Journal of Computer and Communications, 2024, 12, 177-188
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Proposal for Energy Consumption Reduction between Connected Objects in a Network Running on MQTT Protocol

Saidou Haman[1,2*], Djorwe Temoa[2], Eric Michel Deussom Djomadji[1,3], Kolyang[2]

[1]Division of Information and Communications Technology, National Advanced School of Posts, Telecommunications and Information of Communication Technology, University of Yaoundé 1, Yaoundé, Cameroon
[2]Department of Computer Science and Telecommunications Engineering, National Advanced School of Engineering of Maroua, The University of Maroua, Maroua, Cameroon
[3]Department of Electrical and Electronic Engineering, College of Technology, University of Buea, Buea, Cameroon
Email: *saidouhaman@gmail.com

How to cite this paper: Haman, S., Temoa, D., Djomadji, E.M.D. and Kolyang (2024) Proposal for Energy Consumption Reduction between Connected Objects in a Network Running on MQTT Protocol. *Journal of Computer and Communications*, **12**, 177-188.
https://doi.org/10.4236/jcc.2024.1210012

Received: September 7, 2024
Accepted: October 26, 2024
Published: October 29, 2024

Abstract

The "Internet of Things" (IoT) refers to a set of intelligent "objects" that can communicate with each other directly or through a network. The IoT is the embodiment of the idea that everything can be connected anywhere and at any time. The concept can be applied to sectors such as e-health, e-government, automotive, geographic information systems, remote sensing, home networking, e-commerce and climate change mitigation. Unlike the Internet, the IoT has its own constraints, notably those linked to heterogeneity. This divergence is linked to different protocols, technologies and algorithms implemented in these connected objects for their interconnection. It should be noted that IoT devices can communicate with each other using different protocols and dedicated M2M (Machine to Machine) communication technologies. The aim of this work is to find solutions for optimising energy consumption during data exchanges between connected objects, with respect to certain constraints by using firstly this exchange for only Message Queuing Telemetry Transport (MQTT) and secondly the combination of the MQTT protocol and the Constrained Application Protocol (CoAP) protocol to check the quantity of the energy optimized. The MQTT protocol, for example, is one of the most widely used protocols for connected objects. Admittedly, this protocol consumes less energy, but in the situation of a very large number of users, the problem of saturation inevitably arises. In this article, we propose a solution of optimising energy consumption by combining the MQTT protocol with the CoAP protocol which can allow to use the standby mode contrary to the use of MQTT where the broker is always being turning. This solution has not yet been

implemented but is being discussed. In this article, we're going to use the joulemeter which is an application developed by Microsoft to measure and estimate the energy consumption of computers and applications. In our case, we take the example of the "Service Broker for network connections" of the Windows's 10 Operating System, in my own computer to show the difference between the consumption of energy without the standby mode and with standby mode, because with the MQTT, the Broker's MQTT is always on. Now, with the combination MQTT and CoAP, it is possible that we have standby mode and to compare these two cases in term of consumption of an energy. And to do it, we must use the joulemeter that we installed in our computer to simulate it. This is achieved by using the CoAP protocol combined with the MQTT protocol. The aim of our work is to reduce energy consumption in order to solve the problem of saturation of the MQTT by linking it to CoAP protocol by using Joulemeter mentioned above.

Keywords

Internet of Things, Heterogeneity, Message Queuing Telemetry Transport, Constrained Application Protocol, Application-Layer Semantic Gateway, Gateway

1. Introduction

Connected objects, often referred to internet of things [1] use different protocols and algorithms for their interconnection. But there are problems linked to heterogeneity that need to be resolved. Means such as gateways are used to facilitate their interoperability. Interconnection often requires sufficient resources in terms of the number of sensors and clients used. And given the environmental constraints, we need to take action to optimise energy savings. One of these problems is network saturation when these connected objects use the MQTT protocol. Several problems are linked to MQTT network saturation, in particular optimising energy consumption [2], using IoT gateways, monitoring performance indicators and using suitable wireless technologies. Let us know that our manner to optimize the energy between objects is to switch from active mode to standby mode by using joulemeter installed in our computer. Joulemeter which is a software package developed by Microsoft Research, designed to estimate the energy consumption of computer applications and systems [3]. In fact, the use of MQTT does not allow us to move to standby mode, because the broker is always on. But by combining the MQTT protocol to the CoAP, communication between objects will take place in the same way as in a client/server system, so it is when information is requested that the server must take charge of it, and also it's possible to have a system in standby mode when no request is made. Note that this will be done using the Application-Layer Semantic Gateway (ALSG) with a multi-protocol proxy. In this article, so that's how we're going we propose to optimise the energy consumption

of connected objects through the combined use of the MQTT and CoAP protocols [4]-[6] by using joulemeter.

2. State of the Art

This article highlights two application protocols, one of which is Publish/Subcribe used by the MQTT protocol and the other Client/Server by the CoAP protocol. The MQTT protocol and the CoAP protocol will be discussed here.

The development of the Internet of Things is putting the Internet under remarkable pressure. New protocols have been implemented, including application protocols such as MQTT, XMPP, DDS and CoAP. Traditional protocols, such as HTTP, are not suitable, given the constraints in terms of energy consumption and computing capacity. In this section, we will present the application protocols specially developed for IoT devices. These are organised into two groups: Publish/Subscribe protocols (MQTT, XMPP) and client/server protocols (COAP, WebSocket).

The client/server protocol is a transaction model between several programs or processes via a network. In this model, there are mainly two players: a client, making requests to the server. A server, which waits for requests from clients, then processes and responds to them. There are several types of client/server protocols. In this article, we will focus on two protocols: the MQTT protocol and the CoAP protocol.

2.1. The MQTT (Message Queue Telemetry Transport) Protocol

MQTT is a messaging protocol that was introduced by Andy Stanford-Clark of IBM and Arlen Nipper of Arcom (now Eurotech) in 1999 and standardised in 2013 at OASIS [7]. It is based on a client/server architecture where sensors act as clients and then connect to a server called a broker. It is a lightweight protocol that is easy to implement.

It is widely used in IoT device connectivity and M2M (machine-to-machine communication). It uses the Publish-subscribe model to offer transition flexibility and implementation simplicity, as illustrated in **Figure 1** below.

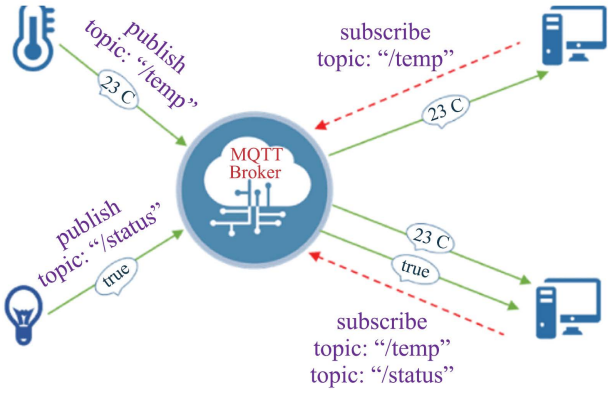


Figure 1. MQTT architecture [8].

2.2. The Constrained Application Protocol (CoAP)

CoAP (Constrained Application Protocol) is a web transfer protocol designed to be ran on constrained devices, which are generally IoT devices. It is based on the REST architecture. REST represents a simpler way of exchanging data between clients and servers via HTTP [9]. The CoAP protocol is a lightweight version of http, based on UDP. This makes it more suitable for IoT applications. It should also be remembered that the CoAP protocol modifies certain HTTP functionalities to meet IoT requirements such as low energy consumption. Consequently, COAP is presented as the protocol by excellence for constrained nodes and networks, including the IoT, in [10].

3. The Problem

The problem is the saturation of the MQTT protocol when we have a very large number of users [2]. We illustrate the difference in energy consumption between the MQTT and CoAP protocols in **Figure 4** using the joulemeter [11]. Admittedly, the MQTT protocol does not consume energy, but the fact that the relationship between the clients and the broker is always in an "activated state" will consume even more energy. In other words, if power consumption is our main constraint, MQTT transfers data over Transmission Control Protocol (TCP) whereas CoAP transfers data over User Data Protocol (UDP). This also makes a huge difference in the power consumption of these two protocols where MQTT consumes more power than CoAP because it has to maintain the connection between the client and the broker in an "always on" state. So, the problem is how to optimise energy consumption to avoid network saturation. With this in mind, our article uses the CoAP protocol to solve this problem.

Figure 2 below shows the respective size in bytes of the messages that can be sent by the CoAP and MQTT protocols, which are 4 bytes and 8 bytes respectively.

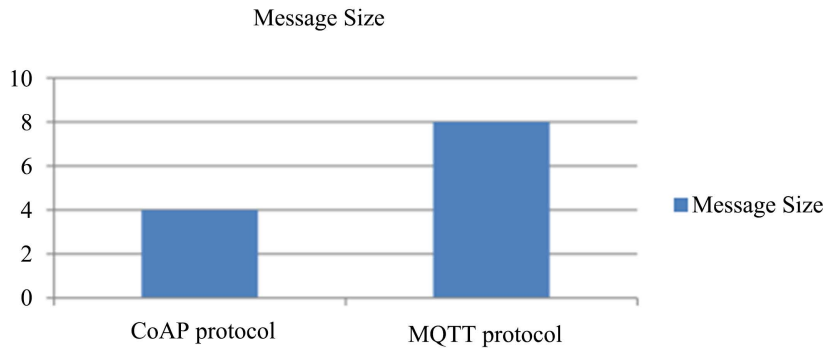


Figure 2. Message size of the CoAP and MQTT protocols [11].

The power or energy used by the client and the server for sending and receiving the first message up to the receiving of last response or message is termed as in the power consumption by the different protocols. It is measured in J or mJ. The CoAP protocol also consumes low power than the MQTT protocol and performs

a better transmission of the messages over a network. The power consumption by these two protocols is illustrated in the graph below [11] (**Figure 3**):

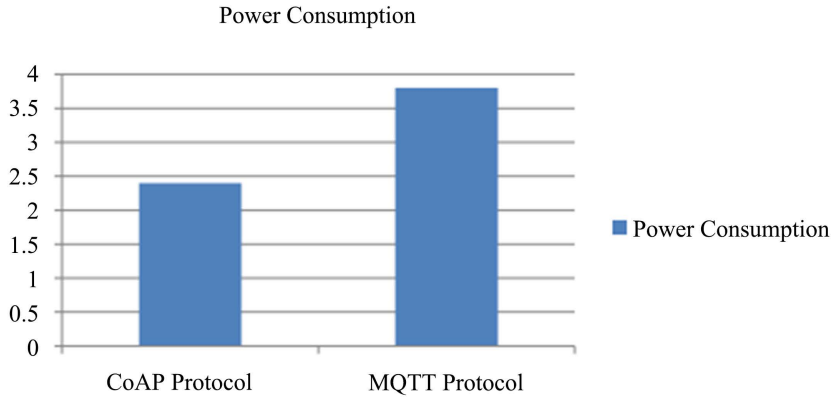


Figure 3. Power consumption by the CoAP and MQTT protocols [11].

4. Methodology

The aim is to implement prerequisites in order to solve the problem of saturation of the MQTT protocol as shown by Mishra *et al.* [12] in their work related to the Stress-Testing MQTT Brokers. These prerequisites are the use of the ALSG gateway [4], whose role is to facilitate communication and interconnectivity between heterogeneous messaging protocols such as MQTT and CoAP. ALSG can receive any information from MQTT or CoAP and, through its Mult-iprotocol Proxy, data are saved from these two protocols and delivered in the same format, which is JSON. In the event that the connected objects want to exchange data with each other, it will no longer be necessary to know the nature of the client. Is this client MQTT or CoAP or is the receiving node that wants information MQTT or CoAP, because everything happens as if we were in a client/server system. And in the case of a client/server system, it's when a request is sent that the server starts looking for it to deliver it.

We find that with our ALSG, the fact that the Broker is always running is no longer relevant. And for more information about our ALSG, we give you its various components and their role as follows:

- Proxy-multi protocol solves the problem of message translation by introducing two additional components, message store, topic router and message broker;
- The Message broker is responsible for representing the different message protocols so that message content is not disrupted.
- The Semantic Annotation Service is the heart of the semantic interoperability and heterogeneous data integration system, which translates raw data and transfers it to a knowledge centre application;
- The Gateway Interface converts the data into JSON, a specific format for linked data (JSON-LD);
- The Message store is responsible for storing all messages until they are successfully sent to the top level of IoT services via the REST interface on ALSG;

- The Topic router is responsible for the creation, maintenance and knowledge sharing of all connected IoT devices.

4.1. The ALSG Gateway (Application-Layer Semantic Gateway)

The architecture is that of ALSG [4], which makes it possible to receive the various data from different clients and route them to recipients in the same JSON-LD format (**Figure 4**).

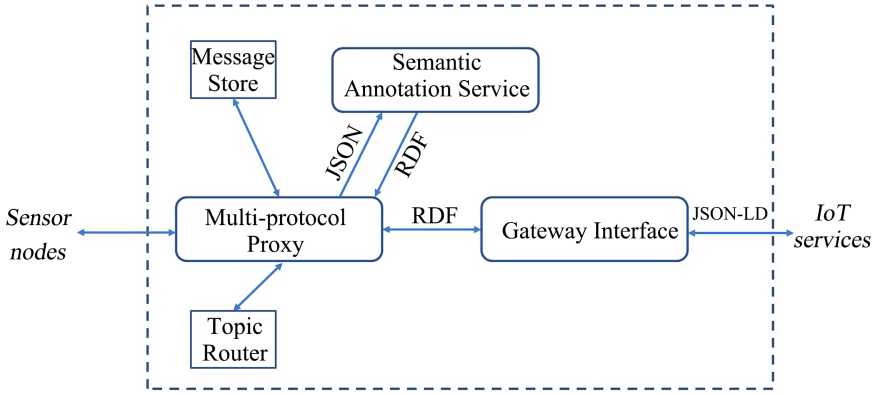


Figure 4. Architecture of the ALSG gateway [4].

4.2. Multi-Protocol Proxy

The Multi-Protocol Proxy receives data from the various CoAP and MQTT protocols for communication. It is the key element in putting the CoAP and MQTT protocols together (**Figure 5**).

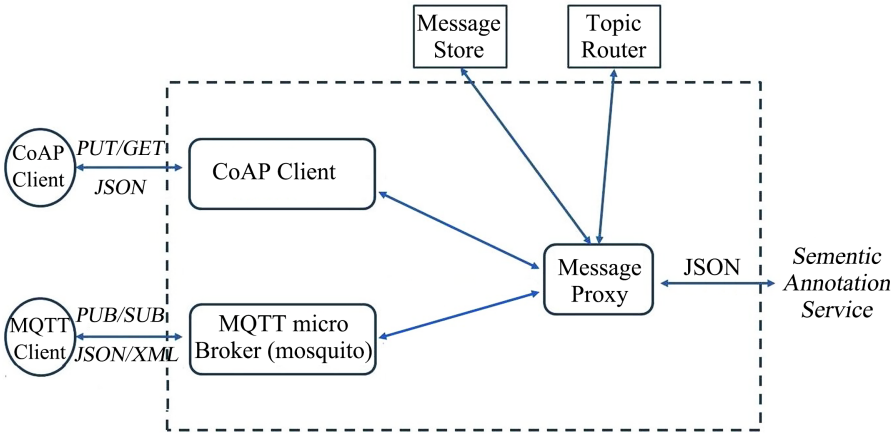


Figure 5. Multi-Protocol Proxy for communication between messaging protocols [4].

4.3. Proposed Architecture

Our solution consists of implementing the ALSG1 architecture shown in **Figure 6**, inspired by the ALSG architecture in order to prevent the Broker from always remaining in an "Activated state" while waiting for messages from publishers and subscribers. Now, with this new architecture, when publishers publish

information in topics, it is stored in the Message Store with an identifier and the Semantic Annotation Service for translation into compatible formats. And now, when subscribers go to subscribe, they can access the right information using their identifiers.

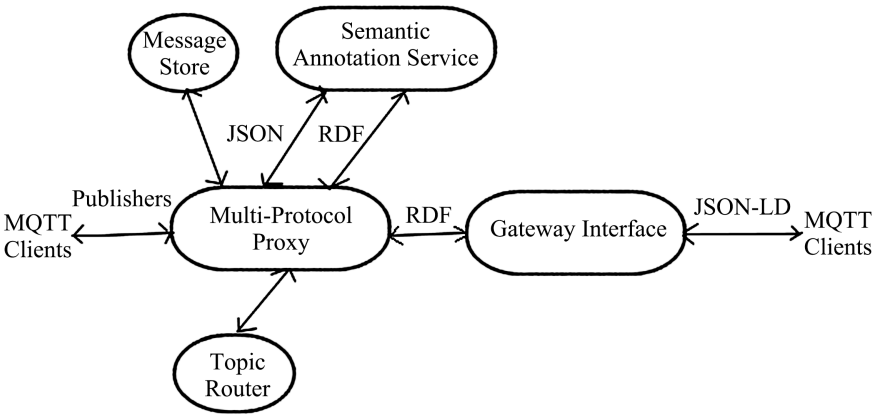


Figure 6. Proposed architecture of the solution; ALSG1.

After presenting the architecture, we'll introduce the platform used for the simulation and implementation of this project which in fact is GitHub. GitHub [3] is a developer platform that allows developers to create, store, manage and share their code. It uses Git software, providing the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project. GitHub can provide the joulemeter [13] tool used in this paper and presented below in **Figure 7**.

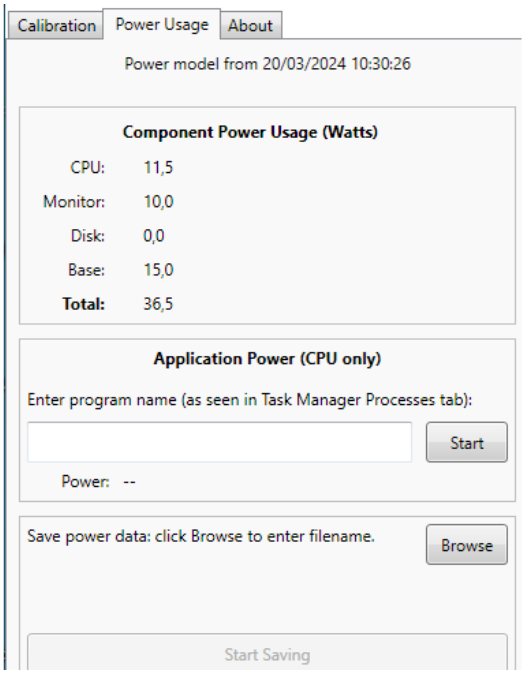
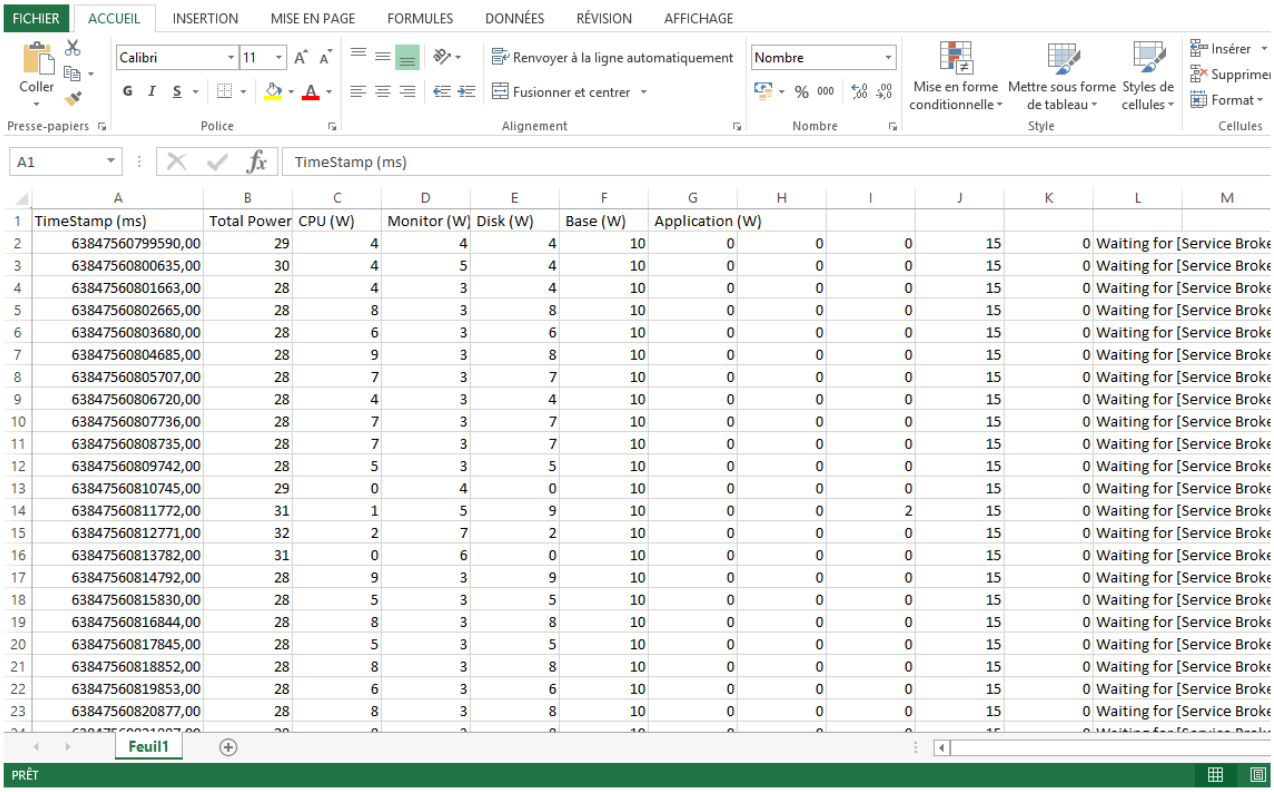


Figure 7. Interface of joulemeter in my own computer.

The simulation importance is not to prove again. Many authors have proposed different king of simulation solution to tests proposed algorithms or designs in various fields like Offloading between WiFi and 4G LTE [14], Networking and elearning [15] and 4G core network virtualization [16] using tools like NS3, Huawei ENSP and others. As sai before, in this work we use GitHub.

Below is an image of the excel file generated after testing the "Broker service for network connections" before and after putting the computer to sleep in **Figure 8**.

In fact, the data in **Figure 8** is obtained by calculating the energy consumption of Broker service for network connections. Simply enter the name of the application in the "start" box in **Figure 7** and click on the "start" button in the joulemeter tool interface. We left it running for a minute to see the power consumption of the "network broker service" application and then switched to standby mode to see the power consumption of the same application.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	TimeStamp (ms)	Total Power	CPU (W)	Monitor (W)	Disk (W)	Base (W)	Application (W)						
2	63847560799590,00	29	4	4	4	10	0	0	0	15	0	Waiting for [Service Broke	
3	63847560800635,00	30	4	5	4	10	0	0	0	15	0	Waiting for [Service Broke	
4	63847560801663,00	28	4	3	4	10	0	0	0	15	0	Waiting for [Service Broke	
5	63847560802665,00	28	8	3	8	10	0	0	0	15	0	Waiting for [Service Broke	
6	63847560803680,00	28	6	3	6	10	0	0	0	15	0	Waiting for [Service Broke	
7	63847560804685,00	28	9	3	8	10	0	0	0	15	0	Waiting for [Service Broke	
8	63847560805707,00	28	7	3	7	10	0	0	0	15	0	Waiting for [Service Broke	
9	63847560806720,00	28	4	3	4	10	0	0	0	15	0	Waiting for [Service Broke	
10	63847560807736,00	28	7	3	7	10	0	0	0	15	0	Waiting for [Service Broke	
11	63847560808735,00	28	7	3	7	10	0	0	0	15	0	Waiting for [Service Broke	
12	63847560809742,00	28	5	3	5	10	0	0	0	15	0	Waiting for [Service Broke	
13	63847560810745,00	29	0	4	0	10	0	0	0	15	0	Waiting for [Service Broke	
14	63847560811772,00	31	1	5	9	10	0	0	2	15	0	Waiting for [Service Broke	
15	63847560812771,00	32	2	7	2	10	0	0	0	15	0	Waiting for [Service Broke	
16	63847560813782,00	31	0	6	0	10	0	0	0	15	0	Waiting for [Service Broke	
17	63847560814792,00	28	9	3	9	10	0	0	0	15	0	Waiting for [Service Broke	
18	63847560815830,00	28	5	3	5	10	0	0	0	15	0	Waiting for [Service Broke	
19	63847560816844,00	28	8	3	8	10	0	0	0	15	0	Waiting for [Service Broke	
20	63847560817845,00	28	5	3	5	10	0	0	0	15	0	Waiting for [Service Broke	
21	63847560818852,00	28	8	3	8	10	0	0	0	15	0	Waiting for [Service Broke	
22	63847560819853,00	28	6	3	6	10	0	0	0	15	0	Waiting for [Service Broke	
23	63847560820877,00	28	8	3	8	10	0	0	0	15	0	Waiting for [Service Broke	

Figure 8. Structure of raw data collected from the simulation platform.

5. Results

The results of the research present the power consumption of the CoAP and MQTT protocols as a function of message sizes.

We recall that when the Broker is in the "Activated state", our server is quickly subjected to the maximum load to which it could be subjected during a peak in activity. This peak activity is the source of significant energy consumption. To mitigate this, we have used the CoAP protocol, where exchanges between the

different entities use client/server mode. As a result, the broker used by the MQTT protocol can now go into standby mode without causing any major impact on communication between the clients and the server. As the broker is in standby mode, we are seeing a considerable reduction in our server's electricity consumption, as it will enter intensive processing only when it receives a request. According to the thesis report "Data centres energy optimization" [17], a server's consumption in standby mode is around 65% of its maximum consumption. In other words, when our server's broker goes into an "inactive state", the potential reduction in energy consumption is around 35%. To confirm this thesis, we installed the "joule meter" in our machine and started it up in "activated mode" for one (01) minute by running the "Service Broker for network connections", and after one (01) minute, the machine went into "standby mode". At the end of this scenario, we obtained the result in the following link:

https://github.com/saidou-supptic/Test_with_joulemeter/blob/main/Test_with_joulemeter.csv. This result enabled to obtain the curve shown in **Figure 9**.

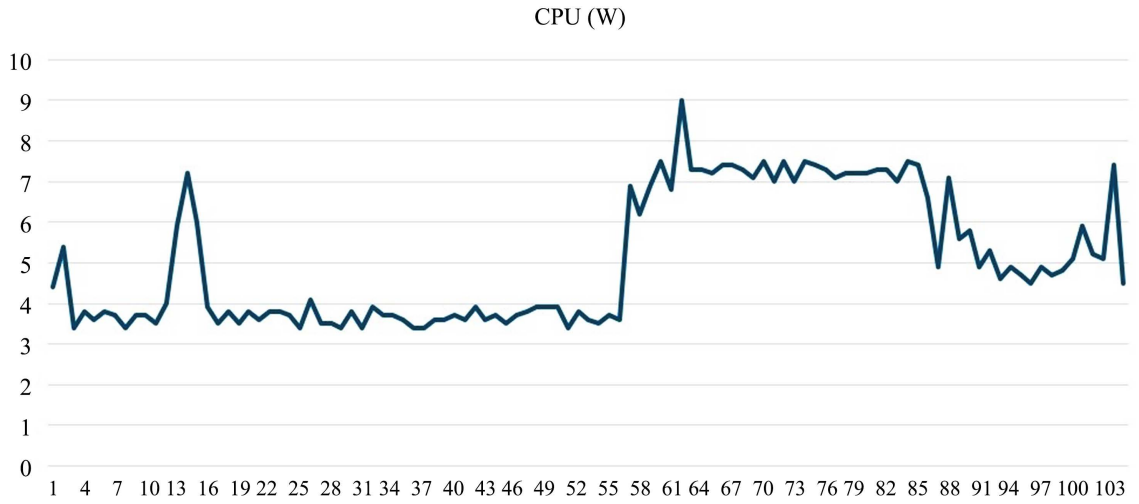


Figure 9. CPU consumption for the "Service Broker for network connections" in "activated mode" and then in "standby mode".

When the machine was in "standby mode" between [16 - 55], energy consumption dropped to around 4.5 W. When it switched to "active mode" between [55 - 85], energy consumption picked up again to around 7.2 W…

The longer the time, the greater the reduction in energy consumption, but here we only carried out our test for one (01) minute.

The following curve is generated from **Figure 8**, showing power consumption before and after the "Network Connection Broker Service" is put on standby

The energy consumption of a machine implementing the CoAP protocol and a machine implementing the MQTT protocol is illustrated in **Figure 10**. We have determined the electrical energy consumption of our server where both the CoAP protocol and the MQTT protocol are implemented, as shown in **Figure 10**.

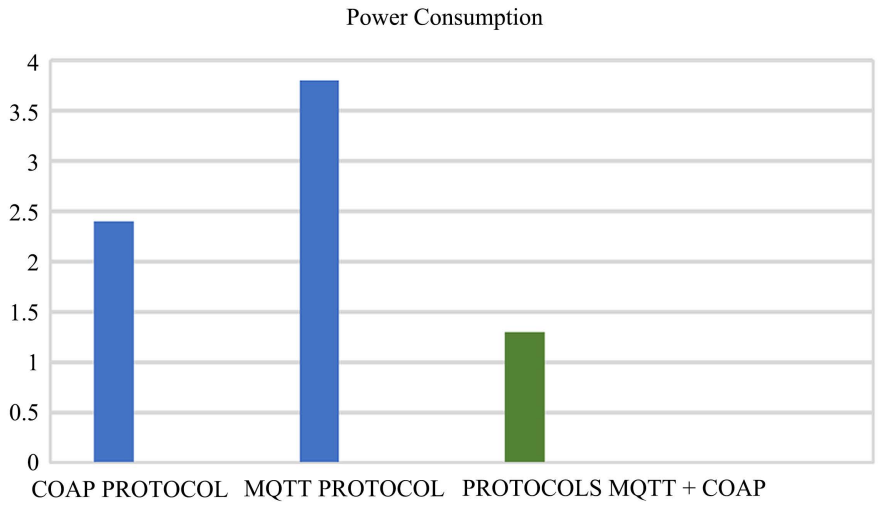


Figure 10. Energy consumption of MQTT protocol, CoAP protocol and merging of MQTT and CoAP protocols [10].

In fact, in **Figure 10**, the respective energy consumption is 2.4 joules for the CoAP protocol and 3.75 joules individually, but by combining the two protocols we obtain an energy consumption of around 1.3 joules.

The aim of our article is to show the energy consumption of the MQTT protocol and that of CoAP. As **Figure 10** shows, the combination of these two protocols helps us to consume less energy.

To sum up, we can have the following table presenting the power consumption for CoAP, MQTT protocol and combined MQTT + CoAP protocols, we can see a reduction of 52 % between MQTT + CoAP and CoAP protocol, and 67.56% between MQTT + CoAP and MQTT protocol.

As shown in **Table 1**, by combining MQTT and CoAP protocols, we can optimise (reduced) energy and have reduction of power consumption greater than 50% compare to each protocol taken individually.

Table 1. Power consumption and comparison between protocols.

CoAP	MQTT	MQTT + CoAP
2.5 Joules	3.7 Joules	1.2 Joule

6. Conclusion

The combination of the MQTT and CoAP protocols can be a solution for avoiding MQTT network saturation. The best practices for avoiding network saturation in the MQTT protocol, such as optimising energy consumption, using IoT gateways, monitoring performance indicators and using suitable wireless technologies, can also be applied to the combination of the MQTT and CoAP protocols. The combination of the MQTT and CoAP protocols, can be used to optimise energy consumption in the IoT using the advantages of each protocol and in our paper, an architecture that combines the two protocols but operates on the CoAP principle

is illustrated in figure. By using MQTT for control messages and CoAP for data messages, it is possible to reduce the load on the MQTT network and optimise energy consumption. CoAP can also be used for multicast messages, reducing the load on the network. In short, the combination of the MQTT and CoAP protocols is a solution for avoiding saturation of the MQTT network, using the advantages of each protocol to optimise energy consumption and avoid network saturation. In summary, CoAP is generally more suitable for IoT applications requiring low power consumption, particularly for battery-powered devices, due to its lightweight design and use of UDP. In contrast, MQTT, while also lightweight, can consume more power due to its TCP-based communication model and the need to maintain reliable connections. The choice between the two protocols will therefore depend on the specific requirements of the application, particularly in terms of reliability, latency and energy consumption.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] IoT-GSI (2011) Initiative "Normes mondiales sur l'Internet des objets" Internet of Things.

[2] Mektoubi, A., Hassani, H.L., Zakari, A. and Malassé, O. (2021) Nouvelle approche de communication sécurisée des objets connectés basée sur le protocole MQTT. *Revue Méditerranéenne des Télécommunications/Mediterranean Telecommunication Journal*, **6**, No. 2.

[3] Williams, A. (2012) GitHub Pours Energies into Enterprise—Raises $100 Million from Power VC Andreessen Horowitz. TechCrunch.

[4] Sihem, B. (2021) Une Approche d'Interopérabilité entre les Objets Intercon-nectés dans le Contexte d'Internet des Objets. http://dspace.univ-khenchela.dz:4000/handle/123456789/4943

[5] Craggs, I. (2022) MQTT vs CoAP for IoT. https://www.hivemq.com/article/mqtt-vs-coap-for-iot/

[6] Craggs, I. (2024) MQTT vs CoAP for IoT. https://www.hivemq.com/article/mqtt-vs-coap-for-iot/

[7] Kobo, H.I., Abu-Mahfouz, A.M. and Hancke, G.P. (2017) A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements. *IEEE Access*, **5**, 1872-1899. https://doi.org/10.1109/access.2017.2666200

[8] Djemai, T. (2021) Étude des protocoles de messagerie pour l'Internet des objets (IoT). *IEEE Journals & Magazine*, **211**.

[9] Zhao, M., Li, H., Li, Y., Fang, L. and Chen, P. (2018) Non-Orthogonal Multi-Carrier Technology for Space-Based Internet of Things Applications. In: Li, B., Shu, L. and Zeng, D., Eds, *Communications and Networking*, Springer, 37-45. https://doi.org/10.1007/978-3-319-78130-3_5

[10] Heller, M. (2018) CoAP: Un protocole d'application pour des milliards de minuscules nœuds Internet. https://www.lemagit.fr/definition/CoAP-Constrained-Application-Protocol

[11] Sharma, A. and Nandal, V. (2020) Comparison between the Messaging Protocols:

CoAP and MQTT Protocol. *Journal of Emerging Technologies and Innovative Research*, **7**, 481-490.

[12] Mishra, B., Mishra, B. and Kertesz, A. (2021) Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements. *Energies*, **14**, Article 5817. https://doi.org/10.3390/en14185817

[13] Nupur, K. and Bhattacharya, A. (2009) Joulemeter: Virtual Machine Power Measurement and Management. MSR Technical Report.

[14] Lyeb, L.A.H., Deussom Djomadji, E.M. and Tonye, E. (2022) A Proposal and Simulation with NS3 of a New Offloading Algorithm between LTE/LTE-Advanced and Wi-Fi. *European Journal of Applied Sciences*, **10**, 59-88. https://doi.org/10.14738/aivp.105.13060

[15] Eric Michel, D.D., Duplex, M.P. and Sone, M.E. (2020) WLAN Simulations Using Huawei eNSP for e-Laboratory in Engineering Schools. *IOSR Journal of Electronics and Communication Engineering*, **15**, 47-70.

[16] Emmanuel, T., Michel, D.D.E. and Agbor, E.O.B. (2024) Virtualization of a 4G Evolved Packet Core Network Using Network Function Virtualization (NFV) Technology with NS3 for Enterprise and Educational Purpose. *American Journal of Networks and Communications*, **13**, 1-18. https://doi.org/10.11648/j.ajnc.20241301.11

[17] Bayati, L. (2021) Data Centers Energy Optimization. 247 p. https://theses.hal.science/tel-03120640v1/document