HTTP vs CoAP

The aim of this laboratory task is to compare HTTPS and CoAP. Wireshark will be used to analyze the way traffic is exchanged when making HTTPS and CoAP GET requests from websites.

The task

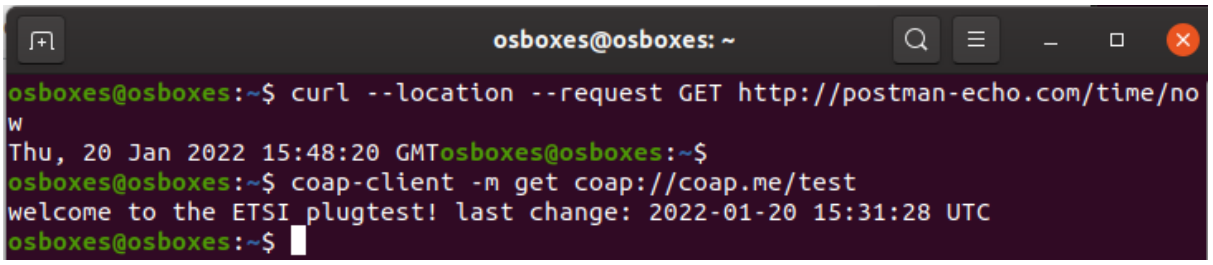
First we launch wireshark on the enp0s3 interface, then we execute the following commands to make GET requests:

```
curl --location --request GET http://postman-echo.com/time/now
coap-client -m get coap://coap.me/test
```

After executing the commands, capturing is stopped and a wireshark filter is applied to only see HTTP and CoAP related packets.

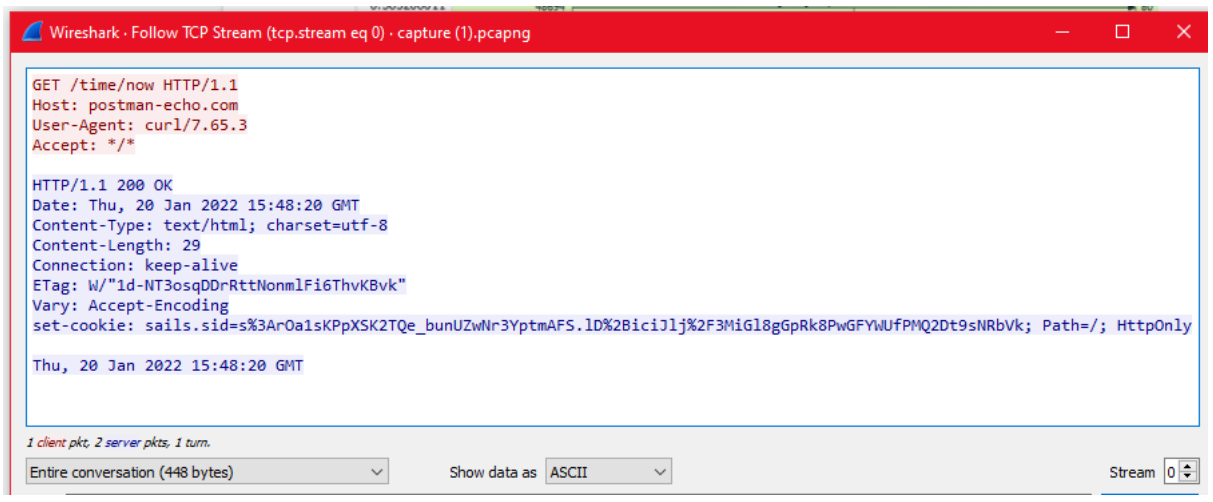
Console output

The commands in the console were executed, and this was the obtained output:



Wireshark results - TCP (HTTP) stream

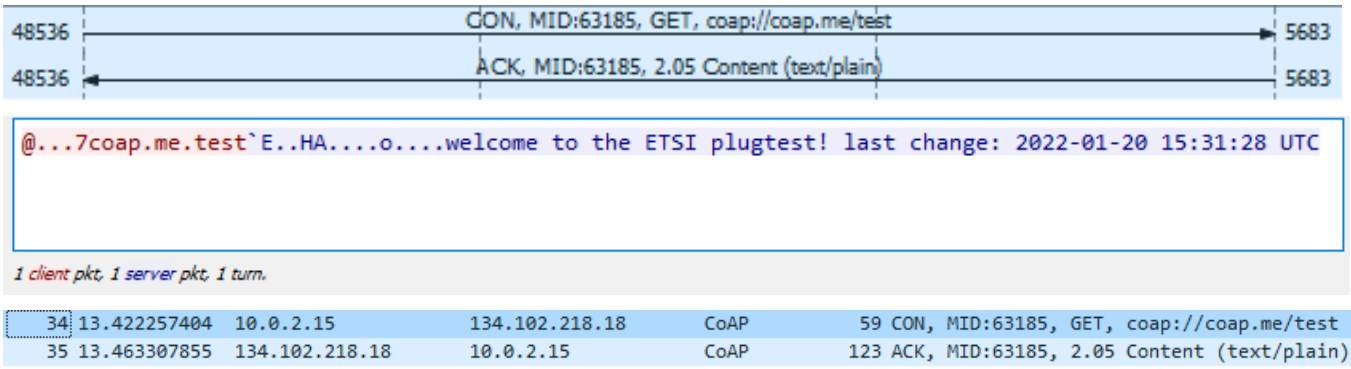
Below we show packets that correspond to the curl HTTP request:



11	0.102430911	10.0.2.15	54.236.132.16	TCP	74 48694 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2924008444 TSecr=0 WS=12
12	0.231677241	54.236.132.16	10.0.2.15	TCP	60 80 → 48694 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
13	0.231712178	10.0.2.15	54.236.132.16	TCP	54 48694 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
14	0.232162187	10.0.2.15	54.236.132.16	HTTP	142 GET /time/now HTTP/1.1
15	0.232564079	54.236.132.16	10.0.2.15	TCP	60 80 → 48694 [ACK] Seq=1 Ack=89 Win=65535 Len=0
16	0.365189588	54.236.132.16	10.0.2.15	TCP	413 80 → 48694 [PSH, ACK] Seq=1 Ack=89 Win=65535 Len=359 [TCP segment of a reassembled PDU]
17	0.365206011	10.0.2.15	54.236.132.16	TCP	54 48694 → 80 [ACK] Seq=89 Ack=360 Win=63881 Len=0
18	0.365534993	54.236.132.16	10.0.2.15	HTTP	60 HTTP/1.1 200 OK (text/html)
19	0.365541212	10.0.2.15	54.236.132.16	TCP	54 48694 → 80 [ACK] Seq=89 Ack=361 Win=63881 Len=0
20	0.366000783	10.0.2.15	54.236.132.16	TCP	54 48694 → 80 [FIN, ACK] Seq=89 Ack=361 Win=63881 Len=0
21	0.366283712	54.236.132.16	10.0.2.15	TCP	60 80 → 48694 [ACK] Seq=361 Ack=90 Win=65535 Len=0
22	0.490552499	54.236.132.16	10.0.2.15	TCP	60 80 → 48694 [FIN, ACK] Seq=361 Ack=90 Win=65535 Len=0
23	0.490578595	10.0.2.15	54.236.132.16	TCP	54 48694 → 80 [ACK] Seq=90 Ack=362 Win=63881 Len=0

Wireshark results - UDP (CoAP) stream

Below we show packets that correspond to the CoAP request:



HTTP analysis

a. What is the response displayed in the Linux terminal? How many characters were shown (counting spaces and punctuation marks)?

The response shows UTC time at the moment of the request. The exact text is:

`Wed, 19 Jan 2022 22:45:26 GMT`

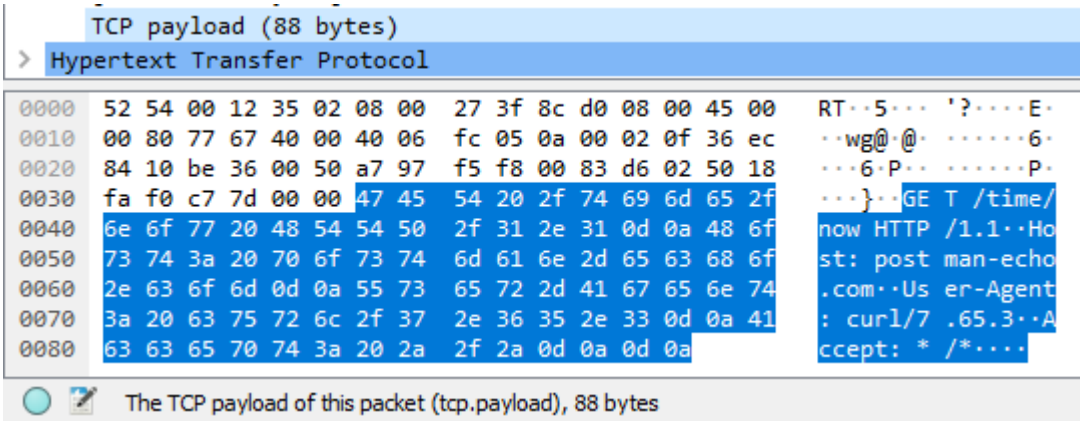
Which is in total 29 characters when including spaces.

b. How many HTTP messages were exchanged?

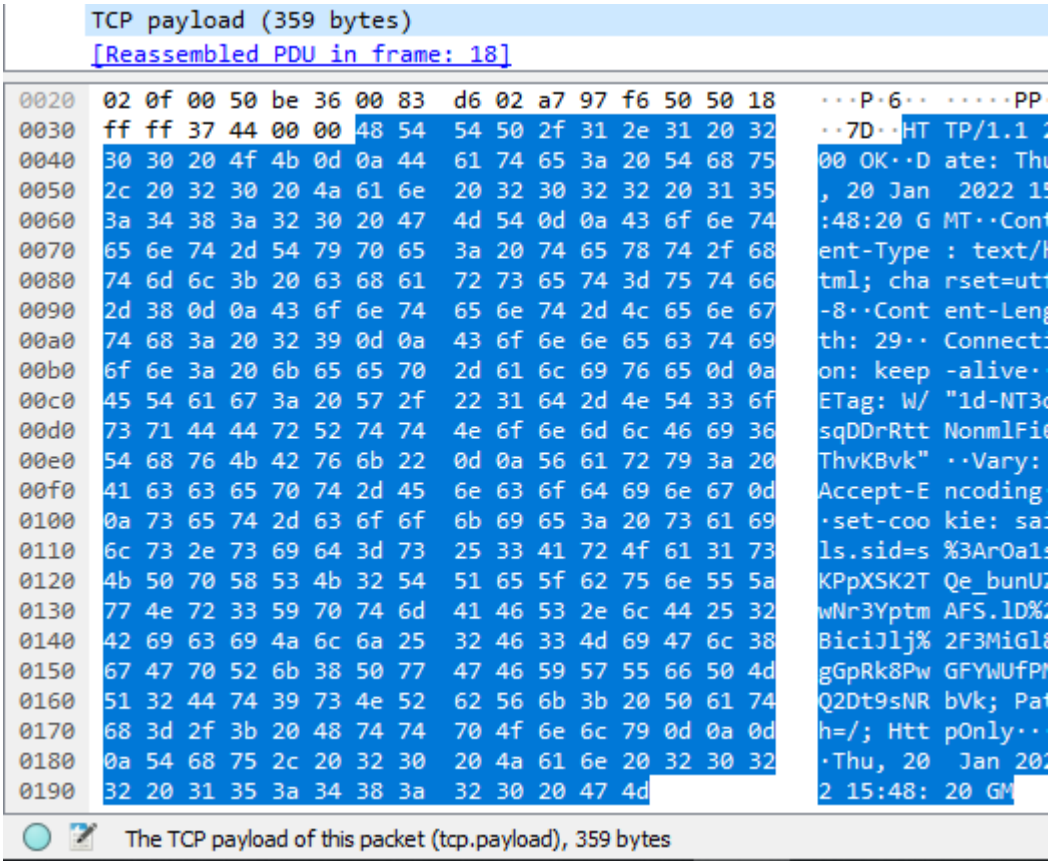
Two, as shown on the screenshots provided in **Wireshark results - TCP (HTTP) stream** paragraph (flow sequence and packet list).

c. How many bytes does the payload of the transport layer protocol need to encode the HTTP request? How many for the HTTP response?

As shown on screenshot - 88 bytes is the size of TCP payload:



TCP payload of the response is 359 bytes:



d. Divide the number of characters displayed in the screen (question 1.a) by the number of bytes included as the transport layer protocol payload of the HTTP response (question 1.d).

29/359 is equal to about 0.08, which means that the message text is only 8% of the used TCP payload byte size.

This is because TCP payload containing the response information also includes different information about the request response, for example: response code, HTTP version, date of response, charset, cookie info and other.

CoAP analysis

a. What is the response displayed in the Linux terminal? How many characters were shown (counting spaces and punctuation marks)?

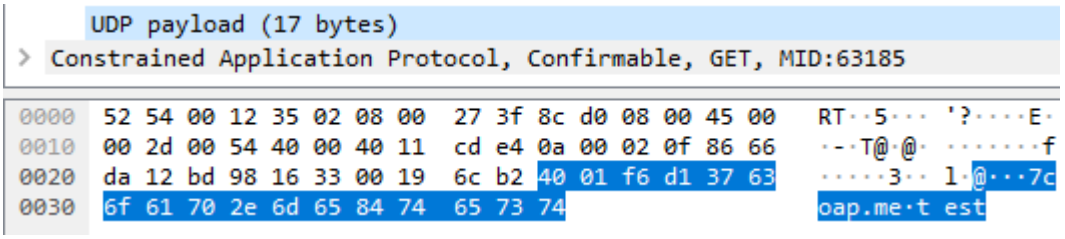
The response is:

`welcome to the ETSI plugtest! last change: 2022-01-20 15:31:28 UTC`

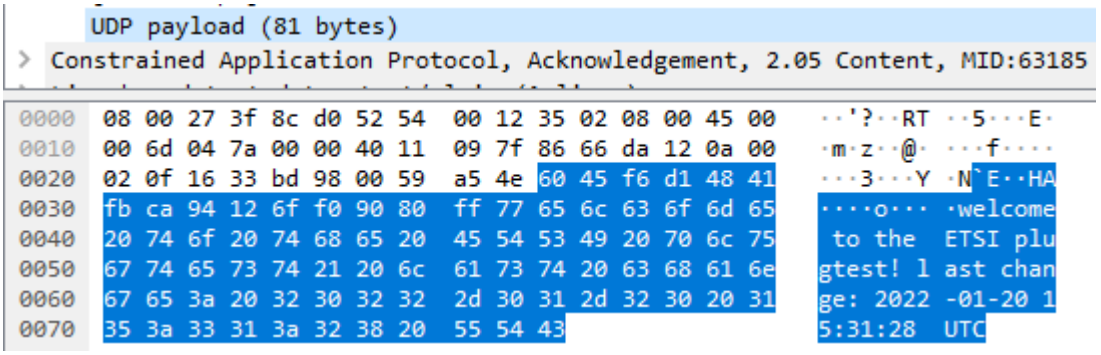
Which contains in total 66 characters.

b. How many bytes does the payload of the transport layer protocol need to encode the CoAP request? How many for the CoAP response?

For the request, the UDP payload contains 17 bytes:



The response has 81 bytes in the UDP payload:



c. Divide the number of characters displayed in the screen (question 2.a) by the number of bytes included as the transport layer protocol payload of the HTTP response (question 2.b).

66/81 = is about 0.815. Which shows that the utilization of the UDP window is about 80%, i.e. 80% of the payload size is the data we are interested in.

d. Considering your answers from questions 1.c. and 2.b., which protocol is more efficient in terms of encoding? Why?

In terms of efficiency, CoAP is definitely better. 80% of the payload size is the data, whereas for the HTTP it is only roughly 8%. This is because CoAP features way less information in the response than HTTP. Furthermore, relevant information is stored as codes rather than full text like in HTTP.

Closing remarks - CoAP vs HTTP

CoAP in general is way more efficient than HTTP. It should be noted though, that CoAP is less reliable than HTTP, because it uses UDP. CoAP in general is very good for devices that have constraints such as limited power, memory and other resources. These kinds of devices are present in the IoT field, so using CoAP is very important in order to save mentioned resources.
One very important feature of CoAP is that it supports multicast, which is important in cases where we need to manage multiple IoT devices. HTTP does not support multicast.

In no way it is recommended though to use CoAP to communicate between e.g. laptops, PCs and servers, as the protocol is less reliable than HTTP and has less options in the overhead.