POLITECNICO MILANO 1863

Internet of things

IoT Challenge #2 MQTT and CoAP in a Wireless Network Exercise PDF

Authors:
Daniel Shala - 10710181
Jurij Diego Scandola - 10709931

Academic Year:
2024 - 2025

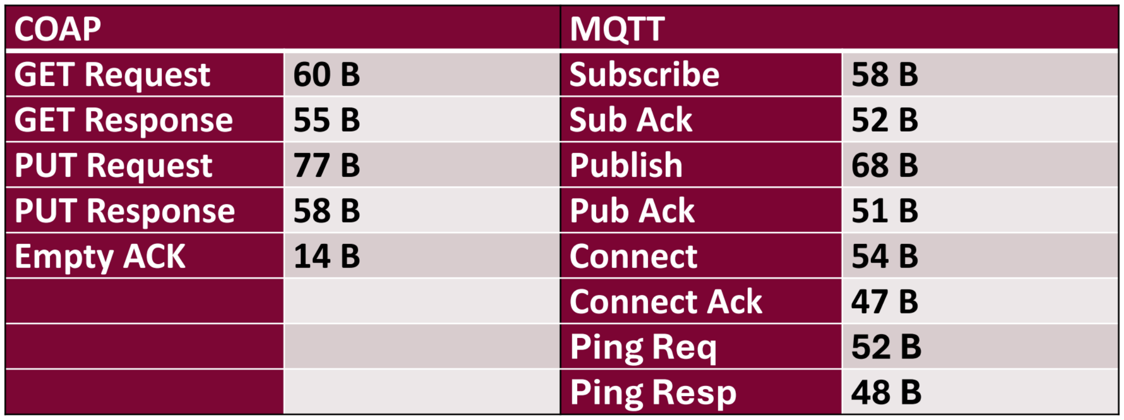
1 Exercise introduction

A wireless IoT network consists of the following devices:

- A battery-powered, Wi-Fi-enabled temperature sensor that measures and transmits temperature data every 5 minutes.
- A battery-powered, Wi-Fi-enabled valve that receives temperature readings from the sensor and computes the average temperature every 30 minutes to decide whether to open or close.
- A Raspberry Pi, connected to the power grid, which only supports MQTT for communication.

The temperature sensor and valve can communicate using either MQTT or CoAP, with a specific pre-defined topic or resource. The topic/resource length is 10 bytes and the payload size is 8 bytes. However, since the Raspberry Pi only supports MQTT, any interaction between the Raspberry Pi and the battery-operated devices must use MQTT. The sensor and valve, however, can communicate directly using CoAP if desired.

Consider the following message sizes (in bytes), which already include header and payload size for the COAP resource or MQTT topic used in the system:



COAP		MQTT	
GET Request	60 B	Subscribe	58 B
GET Response	55 B	Sub Ack	52 B
PUT Request	77 B	Publish	68 B
PUT Response	58 B	Pub Ack	51 B
Empty ACK	14 B	Connect	54 B
		Connect Ack	47 B
		Ping Req	52 B
		Ping Resp	48 B

Assumptions

1. Transmit and Receive cost per bit are:
   - $E_{TX} = 50$ nJ/bit
   - $E_{RX} = 58$ nJ/bit
2. The Wi-Fi network is ideal (no losses).
3. The processing cost on the valve to compute the average temperature every 30 minutes is $E_c = 2.4$ mJ.
4. The sensor and valve start in power-off state.

2 EQ1:

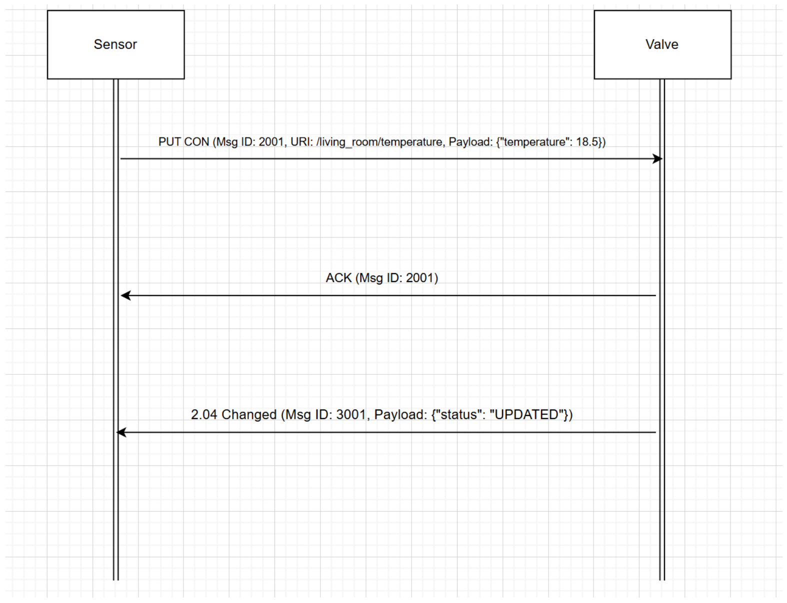
Compute the total energy consumed by the two battery-powered devices over a period of 24 hours in both cases when using COAP (a) and MQTT (b), using each in its most efficient configuration energywise.

a) CoAP case

CoAP (Constrained Application Protocol) is a specialized internet application protocol designed for constrained devices and low-power networks, primarily used in IoT (Internet of Things) environments. It operates over UDP (User Datagram Protocol) to minimize overhead and improve efficiency. CoAP is based on the client-server model and supports request/response interactions, similar to HTTP, but with lower resource consumption.

In this case, we are going to study the communication between the sensor and the valve only; in fact, due to the technology limitations of the Raspberry Pi, it can't use CoAP to communicate.
The model follows a client-server architecture, where the sensor acts as the client sending PUT requests, and the valve serves as the server responding with PUT responses. The valve may also send an empty acknowledgment (ACK) as confirmation of receipt. We understood that no GET requests or responses are sent, as the sensor is programmed to minimize data transmission by only sending the actual data and not unnecessary communication messages. This approach ensures efficiency by eliminating redundant requests.

For communication we will use port 5863, the default port for CoAP communication, used for standard, unencrypted CoAP communication. We won't use TLS port 5864 because, as the text specifies, we need to use the most efficient configuration energy-wise.


Sensor
Valve
PUT CON (Msg ID: 2001, URI: /living_room/temperature, Payload: {"temperature": 18.5})
ACK (Msg ID: 2001)
2.04 Changed (Msg ID: 3001, Payload: {"status": "UPDATED"})

Energy consumption in CoAP case

0. **Initialization (GET Request and Response):**
   - The sensor transmits (ETX0) the GET Request (60 B).
   - The valve receives (ERX0) the GET Request.
   - The valve transmits (ETX0r) the GET Response (55 B).
   - The sensor receives (ERX0r) the GET Response.
1. **PUT Request (Sensor $\rightarrow$ Valve)**
   - The sensor transmits (E_{TX1}) the PUT Request (77 B).
   - The valve receives (E_{RX1}) the PUT Request.
2. **ACK (Valve $\rightarrow$ Sensor)**
   - The valve transmits (E_{TX2}) an ACK (14 B).
   - The sensor receives (E_{RX2}) the ACK.
3. **PUT Response (Valve $\rightarrow$ Sensor)**
   - The valve transmits (E_{TX3}) the PUT Response (58 B).
   - The sensor receives (E_{RX3}) the PUT Response.

On a span of 24 hours, we have:

$$\frac{24\frac{\text{hours}}{\text{day}} \times 60\frac{\text{minutes}}{\text{hours}}}{5\frac{\text{minutes}}{\text{transmission}}} = 288\frac{\text{transmissions}}{day} \tag{1}$$

The values represent the number of transmissions that the sensor makes and the corresponding receptions by the valve. On a daily basis, there would be 288 PUT requests, 288 ACKs, and 288 PUT responses exchanged between the sensor and the valve.

Additionally, every 30 minutes, the valve computes the average value of the data it has received consuming E_c energy. This computed average is stored locally within the valve, as it is not required by any other device on the network that can communicate via CoAP, so it won't be sent.

$$\frac{24\frac{\text{hours}}{\text{day}} \times 60\frac{\text{minutes}}{\text{hours}}}{30\frac{\text{minutes}}{\text{computations}}} = 48\frac{\text{computations}}{day} \tag{2}$$

So we can finally compute energy consumptions as follows:

$$E_{\text{sensor}} = E_{\text{TX1}} + E_{\text{RX2}} + E_{\text{RX3}} \qquad E_{\text{valve}} = E_{\text{RX1}} + E_{\text{TX2}} + E_{\text{TX3}} + E_{\text{c}}$$

$$E_{\text{sensor}} = 288 \times \left[\left(77 \text{ B} \times 8 \frac{bit}{B} \times 50 \frac{nJ}{bit}\right) + \left(14 \text{ B} \times 8 \frac{bit}{B} \times 58 \frac{nJ}{bit}\right) + \left(58 \text{ B} \times 8 \frac{bit}{B} \times 58 \frac{nJ}{bit}\right) \right] = 0,01823 J$$

$$E_{\text{valve}} = 288 \times \left[\left(77 \text{ B} \times 8 \frac{bit}{B} \times 58 \frac{nJ}{bit}\right) + \left(14 \text{ B} \times 8 \frac{bit}{B} \times 50 \frac{nJ}{bit}\right) + \left(58 \text{ B} \times 8 \frac{bit}{B} \times 50 \frac{nJ}{bit}\right) \right] + 48 \times 2,4 \text{ mJ} = 0,1338 J$$

We will also calculate the initialization energy with the hypothesis that the process would happen just once a day; so we have that:

$$E_{\text{tx-get request}} = \left[\left(60 \text{ B} \times 8 \frac{bit}{B} \times 50 \frac{nJ}{bit}\right) \right] = 0,024 \text{ mJ (sensor)}$$

$$E_{\text{rx-get request}} = \left[\left(60 \text{ B} \times 8 \frac{bit}{B} \times 58 \frac{nJ}{bit}\right) \right] = 0,02784 \text{ mJ (valve)}$$

$$E_{\text{tx-get response}} = \left[\left(55 \text{ B} \times 8 \frac{bit}{B} \times 50 \frac{nJ}{bit}\right) \right] = 0,022 \text{ mJ (valve)}$$

$$E_{\text{rx-get response}} = \left[\left(55 \text{ B} \times 8 \frac{bit}{B} \times 58 \frac{nJ}{bit}\right) \right] = 0,02552 \text{ mJ (sensor)}$$

In conclusion, in 24 hours and using CoAP, the sensor will consume 18.73 mJ, while the valve will consume 133.85 mJ.

b) MQTT case

System Components

- **Sensor (Publisher)**: Collects environmental data (e.g., temperature) and periodically publishes it to an MQTT topic every 5 minutes.
- **Valve (Subscriber)**: Subscribes to the sensor's topic to receive temperature data and acts accordingly (e.g., opening/closing based on predefined thresholds). It also publishes its status update every 30 minutes.
- **Raspberry Pi (MQTT Broker)**: Acts as the central broker, managing message distribution, processing data (e.g., computing averages), and potentially forwarding it to external systems.

Flow of Data

- **Sensor → Raspberry Pi (Broker)**: Every 5 minutes, the sensor publishes temperature data to the MQTT topic `sensors/temperature`.

  ```
  Topic: sensors/temperature
  Payload: {"temperature": 22.5}
  ```

  The broker acknowledges the message if QoS ¿ 0.

- **Raspberry Pi (Broker) → Valve**: The valve, previously subscribed to `sensors/temperature`, receives the message and processes it. If needed, it takes action (e.g., opening/closing based on predefined temperature thresholds).

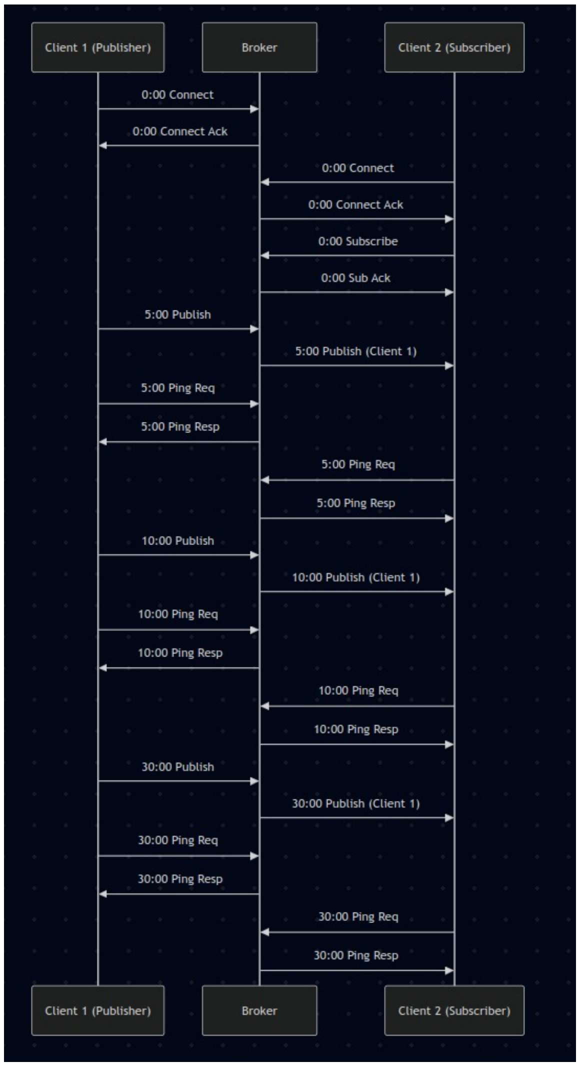
  Example action response:

  ```
  Topic: valve/status
  Payload: {"status": "open"}
  ```

- **Valve → Raspberry Pi**: Every 30 minutes, the valve publishes its status to `valve/status`. The broker may forward this to other subscribed devices.
- **Raspberry Pi → External Systems (Optional)**: The Raspberry Pi may compute aggregated metrics (e.g., average temperature) and publish them to another topic like `dashboard/temperature`.

  Example aggregated data:

  ```
  Topic: dashboard/temperature
  Payload: {"avg_temperature": 22.0}
  ```



Message	**Tx (Sent)**	**Rx (Received)**
CONNECT	1	0
CONNACK	0	1
PUBLISH	288	0
PUBACK (QoS greater than 0)	0	0
Total	**433 Tx**	**145 Rx**

Table 1: Messages Sent and Received Per Day for Sensor (Publisher)

Message	**Tx (Sent)**	**Rx (Received)**
CONNECT	1	0
CONNACK	0	1
SUBSCRIBE	1	0
SUBACK	0	1
PUBLISH	0	288
PUBACK (QoS greater than 0)	0	0
Total	**2 Tx**	**290 Rx**

Table 2: Messages Sent and Received Per Day for Valve (Subscriber)

Energy consumption in MQTT case

Considering the messages tables for MQTT the topic/resource length of 10 bytes, the payload size of 8 bytes and a daily initialization phase, we are able to compute the energy consume of every type of messages, transmitted or received by one party. We will now calculate the energy consumption for each type of message, both for transmission and reception, and sum the results for each device.

Energy Calculation for Sensor (Publisher)

PUBLISH (Tx):

$$\text{Number of bits} = 68\,\text{B} \times 8 = 544\,\text{bits}$$

$$\text{Energy for transmission} = 544\,\text{bits} \times 50\,\text{nJ/bit} = 27,200\,\text{nJ}$$

$$\text{Total energy (288 messages)} = 27,200\,\text{nJ} \times 288 = 7,833,600\,\text{nJ}$$

CONNECT (Tx):

$$\text{Number of bits} = 54\,\text{B} \times 8 = 432\,\text{bits}$$

$$\text{Energy for transmission} = 432\,\text{bits} \times 50\,\text{nJ/bit} = 21,600\,\text{nJ}$$

$$\text{Total energy (1 message)} = 21,600\,\text{nJ}$$

Total Energy for Sensor (Publisher):

$$\text{Total energy (Tx)} = 7,833,600 + 21,600 = 7,855,200\,\text{nJ}$$

CONNACK (Rx):

$$\text{Number of bits} = 47\,\text{B} \times 8 = 376\,\text{bits}$$

$$\text{Energy for reception} = 376\,\text{bits} \times 58\,\text{nJ/bit} = 21,808\,\text{nJ}$$

$$\text{Total energy (1 message)} = 21,808\,\text{nJ}$$

Total Energy for Sensor (Publisher) (Rx):

$$\text{Total energy (Rx)} = 21,808 = 21,808\,\text{nJ}$$

Total Energy for Sensor (Publisher) (Final):

$$\text{Total energy (Tx)} = 7,855,200\,\text{nJ}$$

$$\text{Total energy (Rx)} = 21,808\,\text{nJ}$$

$$\text{Total energy (Sensor)} = 7,855,200 + 21,808 = 7,877,008\,\text{nJ} \quad (\ 7.88\,\text{mJ})$$

Energy Calculation for Valve (Subscriber)

SUBSCRIBE (Tx):

$$\text{Number of bits} = 58\,\text{B} \times 8 = 464\,\text{bits}$$

$$\text{Energy for transmission} = 464\,\text{bits} \times 50\,\text{nJ/bit} = 23,200\,\text{nJ}$$

$$\text{Total energy (1 message)} = 23,200\,\text{nJ}$$

CONNECT (Tx):

$$\text{Number of bits} = 54\,\text{B} \times 8 = 432\,\text{bits}$$

$$\text{Energy for transmission} = 432\,\text{bits} \times 50\,\text{nJ/bit} = 21,600\,\text{nJ}$$

$$\text{Total energy (1 message)} = 21,600\,\text{nJ}$$

PUBLISH (Rx):

$$\text{Number of bits} = 68\,\text{B} \times 8 = 544\,\text{bits}$$

$$\text{Energy for reception} = 544\,\text{bits} \times 58\,\text{nJ/bit} = 31,552\,\text{nJ}$$

$$\text{Total energy (288 messages)} = 31,552\,\text{nJ} \times 288 = 9,090,816\,\text{nJ}$$

CONNACK (Rx):

$$\text{Number of bits} = 47\,\text{B} \times 8 = 376\,\text{bits}$$

$$\text{Energy for reception} = 376\,\text{bits} \times 58\,\text{nJ/bit} = 21,808\,\text{nJ}$$

$$\text{Total energy (1 message)} = 21,808\,\text{nJ}$$

SUBACK (Rx):

$$\text{Number of bits} = 52\,\text{B} \times 8 = 416\,\text{bits}$$

$$\text{Energy for reception} = 416\,\text{bits} \times 58\,\text{nJ/bit} = 24,128\,\text{nJ}$$

$$\text{Total energy (1 message)} = 24,128\,\text{nJ}$$

Total Energy for Valve (Subscriber):

$$\text{Total energy (Tx)} = 23,200 + 21,600 = 44,800\,\text{nJ}$$

Total Energy for Valve (Subscriber) (Rx):

$$\text{Total energy (Rx)} = 9,090,816 + 21,808 + 24,128 = 9,136,752\,\text{nJ}$$

Total Energy for Valve (Subscriber) (Final):

$$\text{Total energy (Tx)} = 44,800\,\text{nJ}$$

$$\text{Total energy (Rx)} = 9,136,752\,\text{nJ}$$

$$\text{Total energy (Valve)} = 44,800 + 9,136,752 = 9,181,552\,\text{nJ} \quad (\ 9.18\,\text{mJ})$$

Total Energy

Total energy = 9.18 + 7.88 = 17.06 mJ

3 EQ2:

Propose at least one solution for decreasing the energy consumption when passing using the Raspberry PI as a broker. Give a rough estimate of the energy saving that could be obtained with your solution: recompute the energy under your proposed configuration.

One effective strategy to reduce energy consumption is to batch the transmission of sensor readings. By encoding each temperature reading using 2 bytes, it is possible to:

- Transmit 4 readings in an 8-byte payload every 20 minutes, or
- Transmit 6 readings in a 12-byte payload every 30 minutes, assuming the payload size can be increased and the MQTT keep-alive is set to 30 minutes.

Let's analyze the energy consumption in the second scenario (30-minute batch) and compare it with the default case (5-minute intervals):

Case 1 – Transmission every 5 minutes
2 MQTT publish operations per 5 minutes (sensor → broker and broker → valve) Over 1 hour: 24 publish operations (2 per 5 min × 12 intervals) Payload size: 68 bytes per publish (including topic, headers, etc.)
Energy consumption:

$$2(2E_{tx} + 2E_{rx}) * 24 + 2E_c$$

$$48 * (2 * 68 * 8 * 50nJ/bit + 2 * 68 * 8 * 58nJ/bit) + 4.8mJ = 10.44mJ$$

Case 2 – Batched transmission every 30 minutes
2 MQTT publish operations per 30 minutes, payload size (68B + 4B = 72B) Over 1 hour: 4 publish operations (2 per half hour × 2 intervals)
Energy consumption:

$$2(2E_{tx} + 2E_{rx}) * 2 + 2E_c$$

$$4 * (2 * 72 * 8 * 50nJ/bit + 2 * 72 * 8 * 58nJ/bit) + 4.8mJ = 5.3mJ$$

In the end the savings in transmission is about 50.77%
The total energy consumption in this scenario is:

$$2 * (Connect + ConnectACK) + SUB + SUBACK + E_{tx,rx}$$

$Total = 2 * 0.043\text{mJ} + 0.047\text{mJ} + 5.3\text{mJ} = 5.43\text{mJ}$
Energy savings on total system: 31.85%