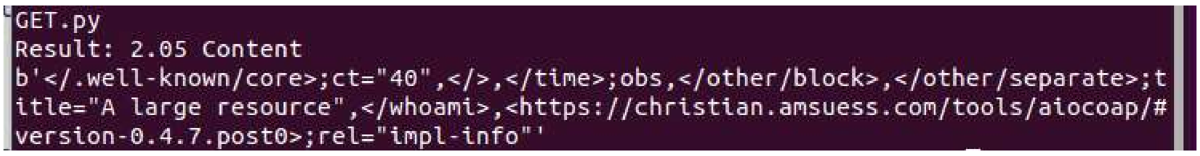
Victor MAYAUD

APPIOT : Lab 1
CoAP

1/ Discovering Server Resources :

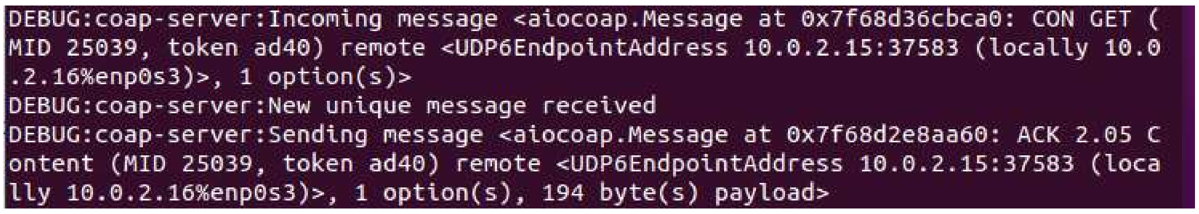
In clientGet.py I changed the Uri by this ‘coap://10.0.2.16/.well-known/core’ and I obtained all the available resources in the server:

client:



```
GET.py
Result: 2.05 Content
b'</.well-known/core>;ct="40",</>,</time>;obs,</other/block>,</other/separate>;t
itle="A large resource",</whoami>,<https://christian.amsuess.com/tools/aiocoap/#
version-0.4.7.post0>;rel="impl-info"'
```

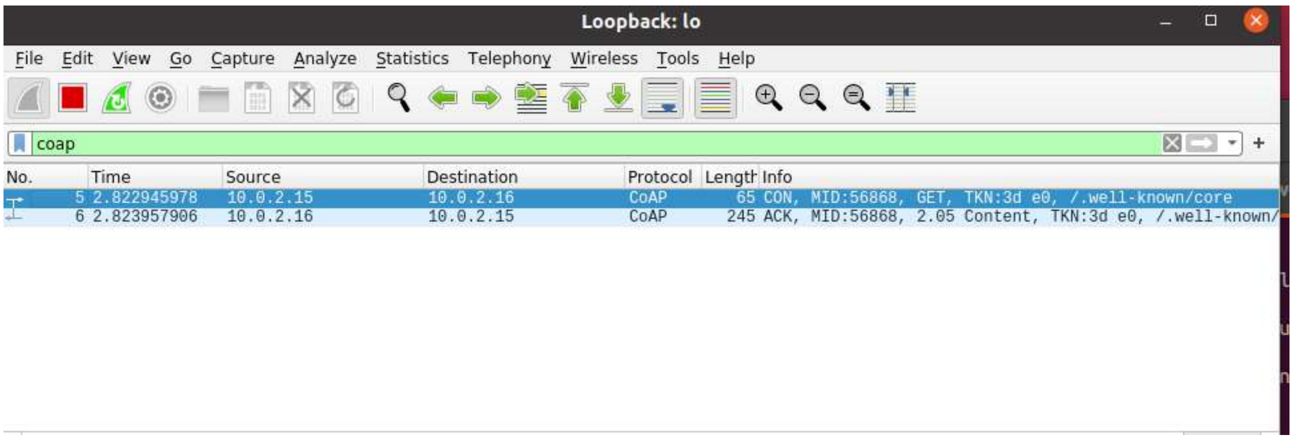
server:



```
DEBUG:coap-server:Incoming message <aiocoap.Message at 0x7f68d36cbca0: CON GET (
MID 25039, token ad40) remote <UDP6EndpointAddress 10.0.2.15:37583 (locally 10.0
.2.16%enp0s3)>, 1 option(s)>
DEBUG:coap-server:New unique message received
DEBUG:coap-server:Sending message <aiocoap.Message at 0x7f68d2e8aa60: ACK 2.05 C
ontent (MID 25039, token ad40) remote <UDP6EndpointAddress 10.0.2.15:37583 (loca
lly 10.0.2.16%enp0s3)>, 1 option(s), 194 byte(s) payload>
```

2/ Analysing Traffic with Wireshark.

After capturing the traffic I obtained this:


Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

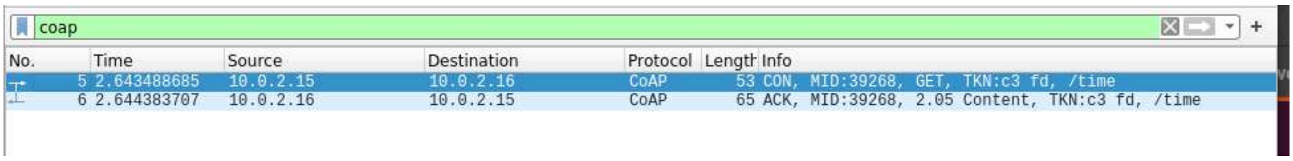
coap

```
No.  Time         Source     Destination Protocol Length Info
   5 2.822945978  10.0.2.15  10.0.2.16   CoAP        65 CON, MID:56868, GET, TKN:3d e0, /.well-known/core
   6 2.823957906  10.0.2.16  10.0.2.15   CoAP       245 ACK, MID:56868, 2.05 Content, TKN:3d e0, /.well-known/
```

In the info section we can see CON, so the messages are confirmable.

For the other resources we have this:

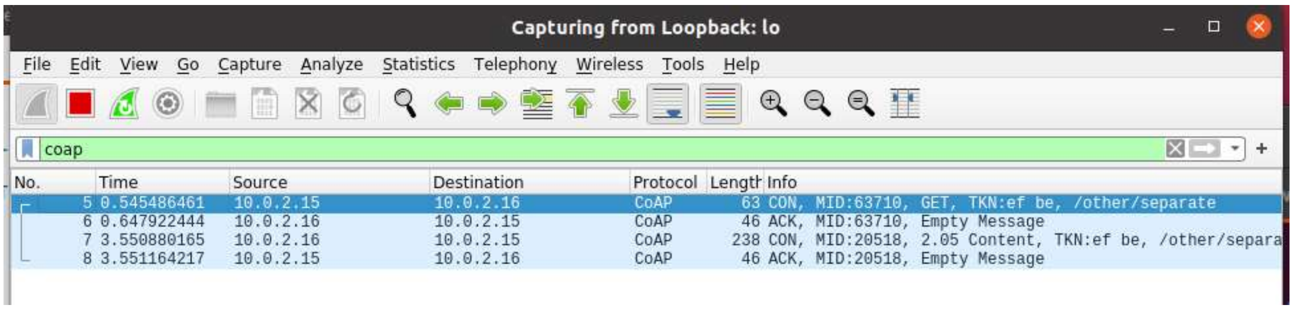
/time:



```
coap
No.  Time         Source     Destination  Protocol Length Info
   5 2.643488685  10.0.2.15  10.0.2.16    CoAP      53 CON, MID:39268, GET, TKN:c3 fd, /time
   6 2.644383707  10.0.2.16  10.0.2.15    CoAP      65 ACK, MID:39268, 2.05 Content, TKN:c3 fd, /time
```

For this request, we have a new token designated as 'c3.' Additionally, the message ID and the length of the response have changed. The token remains consistent for each identical request, but it alters when the request changes. The message ID is updated with each new request, and the response length varies depending on the request specifics.

Here is another example with the request other/separate:



Capturing from Loopback: lo

```
coap
No.  Time         Source     Destination  Protocol Length Info
   5 0.545486461  10.0.2.15  10.0.2.16    CoAP      63 CON, MID:63710, GET, TKN:ef be, /other/separate
   6 0.647922444  10.0.2.16  10.0.2.15    CoAP      46 ACK, MID:63710, Empty Message
   7 3.550880165  10.0.2.16  10.0.2.15    CoAP     238 CON, MID:20518, 2.05 Content, TKN:ef be, /other/separa
   8 3.551164217  10.0.2.15  10.0.2.16    CoAP      46 ACK, MID:20518, Empty Message
```

For this case we have this steps:

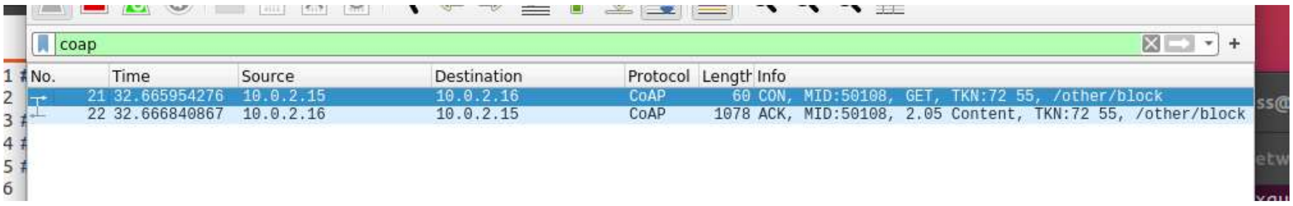
Packet #5: This is a confirmable CoAP GET request sent by the client. It requires an acknowledgment from the server to confirm that the request has been received.

Packet #6: The server sends an acknowledgment (ACK) back to the client. However, this ACK is an empty message, which means it's acknowledging receipt of the request without carrying the actual response data yet.

Packet #7: Subsequently, the server sends the response with the requested content in a separate confirmable message. This pattern is used when the server needs more time to prepare the response or wishes to separate the acknowledgment of the request from the response.

Packet #8: Finally, the client acknowledges the receipt of the response content with another ACK.

for other/block:



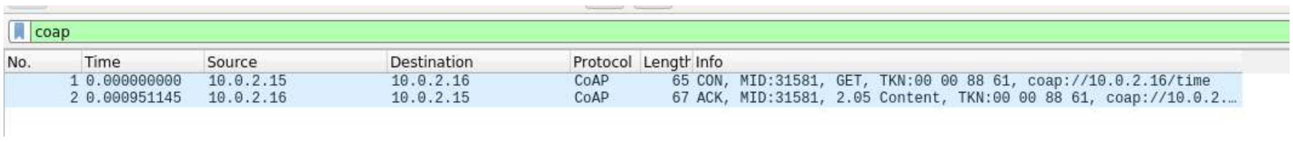
```
coap
No.  Time          Source     Destination  Protocol Length Info
  21 32.665954276  10.0.2.15  10.0.2.16    CoAP       60 CON, MID:50108, GET, TKN:72 55, /other/block
  22 32.666840867  10.0.2.16  10.0.2.15    CoAP     1078 ACK, MID:50108, 2.05 Content, TKN:72 55, /other/block
```

the token changed, the same for the length, the mid and the uri.

For the other resources it is the same the token, mid, uri and the number of packets changed.

3/ GET Requests with and without Confirmation.

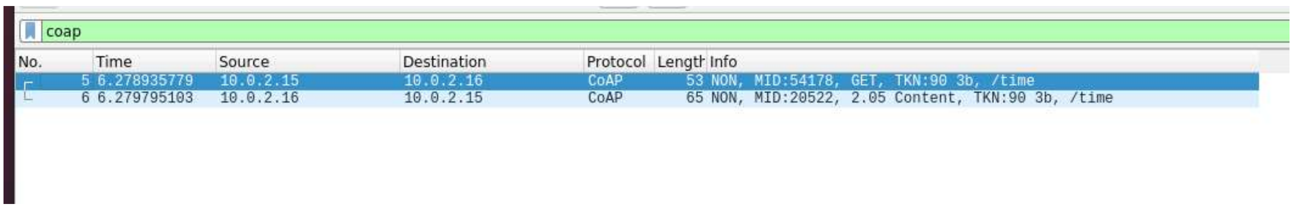
Result of the command aiocoap-client coap://10.0.2.16/time :



```
coap
No.   Time         Source      Destination  Protocol Length Info
    1 0.000000000  10.0.2.15   10.0.2.16    CoAP     65 CON, MID:31581, GET, TKN:00 00 88 61, coap://10.0.2.16/time
    2 0.000951145  10.0.2.16   10.0.2.15    CoAP     67 ACK, MID:31581, 2.05 Content, TKN:00 00 88 61, coap://10.0.2.…
```

The message is confirmable. The client send a request and the server send back the data with the ack.

aiocoap-client --non coap://10.0.2.16/time

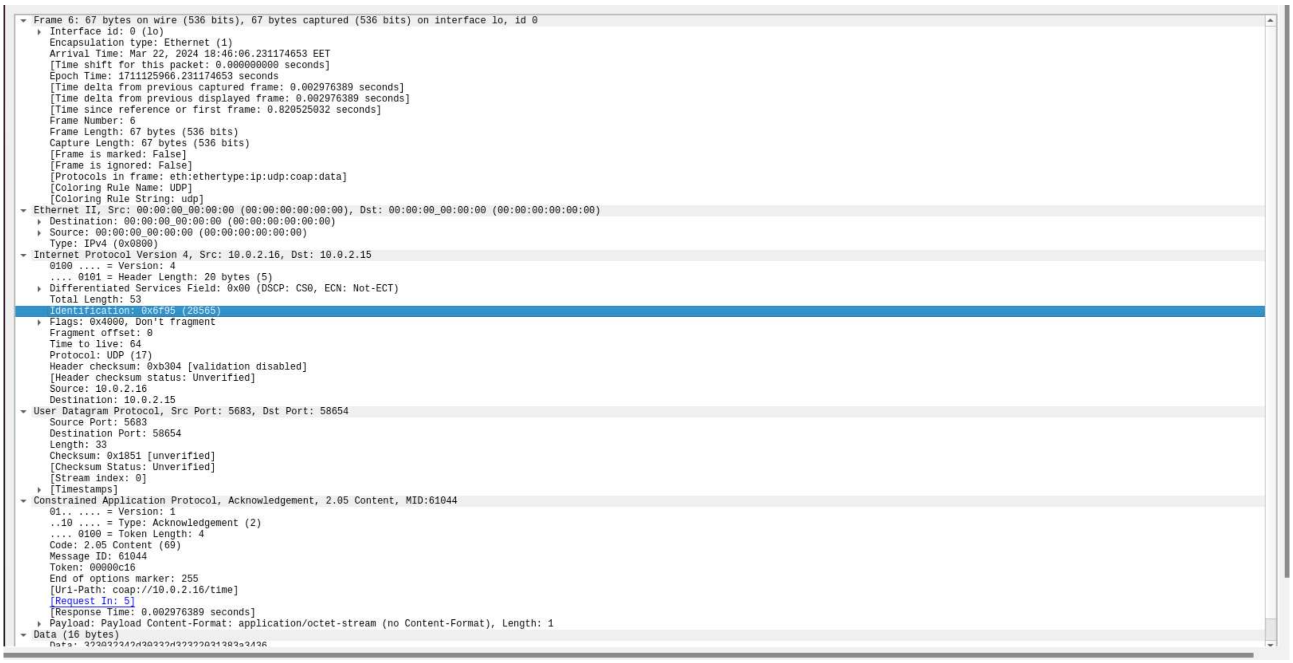


```
coap
No.   Time         Source      Destination  Protocol Length Info
    5 6.278935779  10.0.2.15   10.0.2.16    CoAP     53 NON, MID:54178, GET, TKN:90 3b, /time
    6 6.279795103  10.0.2.16   10.0.2.15    CoAP     65 NON, MID:20522, 2.05 Content, TKN:90 3b, /time
```

The main difference between the two captures is the message type. The first capture shows a confirmable (CON) request that requires an acknowledgment (ACK) from the server, evidenced by the two-packet exchange. The second capture illustrates a non-confirmable (NON) request which doesn't require an ACK, resulting in just two messages: the request and the response.

4/ Message format analysis.

For the request **/time** we have:



```
▾ Frame 6: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface lo, id 0
  ▸ Interface id: 0 (lo)
    Encapsulation type: Ethernet (1)
    Arrival Time: Mar 22, 2024 18:46:06.231174653 EET
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1711125966.231174653 seconds
    [Time delta from previous captured frame: 0.002976389 seconds]
    [Time delta from previous displayed frame: 0.002976389 seconds]
    [Time since reference or first frame: 0.820525032 seconds]
    Frame Number: 6
    Frame Length: 67 bytes (536 bits)
    Capture Length: 67 bytes (536 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:udp:coap:data]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
▾ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
  ▸ Destination: 00:00:00_00:00:00 (00:00:00:00:00:00)
  ▸ Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 10.0.2.16, Dst: 10.0.2.15
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 53
    Identification: 0x6f95 (28565)
  ▸ Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0xb304 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.16
    Destination: 10.0.2.15
▾ User Datagram Protocol, Src Port: 5683, Dst Port: 58654
    Source Port: 5683
    Destination Port: 58654
    Length: 33
    Checksum: 0x1851 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  ▸ [Timestamps]
▾ Constrained Application Protocol, Acknowledgement, 2.05 Content, MID:61044
    01.. .... = Version: 1
    ..10 .... = Type: Acknowledgement (2)
    .... 0100 = Token Length: 4
    Code: 2.05 Content (69)
    Message ID: 61044
    Token: 00000c16
    End of options marker: 255
    [Uri-Path: coap://10.0.2.16/time]
    [Request In: 5]
    [Response Time: 0.002976389 seconds]
  ▸ Payload: Payload Content-Format: application/octet-stream (no Content-Format), Length: 1
▾ Data (16 bytes)
    Data: 323032342d30332d32322031383a3436
```

Frame: Details about the data captured by Wireshark, including the length and bytes on the wire.

Ethernet II: Shows the source and destination MAC addresses, indicating the hardware-level communication.

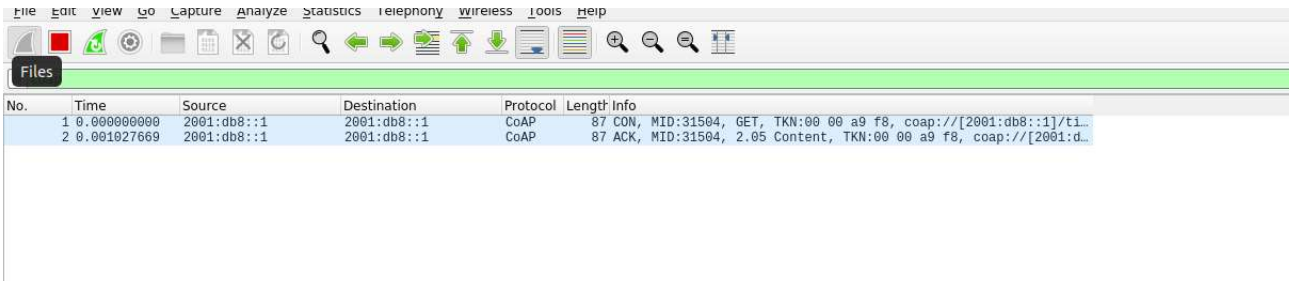
Internet Protocol Version 4 (IPv4): Includes source and destination IP addresses, differentiating services field, identification, flags, fragment offset, time to live (TTL), and protocol (indicating CoAP).

User Datagram Protocol (UDP): Displays the source and destination ports, important for identifying the CoAP messages.

Constrained Application Protocol (CoAP): Displays the protocol-specific information such as the message type (Acknowledgment), the message ID (MID), the request method (GET), the response code (2.05 Content), and the Token (TKN), which is used to match responses with requests.

5/ IPv6 configuration.

With the request "aiocoap-client coap://[2001:db8::1]/time" we have:

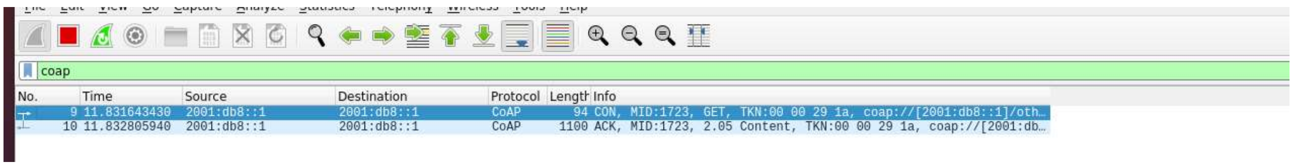

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Files

```
No.   Time         Source        Destination   Protocol Length Info
    1 0.000000000  2001:db8::1   2001:db8::1   CoAP        87 CON, MID:31504, GET, TKN:00 00 a9 f8, coap://[2001:db8::1]/ti…
    2 0.001027669  2001:db8::1   2001:db8::1   CoAP        87 ACK, MID:31504, 2.05 Content, TKN:00 00 a9 f8, coap://[2001:d…
```

The source and destination addresses changed according to the IPv6 and the length is bigger than the request with IPv4.

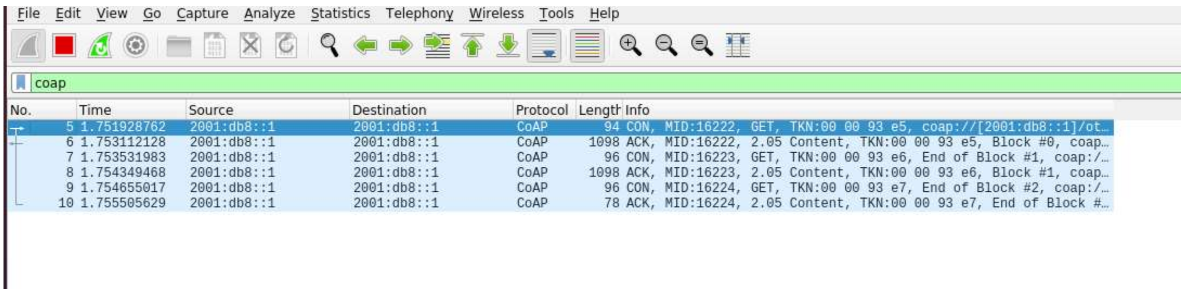
6/ Large block transfer.

For the request **other/block/** with payload equals to 1024 I have this:



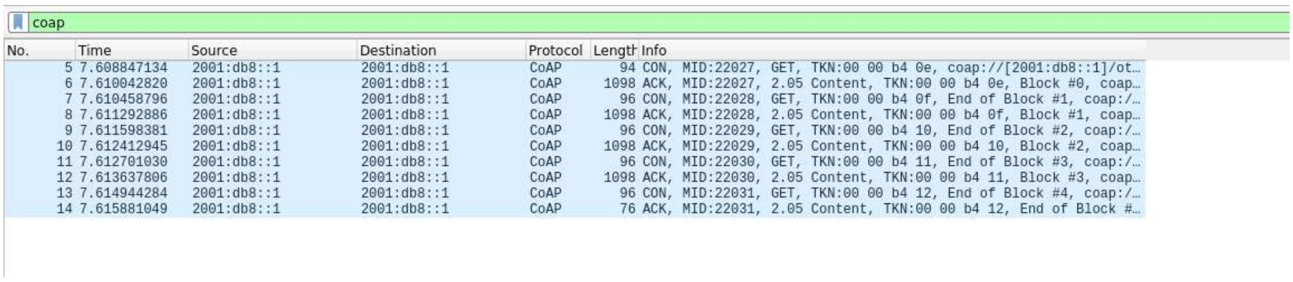
```
No.   Time          Source        Destination   Protocol Length Info
    9 11.831643430  2001:db8::1   2001:db8::1   CoAP        94 CON, MID:1723, GET, TKN:00 00 29 1a, coap://[2001:db8::1]/oth…
   10 11.832805940  2001:db8::1   2001:db8::1   CoAP      1100 ACK, MID:1723, 2.05 Content, TKN:00 00 29 1a, coap://[2001:db…
```

For the request **other/block/** with payload equals to 2048 I have this:


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
coap

```
No.   Time         Source        Destination   Protocol Length Info
    5 1.751928762  2001:db8::1   2001:db8::1   CoAP        94 CON, MID:16222, GET, TKN:00 00 93 e5, coap://[2001:db8::1]/ot…
    6 1.753112128  2001:db8::1   2001:db8::1   CoAP      1098 ACK, MID:16222, 2.05 Content, TKN:00 00 93 e5, Block #0, coap…
    7 1.753531983  2001:db8::1   2001:db8::1   CoAP        96 CON, MID:16223, GET, TKN:00 00 93 e6, End of Block #1, coap:/…
    8 1.754349468  2001:db8::1   2001:db8::1   CoAP      1098 ACK, MID:16223, 2.05 Content, TKN:00 00 93 e6, Block #1, coap…
    9 1.754655017  2001:db8::1   2001:db8::1   CoAP        96 CON, MID:16224, GET, TKN:00 00 93 e7, End of Block #2, coap:/…
   10 1.755505629  2001:db8::1   2001:db8::1   CoAP        78 ACK, MID:16224, 2.05 Content, TKN:00 00 93 e7, End of Block #…
```

For the request **other/block/** with payload equals to 4096 I have this:



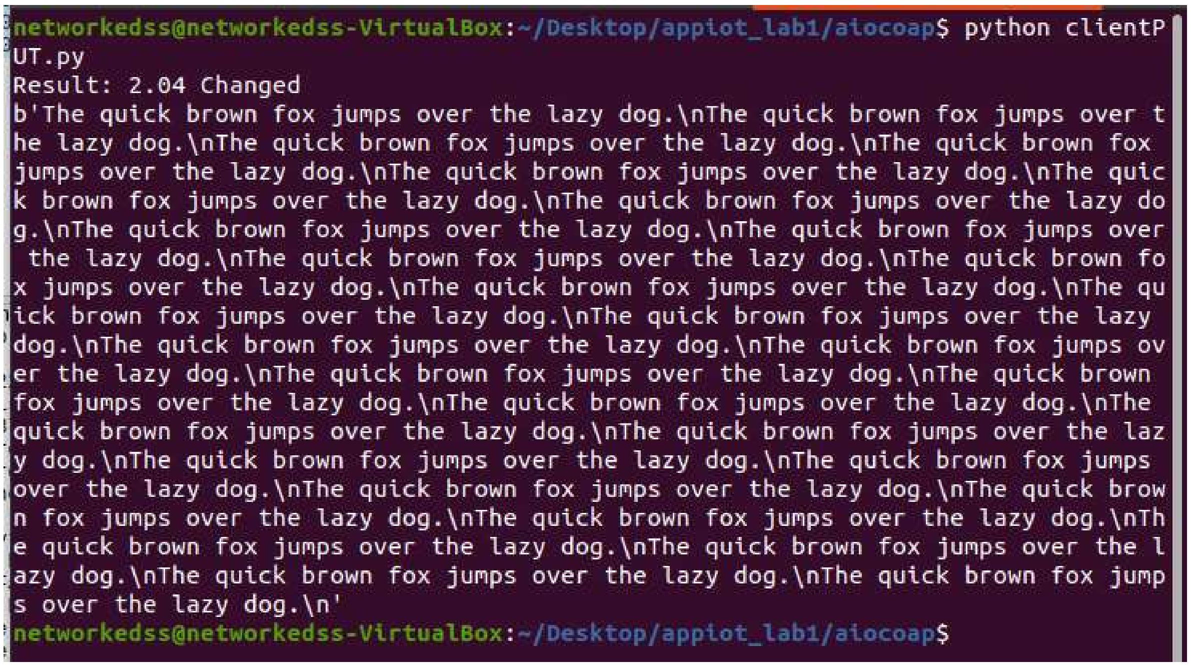
coap

No.	Time	Source	Destination	Protocol	Length	Info
5	7.608847134	2001:db8::1	2001:db8::1	CoAP	94	CON, MID:22027, GET, TKN:00 00 b4 0e, coap://[2001:db8::1]/ot…
6	7.610042820	2001:db8::1	2001:db8::1	CoAP	1098	ACK, MID:22027, 2.05 Content, TKN:00 00 b4 0e, Block #0, coap…
7	7.610458796	2001:db8::1	2001:db8::1	CoAP	96	CON, MID:22028, GET, TKN:00 00 b4 0f, End of Block #1, coap:/…
8	7.611292886	2001:db8::1	2001:db8::1	CoAP	1098	ACK, MID:22028, 2.05 Content, TKN:00 00 b4 0f, Block #1, coap…
9	7.611598381	2001:db8::1	2001:db8::1	CoAP	96	CON, MID:22029, GET, TKN:00 00 b4 10, End of Block #2, coap:/…
10	7.612412945	2001:db8::1	2001:db8::1	CoAP	1098	ACK, MID:22029, 2.05 Content, TKN:00 00 b4 10, Block #2, coap…
11	7.612701030	2001:db8::1	2001:db8::1	CoAP	96	CON, MID:22030, GET, TKN:00 00 b4 11, End of Block #3, coap:/…
12	7.613637806	2001:db8::1	2001:db8::1	CoAP	1098	ACK, MID:22030, 2.05 Content, TKN:00 00 b4 11, Block #3, coap…
13	7.614944284	2001:db8::1	2001:db8::1	CoAP	96	CON, MID:22031, GET, TKN:00 00 b4 12, End of Block #4, coap:/…
14	7.615881049	2001:db8::1	2001:db8::1	CoAP	76	ACK, MID:22031, 2.05 Content, TKN:00 00 b4 12, End of Block #…

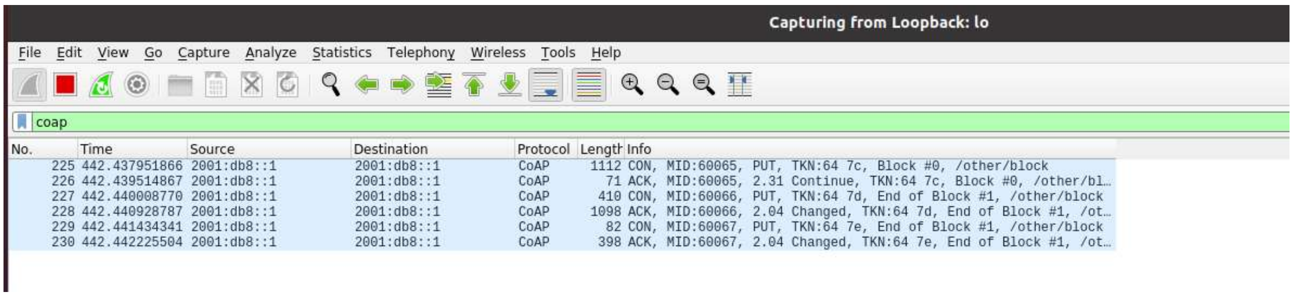
With increasing payload sizes, CoAP uses blockwise transfer to efficiently manage data transmission. For the 1024-byte payload, we see a simple two-message exchange, showing that it fits within a single CoAP message without the need for segmentation. However, with the 2048 and 4096-byte payloads, multiple CoAP messages are involved, indicating that the payload is divided into blocks. This is evident from the "Block" option in the CoAP header, which signifies that the message is part of a sequence of block transfers. The transfer of larger payloads results in more CoAP messages, as the payload must be split into sizes that conform to the network's MTU limits.

7/ PUT Request and Blockwise Handling.

We the script clientPUT.py (payload=1024) I have got this:



```
networkedss@networkedss-VirtualBox:~/Desktop/appiot_lab1/aiocoap$ python clientP
UT.py
Result: 2.04 Changed
b'The quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over t
he lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox 
jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quic
k brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy do
g.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over
 the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fo
x jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe qu
ick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy 
dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps ov
er the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown 
fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe 
quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the laz
y dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps 
over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brow
n fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nTh
e quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the l
azy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jump
s over the lazy dog.\n'
networkedss@networkedss-VirtualBox:~/Desktop/appiot_lab1/aiocoap$
```



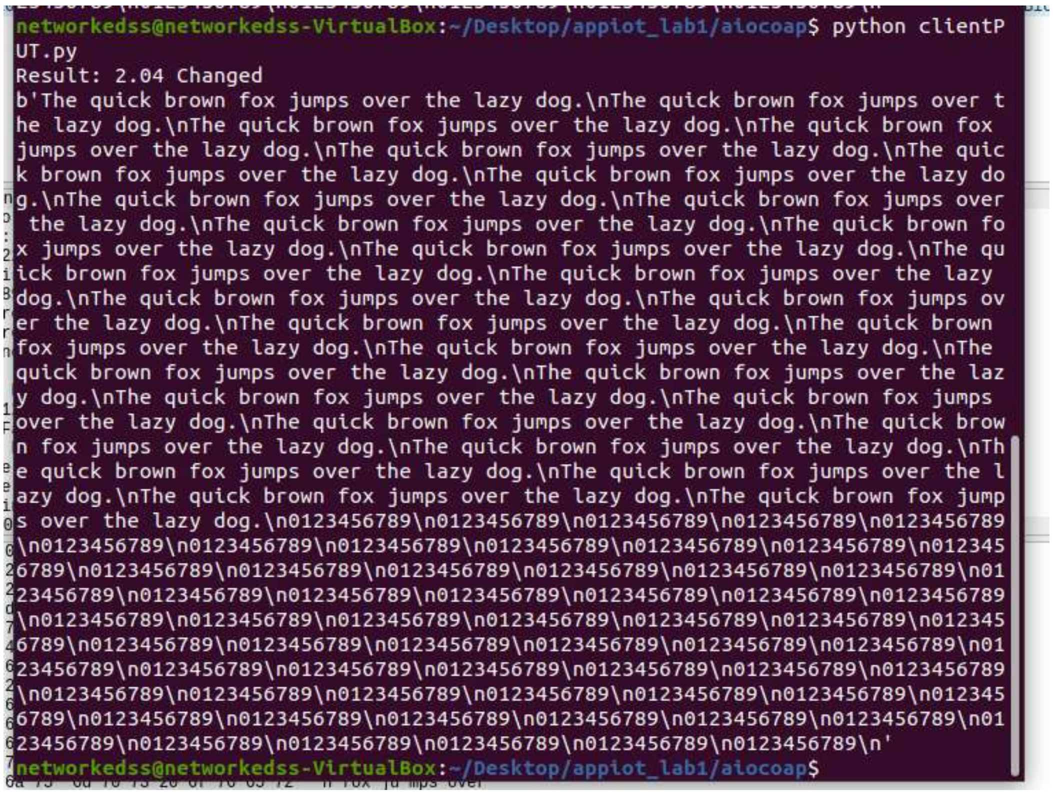
Capturing from Loopback: lo

```
coap
No.   Time          Source        Destination   Protocol Length Info
  225 442.437951866 2001:db8::1   2001:db8::1   CoAP     1112 CON, MID:60065, PUT, TKN:64 7c, Block #0, /other/block
  226 442.439514867 2001:db8::1   2001:db8::1   CoAP       71 ACK, MID:60065, 2.31 Continue, TKN:64 7c, Block #0, /other/bl…
  227 442.440008770 2001:db8::1   2001:db8::1   CoAP      410 CON, MID:60066, PUT, TKN:64 7d, End of Block #1, /other/block
  228 442.440928787 2001:db8::1   2001:db8::1   CoAP     1098 ACK, MID:60066, 2.04 Changed, TKN:64 7d, End of Block #1, /ot…
  229 442.441434341 2001:db8::1   2001:db8::1   CoAP       82 CON, MID:60067, PUT, TKN:64 7e, End of Block #1, /other/block
  230 442.442225504 2001:db8::1   2001:db8::1   CoAP      398 ACK, MID:60067, 2.04 Changed, TKN:64 7e, End of Block #1, /ot…
```

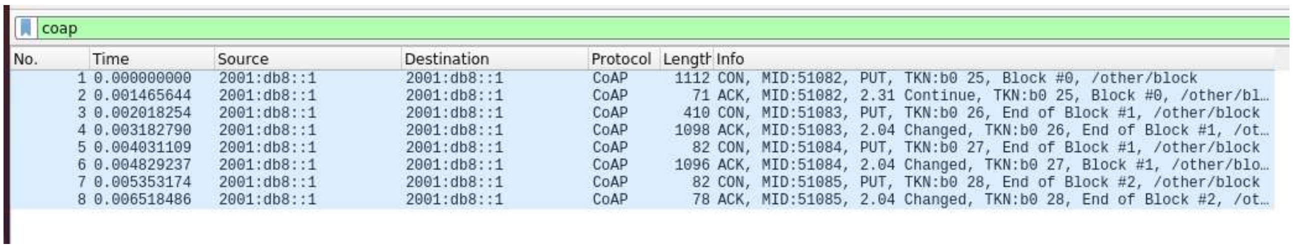
Only 6 packets is travelling.

For different payload we have:

-2048:

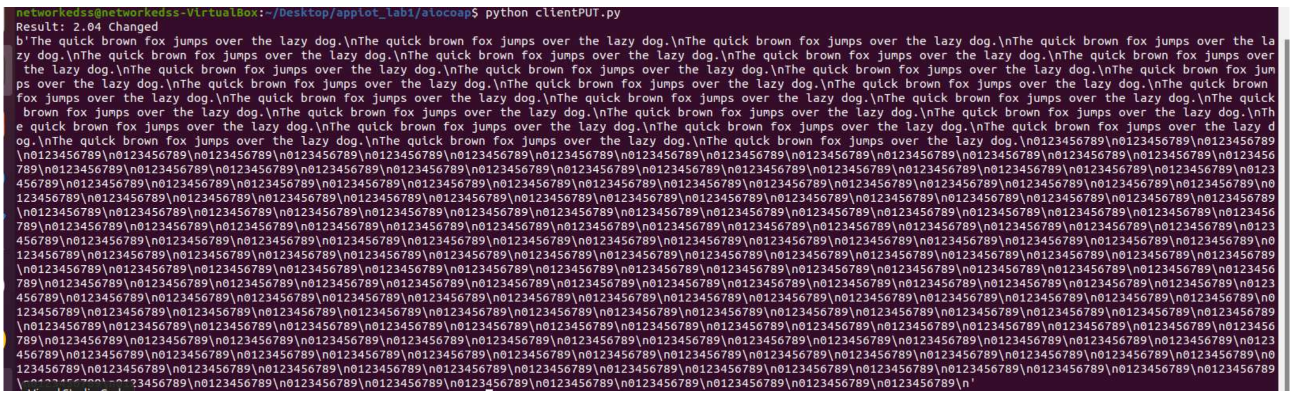


```
networkedss@networkedss-VirtualBox:~/Desktop/appiot_lab1/aiocoap$ python clientP
UT.py
Result: 2.04 Changed
b'The quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over t
he lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox
jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quic
k brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy do
g.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over
 the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fo
x jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe qu
ick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy
dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps ov
er the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown
fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe
quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the laz
y dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps
over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brow
n fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nTh
e quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the l
azy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jump
s over the lazy dog.\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789
\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n012345
6789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n01
23456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789
\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n012345
6789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n01
23456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789
\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n012345
6789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n01
23456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n'
networkedss@networkedss-VirtualBox:~/Desktop/appiot_lab1/aiocoap$
```

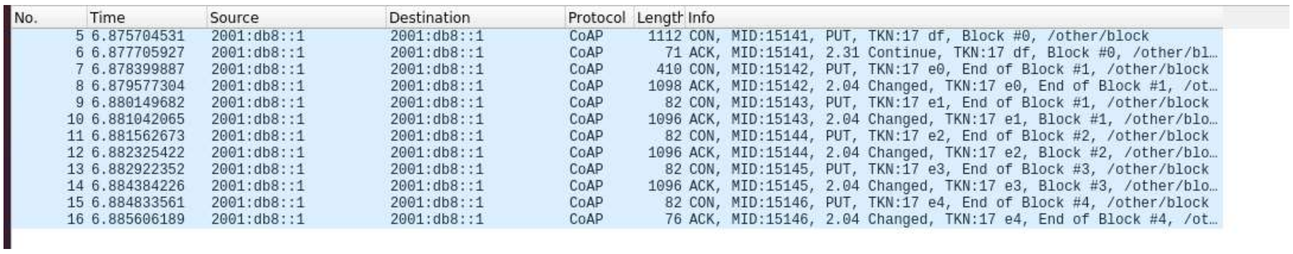


```
coap
No.   Time          Source        Destination   Protocol Length Info
    1 0.000000000   2001:db8::1   2001:db8::1   CoAP     1112 CON, MID:51082, PUT, TKN:b0 25, Block #0, /other/block
    2 0.001465644   2001:db8::1   2001:db8::1   CoAP       71 ACK, MID:51082, 2.31 Continue, TKN:b0 25, Block #0, /other/bl…
    3 0.002018254   2001:db8::1   2001:db8::1   CoAP      410 CON, MID:51083, PUT, TKN:b0 26, End of Block #1, /other/block
    4 0.003182790   2001:db8::1   2001:db8::1   CoAP     1098 ACK, MID:51083, 2.04 Changed, TKN:b0 26, End of Block #1, /ot…
    5 0.004031109   2001:db8::1   2001:db8::1   CoAP       82 CON, MID:51084, PUT, TKN:b0 27, End of Block #1, /other/block
    6 0.004829237   2001:db8::1   2001:db8::1   CoAP     1096 ACK, MID:51084, 2.04 Changed, TKN:b0 27, Block #1, /other/blo…
    7 0.005353174   2001:db8::1   2001:db8::1   CoAP       82 CON, MID:51085, PUT, TKN:b0 28, End of Block #2, /other/block
    8 0.006518486   2001:db8::1   2001:db8::1   CoAP       78 ACK, MID:51085, 2.04 Changed, TKN:b0 28, End of Block #2, /ot…
```

-4096:



```
networkedss@networkedss-VirtualBox:~/Desktop/appiot_lab1/aiocoap$ python clientPUT.py
Result: 2.04 Changed
b'The quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the la
zy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over
 the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jum
ps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown
fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick
 brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nTh
e quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy d
og.\nThe quick brown fox jumps over the lazy dog.\nThe quick brown fox jumps over the lazy dog.\n0123456789\n0123456789\n0123456789
\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456
789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123
456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0
123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789
\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456
789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123
456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0
123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789
\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456
789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123
456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0
123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789
\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456
789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123
456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0
123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789
\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n0123456789\n'
```

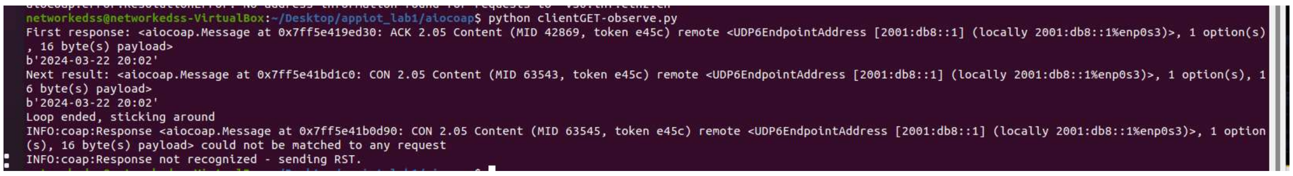


No.	Time	Source	Destination	Protocol	Length	Info
5	6.875704531	2001:db8::1	2001:db8::1	CoAP	1112	CON, MID:15141, PUT, TKN:17 df, Block #0, /other/block
6	6.877705927	2001:db8::1	2001:db8::1	CoAP	71	ACK, MID:15141, 2.31 Continue, TKN:17 df, Block #0, /other/bl…
7	6.878399887	2001:db8::1	2001:db8::1	CoAP	410	CON, MID:15142, PUT, TKN:17 e0, End of Block #1, /other/block
8	6.879577304	2001:db8::1	2001:db8::1	CoAP	1098	ACK, MID:15142, 2.04 Changed, TKN:17 e0, End of Block #1, /ot…
9	6.880149682	2001:db8::1	2001:db8::1	CoAP	82	CON, MID:15143, PUT, TKN:17 e1, End of Block #1, /other/block
10	6.881042065	2001:db8::1	2001:db8::1	CoAP	1096	ACK, MID:15143, 2.04 Changed, TKN:17 e1, Block #1, /other/blo…
11	6.881562673	2001:db8::1	2001:db8::1	CoAP	82	CON, MID:15144, PUT, TKN:17 e2, End of Block #2, /other/block
12	6.882325422	2001:db8::1	2001:db8::1	CoAP	1096	ACK, MID:15144, 2.04 Changed, TKN:17 e2, Block #2, /other/blo…
13	6.882922352	2001:db8::1	2001:db8::1	CoAP	82	CON, MID:15145, PUT, TKN:17 e3, End of Block #3, /other/block
14	6.884384226	2001:db8::1	2001:db8::1	CoAP	1096	ACK, MID:15145, 2.04 Changed, TKN:17 e3, Block #3, /other/blo…
15	6.884833561	2001:db8::1	2001:db8::1	CoAP	82	CON, MID:15146, PUT, TKN:17 e4, End of Block #4, /other/block
16	6.885606189	2001:db8::1	2001:db8::1	CoAP	76	ACK, MID:15146, 2.04 Changed, TKN:17 e4, End of Block #4, /ot…

The Wireshark captures showed that as the payload size increases, CoAP automatically employs blockwise transfer to handle the data. This is necessary because CoAP messages must stay within the size limits of the underlying transport, which for UDP is typically 1280 bytes for the path MTU (Maximum transmission unit). The larger payloads were segmented into blocks, each transferred in separate CoAP messages, as observed in the increasing number of packets captured for the larger sizes. This mechanism ensures reliable and efficient data transmission for constrained environments where large messages could lead to network congestion or loss.

<u>8/ CoAP Observer Functionality.</u>

In putting my ip address and I put the uri **/time** in the script client-observer.py, then I obtained this:



```
networkedss@networkedss-VirtualBox:~/Desktop/appiot_lab1/aiocoap$ python clientGET-observe.py
First response: <aiocoap.Message at 0x7ff5e419ed30: ACK 2.05 Content (MID 42869, token e45c) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s)
, 16 byte(s) payload>
b'2024-03-22 20:02'
Next result: <aiocoap.Message at 0x7ff5e41bd1c0: CON 2.05 Content (MID 63543, token e45c) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s), 1
6 byte(s) payload>
b'2024-03-22 20:02'
Loop ended, sticking around
INFO:coap:Response <aiocoap.Message at 0x7ff5e41b0d90: CON 2.05 Content (MID 63545, token e45c) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option
(s), 16 byte(s) payload> could not be matched to any request
INFO:coap:Response not recognized - sending RST.
```

The script initially set up an observation relationship with a CoAP server resource. As expected, it received and printed the first notification of the current state of the resource. Subsequently, it printed updates whenever the observed resource changed.

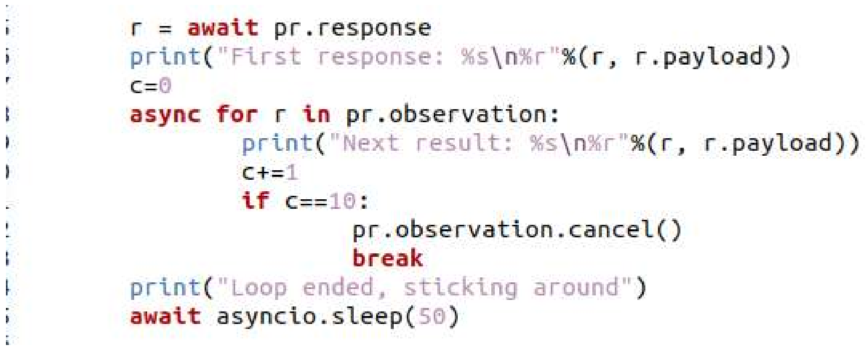
I modified it to cease observation after receiving ten values. To achieve this, I added a counter variable that increments with each received notification. When the counter reached ten, the script executed the observation.cancel() method to stop receiving updates.

I replaced this line of code:

async for r in pr.observation :

print(« Next result :%s\n%r » % (r,r.payload))

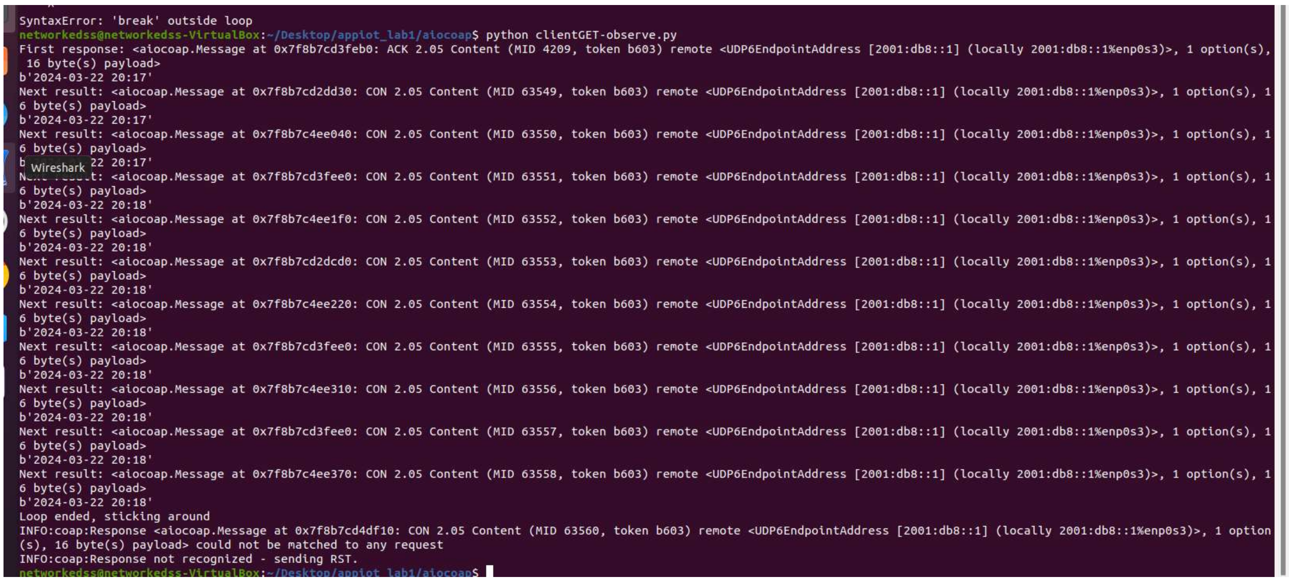
by this :



```
r = await pr.response
print("First response: %s\n%r"%(r, r.payload))
c=0
async for r in pr.observation:
        print("Next result: %s\n%r"%(r, r.payload))
        c+=1
        if c==10:
                pr.observation.cancel()
                break
print("Loop ended, sticking around")
await asyncio.sleep(50)
```

I implemented a counter called c and I stopped after 10 iterations.

Finally I obtain the good result which is :

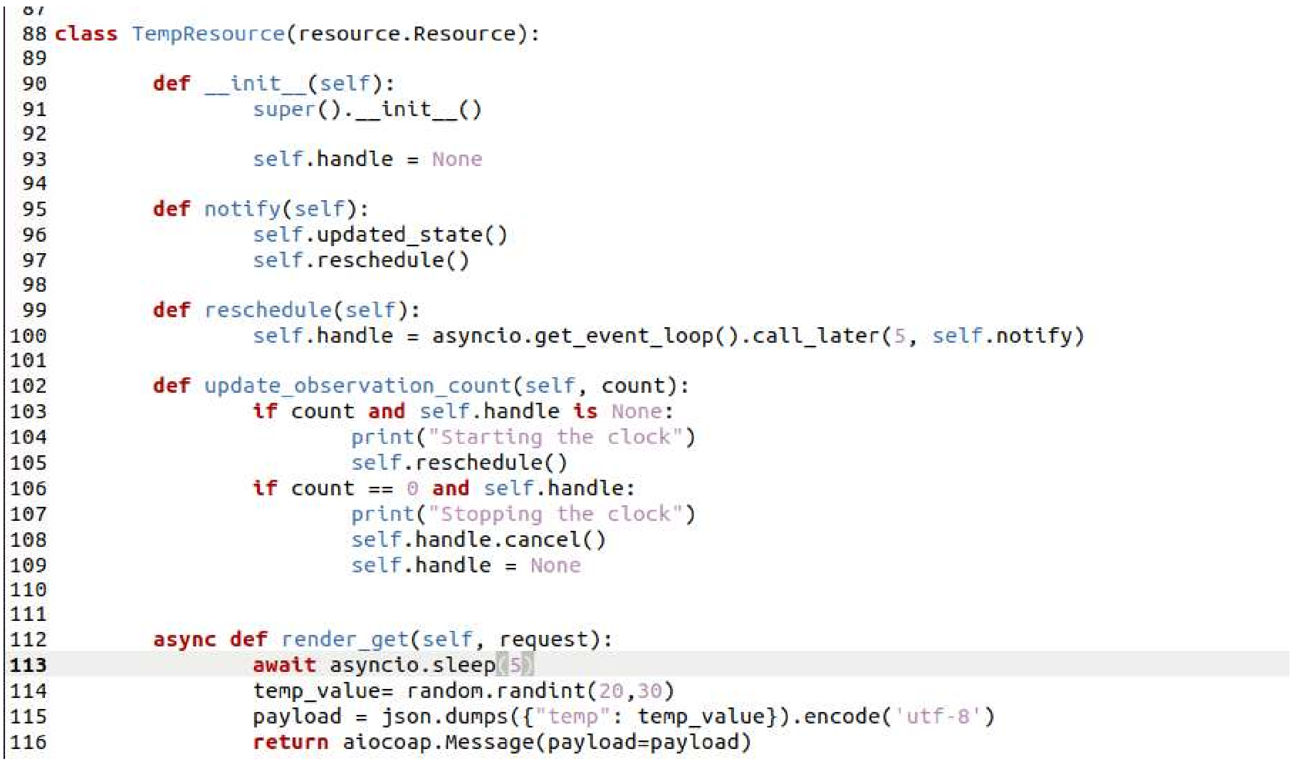


```
SyntaxError: 'break' outside loop
networkedss@networkedss-VirtualBox:~/Desktop/appiot_lab1/aiocoap$ python clientGET-observe.py
First response: <aiocoap.Message at 0x7f8b7cd3feb0: ACK 2.05 Content (MID 4209, token b603) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s),
 16 byte(s) payload>
b'2024-03-22 20:17'
Next result: <aiocoap.Message at 0x7f8b7cd2dd30: CON 2.05 Content (MID 63549, token b603) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s), 1
6 byte(s) payload>
b'2024-03-22 20:17'
Next result: <aiocoap.Message at 0x7f8b7c4ee040: CON 2.05 Content (MID 63550, token b603) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s), 1
6 byte(s) payload>
b[illegible]22 20:17'
N[illegible]t: <aiocoap.Message at 0x7f8b7cd3fee0: CON 2.05 Content (MID 63551, token b603) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s), 1
6 byte(s) payload>
b'2024-03-22 20:18'
Next result: <aiocoap.Message at 0x7f8b7c4ee1f0: CON 2.05 Content (MID 63552, token b603) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s), 1
6 byte(s) payload>
b'2024-03-22 20:18'
Next result: <aiocoap.Message at 0x7f8b7cd2dcd0: CON 2.05 Content (MID 63553, token b603) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s), 1
6 byte(s) payload>
b'2024-03-22 20:18'
Next result: <aiocoap.Message at 0x7f8b7c4ee220: CON 2.05 Content (MID 63554, token b603) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s), 1
6 byte(s) payload>
b'2024-03-22 20:18'
Next result: <aiocoap.Message at 0x7f8b7cd3fee0: CON 2.05 Content (MID 63555, token b603) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s), 1
6 byte(s) payload>
b'2024-03-22 20:18'
Next result: <aiocoap.Message at 0x7f8b7c4ee310: CON 2.05 Content (MID 63556, token b603) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s), 1
6 byte(s) payload>
b'2024-03-22 20:18'
Next result: <aiocoap.Message at 0x7f8b7cd3fee0: CON 2.05 Content (MID 63557, token b603) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s), 1
6 byte(s) payload>
b'2024-03-22 20:18'
Next result: <aiocoap.Message at 0x7f8b7c4ee370: CON 2.05 Content (MID 63558, token b603) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option(s), 1
6 byte(s) payload>
b'2024-03-22 20:18'
Loop ended, sticking around
INFO:coap:Response <aiocoap.Message at 0x7f8b7cd4df10: CON 2.05 Content (MID 63560, token b603) remote <UDP6EndpointAddress [2001:db8::1] (locally 2001:db8::1%enp0s3)>, 1 option
(s), 16 byte(s) payload> could not be matched to any request
INFO:coap:Response not recognized - sending RST.
networkedss@networkedss-VirtualBox:~/Desktop/appiot_lab1/aiocoap$
```

Upon successfully receiving a sequence of ten values, the system is programmed to terminate the observation, thereby stopping any further updates.

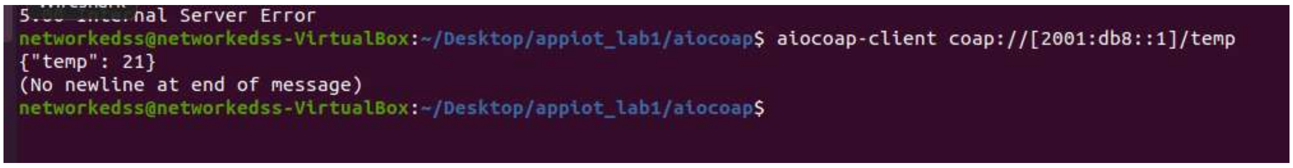
<u>Second Part:</u>

For doing the second part I modified the code server.py. I used the class TimeRessource for doing the new class call TempResource :



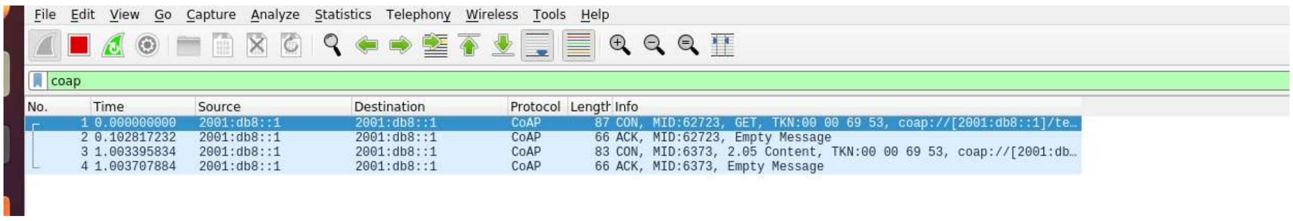
```
class TempResource(resource.Resource):

        def __init__(self):
                super().__init__()

                self.handle = None

        def notify(self):
                self.updated_state()
                self.reschedule()

        def reschedule(self):
                self.handle = asyncio.get_event_loop().call_later(5, self.notify)

        def update_observation_count(self, count):
                if count and self.handle is None:
                        print("Starting the clock")
                        self.reschedule()
                if count == 0 and self.handle:
                        print("Stopping the clock")
                        self.handle.cancel()
                        self.handle = None


        async def render_get(self, request):
                await asyncio.sleep(5)
                temp_value= random.randint(20,30)
                payload = json.dumps({"temp": temp_value}).encode('utf-8')
                return aiocoap.Message(payload=payload)
```

Just the function render_get changed. In the render_get function, a simulated delay of five seconds is introduced using asyncio.sleep(5) to emulate a time-consuming read operation, such as accessing a sensor or a database. A random temperature value between 20 and 30 is generated using random.randint(20, 30). This value is then formatted into a JSON string with json.dumps({"temp": temp_value}). Finally, this JSON string is encoded in 'utf-8' and sent back as the payload of the CoAP message with the content format set to application/json, making the response machine-readable and compliant with common data interchange standards. Finally I obtained this :



```
5.00 Internal Server Error
networkedss@networkedss-VirtualBox:~/Desktop/appiot_lab1/aiocoap$ aiocoap-client coap://[2001:db8::1]/temp
{"temp": 21}
(No newline at end of message)
networkedss@networkedss-VirtualBox:~/Desktop/appiot_lab1/aiocoap$
```

When using wireshark I obtained this :



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	2001:db8::1	2001:db8::1	CoAP	87	CON, MID:62723, GET, TKN:00 00 69 53, coap://[2001:db8::1]/te…
2	0.102817232	2001:db8::1	2001:db8::1	CoAP	66	ACK, MID:62723, Empty Message
3	1.003395834	2001:db8::1	2001:db8::1	CoAP	83	CON, MID:6373, 2.05 Content, TKN:00 00 69 53, coap://[2001:db…
4	1.003707884	2001:db8::1	2001:db8::1	CoAP	66	ACK, MID:6373, Empty Message

we see a CoAP GET request and response sequence. The exchange begins with a CON message indicating a GET request from the client. The server promptly acknowledges this request with an empty ACK message, indicating that it has received the request but the content is not yet ready. After a deliberate delay - often used to emulate data processing or retrieval from a sensor - the server responds with the content in a CON message. Finally, the client sends an ACK to confirm the receipt of the content, completing the transaction.