of successfully completed part ARP resolution. These entries are kept in the cache only for a period of time and then removed.

In our case "permanent" means Entry is Static Arp Entry.

(7) = The Seventh column (Ex. ethernet) represents Hardware type. in our case its ethernet adapter.

Q4. Show the mac addresses for the interfaces of your machine and the ip addresses using ifconfig (or ipconfig), and the routing tables using netstat (or other) commands.

Ans.4 = ① = Running command "ifconfig" ⇒

```
(base) Vikass-MacBook-Pro:LAB 2 vikas$ ifconfig
lo0: flags=8149<UP,LOOPBACK,RUNNING,PROMISC,MULTICAST> mtu 16384
        options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
XHC20: flags=0<> mtu 0
VHC128: flags=0<> mtu 0
XHC1: flags=0<> mtu 0
XHC0: flags=0<> mtu 0
en5: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=4<VLAN_MTU>
        ether 00:e0:4c:68:05:34
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect (none)
        status: inactive
en6: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether ac:de:48:00:11:22
        inet6 fe80::aede:48ff:fe00:1122%en6 prefixlen 64 scopeid 0x9
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect (100baseTX <full-duplex>)
        status: active
en3: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        options=60<TSO4,TSO6>
        ether a2:00:64:49:98:05
        media: autoselect <full-duplex>
        status: inactive
en4: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        options=60<TSO4,TSO6>
        ether a2:00:64:49:98:04
        media: autoselect <full-duplex>
        status: inactive
en2: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        options=60<TSO4,TSO6>
        ether a2:00:64:49:98:00
        media: autoselect <full-duplex>
        status: inactive
en1: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
        options=60<TSO4,TSO6>
        ether a2:00:64:49:98:01
        media: autoselect <full-duplex>
        status: inactive
bridge0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=63<RXCSUM,TXCSUM,TSO4,TSO6>
        ether a2:00:64:49:98:01
        Configuration:
                id 0:0:0:0:0:0 priority 0 hellotime 0 fwddelay 0
                maxage 0 holdcnt 0 proto stp maxaddr 100 timeout 1200
                root id 0:0:0:0:0:0 priority 0 ifcost 0 port 0
                ipfilter disabled flags 0x2
        member: en1 flags=3<LEARNING,DISCOVER>
                ifmaxaddr 0 port 13 priority 0 path cost 0
        member: en2 flags=3<LEARNING,DISCOVER>
                ifmaxaddr 0 port 12 priority 0 path cost 0
        member: en3 flags=3<LEARNING,DISCOVER>
                ifmaxaddr 0 port 10 priority 0 path cost 0
        member: en4 flags=3<LEARNING,DISCOVER>
                ifmaxaddr 0 port 11 priority 0 path cost 0
        nd6 options=201<PERFORMNUD,DAD>
        media: <unknown type>
        status: inactive
ap1: flags=8802<BROADCAST,SIMPLEX,MULTICAST> mtu 1500
        ether a6:83:e7:aa:b7:f2
        media: autoselect
        status: inactive
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether a4:83:e7:aa:b7:f2
        inet6 fe80::1c7a:42c7:e73c:4e85%en0 prefixlen 64 secured scopeid 0x10
        inet 192.168.0.15 netmask 0xffffff00 broadcast 192.168.0.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
p2p0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2304
        ether 06:83:e7:aa:b7:f2
        media: autoselect
        status: inactive
awdl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1484
        ether 1e:66:15:20:d2:80
        inet6 fe80::1c66:15ff:fe20:d280%awdl0 prefixlen 64 scopeid 0x12
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
utun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 2000
        inet6 fe80::ac4a:9b83:5842:66d2%utun0 prefixlen 64 scopeid 0x13
        nd6 options=201<PERFORMNUD,DAD>
(base) Vikass-MacBook-Pro:LAB 2 vikas$
```

(2) - Running command "netstat"

```
(base) Vikass-MacBook-Pro:LAB 2 vikas$ netstat
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  192.168.0.15.50774     dfw28s22-in-f5.1.https ESTABLISHED
tcp4       0      0  192.168.0.15.50773     ec2-13-114-46-23.https ESTABLISHED
tcp4       0     31  192.168.0.15.50772     ec2-3-112-115-14.https LAST_ACK
tcp4       0     31  192.168.0.15.50771     ec2-3-112-115-14.https LAST_ACK
tcp4       0      0  192.168.0.15.50770     209.85.231.105.http    CLOSE_WAIT
tcp4       0      0  192.168.0.15.50769     atl14s08-in-f142.http  ESTABLISHED
tcp4       0      0  192.168.0.15.50761     104.16.30.34.https     ESTABLISHED
tcp4       0     31  192.168.0.15.50758     ec2-3-112-115-14.https LAST_ACK
tcp4       0      0  192.168.0.15.50756     53.224.186.35.bc.https ESTABLISHED
tcp4       0      0  192.168.0.15.50749     ec2-35-175-25-57.https ESTABLISHED
tcp4       0      0  192.168.0.15.50715     104.244.42.8.https     ESTABLISHED
tcp4       0      0  192.168.0.15.50690     151.101.1.69.https     ESTABLISHED
tcp6       0      0  fe80::aede:48ff:.49156 fe80::aede:48ff:.51140 ESTABLISHED
tcp4       0      0  192.168.0.15.49886     oi-in-f188.1e100.5228  ESTABLISHED
tcp4       0      0  192.168.0.15.49863     47.224.186.35.bc.https ESTABLISHED
tcp4       0      0  192.168.0.15.49859     192.168.0.13.8009      ESTABLISHED
tcp4       0      0  192.168.0.15.49858     221.244.190.35.b.4070  ESTABLISHED
tcp6       0      0  fe80::aede:48ff:.49854 fe80::aede:48ff:.51135 ESTABLISHED
tcp6       0      0  fe80::aede:48ff:.49853 fe80::aede:48ff:.51113 ESTABLISHED
tcp4       0      0  192.168.0.15.49836     17.57.144.86.5223      ESTABLISHED
tcp4       0      0  localhost.49281        localhost.49821        ESTABLISHED
tcp4       0      0  localhost.49821        localhost.49281        ESTABLISHED
tcp6       0      0  fe80::aede:48ff:.49536 fe80::aede:48ff:.51123 ESTABLISHED
tcp6       0      0  fe80::aede:48ff:.49512 fe80::aede:48ff:.51130 ESTABLISHED
tcp6       0      0  fe80::aede:48ff:.49178 fe80::aede:48ff:.51127 ESTABLISHED
tcp6       0      0  fe80::aede:48ff:.49157 fe80::aede:48ff:.51124 ESTABLISHED
tcp6       0      0  fe80::aede:48ff:.49152 fe80::aede:48ff:.59602 ESTABLISHED
udp4       0      0  *.64290                *.*
udp4       0      0  192.168.0.15.58742     on-in-f189.1e100.https
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.mdns                 *.*
udp46      0      0  *.mdns                 *.*
udp46      0      0  *.mdns                 *.*
udp46      0      0  *.mdns                 *.*
udp4       0      0  *.mdns                 *.*
udp46      0      0  *.mdns                 *.*
udp4       0      0  *.50552                *.*
udp4       0      0  *.mdns                 *.*
udp4       0      0  *.57621                *.*
udp46      0      0  *.amx-rms              *.*
udp4       0      0  *.amx-rms              *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.55605                *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
udp4       0      0  *.*                    *.*
```

Q5. Extra points (optional): suggest up to three related 'original' experiments (each with up to three short well-defined tasks) to use networking tools or commands to understand socket programming, UDP, TCP, MAC, IP addresses, or ARP/routing tables. Make them clear, brief and include short sample answers. Label them clearly as well (1. a, b, c. and 2. a, b, c, 3. …). Provide references and/or websites as appropriate.

Ans:- Since I am using MacOS, Hence using network commands is different from windows & linux machines.

1)- netstat command is used to obtain listing open sockets i.e. Active internet connections.

a)- "netstat -g" : This command displays information associated with multicast connections (related to Ip address classes)

b)- "netstat -apv TCP" - This command returns only TCP connections on the MACOS Device.

c) = "netstat -apv UDP" = This command return only UDP connections on the MAC os device.

Reference =

https://www.lifewire.com/using-netstat-command-on-mac-4176069

(2) = Arp command can be used to display and modify Arp cache Entries.

a) = "arp -d [Ip address]" = with this command we can delete the specific Ip address from the arp cache table.

b) = "arp -d -a" = with this command we can delete all the entries in arp table.

c) = "arp -s [Ip address] [MAC address]" = using this command we can add an entry manually to the arp cache table.

Reference =

https://krypted.com/mac-os-x/using-the-arp-command/

(3) = Tracert Command = Traceroute is a network diagnostic tool used to track in real time the pathway taken by a packet on an Ip network from source to destination. It reports the Ip addresses of all the routers pinged in between it also records the time taken for each hop the packet makes from source to destination.

a) = "traceroute -M [Host Ip address]"

This command can set the max time to live (max number of hops) used in outgoing probe packets. The default is 30 hops.

(b)= "traceroute [Host Ip] [packetsize]"

This command could be used to increase probe datagram length, by default it is 38 bytes.

(c)= "traceroute -s [src addres]" =

with this command The given Ip address could be used as source address in outgoing probe packets. This feature could be used on hosts with more than one Ip address, a source address could be forced to be something other than Ip address of interface packet is sent on.

Reference=

https://ss64.com/osx/traceroute.html