Analysis of the impact of interaction patterns and IoT protocols on energy consumption of IoT consumer applications

Rodrigo Canek, Pedro Borges, Chantal Taconet

Analysis of the Impact of Interaction Patterns and IoT Protocols on Energy Consumption of IoT Consumer Applications

Rodrigo Canek, Pedro Borges, and Chantal Taconet

SAMOVAR, Télécom SudParis
Institut Polytechnique de Paris, France

Abstract. Nowadays, it is estimated that half the connected devices are related to the Internet of Things (IoT). The IoT paradigm contributes to the increase of the Information Technology energy demand. The energy demand is due on one side to the huge number of IoT devices, and on the other side to the plethora of IoT end user applications consuming data produced by those devices. However, taking into account energy consumption in the development of such applications, consuming data produced by IoT devices is still challenging. There is a lack of knowledge on what are the best practises to develop green IoT applications. The work presented in this paper aims to raise the awareness of application designers concerning the impact of the choice of IoT protocols and interaction patterns on the energy consumption of the applications. For this purpose, we have experimentally analysed the energy consumption of HTTP and MQTT, which are two of the most popular, mature and stable protocols for IoT consumer applications. For the HTTP protocol, we have studied both the publish-subscribe and the request-reply interaction patterns. For MQTT, we have studied the publish-subscribe interaction pattern with the three available Quality of Services. We also examine the impact of message payload on energy consumption. The results show that the publish/subscribe interaction pattern has lower energy consumption (around 92% less) than the synchronous interaction pattern and HTTP consumes 20% more energy than the MQTT protocol for the publish/subscribe interaction pattern. Finally, we show that the payload has a low impact on energy consumption, having a 9% overhead on payloads ranging from 24 to 3120 bytes

Keywords: Middleware · Internet of Things applications · IoT protocols · Interaction patterns · IoT Platforms · Energy Consumption · Green IT

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

1 Introduction

It is estimated that the number of Internet-connected devices will be 29.3 billion in 2023, among them 50% will be IoT devices and 23% smartphones [4]. As the number of IoT systems is one of the main causes of the growth of IT energy consumption [9], handling IoT systems energy-efficiency is from now on a first class imperative [20].

Because of their limited battery lifetime, energy-efficiency has firstly been taken into account in the design of software deployed on IoT devices [12]. However, reducing software energy consumption should not be limited to IoT devices. It has been estimated that around 67 zettabytes of data were generated by IoT devices in 2020 [9]. Part of this volume of data has been consumed by IoT applications. Thus, carefully designing interactions between IoT applications and IoT systems with a energy-efficiency concern is also essential.

Developers still lack knowledge about software energy consumption [18]. Measuring experimentally software and hardware energy consumption participates in providing this knowledge. Several approaches may be used for energy measurement [15]. As most of the libraries that measure energy consumption at the process level only consider the impact of CPU and memory (e.g., [1]), measuring the cost of the interaction between distributed components is still a difficult task. In this study, we propose to experimentally measure the cost of the interactions between IoT consumer applications and IoT systems. These measures will guide IoT consumer application developers in their design choices in terms of energy consumption.

In our experiments, we consider consumer IoT applications connected with a WiFi (802.11n) interface and that use MQTT and HTTP protocols. Those technologies are commonly used for IoT consuming applications placed in different networks from the connected object ones, whereas other networks (e.g. Bluetooth) and protocols (e.g. Zigbee, COAP) are used on the connected object side. The conducted analysis answers the following questions: What is the energy consumption impact of (RQ1) the publish/subscribe interaction pattern vs request/reply, (RQ2) the HTTP protocol vs MQTT for the publish subscribe interaction pattern, (RQ3) the Quality of service (QoS) level (in the case of MQTT) and (RQ4) the size of the payload. From the analysis of the results of the experiments, we propose guidelines to help developers to build low energy consuming IoT applications.

The rest of this paper is structured as follows. Section 2 provides important background concepts on IoT architectures, interaction patterns and IoT protocols. Section 3 investigates the related works concerning the energy consumption of the studied IoT protocols. Section 4 shows the setup of the hardware and software for the experiments and discusses the threats to validity. Section 5 presents the results of the experiments, analyses the results according to the four introduced research questions and provides guidelines for IoT consuming application developers. Finally, Section 6 draws conclusions and perspectives.

2 Consuming IoT applications: architecture, interaction patterns and Protocols

This section introduces the main concepts that will be used throughout the article, concerning IoT distributed architecture and IoT protocols.

2.1 Distributed IoT Architecture

Figure 1 presents a classical IoT system architecture. According to the ISO-IEC IoT reference architecture [10], an IoT system consists of (1) IoT devices (sensors and actuators), (2) end user applications that may consume sensor data (called *IoT consumer applications* in this paper), (3) IoT platforms and IoT gateways, standardized intermediates for interacting with IoT devices that deal with the high degree of hardware and software heterogeneity in IoT environments.

The usage of IoT platforms to support IoT systems is a recent trend: they provide services to deploy and run applications on top of a hardware and/or software suite in different application domains [13]. Their role is to decouple producers from consumers by providing an intermediary layer. Among the platforms, we can cite FIWARE/Orion [6], an IoT platform supported by the European Community, and OneM2M [17] a Machine-2-Machine standard.

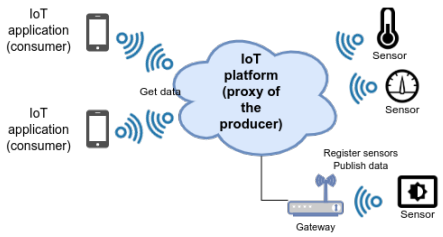

Fig. 1. IoT distributed architecture

2.2 Interaction Patterns

An interaction pattern, or a Message Exchange Pattern [7], defines the structure of the interactions between the two sides engaged in a communication. In the IoT, two high level interaction patterns are commonly used between the consumers of data and the providers of data [3]: Publish-Subscribe and Request-Reply.

Request-Reply As shown in Figure 2, the consumer (i.e. IoT consumer application), sends a request message to the producer (e.g., the IoT platform). The consumer is waiting for a reply from the producer (or a timeout). The producer receives and processes the request and sends the consumer a reply message with a given payload.

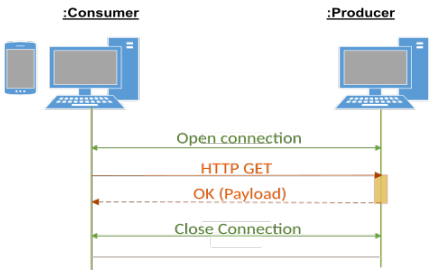

Fig. 2. Request-Reply Pattern

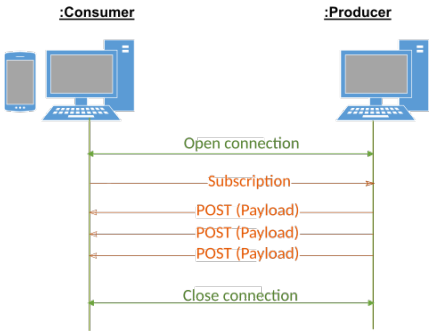

Fig. 3. Publish-Subscribe Pattern

Publish-Subscribe As shown in Figure 3 the consumer defines, with a subscription, what kind of data it is interested in. The consumer is notified whenever there is a message matching the subscription.

2.3 IoT Protocols

We consider the most mature and stable protocols for the interaction between IoT consumer applications and IoT systems [5]: MQTT [16] and HTTP [14].

HTTP is supported by all the IoT platforms. When it comes to using the protocol, it is mostly used in its Request/Reply interaction pattern. However, the FIWARE platform also uses HTTP for the publish-subscribe interaction pattern, where the client is an open listener and the server posts available data (as shown in Figure 3).

MQTT is a lightweight protocol with the publish/subscribe interaction pattern. IoT platforms host MQTT brokers that receive publications from connected objects or gateways. Brokers are responsible to filter incoming messages and distribute them properly according to the message topics. MQTT implements three different models of message exchange known as Quality of Service, where the delivery with QoS 0 being at most once, QoS 1 being at least once, and QoS 2 being exactly once.

In our experiments, MQTT and HTTP are above TCP/IP. TCP handles the connections between the remote processes and reassembles the data in the correct order.The Internet Protocol (IP) [19] is responsible for routing data. It provides fragmentation and reassembly of long datagrams, if necessary according to the **Maximum Transmission Unit** (MTU). We investigate in our experiments whether having a message payload below or above the MTU impacts the energy consumption.

3 Related Work

In this section, we present an analysis of the related works concerning the efficiency of IoT protocols used by IoT consumer applications. We have selected

research papers that include energy consumption measures in the evaluation of HTTP and/or MQTT. We have to mention that, to the best of our knowledge, the number of papers on this subject is low, we only found 4 papers and the measures do not isolate the consumption on the consumer side. Furthermore, none of them study the impact of the interaction pattern.

A synthesis of the study is presented in Table 1. For those related works, the following points have been analyzed. Since our objective is to study the consumer side of an IoT architecture, we indicate whether the study is conducted on the producer side (P), on the consumer side (C), or on both sides. We mention which IoT protocols were compared in the work. We also indicate the experimental conditions: the device where the measure was conducted and the type of network. The last aspect concerns type of the evaluation, analytical or experimental energy evaluation and, if experimental, the tool they have used to measure the energy consumption.

Bandyopadhyay and Bhattacharyya present an analysis of MQTT and CoAP [2]. They examine the resource usage including energy consumption according to the message size and the packet loss ratio. The energy consumption of the most reliable configurations was measured on a Wide Area Network: CoAP and MQTT with QoS 2. They show that with a perfect network without any loss, MQTT with QoS2 is more than ten times more consuming than CoAP. Concerning energy efficiency, they only study MQTT with QoS2. They do not define whether the measures are done on the producer or/and consumer side, which makes it difficult to know which side of the architecture was studied. They also do not mention how the energy consumption was measured.

Toldinas et al. perform a dedicated study of MQTT QoS levels and their energy consumption [21]. They use a ESP-WROOM-02 hardware device connected to the network through Wifi 802.11 and acting both as producer and consumer. For each level of QoS, the remaining battery voltage level was measured using a digital multimeter as an indicator of energy consumption. This study provides a good indication of the percentage increase in energy consumption for each level of the QoS compared to the previous one. However, it does not allow effective energy-consumption conclusions to be drawn about the behavior of a consumer or a producer as the same device is used for both tasks.

Hofer and Pawaska studied the impact of MQTT and HTTP protocols on CPU, RAM, and energy consumption [8]. The device used is a Raspberry Pi connected by Ethernet, that acts both as a producer and a consumer, as a consequence they can not isolate the energy consumption on the consumer side. They do not mention what QoS was used for MQTT. For the energy evaluation, the authors studied the Ampere per second in the device using an oscilloscope. The study proved that MQTT outperformed HTTP RESTful in terms of data overhead which is the amount of extra data needed to be sent to a client (e.g HTTP Headers, MQTT headers, etc), a nearly four times higher throughput. Furthermore, MQTT also had lower resource consumption and significantly lower energy consumption. As HTTP is used with the synchronous interaction pattern and MQTT is used with the publish/subscribe interaction pattern, it is not

possible to isolate the impact of the interaction pattern from the impact of the protocol.

Joshi et al. presented a comparison in terms of protocol impact on throughput and battery consumption between MQTT (QoS not specified), CoAP and HTTP RESTful [11]. The device used was a Raspberry Pi, which acted only as a producer. For energy consumption, the percentage of battery consumption per hour was taken as a reference. However, it was not mentioned how it was calculated. The conclusions are: (i) HTTP consumes more energy than MQTT, and (ii) with the same amount of battery it is possible to send 100 times more messages with MQTT compared to HTTP. Although the work was not dedicated to the study of energy consumption, it lacks details on how the measures were implemented as well as the conditions of the experiment (e.g. network type).

Ref	Network	P/C	MQTT QoS			HTTP		Evaluation
			0	1	2	Sync	Pub/sub	
[2]	WAN em.	?	×	×	√	×	×	?
[21]	Wifi	P+C	√	√	√	×	×	simulation
[8]	Ethernet	P+C	?	?	?	√	×	oscilloscope
[11]	Wifi	P	√	×	×	√	×	calculated
this	Wifi	C	√	√	√	√	√	wattmeter

Table 1. Synthesis of the related work

Compared to the presented works, the experiments we have conducted allow to isolate the cost of the consumer application side of an IoT architecture. Furthermore as we test the HTTP protocol with the request/reply and publish/subscribe interaction patterns, we are able to study the impact of the interaction pattern separately from the impact of the protocol.

4 Experimental methodology

This section presents the methodology used in the experiments. We present the experimental conditions in terms of computer, network, energy measurement tool, software and algorithms in Section 4.1. We continue by presenting the process allowing to isolate the energy consumption of the communication part in Section 4.2. Then we present the experimental plan in Section 4.3. Finally, we discuss the threats that may affect the validity of the experimentation in Section 4.4.

4.1 Experimental setup

Computers and network As shown in Figure 4, three computers were used to perform the experiments. 1) The **Consumer Computer** used for running the

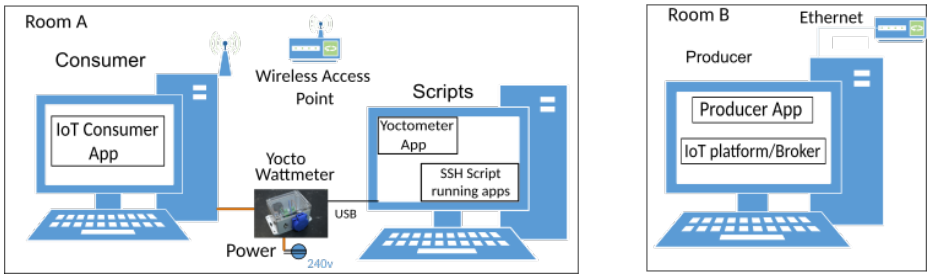

Fig. 4. Experimental setup

consumer application. A wattmeter measures its energy consumption. The Consumer is connected to the network through a Wifi interface. The characteristics of this computer are the following : Dell Latitude E6320 v:01 with 5.68GiB of RAM, a Broadcom (BCM4313 802.11bgn) Wireless Network Adapter driver and Ubuntu 20.10 Operating system. Furthermore, the battery was fully charged and the computer was always plugged to the electricity. 2) The **Producer Computer** used for simulating an IoT platform. It runs a process that produces data. It is a fixed computer connected to the Internet through an Ethernet interface. 3) The **Script Computer** was used (i) running the scripts responsible for starting all applications on the client and server computers, and (ii) for reading the energy consumption measures.

For **MQTT**, we use the Mosquitto broker version 3. The producer and the consumer were developed using the open-source Eclipse Paho library for Java. For **HTTP Request/Reply**, the consumer use HTTP/1.1 with the *java.net.http.HttpClient* Java library. For **HTTP Publish/Subscribe** we also use HTTP/1.1 and the consumer includes an Undertow Server to receive HTTP publications. We have to mention that in a real scenario, the consumer application does not choose the version of the HTTP protocol used by the server neither the configuration of the server concerning the connection management. In this context, the usage of HTTP/1.1 is widely supported by servers and clients whereas other versions such as HTTP//2 are still less common.

Energy consumption measurements Currently there is no library that includes the consumption of the network interface in the energy consumption measurements. Some libraries such as RAPL are able to make energy measurements, but are limited to the CPU and memory consumption. In the case for communications over the internet, the hardware that need to have its energy-consumption measured is the network interface, making it difficult the usage of RAPL in our case. As a consequence, it was decided to use a Yocto wattmeter [22] to measure the energy consumption of the consuming application.

As shown in Figure 4, the Yocto-wattmeter is located between the consumer computer power cable and the wall power outlet. The Yocto-wattmeter is con-

nected to the energy measurement computer via a USB cable. It uses the Yocto software API to read energy consumption measures.

Algorithms We provide below the algorithms used in the experiments.

On the consumer side, Algorithm 1 is used for the Request/Reply interaction pattern, it takes as an input parameter the period between two requests. Algorithm 2 is used for the publish/subscribe interaction pattern and registers the handler to be called on the reception of a notification and runs forever.

Algorithm 1: Consumer Request/Reply

```
Main(period)
begin
    producer ←
     httpInitialisation(URI)
    while true do
        value ←
         producer.getValue()
        sleep(period)
    end
end
```

Algorithm 2: Consumer publish/subscribe

```
Main(void)
begin
    server ←
     initializeServer(URI, handler)
end
handler(receiver)
begin
    value ←
     receiver.getValue()
end
```

On the producer side, the Algorithm 3 is used to simulate IoT data publications. It takes as input the period between two publications and the size of the payload to be sent periodically.

Finally a script runs on the measuring computer. It takes as input the period between two publications, the payload size and the duration of the experiment.The script starts the consumer and the producer, then sleeps for one minute for initialization and consumer warmup purposes. Then, the energy meter on the wattmeter is reset and is ready to start gathering new energy measures for the duration of the experiment. Finally, the script reads the consumed energy on the consumer application from the wattmeter and stops the producer and the consumer.

4.2 Process to isolate the communication energy consumption

Using a wattmeter has the following disadvantage: There is no isolation of the application or any particular process in the measurement, as the wattmeter measures the energy consumption of the computer as a whole. For a proper measurement of the impact of an application, it is necessary to make two measures: (1) the measure of the energy consumption without the application and (2) the measure of the energy consumption with the application.

In Figure 5, we present, for the 5 families of experiments in Table 2, the following measures of energy consumption of the consumer computer:

Algorithm 3: Producer

```
Main(period, payload)
begin
    dest=initializeServer(URI)
    while true do
        dest.send(payload)
        sleep(period)
    end
end
```

- $M_{idle+jvm}$: we start the Consumer computer with the consumer application but without any interaction with the producer application (blue + orange on Figure 5)
- $M_{idle+jvm+interactions}$: we start the Consumer computer with the full consumer application (blue + orange +green on Figure 5)

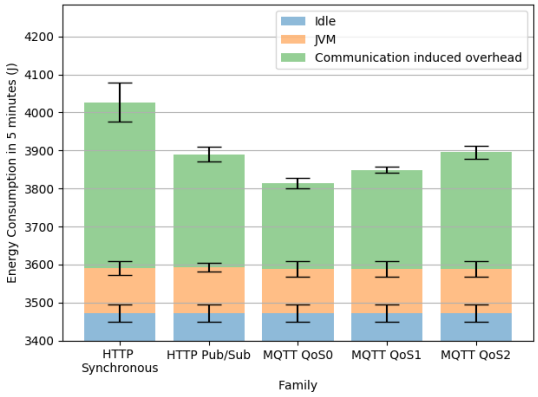

Fig. 5. Energy consumption measures

The results that are presented in Section 5 only show the interaction cost (the upper part in green on Figure 5). We obtain this value with this formula : $M_{idle+jvm+interactions} - M_{idle+jvm}$. As a consequence, the standard deviation of the result is the addition of the standard deviation of the two measures.

4.3 Experimental plan

Table 2 presents the combinations of interaction patterns and protocols for which we have handled the experiments. That gives 5 families of experiments. We measure: (1) the impact of the interaction pattern through families F1 and F2;

Family	Interaction pattern	Protocol
F1	Synchronous	HTTP
F2	Publish/Subscribe	HTTP
F3	Publish/Subscribe	MQTT QoS0
F4	Publish/Subscribe	MQTT QoS1
F5	Publish/Subscribe	MQTT QoS2

Table 2. Families of experiments

(2) the impact of the protocol with families F2 and F3 and (3) the impact of the QoS for MQTT with families F3, F4 and F5.

The message rates used in the experiments were of 1, 2, 4, 8 and 16 messages by second. The tests at 32, 64 and 128 messages per second with both interaction patterns using HTTP started to receive a significantly lower amount of messages, as a consequence we did not keep those results.

The payload used in the experiments were of 24, 48, 240, 1320 and 1560 bytes. We start with 24 bytes, since it is assumed that this payload is about the usual value for an IoT payload. The last two values were chosen considering the MTU, which was measured at 1500 bytes for our experiments. One lower than this value and the other higher for comparison purposes on the energy-consumption influence of such scenario.

We did 125 experiments: 5 families of experiments (see Table 2)* 5 message rates * 5 payloads. For each experiment, we used 30 tests. Three more measures were done with the consumer application also running Wireshark in order to explain the obtained results. As the usage of Wireshark increases the energy consumption of the machine, we do not include those tests for computing the mean and the standard deviation. In total we realized 33*125= 4 125 tests.

Each test had a total duration of 8 minutes. This was organized with one minute for warm up, where the producer started the message exchange with the consumer. Followed by a measurement of the energy consumption for 5 minutes while the producer was exchanging data with the consumer. Finally, two more minutes of sleep time to reset the experiment and the network conditions before starting the following test. 4 125 tests of duration 8 minutes necessitate around one full month of experiments. Additionally, for $M_{idle+jvm}$, 60 tests were realized, we double the number of tests to obtain low standard deviation and confidence intervals.

4.4 Threats to validity

We present below potential threats to the validity of our study and how we propose to minimize their effects.

Computer conditions: The activity of the computer can not be totally controlled, as a consequence we report some discrepancies in the measured values. To reduce these discrepancies, we have shut down or disabled all unnecessary processes of the operating system as well as using the lowest brightness and connecting to the device via ssh to reduce user tampering. In order to minimize the

standard deviation and obtain a more consistent result, each of the experiments were run a total of 30 times.

Network conditions: The conditions of the network while doing the tests were optimal. The gathered data showed that there was no packet loss during the tests and the latency remained low and stable at around 23ms.

Temperature at which the experiments are conducted: During the initial experiments, the climate did not rise above 25 degrees Celsius. However, on some days when the external temperature rose between 28 and 32 degrees, the fluctuations in energy consumption increased. These fluctuations may be due to the need for the equipment cooling systems to increase their output in order to keep the components of the equipment in the correct temperature conditions. To address this threat, the client computer was moved to an air-conditioned room where the computer was always at a cold temperature. This resulted in a reduction of the standard deviations of the measurements, making the results more stable. The experiments realized in the air-conditioned room were for the message rate of 8m/s and the payloads of 1320B, 1560B, and 3120B.

5 Analysis

We organize the analysis of the results of the experiments according to the four tackled research questions presented in the introduction. As an outcome of the analysis, Section 5.5 presents guidelines dedicated to developers of IoT consumer applications.

5.1 (RQ1) Impact of the interaction pattern

For a fair comparison of the interaction patterns, we compared only the results obtained with the HTTP protocol for which we have measured the two interaction patterns.

Figure 6 presents the results of the energy consumption for a 24Bytes payload for both interaction patterns. Table 3 presents in percentage the synchronous pattern overhead over the publish/subscribe pattern. This is a synthesis of all the realized measures (all the message rates).

The results of the experiments show that with the same number of received observations, the synchronous pattern consumes around 92% (mean of all the message rates and payloads results) more energy than the publish/subscribe interaction pattern, being almost two times less efficient. This happens as the client needs to process the request for the server and wait for a reply whereas in pub/sub it will only need to wait for notifications from the server.

5.2 (RQ2) Impact of the application protocol

For the comparison of the protocols, it was desired to do a fair comparison of the two protocols, comparing the MQTT QoS 0 and HTTP Pub/Sub as they both propose an “at most once” semantics. As observed in Figure 7 and in Table 4,

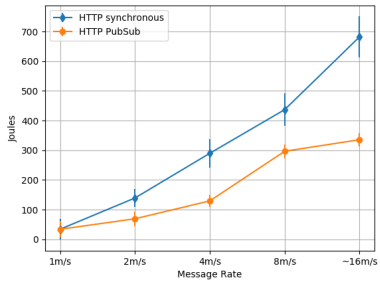

Fig. 6. Energy consumption 24B, Interaction Pattern Comparison

Payload	Overhead in %
24B	+94.03%
48B	+89.96%
240B	+106.90%
1320B	+85.16%
1560B	+85.50%
Mean	+92.31%

Table 3. Synchronous pattern average overhead over the publish/subscribe pattern

MQTT outperforms HTTP in terms of energy consumption and number of bytes by Joule.

We observe that in terms of energy, the MQTT protocol outperforms HTTP by 20% on average while having the same interaction pattern and the same semantics. This happens because of the purpose of each protocol. While HTTP has more processing on top of the data received by the client, as it needs to look into further validations (e.g size variable header, parameters, etc), MQTT is proposed with a more lightweigth structure that, for example, has fixed headers, enabling a less intensive processing by the client.

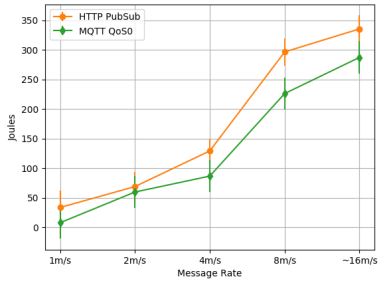

Fig. 7. Energy consumption for a 24B payload, Protocol Comparison

Payload	HTTP vs MQTT in %
24B	+28.07%
48B	+23.40%
240B	+31.05%
1320B	+2.58%
1560B	+18.63%
Mean	+20.75%

Table 4. HTTP vs MQTT average overhead with all the message rates

5.3 (RQ3) Impact of the QoS in MQTT

In Figure 8, we compare the measures of energy consumption for the three MQTT QoS with the 24B payload. Table 5 presents a synthesis of the overheads for all the payloads.

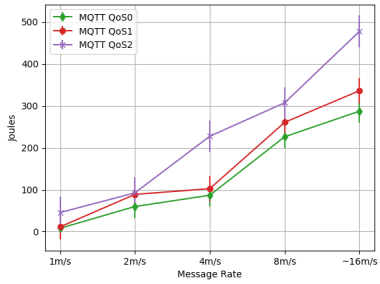

Fig. 8. Energy consumption for a 24B payload, QoS Comparison

Payload	QoS1/QoS0	QoS2/QoS1	QoS2/QoS0
24B	+24.64%	+46.58%	+79.72%
48B	+17.76%	+51.09%	+78.20%
240B	+58.67%	+27.08%	+104.46%
1320B	+12.16%	+88.10%	+111.76%
1560B	+21.83%	+53.31%	+86.45%
Mean	+27.01%	+59.77 %	+92.12%

Table 5. MQTT QoS overheads

Taking into consideration all the measures realized, meaning all the message rates and payloads, the comparison of the QoS shows that QoS 0 consumes around 27% less energy than QoS 1 and 92% compared to QoS 2 with the same number of received observations. QoS 2 consumes 60% more energy than QoS 1. Having similar results in the comparisons of QoS 0, QoS 1 and QoS 2, to what was observed in the related work [21]. The difference is that we are able to measure the consumer side only while they use the same device as producer and consumer and as a result can not differentiate consumer from producer energy consumption.

A deeper look at the results shows that the impact of the QoS using MQTT is related to the amount of messages exchanged during the experiment. QoS 0 had the smallest amount of messages exchanged between the broker and the consumer because of the fire and forget mechanism (Sending messages and not verifying the arrival) it implements and resulted in the lowest energy consumption. MQTT QoS 1 followed a similar path but as it increased the amount of messages, due to the acknowledgments of the client, it resulted in a bigger energy consumption when compared to MQTT QoS 0. Finally, QoS 2 with its bigger amount of messages exchanged between the broker and the client doubled the amount of packets exchanged and ended up almost doubling the amount of energy used.

5.4 (RQ4) Impact of the payload

Figure 9 presents the bytes/Joule for different payloads for the fixed rate of 8 messages per second for the 5 families of experiments.

The usage of a payload up to 3120 bytes, presented a moderate increase in the experiments (mean 9%), having cases with even lower consumption for HTTP. The fragmentation of the messages according to the MTU (1500 bytes) does not seem to have a relevant impact on the energy consumption. The experiment impacted the most by the increase was MQTT QoS 0, having up to 21.89% more energy consumption. Concerning HTTP publish-subscribe and request-reply, both seemed unaffected by the changes in the payload as the payload of

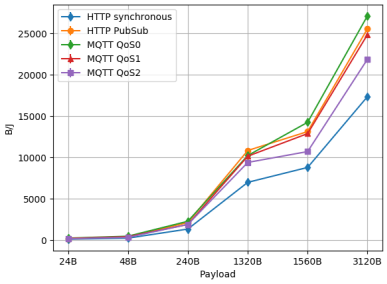

Fig. 9. Bytes Received by Joule Payload Comparison

Family of experiment	Overhead in %
HTTP synchronous	-1.36%
HTTP Pub/sub	-1.32%
MQTT QoS0	+21.89%
MQTT QoS1	+15.45%
MQTT QoS2	+11.24%
Mean	+9,18%

Table 6. Payload overhead from 24Bytes to 3120Bytes

24B was slightly higher than the one with 3120B (Table 6), which had the message broken into 3 fragments according to the size of the MTU. The behaviour of the payload seen in HTTP is further confirmed when checking the amount of TCP connections with 24 bytes and 3120 bytes which remained the same. Furthermore, MQTT is more impacted by the payload because the protocol created only one TCP connection through all the test phases and exchanged messages in this established connection. This causes the payload to become a bigger part of the energy consumption considering that MQTT has a 2 bytes fixed header, while the HTTP header does not have a limit (A limit can be set by the server), but in the case of the tests done, it is around 106 Bytes. Besides those differences by family of experiment, the lesson of this comparison is that the number of Bytes by Joule is augmented significantly for all the families while augmenting the payload. An explanation for this result comes from the cost of the software call stack necessary to handle one message whatever the size of the message is.

5.5 Guidelines for IoT consumer application designers

We provide in this section guidelines for IoT consumer application designers to reduce the energy consumption at the end user device side.

Group several observations in one message We have shown that using different payloads, from 24 up to 3120 bytes, has a small impact on the energy consumption of the application. If the application necessitates multiple sensor observations, we advise combining the different observations into one single message. Some IoT platforms, such as Fiware/Orion, provide the possibility to query (or subscribe to) a group of sensors. This possibility has clearly to be chosen by application developers.

Favor the Publish-Subscribe interaction pattern The comparison of interaction patterns showed that for the same frequency of requests and notifications,

the publish/subscribe pattern consumed on average 92% less energy than the request/reply pattern. As a consequence, we advise to favor the publish/subscribe pattern.

We have to mention that this advice may depend on the IoT application and the IoT platform. If the frequency of requests is far lower than the frequency of publications, the synchronous pattern can be an option because the client can better control the amount of messages being exchanged.

Favor the MQTT protocol over the HTTP protocol For the publish/subscribe pattern, the comparison of the MQTT and HTTP protocols shows that MQTT has 20% less energy overhead in comparison to HTTP. The advice is then to favor the MQTT protocol for the publish/subscribe pattern.

Choose the QoS appropriate for your application If your IoT application supports losing some observations, prefer QoS 0 since it involves less energy consumption. If the application cannot afford to lose observations, use QoS 1 instead of QoS 0 as it provides a complementary service to TCP's reliability by ensuring that each message is received at least once. Keep the QoS2 for exactly once semantics requirement as it presents an overhead of 92% and should be used in conditions that require no duplication of messages.

Guideline example The benefits of guidelines to develop IoT applications can be better seen when viewing with a bigger space of time. As an example, an IoT application running for one year using HTTP pub/sub sending 4 messages per second with a payload of 24B will consume around 31,01 MegaJoules while another application running with the same parameters but using MQTT QoS 0 will consume around 8,40 MegaJoules. Furthermore, if the messages are grouped into a single message, and sent once per second, we can achieve a consumption of around 4,20 MegaJoules for both HTTP pub/sub and MQTT QoS 0 with a payload of 24 Bytes. As an example, for a regular notebook battery with around 360 KiloJoules, an IoT application using HTTP or MQTT and grouping messages could lead to a lifetime of around 31 hours, on the other hand, without grouping and using HTTP synchronous we have around 9 hours of battery (71% less).

6 Conclusions

Energy consumption is a first class concern in the development of future IoT applications. As the amount of devices and the amount of applications related to IoT will keep growing in the near future, there is a new requirement for the developers and the users to regulate and improve the energy consumption of IoT applications not only on the connected object side but also on the consumer application side.

In this paper, we have measured the energy consumption of IoT consumer applications on user devices connected with WiFi (802.11n). We have been able to show the impact on energy consumption on different interaction choices summarized below. The results show that for the same amount of received observations, the publish/subscribe interaction pattern has lower energy consumption (around 92% lower) than the synchronous interaction pattern. We have also shown that, for the publish/subscribe interaction pattern, MQTT consumes less than the HTTP protocol (around 20% less). Finally, we have shown that the payload has a low impact on energy consumption having a 9% overhead from 24 to 3120 bytes payloads. From the above results, we have been able to provide guidelines for IoT consumer application designers, for example we advise developers to favor the publish/subscribe pattern and to group several observations in one message when possible.

As a future work, we plan to investigate the cost of the software call stack to better guide the developers of IoT consumer applications. We plan to investigate the impact of data representation on the cost of marshalling and unmarshalling data in IoT applications. We also plan to follow the guidelines for the design of a middleware used by IoT applications to transparently interact with multiple IoT platforms. Implementing those strategies at the middleware level may have a strong impact for reducing IoT application energy consumption while keeping a low development effort.

References

1. PowerAPI. `http://powerapi.org/`, accessed: 2022-02-15
2. Bandyopadhyay, S., Bhattacharyya, A.: Lightweight internet protocols for web enablement of sensors using constrained gateway devices. In: 2013 International Conference on Computing, Networking and Communications (ICNC). pp. 334–340 (2013). https://doi.org/10.1109/ICCNC.2013.6504105
3. Bouloukakis, G., Georgantas, N., Ntumba, P., Issarny, V.: Automated synthesis of mediators for middleware-layer protocol interoperability in the IoT. Future Generation Computer Systems **101**, 1271–1294 (2019). https://doi.org/https://doi.org/10.1016/j.future.2019.05.064, `https://www.sciencedirect.com/science/article/pii/S0167739X18323586`
4. Cisco Annual Internet Report (2018–2023). `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html` (2020)
5. Dizdarević, J., Carpio, F., Jukan, A., Masip-Bruin, X.: A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. vol. 51. Association for Computing Machinery, New York, NY, USA (Jan 2019). https://doi.org/10.1145/3292674, `https://doi.org/10.1145/3292674`
6. FIWARE: What is fiware? `https://www.fiware.org/`
7. Garbarino, E.: Message exchange patterns (meps) (2013), `https://garba.org/article/general/soa/mep.html#top`
8. Hofer, J., Pawaskar, S.: Impact of the application layer protocol on energy consumption, 4g utilization and performance. In: 2018 3rd Cloudification of the Internet of Things (CIoT). pp. 1–7 (2018). https://doi.org/10.1109/CIOT.2018.8627133

9. directed by Hugues Ferreboeuf, S.P.: Lean ict – towards digital sobriety. https://theshiftproject.org/wp-content/uploads/2019/03/Lean-ICT-Report_The-Shift-Project\textunderscore2019.pdf (2019)
10. ISO/IEC: Internet of things (IoT) - reference architecture. ISO/IEC JTC 1/SC 41 - Internet of Things and Digital Twin p. 84 (8 2018)
11. Joshi, J., Rajapriya, V., Rahul, S., Kumar, P., Polepally, S., Samineni, R., Kamal Tej, D.: Performance enhancement and IoT based monitoring for smart home. In: 2017 International Conference on Information Networking (ICOIN). pp. 468–473 (2017). https://doi.org/10.1109/ICOIN.2017.7899537
12. Munoz, D.J., Montenegro, J.A., Pinto, M., Fuentes, L.: Energy-aware environments for the development of green applications for cyber–physical systems. Future Generation Computer Systems **91**, 536 – 554 (2019). https://doi.org/https://doi.org/10.1016/j.future.2018.09.006, http://www.sciencedirect.com/science/article/pii/S0167739X18307295
13. Nakhuva, B., Champaneria, T.: Study of various internet of things platforms. International Journal of Computer Science & Engineering Survey **6**(6), 61–74 (2015)
14. Nielsen, H., Mogul, J., Masinter, L.M., Fielding, R.T., Gettys, J., Leach, P.J., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Jun 1999). https://doi.org/10.17487/RFC2616, https://rfc-editor.org/rfc/rfc2616.txt
15. Noureddine, A., Rouvoy, R., Seinturier, L.: A review of energy measurement approaches. ACM SIGOPS Oper. Syst. Rev. **47**(3), 42–49 (2013)
16. 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
17. oneM2M: Who we are. https://www.onem2m.org/harmonization-m2m
18. Pang, C., Hindle, A., Adams, B., Hassan, A.E.: What do programmers know about software energy consumption? IEEE Software **33**(3), 83–89 (May 2016). https://doi.org/10.1109/MS.2015.83
19. Postel, J.: Internet Protocol. RFC 791, RFC Editor (09 1981). https://doi.org/10.17487/RFC0791, https://www.rfc-editor.org/info/rfc791
20. Shaikh, F.K., Zeadally, S., Exposito, E.: Enabling technologies for green internet of things. IEEE Systems Journal **11**(2), 983–994 (2017). https://doi.org/10.1109/JSYST.2015.2415194
21. Toldinas, J., Lozinskis, B., Baranauskas, E., Dobrovolskis, A.: MQTT quality of service versus energy consumption. In: 2019 23rd International Conference Electronics. pp. 1–4 (2019). https://doi.org/10.1109/ELECTRONICS.2019.8765692
22. YoctoPuce: Who are we? https://www.yoctopuce.com/EN/aboutus.php, accessed on 17-10-2021