In part one of assignment I have firstly assessed all packets, via packet.summary and selected all DNS packets, and created a CSV file of all pcap7([15+152]%10). The code is is file assignment_1.py

Here the code was as follows:

```
tcp_services = {
    20: "FTP-DATA", 21: "FTP",
    22: "SSH",
    23: "Telnet",
    25: "SMTP", 465: "SMTPS", 587: '
    110: "POP3", 995: "POP3S",
    143: "IMAP", 993: "IMAPS",
    80: "HTTP", 443: "HTTPS",
    139: "NBTSession", 445: "SMB"
}
```

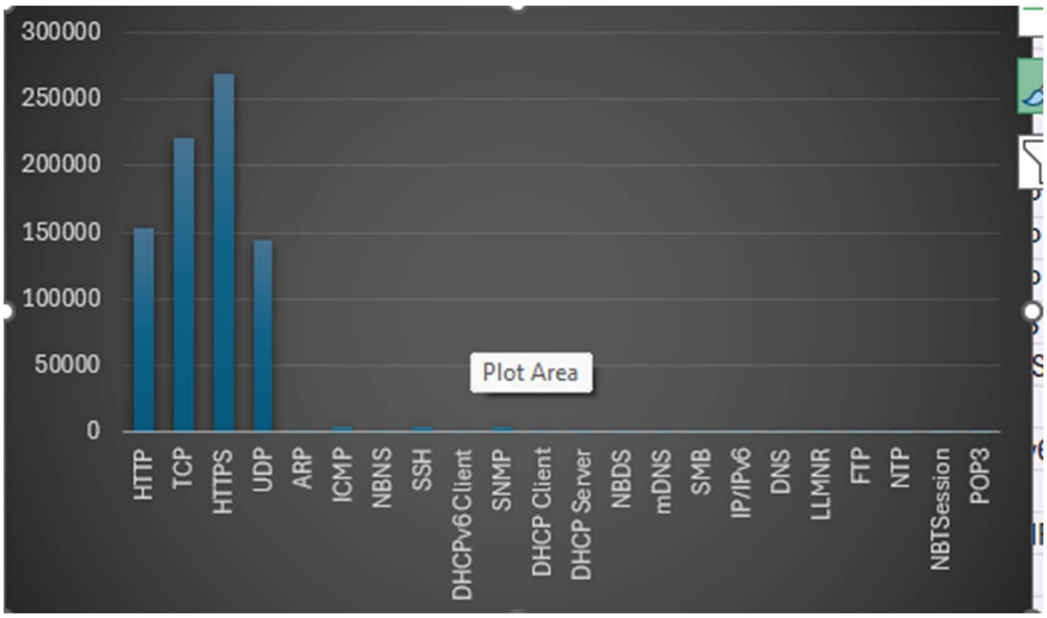
Describing ports that are fetched from TRANSPORT LAYERS AND THEN MAPPING TO CORRECT PROTOCOL DNS/SMTP/HTTPS/MDNS ETC...

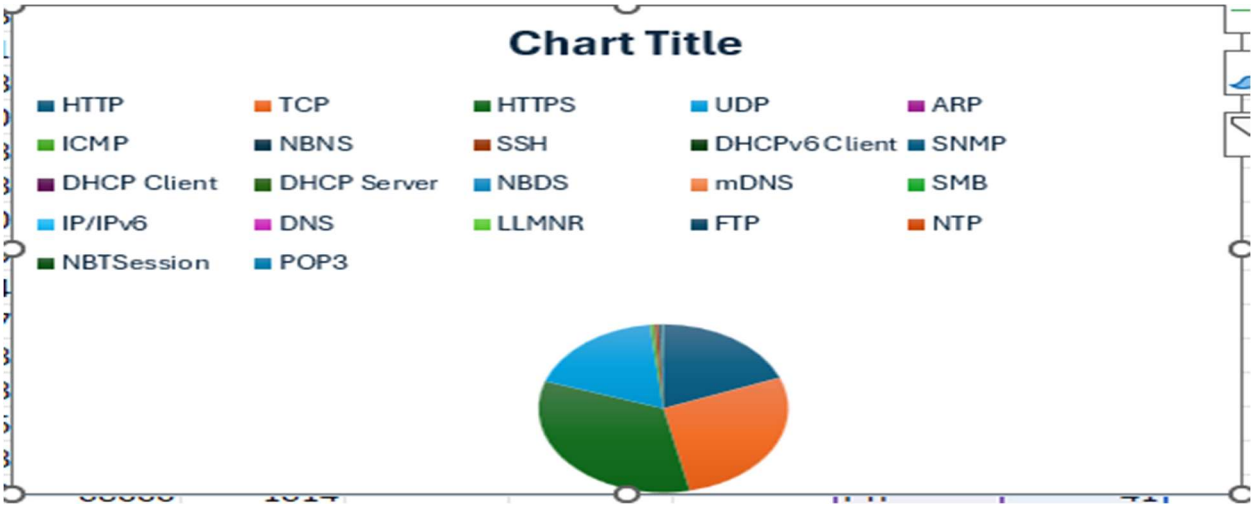
```
    # TCP
    if TCP in pkt:
        sport, dport = pkt[TCP].sport, pkt[TCP].dport
        proto = tcp_services.get(sport, tcp_services.get(dport, "TCP"))
        src_port, dst_port = sport, dport

    # UDP
    elif UDP in pkt:
        sport, dport = pkt[UDP].sport, pkt[UDP].dport
        proto = udp_services.get(sport, udp_services.get(dport, "UDP"))
        src_port, dst_port = sport, dport

    # ICMP / ICMPv6
    elif ICMP in pkt or pkt.haslayer("ICMPv6"):
        proto = "ICMP"

    # ARP
    elif ARP in pkt:
        proto = "ARP"

    # Generic IP/IPv6
    elif IP in pkt or IPv6 in pkt:
        proto = "IP/IPv6"
```

Here we resolve each packet into proper port's ip address.

Next we create a output.csv file that can be found in git repo.

The basic bar chart is as follows:





Then we come to clinet server that parses all the dns packets using valid indexes, and after that there were three options:

One was to use raw socket→but that would require us that we can't add custom header on top of the packet and then design custom protocols, the os does not add tcp/ip packets for such sockets and then for such packets it is mandatory to have the ip at start of packet.

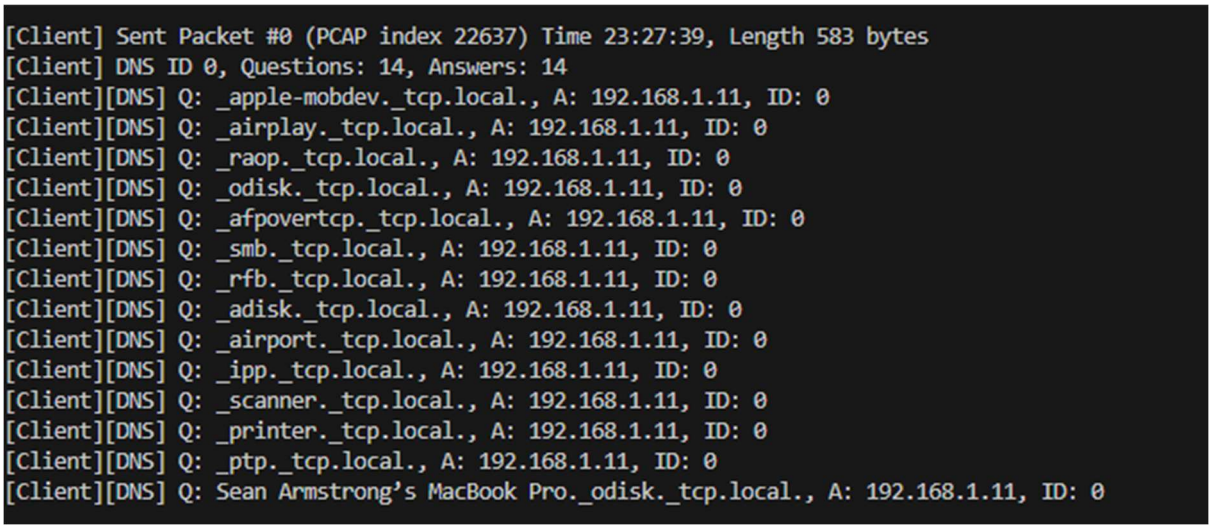
Second was the option of extracting the dns layer from the pckt that was read from pcap via scapy and then use the less used fields like ancount etc that are by default 0 , to send more meaningful header. This also involved changing the raw, tcp ip values of the packet and then use appropriate socket for transfer.. the tcp/ip change is mandatory because the ports/ip addresses in pcap will not be of use in internet via my device.

Third was that both client and server communicating via port 9000 and not standard 53[due to it being occupied in windows by default] over TCP, so that if we add header then the size increases so tcp will not cause issue.We added a custom header at start of raw packet bytes which was a magic(so that verification done properly)#2bytes, followed by a 6 byte header as follows: 1 byte for hh,mm,ss each, 3 bytes for id variable. And 4 bytes to specify length of packet... and then the raw packet.

The server parsed the packets and then created a dns response for all the questions answered in the same way as a true response. More learnings are as follows: I realised there were mDNS packets as well on port 5343 and those had local seeming domain names, and multiple questions per packet all of which have been responded vis resource record of same ip and also the type was initially tried to be type of the question but errors occurred so type was fixed to ipv4... This was a connection oriented talk between server and client. Client parsed the packets and built a report.

Clients ' request payload: the exact packet from pcap in server after which there are layers of TCP/ IP that os adds on client side and removes it at the server layer and hence viewed via sniffing packets between them to view TCP/IP.

The image shows client sending the request...



Client Answering the Response:

```
[Server] Received Packet ID 1 at 22:23:23, length 73 bytes
[DEBUG] Summary: Ether / IP / UDP / DNS Qry b'wikipedia.org.'
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = da:7f:5c:2b:6e:df
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 59
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x457b
     src       = 10.240.26.55
     dst       = 8.8.8.8
     \options   \
###[ UDP ]###
        sport     = 15841
        dport     = domain
        len       = 39
        chksum    = 0x6863
###[ DNS ]###
           id        = 0
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 1
           ra        = 0
           z         = 0
           ad        = 0
           cd        = 0
           rcode     = ok
           qdcount   = 1
           ancount   = 0
           nscount   = 0
           arcount   = 0
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = b'wikipedia.org.'
            |  qtype     = A
            |  unicastresponse= 0
            |  qclass    = IN
           \an        \
           \ns        \
           \ar        \
```

```
[Client] Sent Packet #1 (PCAP index 44981) Time 22:17:57, Length 73 bytes
[Client] DNS ID 1, Questions: 1, Answers: 1
[Client][DNS] Q: wikipedia.org., A: 192.168.1.12, ID: 1
```

Clients' response Payload:

```
[Server] Answering Packet ID 1 with IP 192.168.1.12
###[ DNS ]###
  id        = 1
  qr        = 1
  opcode    = QUERY
  aa        = 1
  tc        = 0
  rd        = 1
  ra        = 0
  z         = 0
  ad        = 0
  cd        = 0
  rcode     = ok
  qdcount   = None
  ancount   = 1
  nscount   = None
  arcount   = None
  \qd        \
   |###[ DNS Question Record ]###
   |  qname     = b'wikipedia.org.'
   |  qtype     = A
   |  unicastresponse= 0
   |  qclass    = IN
  \an        \
   |###[ DNS Resource Record ]###
   |  rrname    = b'wikipedia.org.'
   |  type      = A
   |  cacheflush= 0
   |  rclass    = IN
   |  ttl       = 60
   |  rdlen     = None
   |  rdata     = 192.168.1.12
  \ns        \
  \ar        \
```

Sending Packets… mainly to notice that what ip addresses are sent makes sense::

```
[Client] Sent Packet #1 (PCAP index 44981) Time 22:17:57, Length 73 bytes
[Client] DNS ID 1, Questions: 1, Answers: 1
[Client][DNS] Q: wikipedia.org., A: 192.168.1.12, ID: 1
```

```
[Client] Sent Packet #2 (PCAP index 143275) Time 22:23:53, Length 111 bytes
[Client] DNS ID 2, Questions: 2, Answers: 2
[Client][DNS] Q: Brother MFC-7860DW._pdl-datastream._tcp.local., A: 192.168.1.13, ID: 2
[Client][DNS] Q: Brother MFC-7860DW._pdl-datastream._tcp.local., A: 192.168.1.13, ID: 2
```

```
Loopback / IP / TCP 127.0.0.1:57377 > 127.0.0.1:9000 PA / Raw
Loopback / IP / TCP 127.0.0.1:9000 > 127.0.0.1:57377 A
Loopback / IP / TCP 127.0.0.1:9000 > 127.0.0.1:57377 PA / Raw
Loopback / IP / TCP 127.0.0.1:57377 > 127.0.0.1:9000 A
Loopback / IP / TCP 127.0.0.1:57377 > 127.0.0.1:9000 PA / Raw
Loopback / IP / TCP 127.0.0.1:9000 > 127.0.0.1:57377 A
Loopback / IP / TCP 127.0.0.1:9000 > 127.0.0.1:57377 PA / Raw
Loopback / IP / TCP 127.0.0.1:57377 > 127.0.0.1:9000 A
Loopback / IP / UDP / mDNS Ans b'LAPTOP-JJ0R8OQ9._dosvc._tcp.local.'
Loopback / IPv6 / UDP / mDNS Ans b'LAPTOP-JJ0R8OQ9._dosvc._tcp.local.'
Loopback / IP / UDP / mDNS Qry b'LAPTOP-JJ0R8OQ9._dosvc._tcp.local.'
Loopback / IPv6 / UDP / mDNS Qry b'LAPTOP-JJ0R8OQ9._dosvc._tcp.local.'
Loopback / IP / UDP / mDNS Qry b'LAPTOP-JJ0R8OQ9._dosvc._tcp.local.'
Loopback / IPv6 / UDP / mDNS Qry b'LAPTOP-JJ0R8OQ9._dosvc._tcp.local.'
Loopback / IP / UDP / mDNS Qry b'LAPTOP-JJ0R8OQ9._dosvc._tcp.local.'
Loopback / IPv6 / UDP / mDNS Qry b'LAPTOP-JJ0R8OQ9._dosvc._tcp.local.'
Loopback / IP / UDP / mDNS Ans b'LAPTOP-JJ0R8OQ9._dosvc._tcp.local.'
Loopback / IPv6 / UDP / mDNS Ans b'LAPTOP-JJ0R8OQ9._dosvc._tcp.local.'
Loopback / IP / UDP / mDNS Ans
Loopback / IPv6 / UDP / mDNS Ans
```

The above image shows the sniffer.py result between client and server that shows server on local device via loopback and running on port 9000. And client being assigned port 57377. It also shows the IP AND TCP Layers.

```
[Server] Answering Packet ID 2 with IP 192.168.1.13
###[ DNS ]###
  id        = 2
  qr        = 1
  opcode    = QUERY
  aa        = 1
  tc        = 0
  rd        = 1
  ra        = 0
  z         = 0
  ad        = 0
  cd        = 0
  rcode     = ok
  qdcount   = None
  ancount   = 2
  nscount   = None
  arcount   = None
  \qd        \
   |###[ DNS Question Record ]###
   |  qname     = b'Brother MFC-7860DW._pdl-datastream._tcp.local.'
   |  qtype     = SRV
   |  unicastresponse= 0
   |  qclass    = IN
   |###[ DNS Question Record ]###
   |  qname     = b'Brother MFC-7860DW._pdl-datastream._tcp.local.'
   |  qtype     = TXT
   |  unicastresponse= 0
   |  qclass    = IN
  \an        \
   |###[ DNS Resource Record ]###
   |  rrname    = b'Brother MFC-7860DW._pdl-datastream._tcp.local.'
   |  type      = SRV
   |  cacheflush= 0
   |  rclass    = IN
   |  ttl       = 60
   |  rdlen     = None
   |  rdata     = b'192.168.1.13'
   |###[ DNS Resource Record ]###
   |  rrname    = b'Brother MFC-7860DW._pdl-datastream._tcp.local.'
   |  type      = TXT
   |  cacheflush= 0
   |  rclass    = IN
   |  ttl       = 60
   |  rdlen     = None
   |  rdata     = ['192.168.1.13']
  \ns        \
  \ar        \
```

Domain	IP	ID
'_apple-mobdev._tcp.local.'	'192.168.1.11	0
'_apple-mobdev._tcp.local.'	'192.168.1.12	1
'wikipedia.org.'	'192.168.1.13	2
'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.14	3
'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.15	4
'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.15	5
'reddit.com.'	'192.168.1.11	6
'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.12	7
'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.13	8
'apple.com.'	'192.168.1.14	9
'twitter.com.'	'192.168.1.15	10
'_apple-mobdev._tcp.local.'	'192.168.1.11	11
'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.12	12
'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.13	13
'yahoo.com.'	'192.168.1.14	14
'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.15	15
'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.11	16
'_apple-mobdev._tcp.local.'	'192.168.1.12	17

'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.14	18
'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.15	19
'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.11	20
'Brother MFC-7860DW._pdl-datastream._tcp.local.'	'192.168.1.12	21
'linkedin.com.'	'192.168.1.13	22

This is a reponse to MDNS PACKET. A multicast DNS.