SIT202: Computer Networks and Communication
Leaning Evidence for Active Class Task 4

Name: Kenisha Corera
Student ID: C23020001

Members in this group activity task:
Nishad – C23110001
Kenisha – C23020001
Shekaina – C23110002
Raaid – C23020004
Pavithran – C22060015

Activity 1

1. Group discussion: Assume that you are helping your friend to move his/her house and your responsibility is to take care of moving goods in the kitchen.
   a. What method would you take if your friend needs to pack everything within an hour because the moving truck will be there soon?
      - If your friend has to pack everything in an hour since the moving truck is coming shortly, then efficiency and speed should be the main priorities. Here's a methodical approach:
        - Set Essential Items in Order of Priority: Pack the most important things first, such as dishes, cutlery and regularly used equipment.
        - Make use of the Containers available: Don't stress too much about sorting just use any cartons, bags or containers that are available.
        - Gather Fastly: As soon as you can, stuff everything into boxes without giving too much thought to organize.
        - Reduce the amount of Wrapping: To save time, only lightly wrap delicate goods.

   b. How would your method change if you have more time and your friend would like to make sure each item in the kitchen is safely delivered, and your friend wants to keep those on the same order it was on the previous house?
      - A strategy centered on meticulous packing and organizing would be used if you had more time and your time and your friend wanted to make sure every kitchen item was delivered securely and maintained in the same order as in the previous house:
        - Sort Items into Categories: Put utensils, dishes, pots and pans and appliance in group with related goods.
        - Marker Boxes: Indicate on each box exactly what's inside and which room in belongs in.
        - Use appropriate packaging supplies: To safeguard delicate things use robust boxes, packing paper and bubble wrap.
        - Keep the Peace: Assemble the object in the boxes according to their original arrangement in the kitchen by packing them logically.
        - Make a list: Make an inventory list before the relocation to ensure nothing is forgotten or lots

c. Now, can you relate the above scenarios to the well-known two transport layer protocols? Consider different scenarios including large file transfer and faster communication.

- In networking, the two most well-known transport layer protocols are UDP (user Datagram Protocol) and TCP (Transmission Control Protocols). Here's how we can connect the aforementioned scenarios to these protocols:
- Scenario 1- Fast Packing (Resembling UDP)
    - UDP (User Datagram Protocol): Instead of reliability, this protocol prioritizes speed. It transmits packets in an unconnected manner and makes no assurances on delivery, sequencing or error-checking.
    - In connection with Fast packing: It is similar to UDP in that everything gets packed in less than an hour. Moving everything quickly is the main objective, even if it means that certain things may not be packed precisely or may get mixed up. It's speed not dependability that matters.
- Scenario 2- Thorough Packing (Comparable to TCP)
    - TCP (Transmission Control Protocol): This protocol guarantees packet delivery that is dependable well- organized and error- checked. Prior to data transfer it creates a link and preserves packet order
    - Connection to Cautionary Packing: An approach that is similar to TCP involves taking extra time to pack, guaranteeing that every item is delivered safely and maintained in the same sequence. Making ensuring everything is properly packaged and delivered in the right sequence is the main objective. Even if it takes longer, accuracy and reliability come first.
- Different Scenarios:
  Fast Communication (Similar to UDP)
    - In application such as online gaming and live broadcasting speed is more important than reliability. Since minor packet losses may be tolerated, UDP is a preferable option because of its quicker transmission speed and reduced latency.

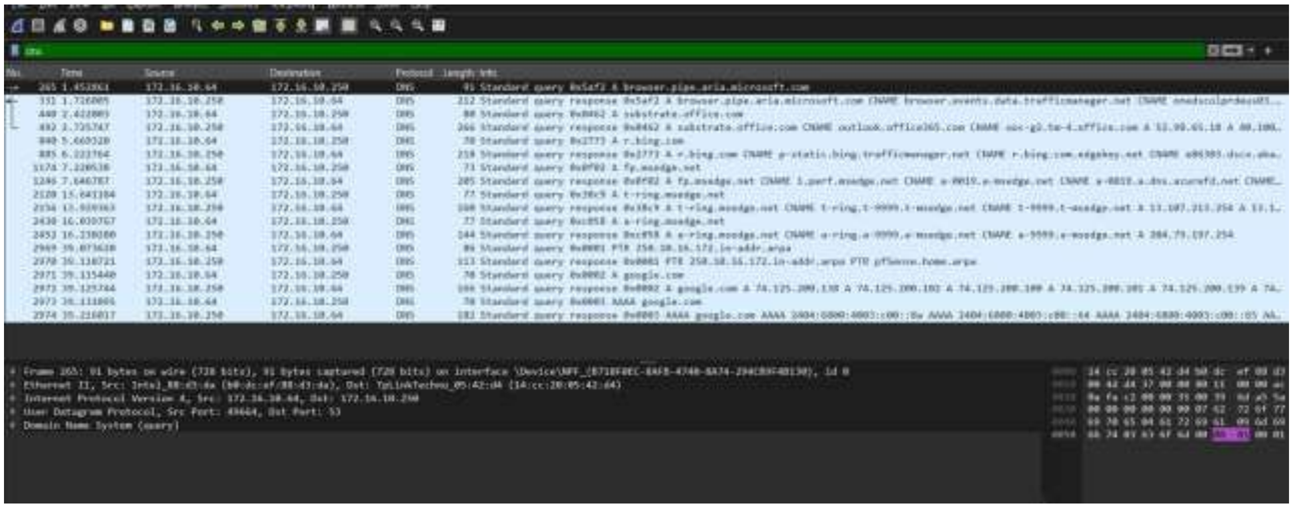
  Large File Transfer (Similar to TCP)
    - To guarantee that the complete file is received accurately while transferring huge files, dependability and order are essential. Since TCP guarantees data integrity and proper sequencing it is perfect for this use case.

To summarize, careless packing is similar to TCP in that it ensures order and dependability at the expense of speed whereas rapid packing is similar to UDP in that it prioritize speed above reliability. Which technique or protocol is best will depend on the many applications or scenarios.

Activity 2

1. Open Wireshark to capture packets, open the web browser and use an application that uses UDP (perhaps DNS). Stop the packet capture.

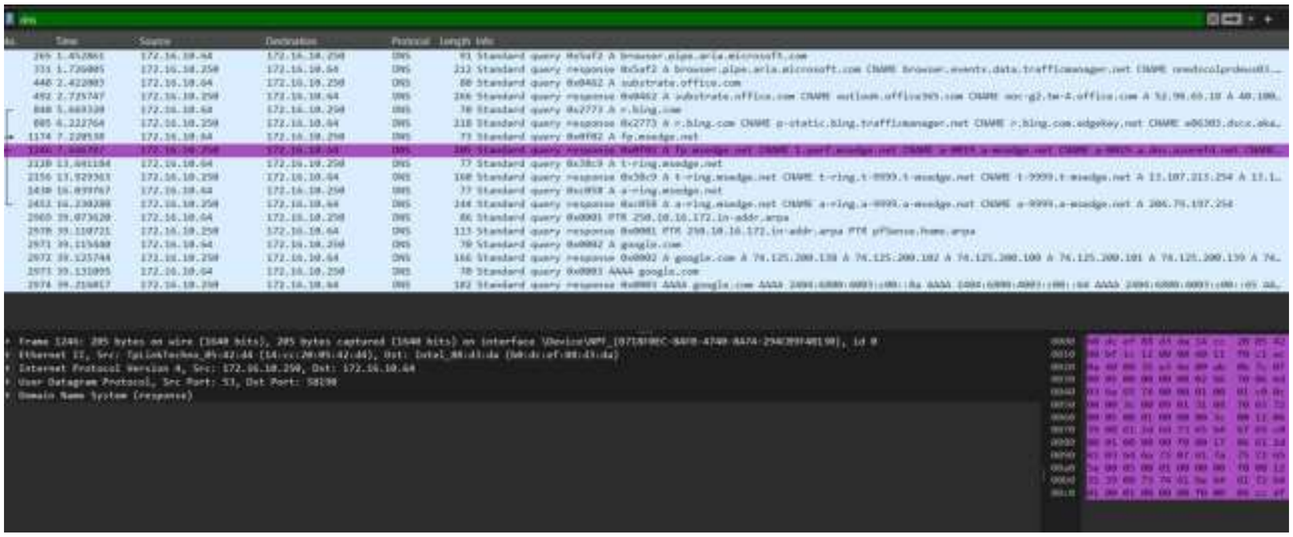
2. Use the filter to display UDP packets (it is UDP segments).



```
dns
No.   Time       Source         Destination    Protocol Length Info
 265  1.453061   172.16.10.64   172.16.10.250  DNS       91 Standard query 0x5af2 A browser.pipe.aria.microsoft.com
 331  1.726005   172.16.10.250  172.16.10.64   DNS      212 Standard query response 0x5af2 A browser.pipe.aria.microsoft.com CNAME browser.events.data.trafficmanager.net CNAME onedscolprdeus03…
 440  2.422003   172.16.10.64   172.16.10.250  DNS       80 Standard query 0x0462 A substrate.office.com
 492  2.725747   172.16.10.250  172.16.10.64   DNS      266 Standard query response 0x0462 A substrate.office.com CNAME outlook.office365.com CNAME ooc-g2.tm-4.office.com A 52.98.65.18 A 40.100…
 840  5.669320   172.16.10.64   172.16.10.250  DNS       70 Standard query 0x2773 A r.bing.com
 885  6.222764   172.16.10.250  172.16.10.64   DNS      218 Standard query response 0x2773 A r.bing.com CNAME p-static.bing.trafficmanager.net CNAME r.bing.com.edgekey.net CNAME e86303.dscx.aka…
1174  7.220538   172.16.10.64   172.16.10.250  DNS       73 Standard query 0x0f02 A fp.msedge.net
1246  7.646787   172.16.10.250  172.16.10.64   DNS      205 Standard query response 0x0f02 A fp.msedge.net CNAME 1.perf.msedge.net CNAME a-0019.a-msedge.net CNAME a-0019.a.dns.azurefd.net CNAME…
2120 13.841184   172.16.10.64   172.16.10.250  DNS       77 Standard query 0x38c9 A t-ring.msedge.net
2156 13.929363   172.16.10.250  172.16.10.64   DNS      160 Standard query response 0x38c9 A t-ring.msedge.net CNAME t-ring.t-9999.t-msedge.net CNAME t-9999.t-msedge.net A 13.107.213.254 A 13.1…
2430 16.039767   172.16.10.64   172.16.10.250  DNS       77 Standard query 0xc858 A a-ring.msedge.net
2453 16.230280   172.16.10.250  172.16.10.64   DNS      144 Standard query response 0xc858 A a-ring.msedge.net CNAME a-ring.a-9999.a-msedge.net CNAME a-9999.a-msedge.net A 204.79.197.254
2969 39.073620   172.16.10.64   172.16.10.250  DNS       86 Standard query 0x0001 PTR 250.10.16.172.in-addr.arpa
2970 39.110721   172.16.10.250  172.16.10.64   DNS      113 Standard query response 0x0001 PTR 250.10.16.172.in-addr.arpa PTR pfSense.home.arpa
2971 39.115440   172.16.10.64   172.16.10.250  DNS       70 Standard query 0x0002 A google.com
2972 39.125744   172.16.10.250  172.16.10.64   DNS      166 Standard query response 0x0002 A google.com A 74.125.200.138 A 74.125.200.102 A 74.125.200.100 A 74.125.200.101 A 74.125.200.139 A 74…
2973 39.131095   172.16.10.64   172.16.10.250  DNS       78 Standard query 0x0003 AAAA google.com
2974 39.216017   172.16.10.250  172.16.10.64   DNS      102 Standard query response 0x0003 AAAA google.com AAAA 2404:6800:4003:c00::8a AAAA 2404:6800:4003:c00::64 AAAA 2404:6800:4003:c00::65 AA…

> Frame 265: 91 bytes on wire (728 bits), 91 bytes captured (728 bits) on interface \Device\NPF_{B718F0EC-8AFB-4740-8A74-294CB9F4B130}, id 0
> Ethernet II, Src: Intel_88:d3:da (b0:dc:ef:88:d3:da), Dst: TpLinkTechno_05:42:d4 (14:cc:20:05:42:d4)
> Internet Protocol Version 4, Src: 172.16.10.64, Dst: 172.16.10.250
> User Datagram Protocol, Src Port: 49664, Dst Port: 53
> Domain Name System (query)

0000  14 cc 20 05 42 d4 b0 dc ef 88 d3
0010  00 4d 44 37 00 00 80 11 00 00 ac
0020  0a fa c2 00 00 35 00 39 6d a5 5a
0030  00 00 00 00 00 00 07 62 72 6f 77
0040  69 70 65 04 61 72 69 61 09 6d 69
0050  66 74 03 63 6f 6d 00 00 01 00 01
```

3. Now you are ready to analyze UDP. The following questions can guide you to conduct your analysis. Make sure that you take enough screenshots and notes. You will use those to prepare your task submissions.

a. Pick one UDP packet from the filtered packets. Examine the number of fields that are in the header of the selected UDP packet. You can take a screenshot of the packet and name and explain the fields in the UDP packet.



```
dns
No.   Time       Source         Destination    Protocol Length Info
 265  1.453061   172.16.10.64   172.16.10.250  DNS       91 Standard query 0x5af2 A browser.pipe.aria.microsoft.com
 331  1.726005   172.16.10.250  172.16.10.64   DNS      212 Standard query response 0x5af2 A browser.pipe.aria.microsoft.com CNAME browser.events.data.trafficmanager.net CNAME onedscolprdeus03…
 440  2.422003   172.16.10.64   172.16.10.250  DNS       80 Standard query 0x0462 A substrate.office.com
 492  2.725747   172.16.10.250  172.16.10.64   DNS      266 Standard query response 0x0462 A substrate.office.com CNAME outlook.office365.com CNAME ooc-g2.tm-4.office.com A 52.98.65.18 A 40.100…
 840  5.669320   172.16.10.64   172.16.10.250  DNS       70 Standard query 0x2773 A r.bing.com
 885  6.222764   172.16.10.250  172.16.10.64   DNS      218 Standard query response 0x2773 A r.bing.com CNAME p-static.bing.trafficmanager.net CNAME r.bing.com.edgekey.net CNAME e86303.dscx.aka…
1174  7.220538   172.16.10.64   172.16.10.250  DNS       73 Standard query 0x0f02 A fp.msedge.net
1246  7.646787   172.16.10.250  172.16.10.64   DNS      205 Standard query response 0x0f02 A fp.msedge.net CNAME 1.perf.msedge.net CNAME a-0019.a-msedge.net CNAME a-0019.a.dns.azurefd.net CNAME…
2120 13.841184   172.16.10.64   172.16.10.250  DNS       77 Standard query 0x38c9 A t-ring.msedge.net
2156 13.929363   172.16.10.250  172.16.10.64   DNS      160 Standard query response 0x38c9 A t-ring.msedge.net CNAME t-ring.t-9999.t-msedge.net CNAME t-9999.t-msedge.net A 13.107.213.254 A 13.1…
2430 16.039767   172.16.10.64   172.16.10.250  DNS       77 Standard query 0xc858 A a-ring.msedge.net
2453 16.230280   172.16.10.250  172.16.10.64   DNS      144 Standard query response 0xc858 A a-ring.msedge.net CNAME a-ring.a-9999.a-msedge.net CNAME a-9999.a-msedge.net A 204.79.197.254
2969 39.073620   172.16.10.64   172.16.10.250  DNS       86 Standard query 0x0001 PTR 250.10.16.172.in-addr.arpa
2970 39.110721   172.16.10.250  172.16.10.64   DNS      113 Standard query response 0x0001 PTR 250.10.16.172.in-addr.arpa PTR pfSense.home.arpa
2971 39.115440   172.16.10.64   172.16.10.250  DNS       70 Standard query 0x0002 A google.com
2972 39.125744   172.16.10.250  172.16.10.64   DNS      166 Standard query response 0x0002 A google.com A 74.125.200.138 A 74.125.200.102 A 74.125.200.100 A 74.125.200.101 A 74.125.200.139 A 74…
2973 39.131095   172.16.10.64   172.16.10.250  DNS       78 Standard query 0x0003 AAAA google.com
2974 39.216017   172.16.10.250  172.16.10.64   DNS      102 Standard query response 0x0003 AAAA google.com AAAA 2404:6800:4003:c00::8a AAAA 2404:6800:4003:c00::64 AAAA 2404:6800:4003:c00::65 AA…

> Frame 1246: 205 bytes on wire (1640 bits), 205 bytes captured (1640 bits) on interface \Device\NPF_{B718F0EC-8AFB-4740-8A74-294CB9F4B130}, id 0
> Ethernet II, Src: TpLinkTechno_05:42:d4 (14:cc:20:05:42:d4), Dst: Intel_88:d3:da (b0:dc:ef:88:d3:da)
> Internet Protocol Version 4, Src: 172.16.10.250, Dst: 172.16.10.64
> User Datagram Protocol, Src Port: 53, Dst Port: 58190
> Domain Name System (response)
```

- The UDP packet has a simple header with four main parts.

1. Source Port (16 bits)
- Value in Frame 1246: 53
- Explanation: This shows the port number of the program or service that sent the packet. Here, port 53 is used by DNS (Domain Name System), so this packet is a DNS response.

2. Destination Port (16 bits)
- Value in Frame 1246: 58190
- Explanation: This shows the port number where the packet should be sent. Port 58190 is likely a temporary port that the client is using to get the DNS response.

3. Length (16 bits)

- Explanation: This shows the total size of the UDP packet, including both the header and the data. While the exact number isn't given, we know the packet is 205 bytes, so this includes the header (8 bytes) and the data.

4. Checksum (16 bits)

- Explanation: The checksum is used to check for errors in the packet. It helps ensure that the data hasn't changed during transmission. The exact value isn't provided, but it's calculated using the header, data, and some parts of the IP header.

Data (Variable Length)

- Explanation: This is the actual information in the packet. In this case, it's a DNS response, which includes the answer to a DNS query. The data size can be found by subtracting the 8-byte header from the total length shown in the length field.

b. Can you identify the length of each UDP header field from this UDP packet? You can indicate the length in bytes. You may want to examine the displayed information in Wireshark's packet content field.

- Each field in the UDP header has a specific length, which is consistent across all UDP packets:
  1. Source Port: 2 bytes (16 bits)
  2. Destination Port: 2 bytes (16 bits)
  3. Length: 2 bytes (16 bits)
  4. Checksum: 2 bytes (16 bits) The total size of the UDP header is 8 bytes.

c. Can you explain the value in the Length field? What exactly this value indicates? You can verify your answer by examining captured UDP packets.

- The Length field in a UDP packet indicates the total size of the packet, including the UDP header and the data payload. The value in the Length field is given in bytes. The Length field value is 205 bytes, it means that the entire UDP packet, including the 8-byte header and the data, is 205 bytes long. The payload data would then be 205 - 8 = 197 bytes.

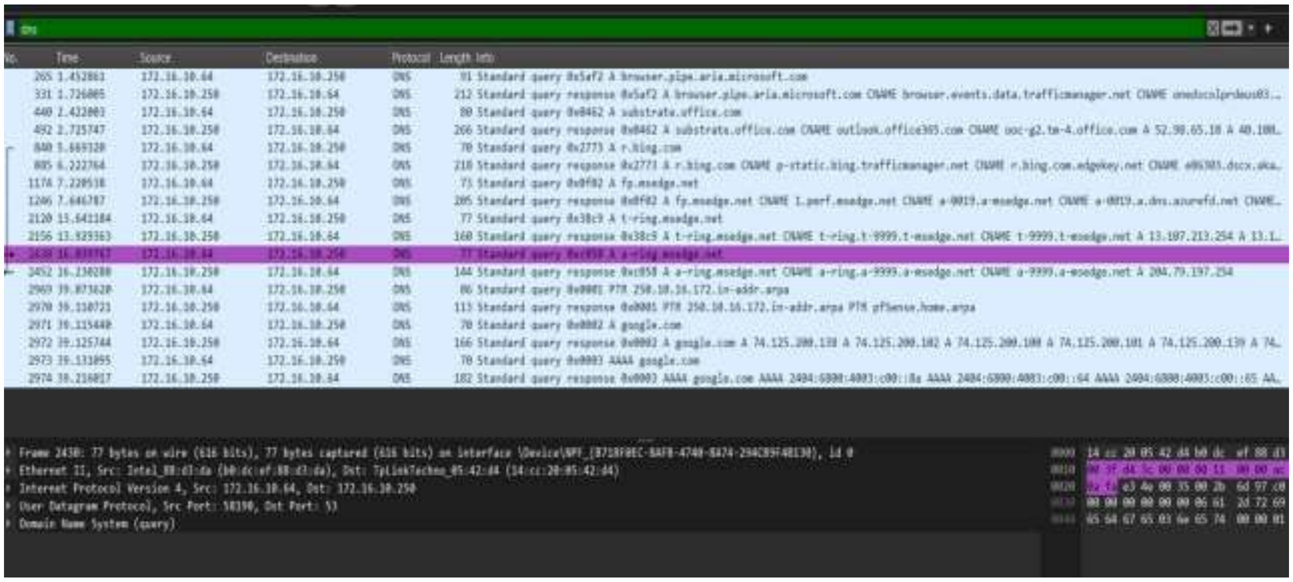
d. Is there a maximum number of bytes that we can include in UDP payload?

- The maximum size of a UDP packet is 65,535 bytes, which includes the header and the payload. Given the header is 8 bytes, the maximum size for the payload is 65,535 - 8 = 65,527 bytes. However, the actual maximum payload size can be limited by the underlying network protocols and infrastructure, such as the Maximum Transmission Unit (MTU) of the network.

e. Can you identify the protocol number given for UDP?

- The protocol number assigned to UDP in the IP protocol field is 17. This number identifies the UDP protocol at the IP layer.

f. Examine two consecutive UDP packets your host sends/receives. Explain the relationship between the port numbers in these two packets.



```
dns
No.   Time        Source         Destination    Protocol Length Info
 265  1.452863    172.16.10.64   172.16.10.250  DNS   91 Standard query 0x5af2 A browser.pipe.aria.microsoft.com
 331  1.726005    172.16.10.250  172.16.10.64   DNS  212 Standard query response 0x5af2 A browser.pipe.aria.microsoft.com CNAME browser.events.data.trafficmanager.net CNAME onedscolprdeus03...
 440  2.422003    172.16.10.64   172.16.10.250  DNS   80 Standard query 0x0462 A substrate.office.com
 492  2.725747    172.16.10.250  172.16.10.64   DNS  266 Standard query response 0x0462 A substrate.office.com CNAME outlook.office365.com CNAME ooc-g2.tm-4.office.com A 52.98.65.18 A 40.100...
 840  5.669320    172.16.10.64   172.16.10.250  DNS   70 Standard query 0x2773 A r.bing.com
 885  6.222764    172.16.10.250  172.16.10.64   DNS  218 Standard query response 0x2773 A r.bing.com CNAME p-static.bing.trafficmanager.net CNAME r.bing.com.edgekey.net CNAME e86303.dscx.aka...
1174  7.220518    172.16.10.64   172.16.10.250  DNS   73 Standard query 0x0f02 A fp.msedge.net
1246  7.646787    172.16.10.250  172.16.10.64   DNS  205 Standard query response 0x0f02 A fp.msedge.net CNAME 1.perf.msedge.net CNAME a-0019.a-msedge.net CNAME a-0019.a.dns.azurefd.net CNAME...
2120 13.641184    172.16.10.64   172.16.10.250  DNS   77 Standard query 0x38c9 A t-ring.msedge.net
2156 13.929363    172.16.10.250  172.16.10.64   DNS  160 Standard query response 0x38c9 A t-ring.msedge.net CNAME t-ring.t-9999.t-msedge.net CNAME t-9999.t-msedge.net A 13.107.213.254 A 13.1...
2430 16.[illegible] 172.16.10.64 172.16.10.250  DNS   77 Standard query 0xc058 A a-ring.msedge.net
2452 16.230288    172.16.10.250  172.16.10.64   DNS  144 Standard query response 0xc058 A a-ring.msedge.net CNAME a-ring.a-9999.a-msedge.net CNAME a-9999.a-msedge.net A 204.79.197.254
2969 39.073620    172.16.10.64   172.16.10.250  DNS   86 Standard query 0x0001 PTR 250.10.16.172.in-addr.arpa
2970 39.110721    172.16.10.250  172.16.10.64   DNS  113 Standard query response 0x0001 PTR 250.10.16.172.in-addr.arpa PTR pfSense.home.arpa
2971 39.115440    172.16.10.64   172.16.10.250  DNS   70 Standard query 0x0002 A google.com
2972 39.125744    172.16.10.250  172.16.10.64   DNS  166 Standard query response 0x0002 A google.com A 74.125.200.138 A 74.125.200.102 A 74.125.200.100 A 74.125.200.101 A 74.125.200.139 A 74...
2973 39.131095    172.16.10.64   172.16.10.250  DNS   70 Standard query 0x0003 AAAA google.com
2974 39.216017    172.16.10.250  172.16.10.64   DNS  182 Standard query response 0x0003 AAAA google.com AAAA 2404:6800:4003:c00::8a AAAA 2404:6800:4003:c00::64 AAAA 2404:6800:4003:c00::65 AA...

> Frame 2430: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface \Device\NPF_{B718F0EC-8AF8-4740-8474-294CB9F48130}, id 0
> Ethernet II, Src: Intel_88:d3:da (b0:dc:ef:88:d3:da), Dst: TpLinkTechno_05:42:d4 (14:cc:20:05:42:d4)
> Internet Protocol Version 4, Src: 172.16.10.64, Dst: 172.16.10.250
> User Datagram Protocol, Src Port: 58190, Dst Port: 53
> Domain Name System (query)

0000  14 cc 20 05 42 d4 b0 dc  ef 88 d3
0010  00 3f d4 5c 00 00 80 11  00 00 ac
0020  0a fa e3 4e 00 35 00 2b  6d 97 c0
0030  00 00 00 00 00 00 06 61  2d 72 69
0040  6e 67 06 6d 73 65 64 67  65 03 6e
```

- In the two UDP packets provided:

1. Frame 2430:
- Source Port: 58190
- Destination Port: 53   - Type: DNS Query

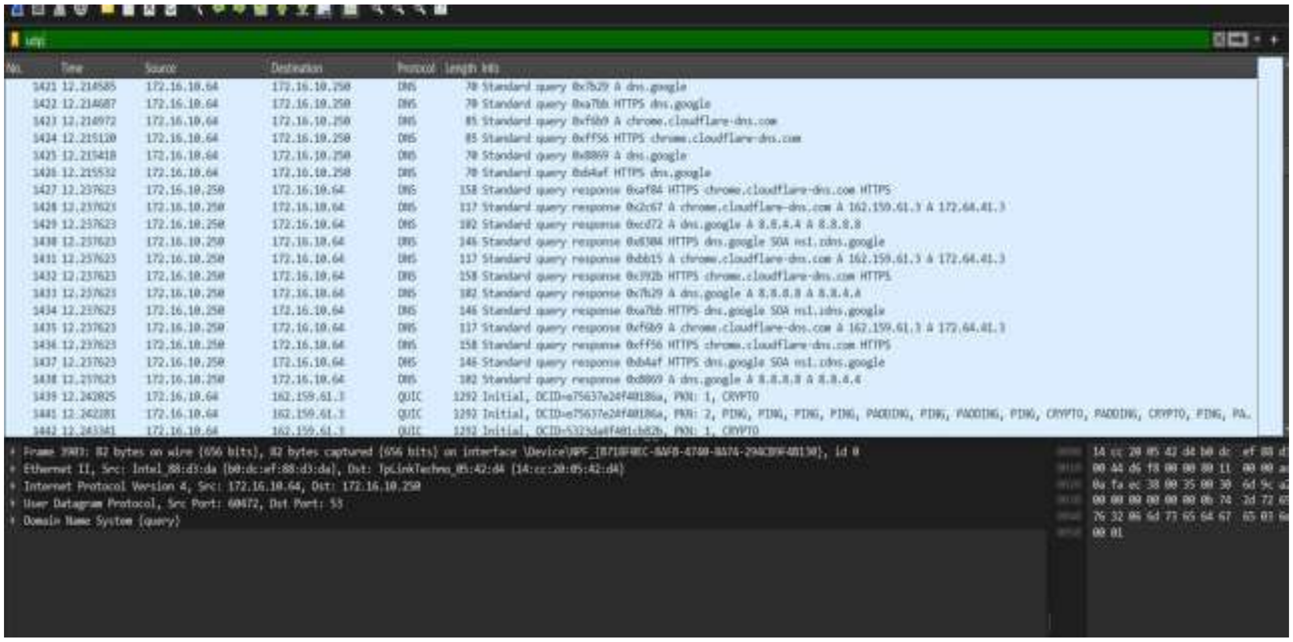
2. Frame 2452:
- Source Port: 53
- Destination Port: 58190   - Type: DNS Response

Relationship Between Port Numbers

- Source and Destination Ports: In the first packet (Frame 2430), the client (at IP 172.16.10.64) sends a DNS query from a high-numbered source port (58190) to the DNS server's standard port (53). In the second packet (Frame 2452), the DNS server (at IP 172.16.10.250) sends a response back to the client's source port (58190).

- Port Reversal: The roles of the source and destination ports are reversed between the query and the response. The client used port 58190 to send the query, and the DNS server responds to that port, while using its standard port 53 as the source.

g. Clear the cache and run another application such as MSTeams while using Web browsing. Capture packets of these two applications. Now you can analyze the captured UDP packets again. Do all UDP packets captured in Wireshark use the same port number? Explain your answer.



```
udp

No.   Time       Source         Destination    Protocol Length Info
1421 12.214585  172.16.10.64   172.16.10.250  DNS        70 Standard query 0x7b29 A dns.google
1422 12.214687  172.16.10.64   172.16.10.250  DNS        70 Standard query 0xa7bb HTTPS dns.google
1423 12.214972  172.16.10.64   172.16.10.250  DNS        85 Standard query 0xf6b9 A chrome.cloudflare-dns.com
1424 12.215120  172.16.10.64   172.16.10.250  DNS        85 Standard query 0xff56 HTTPS chrome.cloudflare-dns.com
1425 12.215418  172.16.10.64   172.16.10.250  DNS        70 Standard query 0x8869 A dns.google
1426 12.215532  172.16.10.64   172.16.10.250  DNS        70 Standard query 0xb4af HTTPS dns.google
1427 12.237623  172.16.10.250  172.16.10.64   DNS       158 Standard query response 0xaf84 HTTPS chrome.cloudflare-dns.com HTTPS
1428 12.237623  172.16.10.250  172.16.10.64   DNS       117 Standard query response 0x2c67 A chrome.cloudflare-dns.com A 162.159.61.3 A 172.64.41.3
1429 12.237623  172.16.10.250  172.16.10.64   DNS       102 Standard query response 0xcd72 A dns.google A 8.8.4.4 A 8.8.8.8
1430 12.237623  172.16.10.250  172.16.10.64   DNS       146 Standard query response 0x6384 HTTPS dns.google SOA ns1.zdns.google
1431 12.237623  172.16.10.250  172.16.10.64   DNS       117 Standard query response 0xbb15 A chrome.cloudflare-dns.com A 162.159.61.3 A 172.64.41.3
1432 12.237623  172.16.10.250  172.16.10.64   DNS       158 Standard query response 0x392b HTTPS chrome.cloudflare-dns.com HTTPS
1433 12.237623  172.16.10.250  172.16.10.64   DNS       102 Standard query response 0x7b29 A dns.google A 8.8.8.8 A 8.8.4.4
1434 12.237623  172.16.10.250  172.16.10.64   DNS       146 Standard query response 0xa7bb HTTPS dns.google SOA ns1.zdns.google
1435 12.237623  172.16.10.250  172.16.10.64   DNS       117 Standard query response 0xf6b9 A chrome.cloudflare-dns.com A 162.159.61.3 A 172.64.41.3
1436 12.237623  172.16.10.250  172.16.10.64   DNS       158 Standard query response 0xff56 HTTPS chrome.cloudflare-dns.com HTTPS
1437 12.237623  172.16.10.250  172.16.10.64   DNS       146 Standard query response 0xb4af HTTPS dns.google SOA ns1.zdns.google
1438 12.237623  172.16.10.250  172.16.10.64   DNS       102 Standard query response 0x8869 A dns.google A 8.8.8.8 A 8.8.4.4
1439 12.242825  172.16.10.64   162.159.61.3   QUIC     1292 Initial, DCID=e75637e24f40186a, PKN: 1, CRYPTO
1441 12.242281  172.16.10.64   162.159.61.3   QUIC     1292 Initial, DCID=e75637e24f40186a, PKN: 2, PING, PING, PING, PING, PADDING, PING, PADDING, PING, CRYPTO, PADDING, CRYPTO, PING, PA.
1442 12.243341  172.16.10.64   162.159.61.3   QUIC     1292 Initial, DCID=5323da0f401cb82b, PKN: 1, CRYPTO

> Frame 3983: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface \Device\NPF_{B710F0EC-0AF0-4740-8A74-294C0F4B130}, id 0
> Ethernet II, Src: Intel_88:d3:da (b0:dc:ef:88:d3:da), Dst: TpLinkTechno_05:42:d4 (14:cc:20:05:42:d4)
> Internet Protocol Version 4, Src: 172.16.10.64, Dst: 172.16.10.250
> User Datagram Protocol, Src Port: 60472, Dst Port: 53
> Domain Name System (query)

14 cc 20 05 42 d4 b0 dc  ef 88 d
00 44 d6 f8 00 00 80 11  00 00 a
0a fa ec 38 00 35 00 30  6d 9c a
00 00 00 00 00 00 0b 74  2d 72 6
76 32 06 6d 77 65 64 67  65 03 6
00 01
```

No, not all UDP packets in Wireshark will use the same port number. Because,

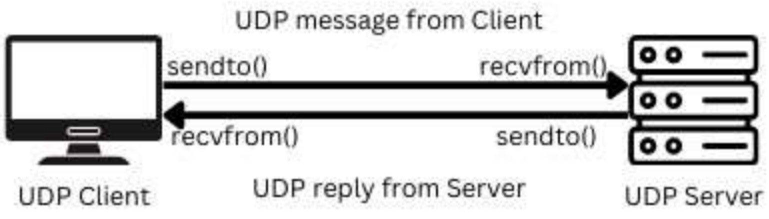
1. Different Apps Use Different Ports: Apps like MSTeams and web browsing use different ports for their UDP communication. For example:
- MSTeams uses various ports for audio, video, and data.
- Web browsing generally uses TCP but might use UDP for things like DNS or streaming, and those will have different ports.
2. Port Numbers Change: UDP packets have source and destination port numbers that help identify the specific app or service. So, packets from MSTeams and web browsing will show different port numbers.

3. Multiple Ports per App: An app might use different ports for different functions. For instance, MSTeams might use different ports for video calls versus chat messages.

So, different port numbers are there for UDP packets from different applications.

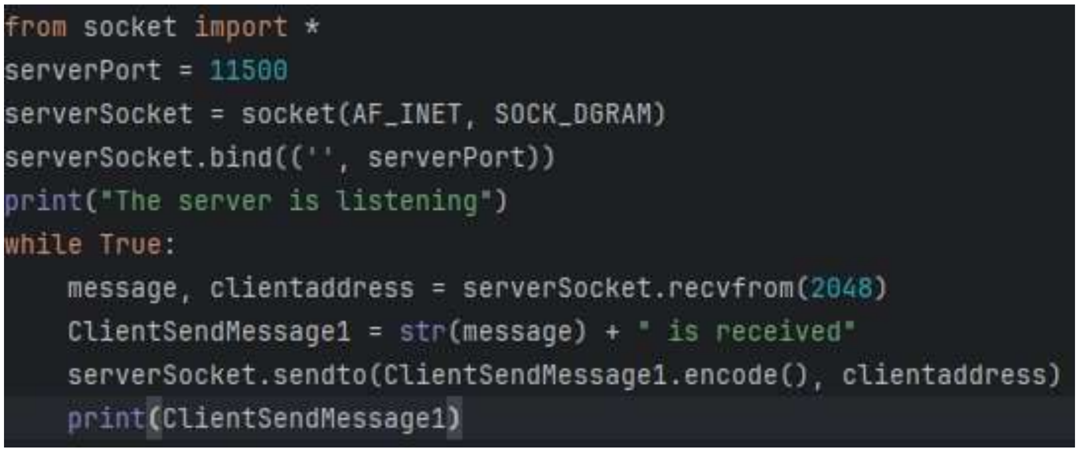
Activity 3

1. First you need to draw a diagram to explain the operation of the client- server program that uses UDP.



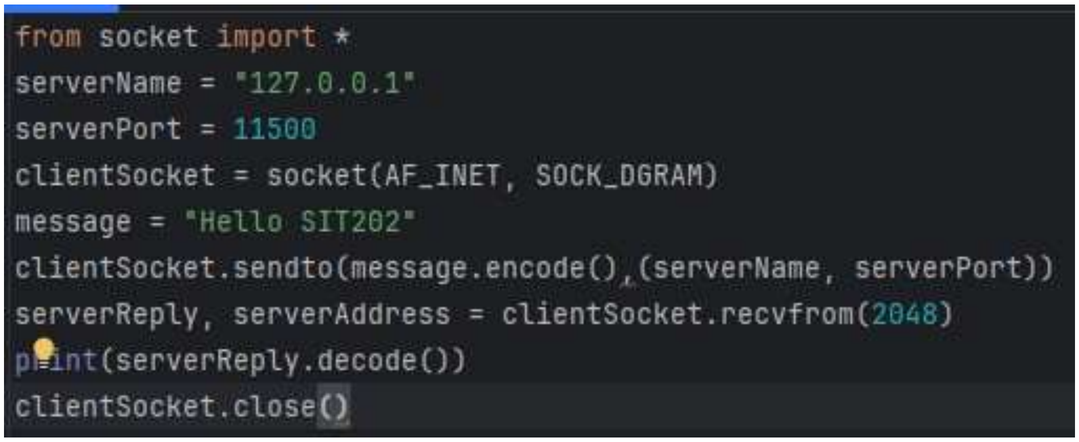
UDP Client – Server diagram

2. Then one of you can develop the client side and the other can develop the server side of the program (you can also develop both end systems and demonstrate the outcome if you chose to do so)



```
from socket import *
serverPort = 11500
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print("The server is listening")
while True:
    message, clientaddress = serverSocket.recvfrom(2048)
    ClientSendMessage1 = str(message) + " is received"
    serverSocket.sendto(ClientSendMessage1.encode(), clientaddress)
    print(ClientSendMessage1)
```

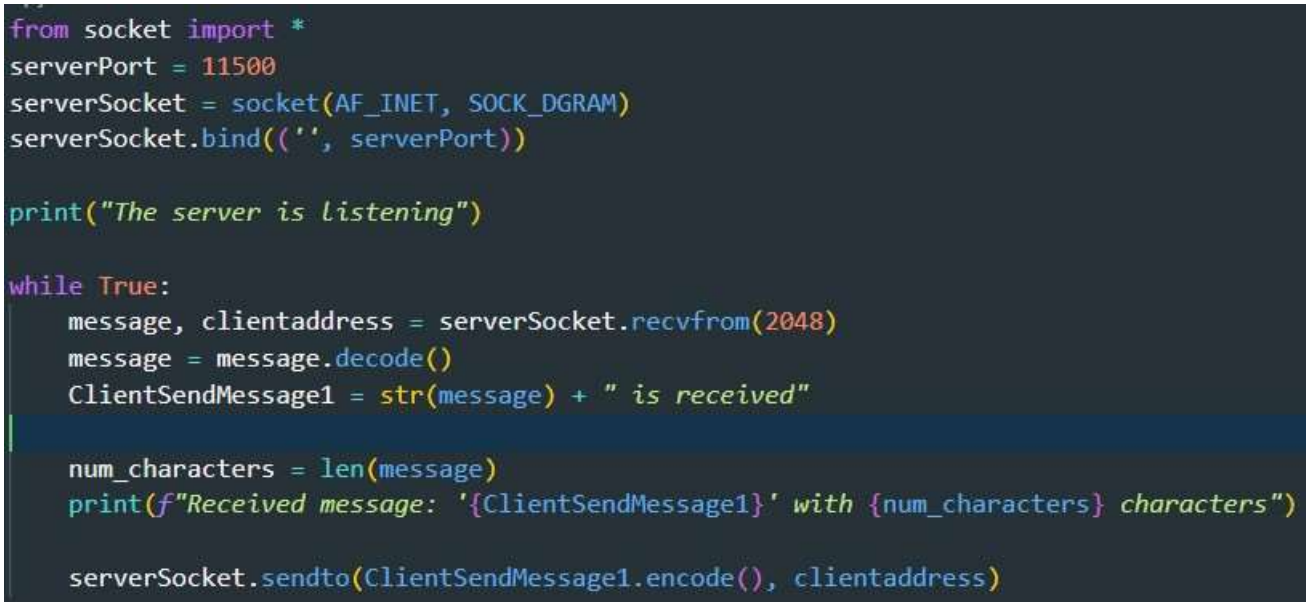
server.py code



```
from socket import *
serverName = "127.0.0.1"
serverPort = 11500
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = "Hello SIT202"
clientSocket.sendto(message.encode(),(serverName, serverPort))
serverReply, serverAddress = clientSocket.recvfrom(2048)
print(serverReply.decode())
clientSocket.close()
```

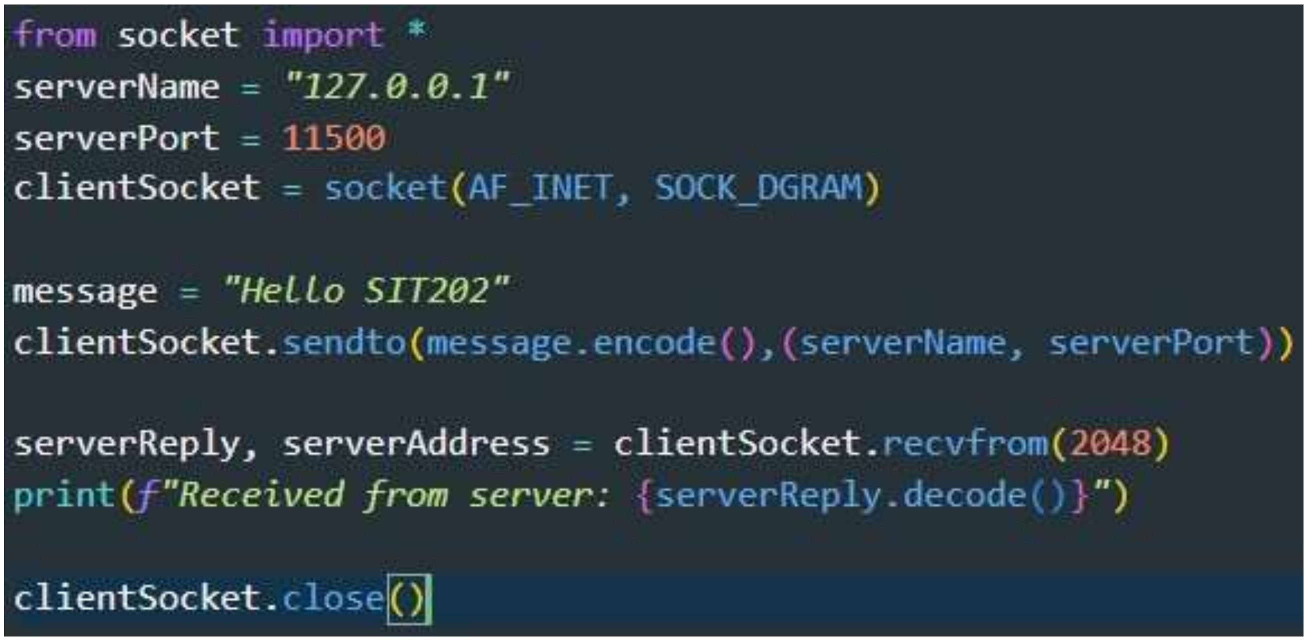
client.py code

3. You need to show the output of the client- server program to demonstrate that your program works as expected. Please make sure to include the server and client codes and output (screenshots) in your Module 3 lesson review.



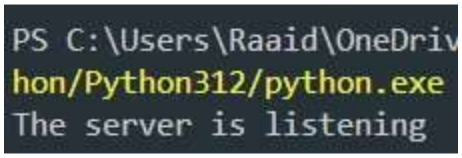
```python
from socket import *
serverPort = 11500
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))

print("The server is listening")

while True:
    message, clientaddress = serverSocket.recvfrom(2048)
    message = message.decode()
    ClientSendMessage1 = str(message) + " is received"

    num_characters = len(message)
    print(f"Received message: '{ClientSendMessage1}' with {num_characters} characters")

    serverSocket.sendto(ClientSendMessage1.encode(), clientaddress)
```

Server.py code



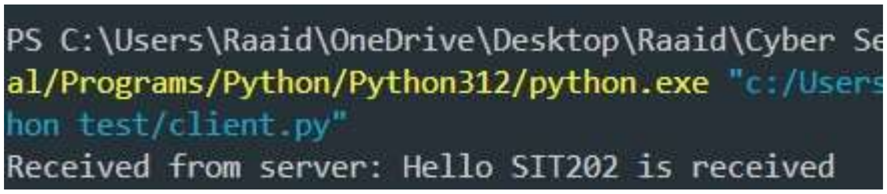
```python
from socket import *
serverName = "127.0.0.1"
serverPort = 11500
clientSocket = socket(AF_INET, SOCK_DGRAM)

message = "Hello SIT202"
clientSocket.sendto(message.encode(),(serverName, serverPort))

serverReply, serverAddress = clientSocket.recvfrom(2048)
print(f"Received from server: {serverReply.decode()}")

clientSocket.close()
```

Client.py code



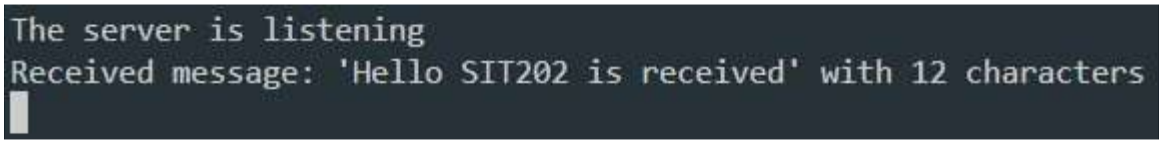
```
PS C:\Users\Raaid\OneDriv
hon/Python312/python.exe
The server is listening
```

Executing server.py and output



```
PS C:\Users\Raaid\OneDrive\Desktop\Raaid\Cyber Se
al/Programs/Python/Python312/python.exe "c:/Users
hon test/client.py"
Received from server: Hello SIT202 is received
```

Executing client.py and output



```
The server is listening
Received message: 'Hello SIT202 is received' with 12 characters
```

Updated output of server.py mentioning client message and character length

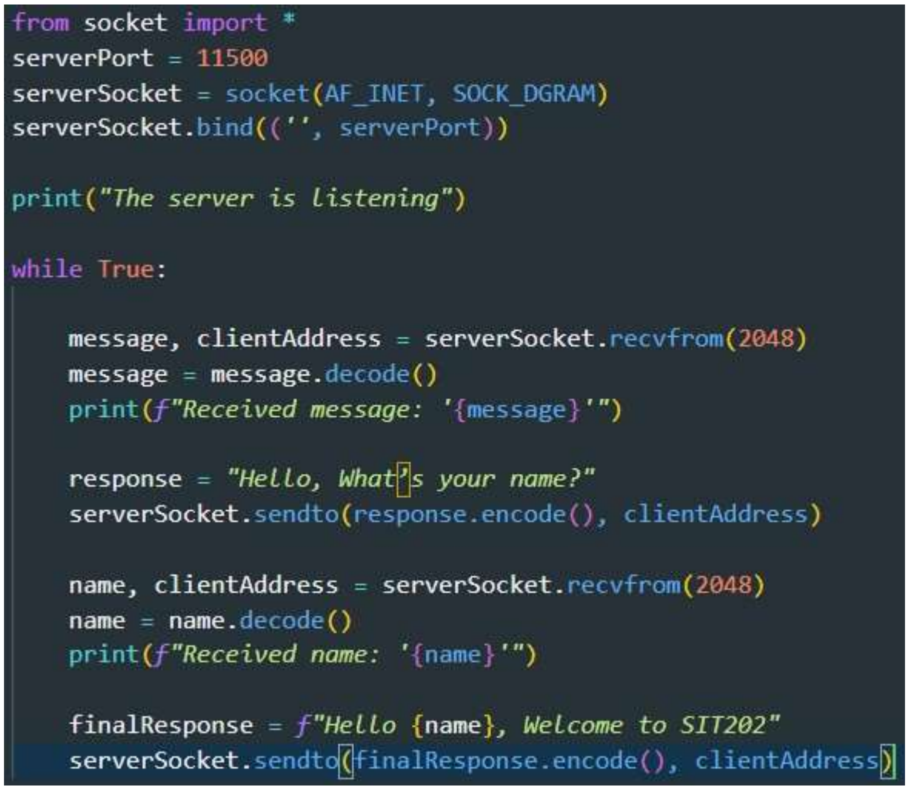
Above and Beyond Tasks

Modified the Python program you wrote to add the following functions:

1. The client sends a message “Hello” to the server
2. Once the server received the full message, sever responds with “Hello, What’s your name?”
3. When the client receives the full message from the server, Client should be able to enter the name via the terminal and send the name to the server.
4. When the server received the full message, server responds with “Hello name, Welcome to SIT202”.

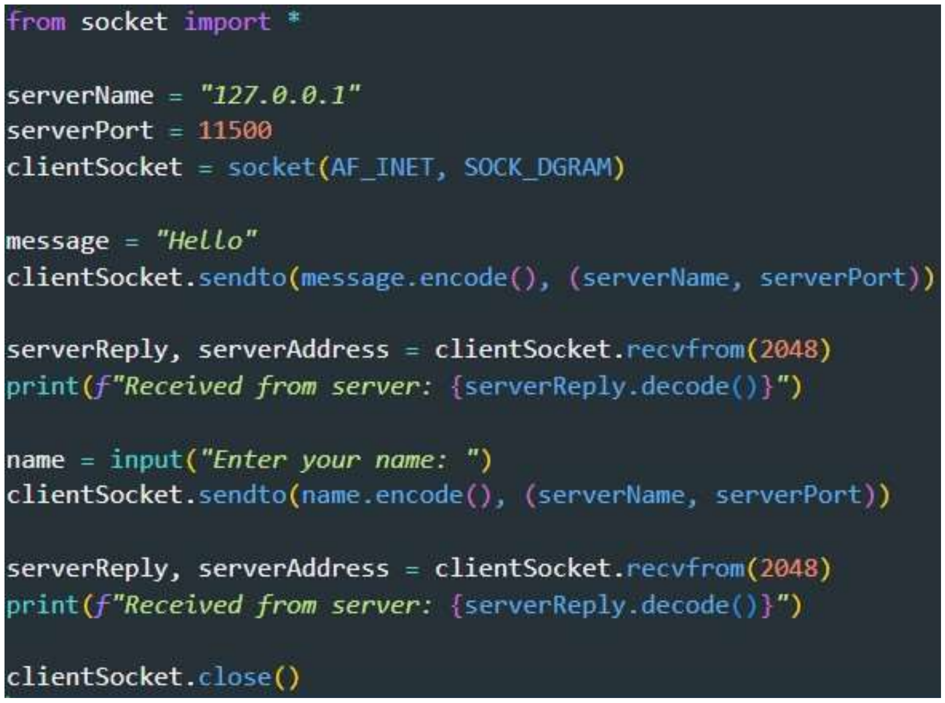
“name” should be the one that is received form the client.

5. All these received messages should be displayed in the relevant terminal. Please make sure to include the server and client codes and output (screenshots) in your task submission.



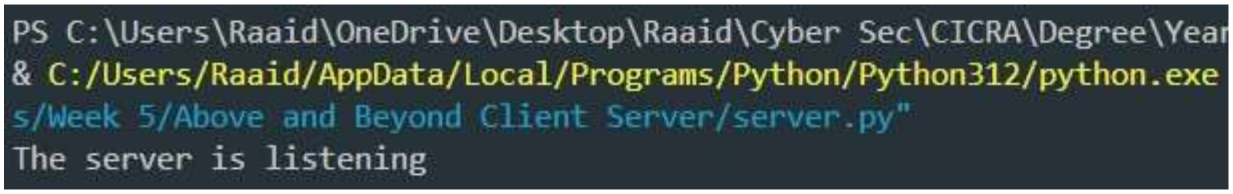
```python
from socket import *
serverPort = 11500
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))

print("The server is listening")

while True:

    message, clientAddress = serverSocket.recvfrom(2048)
    message = message.decode()
    print(f"Received message: '{message}'")

    response = "Hello, What's your name?"
    serverSocket.sendto(response.encode(), clientAddress)

    name, clientAddress = serverSocket.recvfrom(2048)
    name = name.decode()
    print(f"Received name: '{name}'")

    finalResponse = f"Hello {name}, Welcome to SIT202"
    serverSocket.sendto(finalResponse.encode(), clientAddress)
```

Server.py code



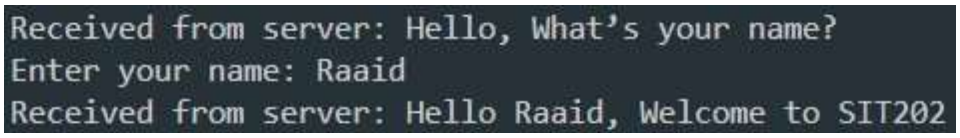
```
from socket import *

serverName = "127.0.0.1"
serverPort = 11500
clientSocket = socket(AF_INET, SOCK_DGRAM)

message = "Hello"
clientSocket.sendto(message.encode(), (serverName, serverPort))

serverReply, serverAddress = clientSocket.recvfrom(2048)
print(f"Received from server: {serverReply.decode()}")

name = input("Enter your name: ")
clientSocket.sendto(name.encode(), (serverName, serverPort))

serverReply, serverAddress = clientSocket.recvfrom(2048)
print(f"Received from server: {serverReply.decode()}")

clientSocket.close()
```

Client.py code



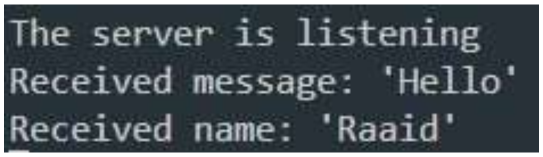
```
PS C:\Users\Raaid\OneDrive\Desktop\Raaid\Cyber Sec\CICRA\Degree\Year
& C:/Users/Raaid/AppData/Local/Programs/Python/Python312/python.exe
s/Week 5/Above and Beyond Client Server/server.py"
The server is listening
```

Executing server.py and output



```
Received from server: Hello, What’s your name?
Enter your name: Raaid
Received from server: Hello Raaid, Welcome to SIT202
```

Executing client.py and output



```
The server is listening
Received message: 'Hello'
Received name: 'Raaid'
```

Updated output of server.py mentioning client message and name.