A middleware architecture for mastering energy consumption in internet of things applications

Pedro Victor Borges Caldas da Silva, Chantal Taconet, Sophie Chabridon, Denis Conan, Everton Cavalcante

A Middleware Architecture for Mastering Energy Consumption in Internet of Things Applications

Pedro Victor Borges
Samovar, Télécom SudParis
Institut Polytechnique de Paris
France
pedro.borges@telecom-sudparis.eu

Chantal Taconet
Samovar, Télécom SudParis
Institut Polytechnique de Paris
France
chantal.taconet@telecom-sudparis.eu

Sophie Chabridon
Samovar, Télécom SudParis
Institut Polytechnique de Paris
France
sophie.chabridon@telecom-sudparis.eu

Denis Conan
Samovar, Télécom SudParis
Institut Polytechnique de Paris
France
denis.conan@telecom-sudparis.eu

Everton Cavalcante
Federal University of Rio Grande do Norte
Natal, Brazil
everton.cavalcante@ufrn.br

***Abstract*—The Internet of things (IoT) has been identified as a significant contributor to increasing Information and Communication Technology (ICT) energy consumption in the future. Carefully designing interactions between client applications and systems in the IoT may positively impact energy saving. For this purpose, IoT middleware, the underlying software that manages those interactions, should consider energy efficiency strategies a first-class concern. Furthermore, adding an energy-awareness capability to the middleware could significantly contribute to deepening developers' understanding of energy consumption by applications and helping them minimize the energy demand. In this paper, we propose energy-efficiency strategies and integrate them into an IoT middleware. We also offer a model to calculate the energy consumption of the interactions between an IoT consumer application and an IoT system, which the middleware could use to choose the best strategy to constrain the application's energy consumption to a given budget.**

***Index Terms*—Internet of Things, middleware, energy consumption, energy efficiency, energy-awareness**

I. Introduction

In 2019, the energy consumption of Information and Communication Technology (ICT) reached 5% of the worldwide electricity consumption, with estimates to almost double by 2025 [1]. The relentless increase in ICT energy demand at a time when energy efficiency is a priority concern for society is a paradox.

The world is experiencing a massive surge of things connected to the Internet known as the *Internet of Things* (IoT). By the end of 2025, estimates indicate the world will comprise nearly 100 billion IoT devices [2], [3]. Furthermore, the increase in the number of IoT systems requires dealing with energy-awareness and energy efficiency in the IoT context as first-class concerns [4]. In the IoT, energy efficiency has first been considered in the design of software deployed on IoT devices [5]. However, reducing software energy consumption in IoT may be broader than at the device level. IoT devices generated around 67 zettabytes of data in 2020 [6], and IoT applications consume part of such a volume of data. Therefore, carefully designing interactions between consumer applications and IoT systems with an energy-efficiency concern is also essential.

Designing and implementing IoT applications is complex because it involves different challenges, such as adequately identifying various stakeholder roles at the different phases of application development, dealing with heterogeneity in IoT systems, and handling a vast amount of data from disparate devices [7]. Moreover, energy efficiency is a new requirement that IoT system designers need to address. This aspect is even more challenging because developers still lack knowledge about software energy consumption [8].

Developers can alleviate the complexity of IoT applications by relying on *IoT middleware* that abstracts the heterogeneity of devices and protocols [9]. IoT specialized middleware provides (i) abstractions for accessing volatile physical devices and managing the data produced by these devices, (ii) virtualization and aggregation functions of many devices into IoT systems, and (iii) interoperability for managing software entities [10]–[13]. To facilitate the communication and data flow between IoT applications and devices, developers can also use *IoT platforms* [14]. These platforms provide interfaces, interaction patterns, communication protocols, and computational capabilities to support IoT application development. They also include services that provide functionalities such as device discovery, context management, and data analysis [15]. IoT client applications can take advantage of the capabilities of IoT platforms and middleware to access IoT data.

Noureddine et al. [16] highlight that middleware for distributed applications could contribute to optimizing or reducing the energy consumption of hardware devices, software

This work is a contribution to the Energy4Climate Interdisciplinary Center (E4C) of the Institut Polytechnique de Paris and École des Ponts ParisTech, supported by 3rd Programme d'Investissements d'Avenir [ANR-18-EUR-0006-02]. It has been partially funded by the "Futur & Ruptures" program from Institut Mines-Télécom, Fondation Mines-Télécom, and Institut Carnot.

services, and the platform itself. IoT middleware should hence not consider energy efficiency solely as a non-functional requirement at the application level. Instead, it must be at the solution's core because the middleware is expected to be shared by many applications and to offer facilities to ease application development. This capability is even more relevant considering that application developers and users often have limited knowledge of how much energy their software consumes and which parts use the most energy [8].

IoT middleware should ensure that applications keep energy consumption at only what is necessary for the expected functionalities. For this purpose, the middleware should be carefully designed with *energy-efficient mechanisms*, e.g., algorithms, protocols, and interaction patterns. To go a step further and provide energy consumption estimations and measures to the IoT applications and end-users, it is also essential to consider *energy-awareness*, i.e., understanding the energy consumption and how efficient each mechanism is [17]. For this purpose, an energy-aware IoT middleware provides a service that can be used by an IoT application to precisely know the energy consumption of the interactions between the application and the IoT system. This facility can contribute to easing the task of decreasing energy consumption and limiting the energy to a given budget.

The primary research question of this work is: *What are the strategies to be proposed by an IoT middleware to reduce the energy consumption of IoT client applications?* To address this research question, we have defined and integrated energy efficiency strategies in an IoT middleware and evaluated the energy consumption of IoT applications with and without those strategies. We also proposed a model to quantitatively estimate the energy consumption of the interactions between a client application and the IoT system. In this paper, we demonstrate how our energy model can be used within the IoT middleware to choose the appropriate strategies to not drift away from the energy consumption of an IoT client application from a given budget. The results of our computational experiments revealed a decrease of up to 60% in energy consumption by applying energy efficiency strategies at the middleware level compared to not using them.

The remainder of this paper is structured as follows. Section II analyzes energy strategies in existing IoT middleware. Section III describes the software architecture of middleware for IoT client applications. Section IV presents energy strategies and their integration in an IoT middleware for client applications. Section V shows the evaluation to compare the middleware with and without the integration of energy-efficiency strategies. Section VI gives the conclusion and future steps of this work.

II. Related Work

This section presents an analysis of related works concerning IoT middleware and energy efficiency. We have studied IoT middleware proposals and have identified the most relevant energy efficiency strategies that would help reduce the energy consumption of IoT applications.

Many strategies middleware uses for providing energy efficiency concern *network adaptation*. Network adaptation refers to introducing new protocols or modifying existing ones, making network optimizations (e.g., choosing the network technology), and reducing network usage at the middleware level. For example, Al-Roubaiey et al. [18] modify the Data Distribution Service (DDS^{TM}) protocol [19] to improve energy efficiency in a sensor network, choosing the nearest nodes to interact with. Akkermans et al. [20], Padhy et al. [21], and Sarkar et al. [22] propose each a protocol adaptation by using new algorithms to select a subset of requested sensors while keeping an acceptable accuracy or to predict sensor values. Akkermans et al. [20] particularly explore the capabilities of IPv6 multicast to replace client-server interactions with publish-subscribe interactions, resulting in lower network and energy consumption. Switching the communication protocol does result in significant improvements. In this paper, we propose several strategies related to network adaptation for grouping requests into one message and changing the communication protocols or modifying the interaction pattern.

The *data filtering* capability offered by some middleware allows for pre-processing data to reduce the amount of transmitted data or limit the size of the messages according to specific criteria, such as network load, CPU usage, and energy usage. Oliveira et al. [23], Li et al. [24], Podnar Zarko et al. [25], Al-Madani and Shabra [26], and Ramachandran et al. [27] use data filtering strategies to continuously remove unused data to reduce the amount of data to be processed/transmitted, thereby reducing the energy consumption of the system. For example, Oliveira et al. [23] provide an energy-aware data collection that can reduce the amount of data processed and sent to the network while maintaining an accurate data flow for applications. This strategy is particularly suitable for applications that need high quality of service (QoS), such as real-time applications. Our work allows data filtering based on the frequency of updates by adapting the refresh time.

Our contributions rely on an energy model we propose to compute the energy consumption due to interactions. This mathematical model may set parameters such as the frequency of updates and grouping of requests that enable the application to respect an energy budget. It also provides interaction energy-awareness at the application level that may be used for end-user energy-awareness.

In the literature, we can observe four main targets of energy-awareness: (i) *end-user application*, representing the device used by the user to access an IoT application; (ii) *connected objects*, such as sensors and actuators; (iii) *server*, referring to middleware usually deployed in fogs or clouds; and (iv) *sensor network*, related to techniques to reduce energy consumption over the whole sensor network. TThe middleware presented by Aazam et al. [28] and Pasricha [29] save energy at the server level. For instance, the one described by Aazam et al. [28] processes data at the cloud/fog level, and only the necessary data are sent to the server. Other studies target the end-user application side or the connected object side. For example, Cecchinel et al. [30] propose a deep sleep of connected objects

to save device battery. Al-Roubaiey et al. [18], Cecchinel et al. [30], Sarkar et al. [22], Li et al. [24], Al-Madani et al. [26], and Huang et al. [31] propose to reduce the energy consumption in the sensor network, e.g., by changing communication among nodes in the same network.

The middleware in the IoT system is deployed in different locations. Song et al. [32], Padhy et al. [21], Pasricha [29], and Aazam et al. [33] propose deploying the middleware at the end-user application side. On the other hand, Kalbarczyk and Julien [34], Shekhar et al. [35], and Podnar Zarko et al. [25] propose a middleware deployed in both the end-user applications and the cloud. Jeon and Jung [36], Sarkar et al. [22], Li et al. [24], Huang et al. [31], and Ramachandran et al. [27] deploy the middleware in the connected objects. It is also possible to notice that servers have been used to deploy the middleware without being the energy-efficiency target in the proposals.

Al-Roubaiey et al. [18], Cecchinel et al. [30], Al-Madani et al. [26], and Ramachandran et al. [27] present middleware on the cloud side. Aazam et al. [28], Akkermans et al. [20], Oliveira et al. [23], and Banouar et al. [37] present middleware deployment in a gateway, while Mukherjee et al. [38] deploy their middleware in the gateway and the connected object. The choice to deploy the middleware on the cloud, gateway, or other parts of a system is highly influenced by the implemented solution, and there is no rule to choosing the right place. Furthermore, Kalbarczyk and Julien [34], Shekhar et al. [35], and Podnar Zarko et al. [25] deploy their middleware in multiple places so that the placement of the middleware is related to how the middleware energy-awareness or energy efficiency strategy works.

The strategies proposed in this paper target reducing energy consumption on the end-user application side. The impact would be to save energy on mainly mobile devices with battery-constrained devices such as smartphones.

III. IoT Middleware Architecture

This section first describes the IoT systems' distributed architecture considered in this work. Next, we present the software architecture of an IoT middleware for IoT consumer applications.

Fig. 1 illustrates the distributed architecture of classical IoT systems such as those considered in this work. An IoT system consists of (i) IoT devices (sensors and actuators), (ii) gateways, (iii) IoT platforms, and (iv) end-user applications. This paper focuses on applications that consume sensor data we call *IoT consumer applications*. In our experiments (see Section V), we interact with FIWARE/Orion [39], a well-known IoT platform.

IoT middleware, which handles the interactions between the distributed components of the IoT system, is present in all the distributed components of the IoT system. This work focuses on the middleware supporting the interactions of applications with IoT platforms. This middleware is deployed on the end-user device. Our goal is to reduce the energy consumption of

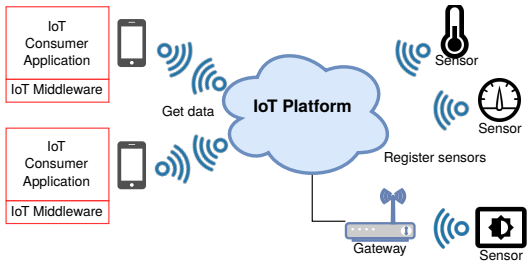

Fig. 1. IoT distributed architecture

IoT applications by integrating energy-efficient strategies and energy-awareness mechanisms within the IoT middleware.

As Disdarevi et al. [40] discuss, the protocols commonly used for the interactions between IoT consumer applications and platforms are HTTP [41], which brings to play the synchronous client/server interaction pattern, and MQTT [42], which brings to play the asynchronous publish/subscribe interaction pattern. In the synchronous pattern, a client application consumes IoT data by sending a request to the IoT platform, which responds to the request. In the publish/subscribe pattern, a consumer application registers to the IoT platform and asynchronously receives publications when they are available [43].

Fig. 2 describes the components of IoTvar, a middleware library deployed on IoT consumer applications that manage all the interactions with IoT platforms [44], [45]. IoTvar relies on declaring variables automatically mapped to sensors whose values are transparently updated with sensor observations through proxies on the client side. IoTvar components are organized in layers. Some components are specific to IoT platforms and should be adapted to the platform APIs and data model.

All the IoT platforms share the two bottom layers. The *protocol layer* defines components that interact with protocols commonly used between IoT consumer applications and IoT platforms such as HTTP and MQTT. When needed, developers can add other protocols. The *interaction layer* provides two components that manage the interaction patterns, i.e., request/reply and publish/subscribe, regardless of the IoT platform.

The top layers contain components specific to IoT platforms: the *API layer* and the *unmarshaller layer* act as a glue that maps the API calls and data structures to the IoT platforms. The *discovery layer* provides functionalities for discovering devices and managing filtering mechanisms.

IV. Energy strategies for an IoT middleware

Section IV-A describes the energy efficiency strategies we propose at the middleware level for IoT consumer applications. Next, Section IV-B introduces the concept of energy budget at the middleware level and how it can be controlled using an energy model. Finally, Section IV-C presents the architecture of the IoTvar middleware with the necessary modifications to support energy-efficient mechanisms and budget management.

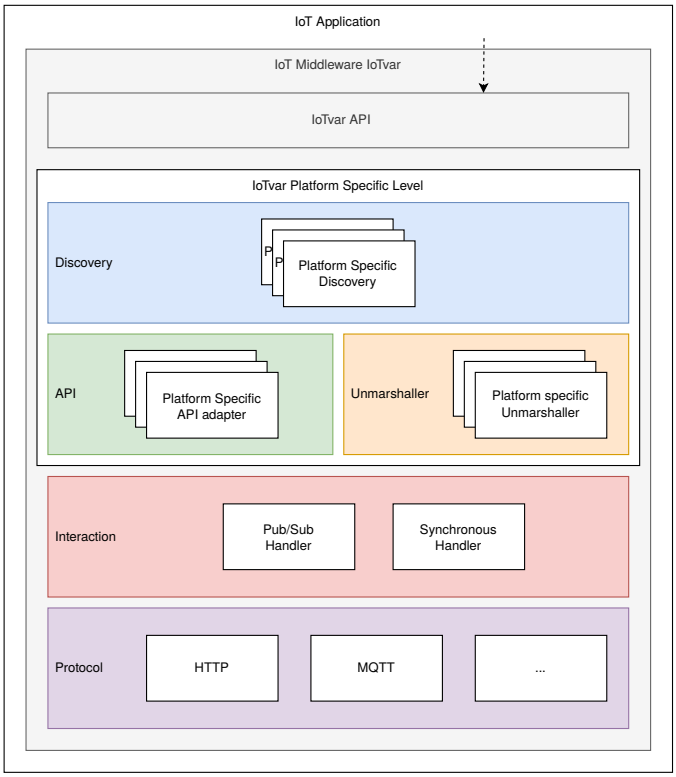

Fig. 2. A generic architecture for IoT middleware

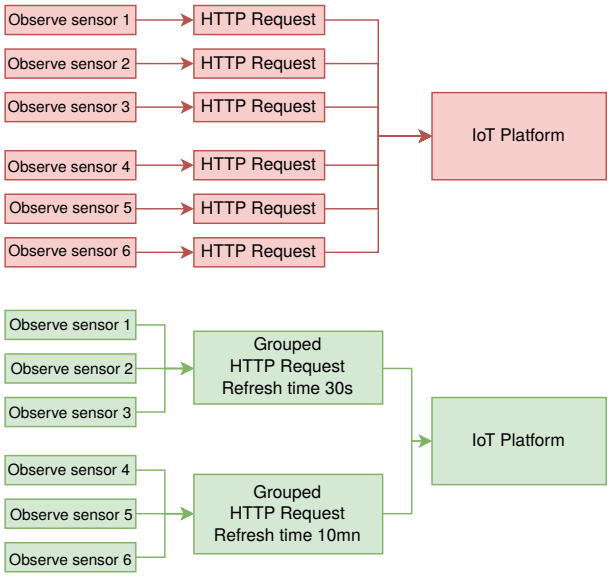

Fig. 3. Grouping sensor observations into one message

A. *Energy-efficient Strategies*

IoT middleware is a relevant software level for improving energy efficiency as it is where many applications may share energy-efficient interaction practices. Although many strategies have been proposed by IoT middleware (see Section II), few of them are dedicated to IoT middleware for IoT consumer applications. Therefore, we offer a set of strategies that could be implemented by IoT middleware in providing energy efficiency to IoT consumer applications: ① communication protocol choice, ② message grouping, ③ refresh-time adaptation, and ④ interaction pattern switch.

We implemented two of those strategies in IoTvar, namely strategies ② and ③. We have not implemented strategies ① and ④ as they depend on the availability of protocols and interaction patterns on the IoT platforms. The IoT platform did not provide grouping sensors for the two interaction patterns in our experiments. We present the results of a quantitative evaluation of strategy ② in Section V-B, while Canek et al. [46] evaluated other strategies.

1) Communication protocol choice: IoT middleware for consumer applications may abstract several protocols and choose among the protocol proposed by the IoT platform that provides the best energy efficiency. For example, an IoT application that wants to communicate with an IoT platform using the publish/subscribe pattern that both MQTT and HTTP support could lead to the middleware automatically choosing MQTT because it is known to be a more energy-efficient protocol. Canek et al. [46] show that the gain over a Wi-Fi network is 20% of energy consumption.

2) Message grouping: Canek et al. [46] showed that HTTP on a Wi-Fi network, for the same number of messages, using different payloads, from 24 up to 3120 bytes, has a small impact on the energy consumption of the application [46]. If the application requires multiple sensor observations, this strategy enables the middleware to automatically combine the different observations into one single message (see Fig. 3. This has a positive impact because it reduces the number of messages sent in a given time while increasing the payload sent in each request.

Some IoT platforms, such as FIWARE/Orion, provide the possibility to query (or subscribe to) a group of sensors. Implementing this feature is both time-consuming and error-prone for a developer. This is typically the middleware's role in bringing this technical feature into play. Furthermore, when a host runs multiple IoT consumer applications, the middleware is the right place to factor in and take advantage of this grouping capability.

3) Refresh time adaptation: Updates to sensor observations are made periodically by the middleware. In a publish/subscribe interaction pattern, limiting updates to a given refresh time requires a mechanism in the IoT platform to impose a minimum delay between two notifications. In a request/response mode, requests are sent from the middleware to the IoT platform at the refresh time period. In either interaction mode, the data refresh time must be defined. The lower the refresh time is, the more energy the middleware consumes. The middleware can manage this to consider a maximum energy budget set by the user and then increase the refresh time if necessary to reduce energy consumption. This trade-off between data freshness and energy consumption can be the basis for a more sophisticated adaptation algorithm. For example, freshness can be adapted to the variability/stability

of sensor observations and the resolution required by the application.

4) Interaction pattern switch: Depending on the required refresh time and the frequency of the notifications, using the publish/subscribe interaction pattern may be more or less efficient than the request/reply one. The principle of the interaction pattern switching strategy is to use the energy consumption model to decide to change the interaction pattern dynamically.

B. From Energy-efficiency Strategies to Energy Budget Management

We propose a model that quantitatively estimates the energy consumption of IoTvar according to applied energy-efficient strategies. At the middleware level, energy consumption comes from the interactions necessary to update the sensors' data required by IoT consumer applications and the treatment of data, e.g., marshaling/unmarshalling.

In this work, we formulate the estimation of energy consumption E (in Joules) as depending mainly on (i) the refresh time chosen by an application for each sensor and (ii) the grouping factor, i.e., the number of sensors with similar refresh time that may be grouped:

$$E = \sum_{i=0}^{nb_G} \frac{(C_V * nb_{V_{G_i}}) + (C_{net} * M_{netS} * M_{netI}) + C_{cpu}}{R_{G_i}}$$

with:

- nb_G is the number of groups of sensors that are managed by IoTvar (one group per similar refresh time);
- R_{G_i} is the refresh time of a group (in seconds);
- C_V is a constant energy value for each sensor inside a group. It represents the cost of the small processing that each sensor observation adds within a group (e.g., for unmarshalling purposes);
- $nb_{V_{G_i}}$ is the number of sensors inside the group;
- C_{net} is the energy value (a constant) consumed for making a request for a group;
- M_{netS} is a modifier for the network calculation that depends on the latency, package loss, and reachability of the IoT platform;
- M_{netI} is a modifier that depends on the network interface being used (cellular, WiFi, or Ethernet);
- C_{cpu} is a constant given for the CPU processing of one group of sensors in IoTvar.

Thanks to this model, the middleware may communicate energy consumption to the application through an API, thus providing to the application, and finally to the end-user, runtime energy-awareness. In addition, provided that the end-user allows reducing the refresh time of some variables, the strategies to be applied by the middleware could be adapted when the energy consumption exceeds an energy budget desired by the end user. For example, IoTvar could modify the refresh time of some variables in the objective to reduce energy consumption. This modification may also lead to a better grouping factor.

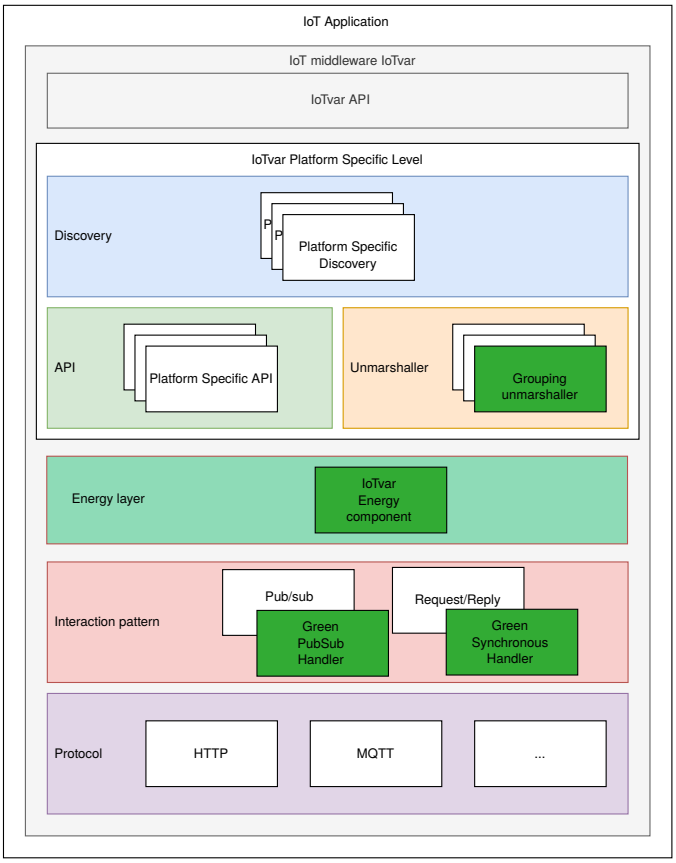

Fig. 4. Energy components in the IoTvar architecture

C. IoTvar Architecture for Energy Efficiency Strategies

The architecture of IoTvar (see Fig. 4) is built on the generic architecture presented in Section III to integrate the energy-efficient strategies and to manage the energy budget of IoT consumer applications. The architecture of IoTvar includes the *Energy layer* with a component and changes the *Grouping unmarshaller*, *Green PubSub Handler*, and *Green Synchronous Handler* components.

The *Grouping unmarshaller* is specialized to unmarshal a group of sensors, and the *Pub/Sub* and *Synchronous Handlers* have been adapted to group several sensors in one notification or request. The *Energy layer* contains the *IoTvar Energy* component detailed in Fig. 5. This component receives input from a configuration file (see Table I) defining a particular energy budget. The IoTvar energy component is decomposed into three sub-components. The *Energy-Awareness* component monitors the network status (package loss, latency, availability of the IoT platform, and type of network, e.g., Wi-Fi, Ethernet, or 3G/4G/5G. It also interprets the energy budget requirements and estimates energy consumption using the energy consumption formula E. The *Decision* component chooses the strategies to apply. Finally, the *Energy-Efficiency* component applies the decision in terms of communication protocol choice, message grouping, refresh time control, and interaction pattern switching.

TABLE I
IOTVAR CONFIGURATION FILE PROPERTIES

Property	Behavior	Default value
Status refresh period (in seconds)	Period for recalculating the status of the middleware (network and sensor conditions)	10
Maximum refresh time increase (in seconds)	This configuration imposes a limit to how much IoTvar will increase the refresh time. When IoTvar increases the refresh time the QoS will be lowered, thus this configuration will limit this degradation of QoS up to a maximum (by refresh period).	10
Use energy model	This property indicates if IoTvar should use the energy model to calculate the energy being consumed considering the sensors declared in the code. If it is `false`, IoTvar does not report the estimation of the energy being used.	`false`
Use energy budget	If the energy model is `true` and this property is `true`, then IoTvar will start to consider the budget provided in the property *Energy Budget*.	`false`
Energy budget (in Joules for five minutes)	This value is used by IoTvar to lower the amount of energy calculated until reaching the desired budget. This value is given in Joules and is the budget for five minutes.	400

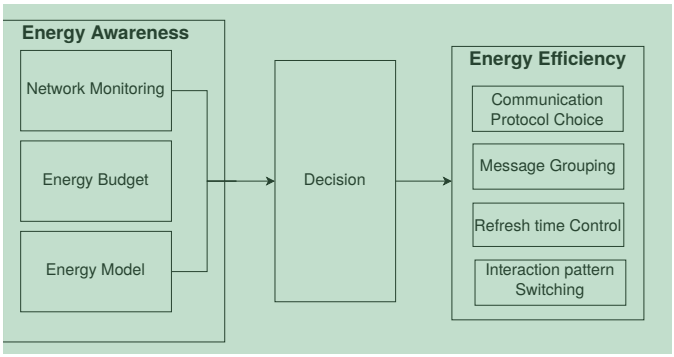

Fig. 5. IoTvar energy component

V. EVALUATION

We implemented the proposed architecture for interactions with the *FIWARE/Orion* IoT platform. It handles the *message grouping* and *refresh time adaptation* strategies and *energy budget management* for the *request/reply* interaction pattern.

This section reports a quantitative evaluation of the IoTvar middleware using the *message grouping* strategy. This evaluation comprised a performance assessment that considers the same application (i) written without IoTvar, i.e., directly accessing the IoT platform, (ii) written with IoTvar without energy-efficiency strategies, and (iii) written with IoTvar using the energy strategies. The evaluation aimed at measuring the impacts of the strategies in terms of CPU usage and energy consumption.

The IoT consumer application used in the performance evaluation receives and displays the data that is returned from a temperature sensor around the Eiffel Tower in the console. The tests done with this application varied the number of sensors observed by the application from 25 to 200 in a step of 25 sensors (25, 50, ..., 200), with a refreshing time of one second for each sensor, and having a small local history of the ten latest values for each sensor. The payload of the data of one sensor is around 117 bytes, while it is around 10,282 bytes for 200 sensors.

We collected the performance measurements (CPU usage and energy consumption) by wrapping the energy-efficient synchronous handler method. This is implemented using AspectJ [47], which allows collecting performance measures around the most critical methods of the IoTvar structure. This enabled us to measure the CPU and memory usage only for the data processing and communication methods without modifying the middleware code.

Tools such as RAPL [48] currently can perform energy measurements but are still limited to CPU and RAM energy consumption. So, no energy measurement software includes the consumption of the network interface in the energy consumption measurements. Consequently, we used a Yocto wattmeter [49] to measure the energy consumption of the whole computer. The wattmeter counter is reset at the beginning of the tests, and the total energy consumed is retrieved at the end and saved in a file along with CPU consumption. Because the wattmeter measures the consumption of the whole computer, we had another test to get the energy consumption of the computer running without the application (idle consumption) to isolate its consumption.

For quantitative analysis purposes, we performed hypothesis testing. Our null hypotheses is that there are no statistically significant differences regarding CPU and energy consumption when using IoTvar with energy efficiency strategies and not using them. To decide the test to use, we first performed the Shapiro-Wilk test [50] to verify if a sample follows a normal distribution. For the significance level $\alpha = 0.05$, we noticed that the values did not follow a normal distribution, thus leading us to perform a non-parametric hypothesis test.

We used the Mann Whitney's U-test to check for a statistically significant difference between the analyzed samples (IoTvar with and without energy-efficiency strategies). Following these results, we ran an effect-size statistical analysis over the testing data using the A-index by Delaney and Vargha interpreted with Hess and Kromrey magnitude levels (negligible, small, medium, and large).

A. Setup

Fig. 6 illustrates the setup of the tests. A server computer with i7-4770K CPU at 3,9GHz, 16 GB of volatile memory, and

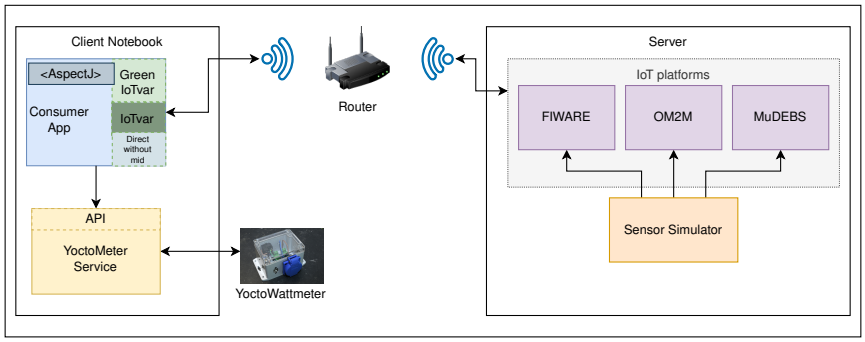

Fig. 6. Experimental setup

Ubuntu 16.04 executed the IoT platform and data producers, which simulates IoT sensors that send data through the platform. The computer running the client application consuming IoT data had an i7-8665U CPU at 1.90GHz, 32GB of volatile memory, and Debian 9. Furthermore, we plugged the client computer into a YoctoPuce YoctoWatt wattmeter to collect energy-related measures. The client application communicated with the server through a locally isolated Wi-Fi network.

Data for each measure (CPU and energy) are collected for 30 executions of five-minute testing. The first minute of the test is the warm-up phase, which was not recorded: it ensures that the class loading is complete in the Java Virtual Machine (JVM) to avoid interference in the results [51]. The last four minutes constituted the effective run phase. When running the tests, the IoT application created the number of sensors for the specific test and called the FIWARE/Orion platform to get the sensor observations.

B. Results

Fig. 7 and Fig. 8 show the CPU usage and energy consumption of IoTvar upon varying the number of sensors. Table II provides the numbers. For 25 sensors, the *message grouping* strategy lowers CPU usage of IoTvar from 35 seconds to less than one second and energy consumption from 1,150 to 1,000 Joules. For 200 sensors, the increase in CPU usage is more than 99%, and the increase in energy consumption is more than 45%. As a result, using IoTvar (compared to direct access to the IoT platform) has a cost, but message grouping reduces CPU usage and energy consumption on the application.

Concerning the statistical analysis of the data, we obtained that our null hypotheses were rejected for all the scenarios (p-value $< \alpha = 0.05$), thus leading us to conclude that not using the energy efficiency strategies for IoT middleware inside IoTvar brings an impact on these resources. The p-value using the A-index test which interprets the values prove a *large* effect for all the scenarios. Therefore, the impact of not using energy efficiency strategies causes a large impact on the resource consumption of the IoT application using an IoT middleware such as IoTvar.

C. IoTvar Energy Model Calibration

Section IV-B presented the energy model that IoTvar implements. Concerning energy model calibration, we initiated an exploratory study. We have performed a campaign of measures for a given end-user device to calibrate the constants. The calibration shall be done for each new type of computer.

The model constants and modifiers were obtained by calibration: several experiments until the model could estimate the energy consumption without deviating from the values given by the power meter tests. Fig. 9 shows the results of the tests

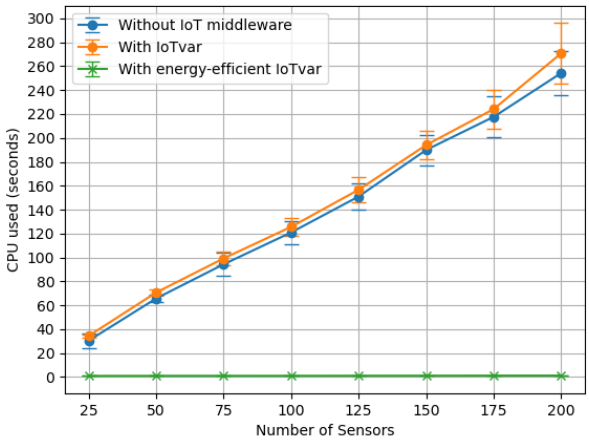

Fig. 7. IoTvar CPU usage using energy efficiency strategies

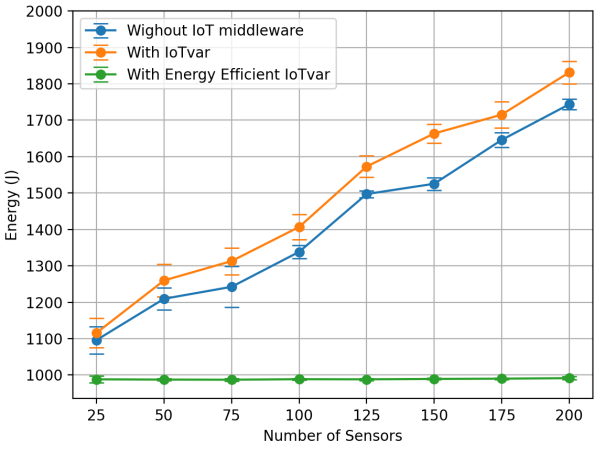

Fig. 8. IoTvar energy consumption using energy efficiency strategies

TABLE II
INCREASE OF JOULES USING IOTVAR WITHOUT ENERGY EFFICIENCY STRATEGIES

Metric	Number of IoTvar sensors							
	25	**50**	**75**	**100**	**125**	**150**	**175**	**200**
CPU usage	97.36%	98.68%	99.04%	99.23%	99.36%	99.46%	99.5%	99.57%
Energy consumption	11.44%	21.62%	24.82%	29.73%	37.17%	40.53%	42.29%	45.87%

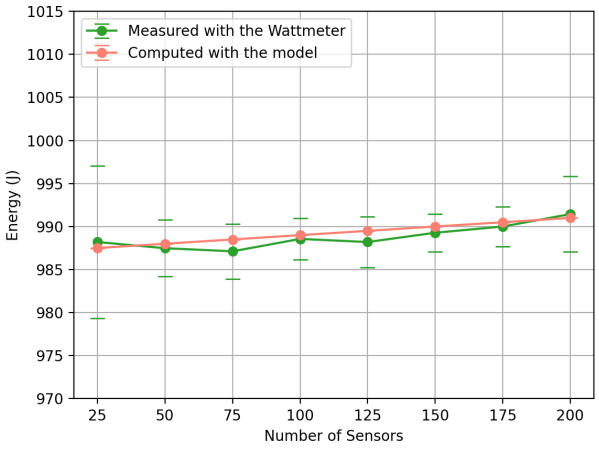

Fig. 9. IoTvar energy consumption using the energy model

TABLE III
VALUES OF CONSTANTS AND MODIFIERS OF THE ENERGY MODEL

Variable	C_V	C_{net}	M_{netS}	M_{netI}	C_{cpu}
Value	0.02	90.0	1.0	10.0	87.0

with the Wattmeter along with an estimation of the energy consumption for five minutes.

D. Discussion

The results of the tests have shown that the *message grouping* strategy significantly lowers the amount of energy consumed by IoTvar, especially when the refresh time of sensors enables a good level of grouping. Moreover, the implemented energy-awareness enables IoTvar to have further knowledge of the energy context of the application, e.g., network status, sensor energy consumption, etc. This energy context information is used to provide enough information to build an energy model to estimate the energy consumption of the middleware.

Nonetheless, there are still some limitations to the results. First, we run the tests using only the Wi-Fi interface, leaving a gap to other types of communication, such as 3G and 4G, which are popular in IoT consumer applications. Second, the performance evaluation is specific to a computer hardware and software architecture. Third, the model calibration is specific to a computer and has to be calibrated for each new computer IoTvar is deployed on. Fourth, the performance evaluation was made to be comparable with the tests of the IoTvar middleware without the energy efficiency strategies, which limits the statistical analysis to a smaller number of sensors than the IoTvar grouping strategy can support.

IoTvar supports multiple platforms, each with its own set of functionalities and specific APIs. Implementing the message grouping strategy was made possible by the availability of this capability in the FIWARE/Orion platform. Such implementation in other platforms should be the first step towards energy efficiency in IoT consumer applications.

VI. CONLUSIONS

The world is still experiencing a greater need for energy production caused by the evolution of technology and the expansion of many technological areas, such as the digital sector. In particular, energy usage in the IoT domain presents a significant challenge as the number of devices has increased over the years. Consequently, many IoT applications are being developed and must interact with an IoT system's heterogeneous components. Those IoT applications need to be energy-efficient.

Designing IoT applications that include energy efficiency strategies is complex as such applications must deal with heterogeneity and massive data. This paper showed how IoT middleware could help to reduce this complexity while introducing energy efficiency and energy-awareness into IoT applications. We have presented (i) the architecture of the IoTvar middleware, which integrates energy efficiency strategies and energy-awareness mechanisms, and (ii) how we have implemented the message grouping and refresh time adaptation strategies and the management of an energy budget. We evaluated the message grouping strategy through an experiment using the IoTvar middleware, and the results indicate that this strategy does lower energy consumption significantly. The results reached a maximum percentage change of around 60% less energy consumption when using the strategy. We also proposed other strategies for switching the interaction pattern or choosing the communication protocol. We need to evaluate them to confirm their efficiency, but we know they are expected to further increase energy efficiency in IoT middleware.

Our evaluation of IoTvar considered only the IoT consumer application device. Future work intends to support distributed applications and enable IoT middleware-level cooperation to provide energy-awareness in a multi-component system. This can be done by implementing communication between the different IoT applications through the IoT middleware running inside the application or an external middleware that will coordinate the communication and exchange of energy information. This information is related to network conditions,

CPU usage, energy budget, and other resources relevant to the energy efficiency of the IoT middleware. Furthermore, evaluating and showing the impacts of such energy features is imperative.

The awareness provided by the IoTvar middleware is automatically enabled with no input from the user. Shah et al. [52] explain that "the maintenance of the balance between the comfort index and power consumption is also a significant issue." As awareness influences energy efficiency, which impacts the quality of the applications (e.g., freshness), it should not be transparent. The users should at least be in the loop and be informed of what is happening in the application. The energy-awareness of the user may have a positive impact by reducing energy consumption in the future, thus contributing to IoT sustainability.

REFERENCES

[1] Shift project, "Environmental impact of digital: 5-year trends and 5G governance," Shift project, Tech. Rep., March 2021.

[2] T. M. Attia, "Challenges and opportunities in the future applications of iot technology," in *2nd Europe - Middle East - North African Regional Conference of the International Telecommunications Society (ITS)*. Calgary: International Telecommunications Society (ITS), 2019.

[3] T. Okrasinski, T. Fleming, A. Moore, G. Gines, S. Kelly, S. Moore, and M. Shackleton, "Smarter 2030," Global eSustainability Initiative (GeSI), Brussels, Belgium, Tech. Rep., 2015. [Online]. Available: https://smarter2030.gesi.org/

[4] F. K. Shaikh, S. Zeadally, and E. Exposito, "Enabling technologies for green internet of things," *IEEE Systems Journal*, vol. 11, no. 2, pp. 983–994, 2017.

[5] D.-J. Munoz, J. A. Montenegro, M. Pinto, and L. Fuentes, "Energy-aware environments for the development of green applications for cyber–physical systems," *Future Generation Computer Systems*, vol. 91, pp. 536 – 554, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18307295

[6] S. P. directed by Hugues Ferreboeuf, "Lean ict – towards digital sobriety," https://theshiftproject.org/wp-content/uploads/2019/03/Lean-ICT-Report_The-Shift-Project_2019.pdf, 2019.

[7] P. Patel and D. Cassou, "Enabling high-level application development for the internet of things," *Journal of Systems and Software*, vol. 103, pp. 62–84, 2015.

[8] C. Pang, A. Hindle, B. Adams, and A. E. Hassan, "What do programmers know about software energy consumption?" *IEEE Software*, vol. 33, no. 03, pp. 83–89, may 2016.

[9] M. A. Chaqfeh and N. Mohamed, "Challenges in middleware solutions for the internet of things," in *2012 International Conference on Collaboration Technologies and Systems (CTS)*, 2012, pp. 21–26.

[10] G. S. Blair, D. C. Schmidt, and C. Taconet, "Middleware for internet distribution in the context of cloud computing and the internet of things - editorial introduction," *Ann. des Télécommunications*, vol. 71, no. 3-4, pp. 87–92, 2016.

[11] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "Iot middleware: A survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.

[12] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, 2016.

[13] G. Bouloukakis, N. Georgantas, P. Ntumba, and V. Issarny, "Automated synthesis of mediators for middleware-layer protocol interoperability in the iot," *Future Gener. Comput. Syst.*, vol. 101, pp. 1271–1294, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2019.05.064

[14] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A gap analysis of Internet-of-Things platforms," *Computer Communications*, vol. 89-90, pp. 5–16, Sep. 2016.

[15] P. P. Ray, "A survey of iot cloud platforms," *Future Computing and Informatics Journal*, vol. 1, no. 1, pp. 35 – 46, 2016.

[16] A. Noureddine, R. Rouvoy, and L. Seinturier, "A review of energy measurement approaches," *SIGOPS Oper. Syst. Rev.*, vol. 47, no. 3, p. 42–49, nov 2013. [Online]. Available: https://doi.org/10.1145/2553070.2553077

[17] M. G. Hassan, R. Hirst, C. Siemieniuch, and A. Zobaa, "The impact of energy awareness on energy efficiency," *International Journal of Sustainable Engineering*, vol. 2, no. 4, pp. 284–297, 2009. [Online]. Available: https://doi.org/10.1080/19397030903121968

[18] A. Al-Roubaiey, T. Sheltami, A. Mahmoud, and A. Yasar, "Eatdds: Energy-aware middleware for wireless sensor and actuator networks," *Future Generation Computer Systems*, vol. 96, pp. 196–206, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18331777

[19] G. Pardo-Castellote, "Omg data-distribution service: Architectural overview," in *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, 2003, pp. 200–206.

[20] S. Akkermans, R. Bachiller, N. Matthys, W. Joosen, D. Hughes, and M. Vučinić, "Towards efficient publish-subscribe middleware in the iot with ipv6 multicast," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[21] S. Padhy, H.-Y. Chang, T.-F. Hou, J. Chou, C.-T. King, and C.-H. Hsu, "A middleware solution for optimal sensor management of iot applications on lte devices," in *Quality, Reliability, Security and Robustness in Heterogeneous Networks*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, J.-H. Lee and S. Pack, Eds. Switzerland: Springer International Publishing, 2017, vol. 199, pp. 283–292.

[22] C. Sarkar, V. S. Rao, R. Venkatesha Prasad, S. N. Das, S. Misra, and A. Vasilakos, "Vsf: An energy-efficient sensing framework using virtual sensors," *IEEE Sensors Journal*, vol. 16, no. 12, pp. 5046–5059, 2016.

[23] E. A. de Oliveira, F. Delicato, and M. Mattoso, "An energy-aware data cleaning workflow for real-time stream processing in the internet of things," in *Anais do IV Workshop de Computação Urbana*. Porto Alegre, RS, Brasil: SBC, 2020, pp. 71–83. [Online]. Available: https://sol.sbc.org.br/index.php/courb/article/view/12354

[24] W. Li, F. C. Delicato, P. F. Pires, Y. C. Lee, A. Y. Zomaya, C. Miceli, and L. Pirmez, "Efficient allocation of resources in multiple heterogeneous wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 74, no. 1, pp. 1775–1788, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731513002104

[25] I. Podnar Zarko, A. Antonic, and K. Pripužic, "Publish/subscribe middleware for energy-efficient mobile crowdsensing," in *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, ser. UbiComp '13 Adjunct. New York, NY, USA: Association for Computing Machinery, 2013, p. 1099–1110. [Online]. Available: https://doi.org/10.1145/2494091.2499577

[26] B. M. Al-Madani and E. Q. Shahra, "An energy aware plateform for iot indoor tracking based on rtps," *Procedia Computer Science*, vol. 130, pp. 188–195, 2018, the 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705091830379X

[27] G. Ramachandran, J. Proença, W. Daniels, M. Pickavet, D. St, D. Staessens, C. Huygens, W. Joosen, and D. Hughes, "Hitch hiker 2.0: a binding model with flexible data aggregation for the internet-of-things," *Journal of Internet Services and Applications*, vol. 7, 04 2016.

[28] M. Aazam, S. U. Islam, S. T. Lone, and A. Abbas, "Cloud of things (cot): Cloud-fog-iot task offloading for sustainable internet of things," *IEEE Transactions on Sustainable Computing*, pp. 1–1, 2020.

[29] S. Pasricha, "Overcoming energy and reliability challenges for iot and mobile devices with data analytics," in *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*, 2018, pp. 238–243.

[30] C. Cecchinel, F. Fouquet, S. Mosser, and P. Collet, "Leveraging live machine learning and deep sleep to support a self-adaptive efficient configuration of battery powered sensors," *Future Generation Computer Systems*, vol. 92, pp. 225–240, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18305740

[31] Z. Huang, K.-J. Lin, and L. Han, "An energy sentient methodology for sensor mapping and selection in iot systems," in *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, 2014, pp. 1436–1441.

[32] Z. Song, M. Le, Y.-W. Kwon, and E. Tilevich, "Extemporaneous micro-mobile service execution without code sharing," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2017, pp. 181–186.

[33] M. Aazam, S. Zeadally, and E. F. Flushing, "Task offloading in edge computing for machine learning-based smart healthcare," *Computer Networks*, vol. 191, p. 108019, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621001298

[34] T. Kalbarczyk and C. Julien, "Omni: An application framework for seamless device-to-device interaction in the wild," in *Proceedings of the 19th International Middleware Conference*, ser. Middleware '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 161–173. [Online]. Available: https://doi.org/10.1145/3274808.3274821

[35] S. Shekhar, A. Chhokra, H. Sun, A. Gokhale, A. Dubey, and X. Koutsoukos, "Urmila: A performance and mobility-aware fog/edge resource management middleware," in *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*, 2019, pp. 118–125.

[36] S. Jeon and I. Jung, "Mint: Middleware for cooperative interaction of things," *Sensors*, vol. 17, no. 6, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/6/1452

[37] Y. Banouar, T. Monteil, and C. Chassot, "Analytical model for adaptive qos management at the middleware level in iot," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, 2017, pp. 1201–1208.

[38] A. Mukherjee, N. Dey, and D. De, "Edgedrone: Qos aware mqtt middleware for mobile edge computing in opportunistic internet of drone things," *Computer Communications*, vol. 152, pp. 93 – 108, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366419315750

[39] FIWARE, "What is fiware?" https://www.fiware.org/.

[40] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration," vol. 51, no. 6. New York, NY, USA: Association for Computing Machinery, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3292674

[41] H. Nielsen, J. Mogul, L. M. Masinter, R. T. Fielding, J. Gettys, P. J. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, Jun. 1999. [Online]. Available: https://rfc-editor.org/rfc/rfc2616.txt

[42] OASIS, "MQTT version 3.1.1 plus errata 01," https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf, 12 2015, accessed on 21-05-2021.

[43] A. Reeve, "Chapter 12 - data integration patterns," in *Managing Data in Motion*, ser. MK Series on Business Intelligence, A. Reeve, Ed. Boston: Morgan Kaufmann, 2013, pp. 79–85.

[44] P. V. Borges, C. Taconet, S. Chabridon, D. Conan, E. Cavalcante, and T. Batista, "Taming internet of things application development with the iotvar middleware," *ACM Trans. Internet Technol.*, feb 2023. [Online]. Available: https://doi.org/10.1145/3586010

[45] "IoTvar git repository," https://gitlabev.imtbs-tsp.eu/m4iot/iotvar/, accessed: 2023-04-17.

[46] R. Canek, P. Borges, and C. Taconet, "Analysis of the Impact of Interaction Patterns and IoT Protocols on Energy Consumption of IoT Consumer Applications," in *DAIS 2022: 17th International Conference on Distributed Applications and Interoperable Systems*, ser. Lecture Notes in Computer Science. Lucca, Italy: Springer, Jun. 2022, pp. 1–17. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03710735

[47] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold, "An overview of aspectj," in *Proceedings of the 15th European Conference on Object-Oriented Programming*, ser. ECOOP '01. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 327–353.

[48] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan, "Power-management architecture of the intel microarchitecture code-named sandy bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, 2012.

[49] YoctoPuce, "Who are we?" https://www.yoctopuce.com/EN/aboutus.php, accessed on 17-10-2021.

[50] S. S. Shapiro and M. Wilk, "An analysis of variance test for normality," *Biometrika*, vol. 52, no. 3-4, pp. 591–611, 1965.

[51] A. Georges, D. Buytaert, and L. Eeckhout, "Statistically rigorous java performance evaluation," *SIGPLAN Not.*, vol. 42, no. 10, p. 57–76, oct 2007. [Online]. Available: https://doi.org/10.1145/1297105.1297033

[52] A. Shah, H. Nasir, M. Fayaz, A. Lajis, and A. Shah, "A review on energy consumption optimization techniques in iot based smart building environments," *Information*, vol. 10, no. 3, p. 108, Mar 2019. [Online]. Available: http://dx.doi.org/10.3390/info10030108