Noname manuscript No.
(will be inserted by the editor)

Energy characterization of IoT systems through design aspect monitoring

Alexios Lekidis · Panagiotis Katsaros

Received: date / Accepted: date

Abstract The technological revolution brought by the Internet of Things (IoT) is characterized by a high-level of automation based, to a large extent, on battery autonomy. Important risks hindering its wide adoption, though, are associated with device battery lifetime, which is affected by system design aspects such as connectivity, data processing and storage, as well as security protection against cyber threats. Even though simulation can help for the energy cost estimation of IoT applications before their actual deployment, it is still challenging, and extensive effort is required to converge to a feasible architectural deployment scenario. This article introduces a method to address this challenge by estimating the energy cost of the IoT design aspects and identifying the feasible deployment scenarios, for an IoT system architecture. The method is illustrated on a smart city application that consists of subsystems for building management and intelligent transportation. These two subsystems employ a variety of IoT devices connected to an Orion border router. We estimate the feasibility of various architectural deployments with respect to the system requirements and conclude to those that are possible, as feedback to the system designers. The case study results include quantitative metrics for the evaluation of system requirements using a new aspect monitoring technique and the well-established Statistical Model Checking approach.

Alexios Lekidis
School of Informatics,
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece E-mail: alekidis@csd.auth.gr

Panagiotis Katsaros
School of Informatics,
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece E-mail: katsaros@csd.auth.gr

Keywords Internet of Things · Rigorous system design · Energy cost estimation · Smart city testbed

1 Introduction

IoT is an emerging technology trend that shifted application design towards a networked embedded system architecture, providing automation capabilities across heterogeneous devices and systems. Automation was gradually linked with autonomy characteristics based on battery portability for IoT devices. These two characteristics, though, are conflicting with each other, as data processing and network connectivity are vital for automation, but limit substantially the autonomous operation.

Hence, engineers have to invest extensive time and effort to find the right trade-off between them, usually by a trial-and-error approach. Specifically, the requirements on which a system's design is based, often imply only limited autonomy and hence they are usually reduced iteratively, in order to reach a viable battery lifetime. This happens due to the limited availability of tools and adequate techniques for measuring and optimizing the energy cost of various IoT design aspects, such as availability, reliability, performance and security with respect to the variety of IoT cyber threats [1].

The requirements focusing on the need for a low energy footprint, for the IoT device, usually incur an extensive implementation effort towards finding a feasible system architecture. Additionally, the absence of universal IoT standardization [4] makes even establishing proper requirements a hard task. The difficulty of IoT standardization initiatives lies in the heterogeneity of IoT devices, as well as their massive expansion and usage in diverse application domains. Instead, experts

are lately shifting from requirements towards the definition of design aspects that impact IoT applications [5]. The most significant among these aspects are the availability, reliability, dynamicity and security of IoT applications.

The IoT energy cost estimation challenge is handled in literature with hardware-based energy calculation techniques [12, 24] that provide a detailed analysis of consumption, for the devices of the IoT architecture. However, these techniques are not supported by default by the IoT devices and hence require hardware modifications and calibration that is specific for each different device. Thus, such approaches do not constitute a generic method, which led to the introduction of software-based energy estimation techniques [10] that add minimal overhead to the IoT application processing and maintain the IoT architecture unaffected. Martinez et al. [17] present a technique that aims at profiling the energy consumption in different IoT device operating modes. However, this technique does not provide support to the application designer, if he wants to estimate the energy cost for operations defined by the system requirements. Such cost estimation would allow the prediction of feasible application deployment scenarios during the early system design stages. The lack of energy estimation techniques combined with the misleading belief that hardware-based techniques are more accurate, as they are linked directly to the hardware, make application designers hesitant towards the adoption of software-based techniques.

Apart from the proposed software-based energy calculation technique though, Martinez et al. [17] also introduce an energy estimation direction. According to this perspective, we should aim at gathering and characterizing all parameters and scenarios that impact the energy consumption for an IoT device. The approach should be generic across all vendors and models of IoT devices and should be also modular to cover new IoT devices that are made available in the market. Inspired by this direction, our article introduces a new method that allows to: (1) characterize the energy cost of the IoT design aspects with respect to the system's requirements and (2) estimate feasible IoT application deployment scenarios that satisfy these requirements.

The effectiveness of our method is illustrated by experimenting with a smart city scenario that consists of a Building Management System (BMS) and an Intelligent Transport System (ITS). Both types of systems have associated committees for defining application and system requirements (e.g. European Transport Standards Committee - ETSI[1]). Our method allows application designers to evaluate the feasibility of system requirements that are linked to IoT design aspects such as connectivity, data processing, storage, security (authentication and encryption), as well as dynamicity of the IoT environment. The method has been implemented for IoT systems that are based on the Contiki OS [6] [16], but with appropriate adjustment it can be also ported to another IoT system architecture. The concrete contributions of this article are:

- the semantics and context of monitoring IoT design aspects;
- identification of system configurations that satisfy energy requirements using IoT design aspect monitoring;
- the smart city case-study scenario including the BMS and ITS subsystems;
- quantitative metrics for the comparison of IoT aspect monitoring and Statistical Model Checking (SMC) [20] techniques

The rest of this article is organized as follows. Section 2 provides a brief introduction to the IoT design aspects, the rigorous model-based design using the component based framework called BIP [2], and the factors contributing to the energy cost calculation. Section 3 provides an overview of the proposed method and its semantics and demonstrates the techniques and algorithms to validate it. Section 4 evaluates the method in our experimental smart city architecture to derive accurate energy estimations of the IoT design aspects. Finally, Section 5 states the conclusions and the perspectives for future work.

2 Preliminaries

2.1 IoT design aspects

IoT applications rely on various design aspects to provide automation across heterogeneous devices and systems. The main difference with application design for other systems is that in IoT, design aspects have to be guaranteed throughout the entire IoT device architecture [16]. Fig. 1 provides an overview of the main layers of an IoT device architecture. Based on the shown layers the main requirements that arise are:

i) Availability: device communication overhead should be limited to a suitably low level, in order the device to last for a reasonably long period of time.

This requirement is related to device connectivity aspect, since a significant amount of device energy consumption depends on the frequency of radio transmissions/receptions (radio communication is provided by the Hardware layer of Fig. 1). The relevant parameters for this requirement [14] are summarized in Table 1.

[1] https://www.etsi.org/

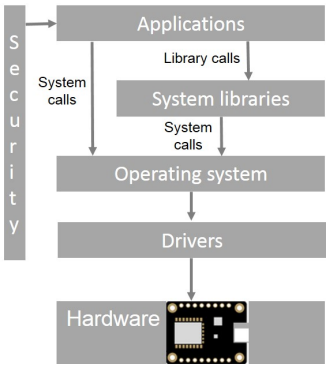

Fig. 1 IoT device layers

Parameter	Description	Values
ASP_{CONN_A}	The duty-cycling protocol value	Contiki-MAC X-MAC LPP nullRDC
ASP_{CONN_B}	The duty-cycling frequency	[2-32] Hz
ASP_{CONN_B}	The packet re-transmission number	[0-5] Z
ASP_{CONN_C}	The web-service protocol	CoAP MQTT HTTP
ASP_{CONN_D}	The network packet header size	[32-64] bytes
ASP_{CONN_E}	The wireless network interference	[0-1]

Table 1 Connectivity aspect parameters

ii) Performance: computations have to respect the resource constrained nature of IoT devices to avoid consuming all their resources.

This requirement is related to the processing and data storage aspects (Operating system layer of Fig. 1). The dependence on these aspects is implied by the CPU and memory constraints of IoT devices. Examples of efficient processing and storage include the cloud resources for computation [19] and the static (i.e. malloc library) or dynamic memory allocation (i.e. mmem and memb libraries) [15]. The parameters of Table 2 control the energy cost for IoT devices with respect to the mentioned aspects.

Parameter	Description	Values
ASP_{PROC_A}	Available IoT resources	Temperature Humidity Motion Light Accelerometer
ASP_{PROC_B}	Routing protocol for data forwarding	RPL CORPL CARP none
ASP_{PROC_C}	Memory block management module	Malloc mmem memb

Table 2 Processing aspect parameters

iii) Security: IoT devices must be resilient to the presence of cyber attacks.

This requirement is related to the security aspects (Security layer of Fig. 1). The presence of cyber threats is related to abnormal system behavior caused by malicious actors. Examples of such behavior include the Denial of Service (DoS) [13], spoofing or frame replay [21]. Cyber protection should respect the energy constraints of IoT devices. Hence, Table 3 summarizes the parameters controlling the energy cost.

Parameter	Description	Values
ASP_{SEC_A}	Security level of IoT application	SL-0 SL-1 SL-2
ASP_{SEC_B}	Implemented security protocols	TLS DTLS IPSec
ASP_{SEC_C}	Encryption key size	128 bits 192 bits 256 bits

Table 3 Security aspect parameters

iv) Dynamicity: IoT systems should handle dynamic changes in the environment or the individual devices.

The ability to adapt to the addition of new resources at the device level or to adjustments in the network and environment's topology is an important characteristic of IoT systems. Dynamicity covers the entire IoT system architecture, thus affecting energy consumption. Our method (Section 3.1) assumes a configurable application description and an energy-parameter configuration to allow analysis of the energy consumed in these scenarios. Moreover, a related approach to handle architecture dynamicity is under development [3] to further support this aspect.

2.2 Energy cost computation

We associate an energy cost to every IoT device and for each aspect category, which varies according to the parameters of the previous section. Quantifying this cost though, is challenging due to: 1) the heterogeneity of the IoT systems and 2) the lack of methods for identifying how the system requirements contribute to the energy consumption. Specifically, existing methods [17] focus on calculating the energy footprint, which is used to build a profile of energy consumption over time, for each operating mode of an IoT device. However, building an energy profile does not provide insight on how much each requirement contributes to the overall energy consumption. Consequently, as requirements are specified in the early design phase, an application designer cannot reason about the feasible deployment scenarios on an IoT architecture. This shortcoming incurs

extensive time overhead in the application development and maintenance stage [22].

A potential solution to this problem, would be to derive the contribution of IoT application parameters to the energy cost. Application parameters are chosen according to the system requirements and belong to the design aspect categories of the previous section. Moreover, the direct association of parameters to design aspects, allows an IoT application designer to only focus on computing the energy cost attributed to the application parameters. Then, the designer can reason about the energy cost of design aspects before the application is actually deployed on the target IoT architecture. In [14,15], we proposed a framework to support this direction that models the energy-oriented behavior of an IoT application. This allows to identify the application parameters associated with the energy cost. Furthermore, we provided equations, in order to compute the overall contribution of the application parameters to the energy cost. We hereby provide the three main equations of this framework and explain thoroughly their coefficients and terms in Appendix A.

1. The total energy cost E_{total} (in Joule) is:

$$E_{total} = E_{ASP_i} + E_{LPM} + E_{PER} \quad (1)$$

where the energy terms with $E_{ASP} = E_{ASP_{CONN}} + E_{ASP_{PROC}} + E_{ASP_{SEC}}$ are associated with the IoT design aspects and are presented in Appendix A. The remaining terms depend on the device characteristics. Specifically, E_{LPM} depends on the implementation of energy saving mode, where the device wakes up and performs all the associated functionalities either periodically or upon asynchronous events. E_{PER} indicates the energy consumed by the device peripherals.

2. The duty cycle reflects the percentage of time that an IoT device remains in an operating mode:

$$D_y = \frac{\sum_{i=1}^{N_y} I_y * V_y * \Delta t_{y_i}}{E_{total}} \quad (2)$$

where y indicates the device's operating mode, N_y the relative number of occurrences that the device has visited the operating mode y, Δt_y indicates the time intervals in which the device remains in an operating mode y and I, V indicate respectively the current (in Ampere) and voltage (in Volts).

3. The lifetime reflects the total time duration during which a device operates autonomously:

$$lf = \frac{C_{batt} * V_{cc}}{E_{total}} \quad (3)$$

where C_{batt} indicates the overall capacity of the battery for autonomous operation (in Ampere hours) and V_{cc} the operating voltage (in Volts).

Based on these equations, we can derive the energy cost of design aspects that impact energy consumption in IoT devices. An example that refers to Zolertia Zoul devices[2] is depicted in Fig. 2. In the shown graph, we illustrate the relation between the energy cost of the design aspects [3]. The presented operating modes include the idle mode, when a device is in standby ($y = LPM$), the transmission mode, when it is sending data through the radio ($y = Tx$), the reception mode, when it receives data through the radio ($y = Rx$) and the system processing mode ($y = CPU$), when the device utilizes its CPU and memory resources, for data processing. The energy cost measurements for this example are based on our testbed university deployment that is presented in Section 4.1. The detailed method used to obtain such measurements and compute the energy cost, as in Fig. 2, is called energy characterization and it is presented in Section 3.2.

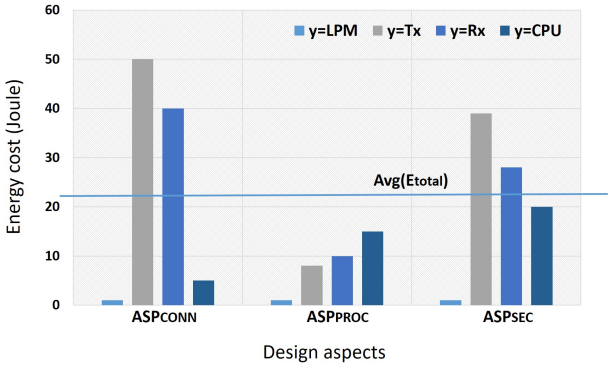

Fig. 2 Energy cost of each IoT design aspect for Zolertia Zoul devices deployed in our university testtbed

2.3 Rigorous system design using BIP

BIP (Behavior-Interaction-Priority) [2] is a highly expressive, component-based framework with rigorous semantic basis. It allows the construction of complex, hierarchically structured models from atomic components, which are characterized by their behavior and interfaces. Such components are transition systems enriched with data. Transitions are used to move from a source to a destination location. Each time a transition is taken, component data (variables) may be assigned with new values, which are computed by user-defined functions (in C/C++). Atomic components are composed by layered application of interactions and priorities. Interactions express synchronization constraints and define the transfer of data between interacting components. A set of atomic components can be composed

[2] https://zolertia.io/zoul-module/

[3] The considered design aspects here do not include the dynamicity aspect contribution as it is still under development

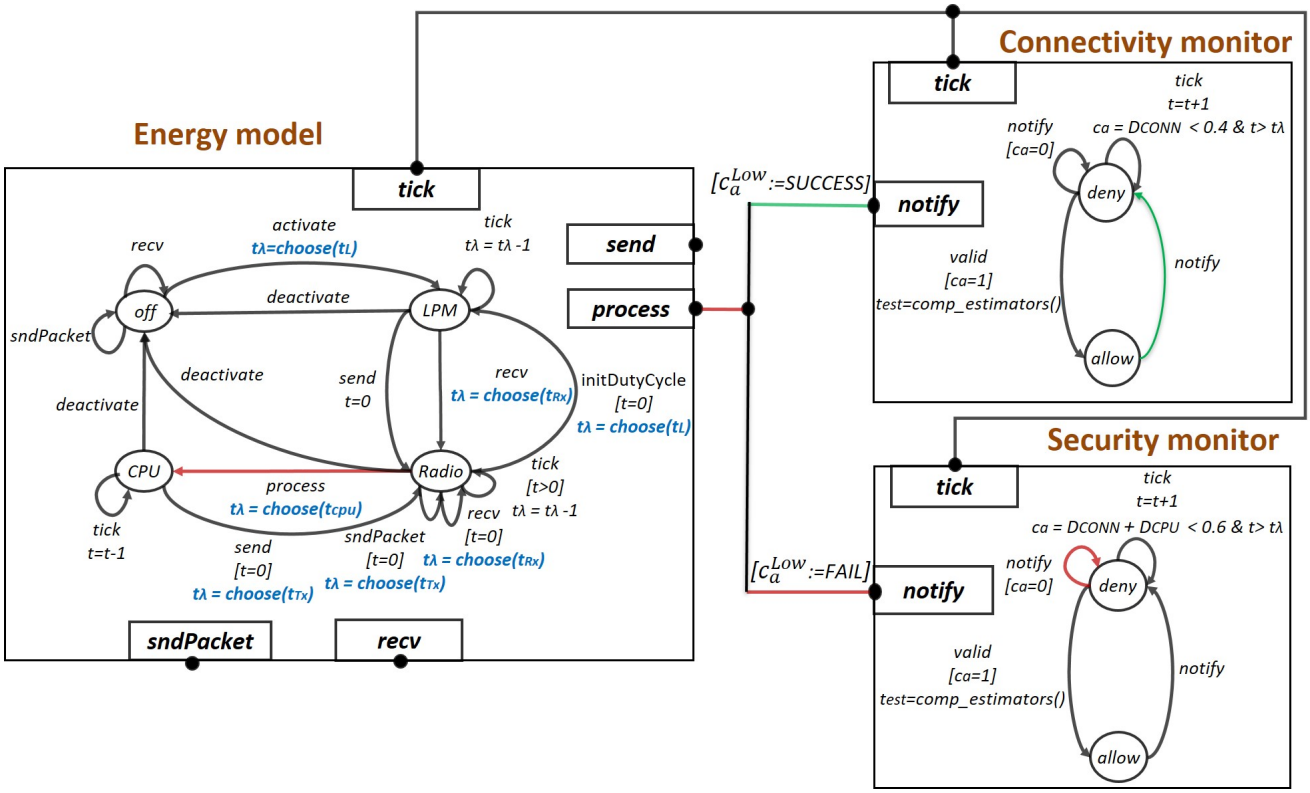

Fig. 3 BIP components example for modeling energy cost and monitoring IoT aspects

into a generic compound component by the successive application of connectors, representing sets of interactions, and priorities.

Connectors can export their ports (similarly to atomic components) for building hierarchies of connectors as illustrated in Fig. 3. They can also use data variables, in order to compute transfer functions associated with interactions. Computations take place iteratively either upwards (up) or downwards (down) through the connectors hierarchy levels, but computed values are not stored between the execution of two interactions (connectors are stateless). Exported ports may also have associated variables, which are mainly used to store results from the computation of transfer functions.

BIP is supported by a rich toolset including tools that are used to verify stochastic systems, through the Statistical Model Checking (SMC) technique [20]. SMC was proposed as a means to cope with the scalability issues in numerical methods for the analysis of stochastic systems. The SMC of BIP models is automated by the SMC-BIP tool [20]. For model checking system requirements, they have to be formalized as stochastic temporal properties expressed in Probabilistic Bounded Linear Temporal Logic (PBLTL) [11]. The SMC-BIP tool accepts as inputs PBLTL properties, a model in BIP and a couple of confidence parameters. When its execution is completed, SMC-BIP provides a verdict in the form of probability for the property to hold true. Since the approach is designed for the validation of bounded LTL properties, it is guaranteed to terminate in finite time.

3 Energy-aspect monitoring

3.1 Method Overview

Our method consists of two phases: (I) model construction and (II) energy estimation. Three inputs by the IoT designer are assumed: an energy-aware parameter configuration file in XML format [14], the application design description in a domain-specific language (DSL) [16] and the system requirements in plain text form. The inputs, the method's flow and its outputs, as well as the energy estimation approach that is presented in this paper are illustrated in Fig. 4. Moreover, an enhanced level of flexibility and modularity is supported by the tools and transformation techniques that were developed to automate the individual design steps. We hereby provide insight into the method's main phases.
I) Model construction: this phase leads to the composition of the IoT system model. The model is composed progressively by specifying first the behavior of the application modules in a DSL-based description [16], which is used to generate an Application Model in

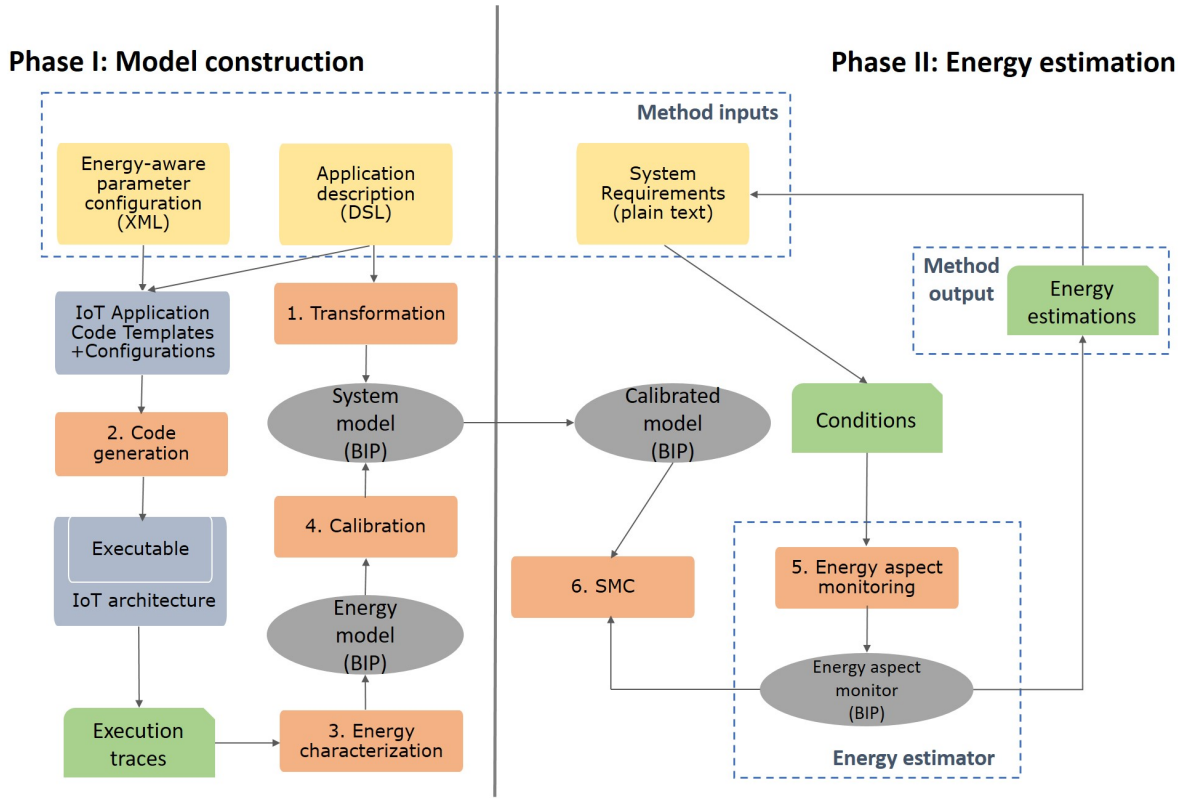

Fig. 4 Design phases of the proposed method

BIP. Our DSL implementation supports the definition of Contiki code application modules. The model is later enhanced with the OS/kernel model, which is composed from a library of BIP components. The two models are then composed by incorporating information specified in the DSL description, for how the application modules are deployed onto the IoT system's devices.

II) Energy estimation: the system model is enhanced with an energy model that represents the energy flow in the system. The energy model is calibrated using probabilistic distributions. Distributions are derived from an energy characterization procedure, in which the execution traces on the IoT architecture are fitted into a distribution using appropriate methods, such as moments matching and maximum likelihood (more details on this are given in Appendix B). The calibrated model is then used for the estimation of tight energy bounds, for the aspect monitoring or SMC techniques. In the following part of this article, we give emphasis to the demonstration of the newly introduced aspect monitoring technique, but the two techniques are compared through quantitative evaluation metrics, in Section 4.3.

3.2 Energy characterization

Energy characterization in our method (Step 3 of Fig. 4) is based on measurements that sum up the energy cost of each design aspect (Section 2.1) on the IoT architecture. The energy cost is computed by enhancing the powertrace library of the Contiki OS [7] with the equations described in Section 2.2. Each design aspect has associated code templates in the Contiki OS, which reflect how atomic operations are implemented by the OS. The code templates are used to generate application code that is subsequently executed on the target IoT architecture. Since each code template describes a different atomic operation, the whole set of templates for an IoT application includes all required functionalities that the application can have. Such functionalities are message transmission or reception, CPU processing, memory storage, security mechanisms or interaction with sensors/actuators. An example of a code template for message transmission with the CoAP protocol [23] is illustrated in Listing 1.

Specifically, Listing 1 includes the *powertrace_start* command in line 11, which activates the radio. In lines 21-22 the CoAP transmitted message is initiated and hence the message header and payload are initialized as shown in lines 23-24. Finally, a CoAP message request

```
1  #include "contiki.h"
2  #include "powertrace.h"
3  #include "sys/log.h"
4  #define LOG_MODULE "CoAP"
5  #define LOG_LEVEL LOG_LEVEL_INFO
6  PROCESS(power, "powertrace example");
7  AUTOSTART_PROCESSES(&power);
8  PROCESS_THREAD(power, ev, data)
9  {
10   static struct etimer et;
11   rtimer_clock_t start;
12   PROCESS_BEGIN();
13   /* Start powertracing */
14   int RTIMER = 1; // 1 second reporting cycle
15   powertrace_start(CLOCK_SECOND * RTIMER);
16   etimer_set(&et, CLOCK_SECOND*t);
17   while(1) {
18
19     if(etimer_expired(&et)) {
20       start = RTIMER_NOW();
21       coap_init_message(request, COAP_TYPE_CON,
22       COAP_POST, 0);
23       coap_set_payload(request, (uint8_t *)msg,..
24        sizeof(msg) - 1);
25       COAP_BLOCKING_REQUEST(&server_ipaddr,..
26        REMOTE_PORT, request, client_chunk_handler);
27       LOG_INFO("CoAP transmission request (%u)\n",..
28        RTIMER_NOW() - start);
29   }
30 }
31 PROCESS_END();
```

Listing 1 Contiki CoAP transmission code template

is initiated during which the device switches in transmission ($y = Tx$) mode. Using the RTIMER_NOW() function of the Contiki OS we can measure the overall time duration, for the execution of the CoAP message request, which is logged in a file as shown in lines 27-28. Finally, the logged time duration along with the current and voltage values for the device's transmission mode are provided to the Equation (5), in order to compute the energy cost of a message transmission.

Once all code templates are developed, they are executed on the devices of the IoT architecture, which allows to obtain a measurement for the energy consumption. Then, the energy measurements of the code templates belonging to each design aspect are combined, in order to describe the IoT device energy footprint, in each operating mode. An example of such a footprint is illustrated in Fig. 2. As a next step, the code templates are used to generate executable code for each device of the IoT architecture. Then, we measure the energy while the IoT application is executed on the device. This characterizes the evolution of the energy footprint over time, which exhibits a non-deterministic behavior, since Contiki is an event-driven OS. Based on the gathered energy measurements, we apply a distribution fitting tool-supported method (Appendix B), in order to characterize the energy footprint over time. In phase II of our method, the probability distributions along with the energy model are used to estimate the feasibility of satisfying the system requirements on the employed IoT architecture.

3.3 Aspect monitoring context

Aspect conditions are derived from the IoT system design aspects (Section 2.1) and are evaluated through aspect monitors.

Definition 1 (Aspect condition)
An aspect condition c is specified by the formula $c : \{B, B_{ENERGY}, V, Par, M_{ASP}, f(V, \epsilon), P\}$, where:

- $B \supseteq \{B_1, ...B_N\}$ the model components that participate in the hierarchical interaction with the aspect monitor.
- B_{ENERGY} is the energy model that represents the energy operations in the IoT system model.
- $V \supseteq \{V_1, ...V_M\}$ is a set of variables used to evaluate the condition. These variables may belong to the components in B or to the aspect monitor. They are provided to the monitor through the upstream function of the hierarchical connector (Fig. 3).
- Par is a set of values for the used parameters of the energy-aware parameter configuration that define the simulated scenario.
- $M_{ASP} \supseteq \{M_{CONN}, M_{PROC}, M_{SEC}\}$ is the monitor(s) that is (are) instantiated to validate the aspect condition (see Definition 3).
- ϵ is a condition value that is derived based on the system requirements.
- $f(V, \epsilon)$ is a function computing the energy equations (Section 2) and using the aspect variables, in order to evaluate them against the condition value.
- P is a priority level for each condition that depends on how important it is, for the designer of the IoT system (see Definition 2).

Aspect conditions are described as: $c = (c_a^P, c_b^P, ..., c_{final}^P)$, where the subscript depicts their value for each condition of the system and the superscript their priority level. Additionally, c_{final} indicates the last condition.

Definition 2 (Priority level)
A priority level P is provided to describe the importance for each aspect condition of the IoT system, with $P = \{Low, Medium, High, Critical\}$.
Each priority level has an associated weight W as an importance indicator, with $W \supseteq \{W_{Low} = 0.25, W_{Medium} = 0.5, W_{High} = 0.75, W_{Critical} = 1\}$. Conditions are further categorized according to their priority level into sets, with S_{total} indicating the entire list of existing conditions for the IoT system: $S_{total} = S_{Low} + S_{Medium} + S_{High} + S_{Critical}$.

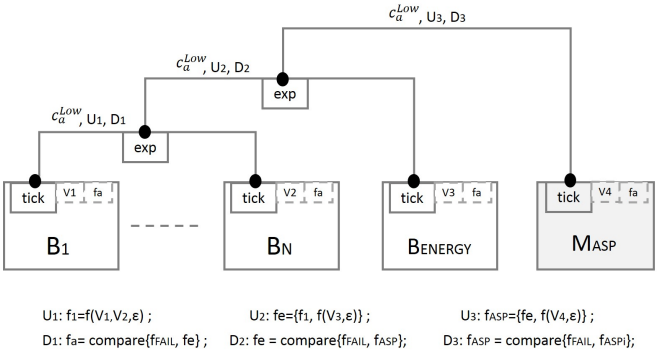

Fig. 5 Hierarchical connection of aspect monitors with the IoT system model

Definition 3 (Aspect monitor)
Monitors are based on the IoT design aspects and are initialized according to the system requirements and conditions. Each monitor is associated with the energy cost equations of Section 2.2:
M_{CONN}: $[D_{Tx}, D_{Rx}, lf, E_{ASP_{CONN}}, E_{total}]$
M_{PROC}: $[D_{CPU}, E_{ASP_{PROC}}, E_{total}, lf]$
M_{SEC}: $[D_{Tx}, D_{Rx}, D_{CPU}, E_{ASP_{SEC}}, E_{total}, lf]$

Aspect monitors are connected hierarchically to the monitored system and the energy model through the exported port *tick* (see Fig. 3). Each hierarchical connector exports an additional *exp* port to allow the connection of other monitor components. Fig. 5 illustrates the hierarchical connection of all system components.

Every hierarchical connector has an associated upstream transfer function U (with $U1$, $U2$, $U3$) to indicate the flow of data during the "up" action of an interaction and likewise a downstream transfer function D (with $D1$, $D2$, $D3$) to indicate the flow of data during the "down" action.

Definition 4 (Scenario parameters)
The selected values for the energy-aware configuration parameters define a scenario SC that is executed by the IoT system model. Scenario parameters are given by $Par_{SC} = (ASP_{i_A}, ASP_{i_B}, ..., ASP_{i_N})$. This definition includes all the IoT system design aspects with $ASP_i : (ASP_1 : CONN, ASP_2 : PROC, ASP_3 : SEC)$.

Scenarios are executed on a per aspect basis with the following assumption: if an aspect is selected all the other aspects remain in their default values. To automate scenario execution, Algorithm 1 is used.

Definition 5 (Condition evaluation)
When the set of variables V required to compute a condition is collected, each individual monitor computes the condition through the function $f(V, \epsilon)$. Then, a condition verdict is submitted with
$f(V, \epsilon) = (f_{SUCCESS} \rightarrow 1, f_{FAIL} \rightarrow 0)$

The first monitor that gives a negative verdict will cause the condition to be marked as invalidated. If all the monitors provide a positive verdict, the condition is marked as valid. The condition verdict is stored through the downstream function D in all participating components in the interaction.

The overall verdict of a scenario is given by a synthesis of all the verdicts, for the conditions associated with the system, based on the defined priorities. B_E performs this synthesis when the set of values for the parameters of the energy-aware parameter configuration changes.

$$f_{S_{total}} = \frac{\sum_{\forall i \in S_Y} W_Y * f(V, e)}{\sum_{\forall i \in S_{total}} f_{success}} \quad (4)$$

where y indicates the priority levels as in Definition 2.

Data: $ASP_1 : CONN, ASP_2 : PROC, ASP_3 : SEC$
Result: SUCCESS, FAIL
while ASP_i **do**
  if A **then**
    $ASP_{i_{A'}} = select(A)$
    $ASP_{i_{B'}}$ = 'RPL'
    $ASP_{i_{C'}}$ = 'Mmem'
  else if B **then**
    $ASP_{i_{A'}}$ = 'Temperature'
    $ASP_{i_{B'}} = select(B)$
    $ASP_{i_{C'}}$ = 'Mmem'
  else if C **then**
    $ASP_{i_{A'}}$ = 'Temperature'
    $ASP_{i_{B'}}$ = 'RPL'
    $ASP_{i_{C'}} = select(C)$
end
$f_{S_{total}} := run\_model(ASP_{i_{A'}}, ASP_{i_{B'}}, ..., ASP_{i_{N'}})$

Algorithm 1: Scenario selection

Definition 6 (Scenario verdict)
The evaluation of a condition during the upstream function is followed by setting the computed value to the associated monitors during the downstream function. As a result, each component participating in the interaction will have a condition verdict (Definition 5).

Following the presence of the condition verdict, the monitors will execute the transition that is specified by the respective guard with $valid : g \rightarrow SUCCESS$ and $notify : g \rightarrow FAIL$. Even in the case of executing the *valid* internal transition, *notify* will be executed right after in the aspect monitor to ensure an uninterrupted system function.

Example 1 (Condition logging)

Aspect parameters	Condition	Verdict
$ContikiMAC, 8, 4, MQTT, 48, 0$ $Temperature, RPL, Mmem$ $SL-0, TLS, 128$	$c_1^{Low} \rightarrow FAIL$ $c_2^{High} \rightarrow SUCCESS$ $c_3^{High} \rightarrow SUCCESS$	$f_{S_{total}} = SUCCESS$

Table 4 Aspect condition logging example

Logging of the condition verdicts is handled by B_{ENERGY}, which includes in the logging file the details that are depicted in Table 4. Let us assume that a particular scenario was executed and the monitors have computed three conditions with:

Condition 1 *(c_1^{Low}): The energy cost of security aspects should be lower than all other aspects.*
Condition 2 *(c_2^{High}): The IoT node is sustained for two consecutive working days on battery power.*
Condition 3 *(c_3^{High}): The processing time of security operations is not higher than 60% of the overall duty cycle.*

The condition evaluation for c_1 is illustrated in Fig. 5. The rest of the condition verdicts along with the overall verdict (Definition 5) that is logged is presented in Table 4.

Having all the conditions in the logging file, allows the system designer to investigate the failed conditions and take actions towards: 1) making enhancements in the IoT system or 2) adapting the priority levels for the conditions to re-calculate the total score.

4 Smart city case-study

4.1 Scenario overview

In this section, we demonstrate the method in a smart city scenario that is illustrated in Fig. 6. The scenario consists of a Building Management (BMS) subsystem and an Intelligent Transport (ITS) subsystem exchanging information via an Orion IPv6 border router [4]. The BMS subsystem that is installed in a building receives real-time updates about the weather, traffic conditions and road works from the city roads, where the ITS subsystem is installed. The received ITS updates allow the BMS subsystem to inform the employees working in the building about traffic/weather conditions in remote locations that are part of their daily commute. Additionally, the BMS subsystem can also optimize the building resources (e.g. working hours, temperature) according to the received updates.

The subsystems of the smart-city scenario belong to different IoT application domains, therefore they have diverse requirements, for ensuring a smooth operation. An example where the requirements differ is the connectivity aspects, for which the BMS subsystem collects small payloads with temperature and light data from the building rooms, whereas the ITS subsystem consists of vehicles that broadcast larger data payloads with traffic and road conditions. Additionally, the presence of private (e.g. driver) data inside the ITS subsystem raises the need for strong security mechanisms (ETSI compliant [5]) for data exchange, whereas a weaker security protection is applied to the BMS subsystem. Finally, the dynamicity aspects should be also considered in the ITS application, as vehicles dynamically join and leave the system, in contrast to the BMS subsystem, which is part of a static deployment. In the following, we provide more technical details about each subsystem of the smart-city scenario and their corresponding energy-oriented requirements.

BMS subsystem

We have deployed and configured the BMS subsystem as an IoT testbed in a university building (Fig. 7)[6]. The setup consists of five Zolertia Zoul nodes and an Orion border router for IPv6 network configuration and for data exchange with the ITS subsystem (Fig. 6). Four Zolertia Zoul nodes are configured as Contiki server nodes, which have been deployed in separate floors of the university building. Server nodes have temperature and light sensors attached to them, in order to collect measurements from each floor and transmit them every single minute to a central BMS controller located near the main building entrance of the ground floor. The BMS controller acts as a Contiki client, in order

[4] https://zolertia.io/product/orion-router/

[5] https://www.etsi.org/e-brochure/Work-Programme/2017-2018/files/basic-html/page17.html

[6] The testbed deployment can be also used for other use-cases, due to its small scale and the configuration flexibility of the Contiki OS.

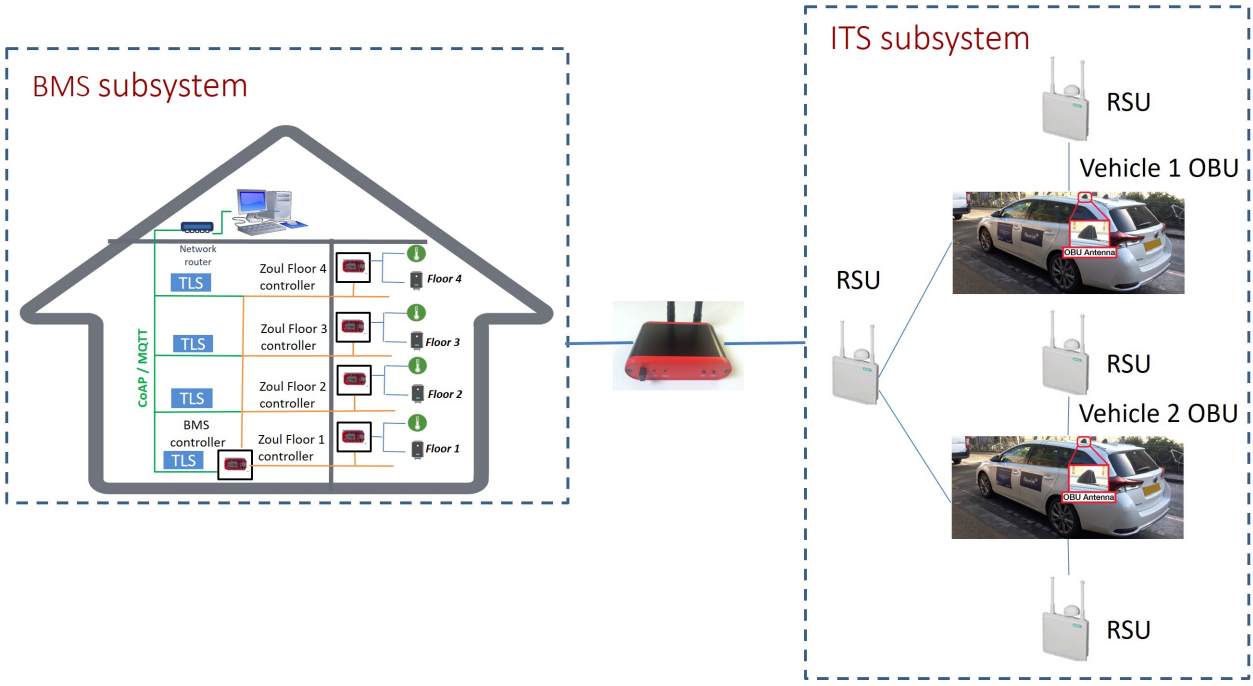

Fig. 6 University building testbed interconnection with an Intelligent Transport System

to: (i) monitor the temperature and light conditions of each floor and (ii) set the temperature level inside the building and switch on the lights during working hours, according to the received updates from the ITS subsystem. The upper floor nodes, which are distant from the BMS controller use the lower floor nodes as relay, in order to forward the data to it. The Zoul nodes execute the application code that is generated using the method presented in Fig. 4.

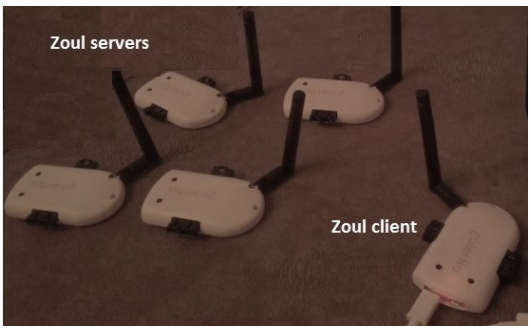

Fig. 7 The testbed deployed in the university building

The energy-oriented requirements for the BMS subsystem concern with the highest possible availability of the Contiki server nodes for transmitting temperature and humidity data to the BMS controller, in order to sustain continuous real-time monitoring for each floor. The Zolertia Zoul nodes are re-charged during the building maintenance that is scheduled to take place every 3 days, which means that nodes should operate continuously on battery power within this period. Additionally, as the lower floor nodes are relaying data to the BMS controller, the performance and storage aspects have to be considered. Secure data exchange mainly concerns with the BMS controller that has a web server installed, for remote building access. To this respect, the security measures must not add substantial processing overhead and energy cost, for the BMS subsystem. We have therefore implemented a lightweight TLS security library, which allows encrypted communication by establishing a TLS handshake between nodes, as well as using symmetric session keys of various sizes (128, 192 and 256 bits), according to the Advanced Encryption Standard (AES).

The energy-oriented requirements for the BMS subsystem have been expressed as in the three conditions of Example 1 (Section 3.3) and have been appropriately customized for the BMS subsystem.

Condition A (c_a^{Low})***:*** *The energy cost of security aspects should be lower than all other aspects.*
Condition B (c_b^{High})***:*** *The Zolertia Zoul nodes are sustained for three consecutive working days on battery power.*
Condition C (c_c^{High})***:*** *The processing time of security operations is not higher than 60% of the overall duty cycle.*

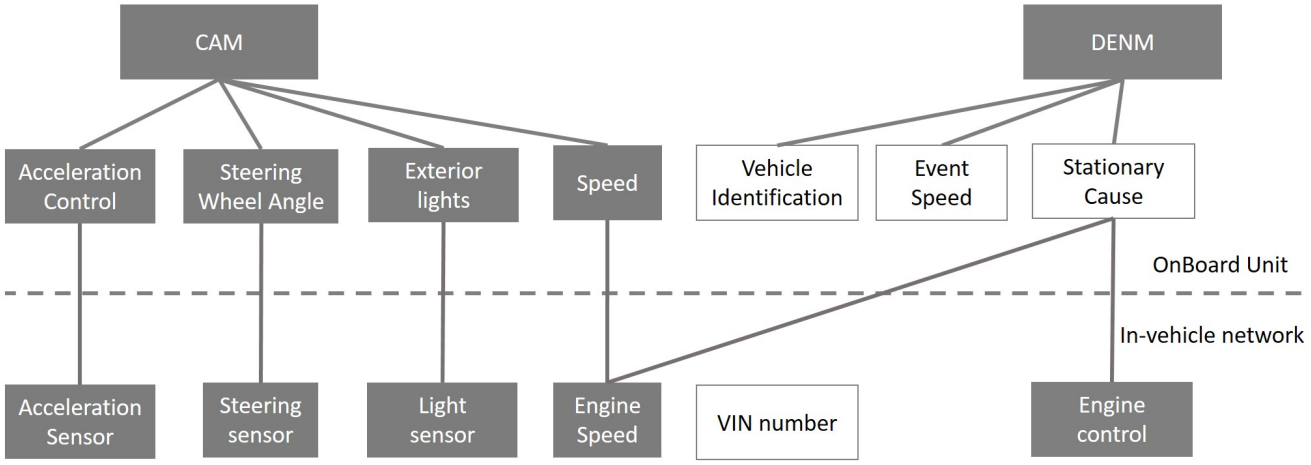

Fig. 8 OBU message mapping to the in-vehicle network

ITS subsystem

The ITS subsystem is driven by data from an existing setup that is described in [18]. In this case, the authors have provided real-time data from driving two Toyota Auris vehicles in a test track around the city center of Bristol, UK. Each vehicle has an On-Board Unit (OBU) and the setup includes four RoadSide Units (RSUs) placed at fixed locations around Bristol's city center. The OBUs obtain in-vehicle network information and broadcast them every 10ms to nearby vehicles and RSUs through Cooperative Awareness Messages (CAM) and Decentralized Environmental Notification Messages (DENM). These messages are defined in the ETSI ITS standards EN 302 637-2 [9] and EN 302 637-3 [8]. The OBU CAM and DENM messages contain information obtained from the in-vehicle network components (i.e. Electronic Control Units-ECUs), as illustrated in Fig. 8. Likewise, RSUs also use CAM and DENM messages to broadcast information upon important incidents, such as traffic jams, stationary vehicles and road hazards (e.g. road accidents, environmental conditions, road works).

The ITS setup of [18] was reproduced using our Contiki-based IoT architecture. This process took place through replaying the packet captures in the Instant Contiki Virtual Machine environment [7], such that the ITS network communication flow is reflected at the same time with the BMS subsystem communications and the in-between interactions between the two subsystems. Moreover, the energy consumption was computed based on the timing measurements from the ITS setup using the Linux native dstat library [8]. The obtained dstat timing measurements concerned with the IoT design aspects of the ITS setup, e.g. the total amount of transmitted and received bytes for the connectivity aspects (more details for the energy cost calculation based on these measurements are presented in Appendix A).

The energy-oriented requirements for the ITS subsystem indicate that the OBUs should operate continuously for at least one day. At the end of the day, the vehicles are re-charged, which allows the OBUs to recover their full battery capacity. Regarding the security aspects, the ETSI standard states that 256-bit encryption has to be used together with appropriate authentication mechanisms. This corresponds to level SL-2, for the security aspect parameters of our method (Section 2.1). Therefore, we have extended the original ITS setup of [18] by enabling the TLS protocol in all connections. Similarly to the BMS subsystem, this extension was based on the same lightweight TLS library. The ITS subsystem requirements are given as follows:

Condition D (c_d^{High})***:*** *The ITS nodes are sustained for one working day on battery power.*
Condition E (c_e^{Low})***:*** *Security aspects should add minimal energy cost to the ITS nodes (i.e. OBUs, RSUs).*

4.2 Experiments

For the ITS subsystem part of the smart city scenario, we configured the Instant Contiki Virtual Machine environment to communicate through an IPv6 LAN network with the Orion border router of the BMS subsystem. The BMS subsystem deployment and the availability of node connections were then checked by: (i) logging in the Orion border router homepage and obtaining the IPv6 node addresses, and (ii) verifying if

[7] https://sourceforge.net/projects/contiki/files/Instant Contiki/
[8] https://linux.die.net/man/1/dstat

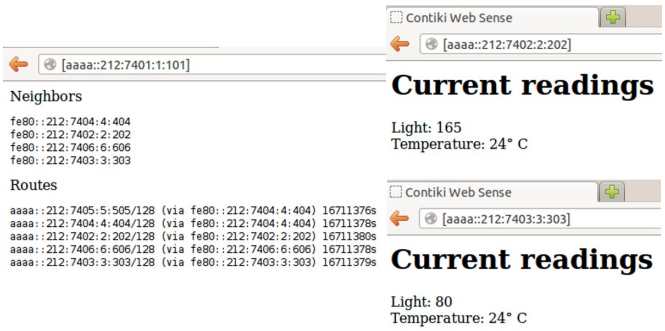

Fig. 9 Building testbed IPv6 addresses and temperature, light readings

the obtained temperature and light readings from the installed nodes were correct. Fig. 9 illustrates the results of the aforementioned configuration setup test.

Two sets of experiments were conducted for the five conditions of Section 4.1 based on aspect monitors for evaluating them (i.e. SUCCESS or FAIL), as well as based on SMC, in order to estimate the probabilities of satisfying them. The threshold for the probability of satisfying a condition through SMC was set to 70%. Hence, an SMC probability result above 70% would denote that a condition is satisfied. The results of these experiments are presented henceforward for each condition separately.

Condition A (c_a^{Low}). This condition is considered of low priority, since if violated it has limited consequences and it is not one of the requirements stemming from the essential system operational context. To evaluate c_a^{Low}, all proposed monitors are required, as there is need to compute the energy cost of the connectivity and processing aspects, in order to compare it with the energy cost of security aspects. The energy cost of the connectivity, processing and security aspects is respectively computed according to Equations (5), (6) and (7) of Appendix A. Monitor evaluation was performed according to the hierarchy shown in Fig. 3. Following the evaluation with the aspect monitors we found that the final verdict for this condition ($f_{S_{total}}$ in Section 3.3) is FAIL, as the Zolertia Zoul nodes dedicate large spans for message encryption/decryption and for the TLS handshake. Moreover, the SMC experimental results for c_a^{Low} showed a 60% probability for satisfying this condition, which is below the set threshold. All results for condition c_a^{Low} are summarized in Table 5.

Condition B (c_b^{High}). From the BMS subsystem requirements, it is expected to ensure continuous operation of the Zolertia Zoul nodes for three consecutive days without intermediate charge, which renders this condition of high priority. Two aspect monitors were found to affect the verdict for this condition, namely the connectivity and security monitors that are based on Equation (3) of Section 2.2, whereas the processing monitor verdict satisfied the condition in all cases, meaning that it does not have substantial effect to the condition. Regarding the connectivity monitor, we observed that it yielded a FAIL verdict in most of the BMS subsystem parameter scenarios (Section 3.3) causing a FAIL for the $f_{S_{total}}$ verdict. However, for the scenarios with the Contiki-MAC and X-MAC parameter values for the RDC protocol, up to 14 Hz RDC frequency and below 0.2 interference, the connectivity monitor verdict was SUCCESS (Fig. 10). On the other hand, the security monitor verdict only satisfied the condition for the SL-0 security level with a 128 bits session key size. The exact probability for satisfying condition c_b^{High} was found by SMC experiments to be 55%, hence below the set threshold. The results though provided us insight for the variation of the device lifetime based on the connectivity aspect parameters, such as the RDC protocol that is illustrated in Fig. 11. The results for condition c_b^{High} are also summarized in Table 5.

Condition C (c_c^{High}). This condition is of high priority, since it is connected to the BMS subsystem requirement for the availability of temperature and light data. In particular, if a substantial amount of time is devoted to encryption/decryption and authentication operations, the latency in data exchange between the Zolertia Zoul nodes is increased accordingly. The experiments for this condition aimed to identify the configuration that offers adequate protection against cyber threats, while being able to be ported to the Zolertia Zoul nodes. We mainly focused on the security monitor for calculating the duty cycle of security operations using Equation (2) of Section 2.2. We could achieve adequate encryption (128 symmetric key size) and implement an authentication scheme (i.e. security level SL-0), but security levels SL-1 and SL-2 and their demands for message encryption/decryption induced an overly high processing overhead, for the BMS nodes, which increased significantly the duty cycle of security operations and resulted in violating this condition. The experiment results from the aspect monitoring are shown in Fig. 12. Based on the evaluation of the security monitor, this condition led to a $f_{S_{total}}$: SUCCESS verdict. For a more accurate verdict, we also experimented with SMC that yielded an overall 71% probability for satisfying the condition, i.e. higher than the set threshold. The results for condition c_c^{High} are summarized in Table 5.

Condition D (c_d^{High}). For condition c_d^{High}, the priority level was set to high, since maintaining the battery charge of ITS nodes at an adequate level, until they can be recharged, stems from an operational re-

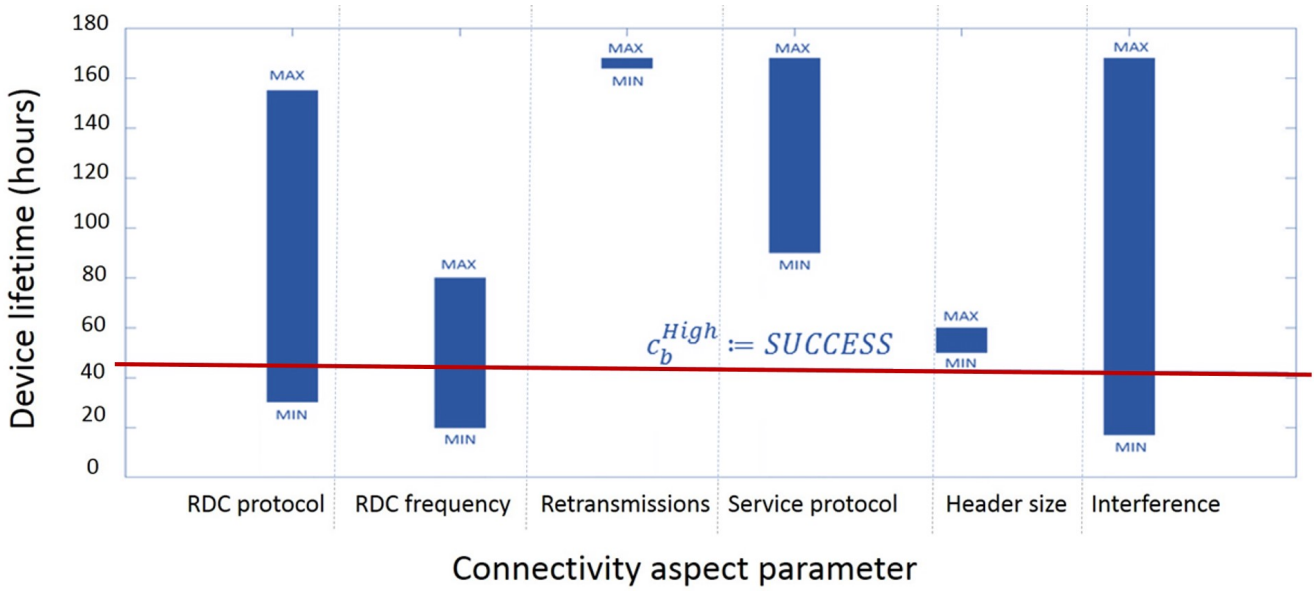

Fig. 10 Connectivity aspect parameter contribution for condition B

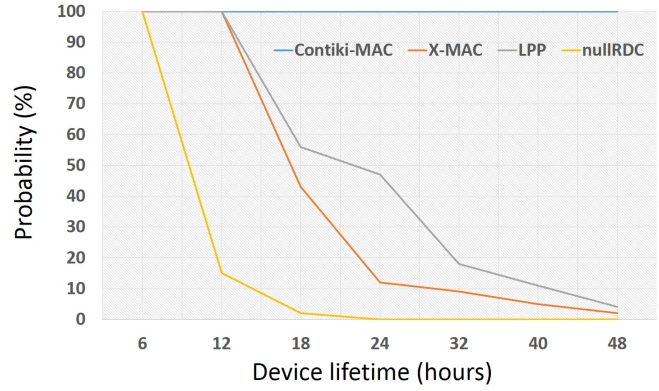

Fig. 11 SMC probabilities for the case-study condition B

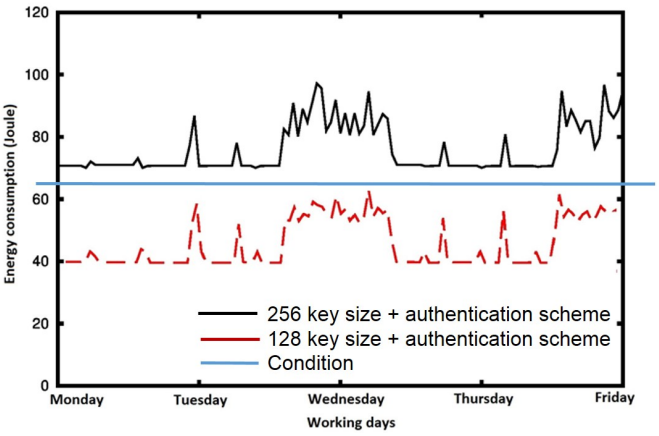

Fig. 12 Energy footprint for condition C with aspect monitoring

striction. The evaluation of this condition required all aspect monitors according to the hierarchy of Fig. 3. As an outcome of aspect monitoring experiments, the battery lifetime of the OBU and RSU nodes lasted for more than one working day (≈ 16 hours and 32 minutes) and thus the evaluation result was a $f_{S_{total}}$: SUCCESS verdict. From the SMC, the probability of satisfying this condition was estimated to be about 90%. In comparison to the BMS subsystem, the battery lifetime of the OBUs and RSUs is significantly lower (see Condition B), even though we used the same Zolertia Zoul nodes for the energy cost calculation. Their main differences lie in (i) more frequent transmissions of the OBUs and RSUs (every 10 ms) and (ii) the larger size of CAM and DENM message (more than 5 times) in comparison with the temperature and light data messages. Moreover, the tighter security requirements of OBUs and RSUs increase significantly the energy cost of security aspects ($E_{ASP_{SEC}}$) and hence also the total energy required (E_{total}). The overall increase of E_{total} results in less battery lifetime as calculated using Equation (3) in Section 2.2. The results for condition c_d^{High} are summarized in Table 6.

Condition E (c_e^{Low}). This is a low priority condition, since there is no specific requirement or policy of enforcement to the ITS subsystem.The condition is related to the energy cost for security aspects and as expected, we relied on the security monitor for its evaluation. The energy cost for the security aspect monitor was calculated according to Equation (7) of Appendix A. The results from the aspect monitoring, as well as those from the SMC experiments showed that the con-

dition is not satisfied, which happens due to the substantial energy cost that was induced to the subsystem, for the security level SL-2. Specifically, the aspect monitoring technique provided an $f_{S_{total}}$: FAIL verdict and the SMC resulted in a 10% probability for satisfying this condition, which is far below of the set probability threshold. The additional energy cost for the security aspects is linked to the encryption/decryption mechanisms, as well as to the TLS handshake and authentication mechanisms that should be applied every time that a vehicle enters the ITS subsystem area. The condition was found to be satisfied only in the scenario of applying the security level SL-0 with a 128 symmetric key size for encryption/decryption of the exchanged messages, but this scenario does not comply with the aforementioned ETSI policy, as it would allow data eavesdropping from unauthorized and potentially malicious entities. The results for condition c_e^{Low} are summarized in Table 6.

4.3 Evaluation metrics

We proposed a method that includes both an aspect monitoring and an SMC technique, for validating system requirements and deployment scenarios. Due to the diverse requirements of the BMS and ITS subsystems we conducted two separate sets of experiments, one for each subsystem. Both sets of experiments were based on three evaluation metrics, in order to facilitate the selection between them, according to the IoT application needs: (i) the overall CPU time for the evaluation, (ii) the average memory that is required for the computations and (iii) the final verdict for the condition that is evaluated. The final verdict for aspect monitoring ($f_{S_{total}}$ in Definition 5) was based on a statistical average of the verdicts for each executed scenario (Definition 4) and a statistical average of the probabilities for each scenario for SMC.

Table 5 summarizes the results from the experiments for the BMS subsystem with the aspect monitoring and SMC techniques. The results refer the used evaluation metrics. Likewise, Table 6 provides the results from the experiments for the ITS subsystem in regard with the respective evaluation metrics.

In overall, we consider SMC a better tool for obtaining the exact probability and statistical confidence level, for each condition, as well as for analyzing the fluctuation of each condition verdict. On the other hand, aspect monitoring reaches its verdict faster and it has on average similar memory footprint (Table 5 and 6), but it is a result-driven technique that only provides the scenario parameters (Definition 4) as contextual information. Hence, further investigation for the scenarios with failed verdicts is not possible, as with SMC. Finally, for certain conditions, as for example c_c^{High} and c_d^{High}, SMC takes overly long time to reach a verdict, which is not available for IoT designers, who usually wish to validate the system requirements rapidly.

Condition	Aspect monitoring	SMC
c_a^{Low}	CPU time: 3h 5min Av. Memory: 930 MB $f_{S_{total}}$: FAIL	CPU time: 26h 3min Memory: 960 MB SMC Verdict: 60%
c_b^{High}	CPU time: 2h 34min Av. Memory: 978 MB $f_{S_{total}}$: FAIL	CPU time: 23h 48min Av. Memory: 893 MB SMC Verdict: 55%
c_c^{High}	CPU time: 3h 4min Av. Memory: 727 MB $f_{S_{total}}$: SUCCESS	CPU time: 44h 23min Av. Memory: 920 MB SMC Verdict: 71%

Table 5 Aspect condition logging for the BMS subsystem

Condition	Aspect monitoring	SMC
c_d^{High}	CPU time: 4h 14min Memory: 1126 MB $f_{S_{total}}$: SUCCESS	CPU time: 37h 3min Memory: 1182 MB SMC Verdict: 90%
c_e^{Low}	CPU time: 3h 46min Memory: 1109 MB $f_{S_{total}}$: FAIL	CPU time: 26h 2min Memory: 1151 MB SMC Verdict: 10%

Table 6 Aspect condition logging for the ITS subsystem

5 Conclusion

We presented a method for evaluating the energy footprint for a variety of IoT design aspects. The method supports the optimal configuration of IoT applications according to their architectural deployment. Additionally, it follows the principles of rigorous system design by using the BIP component framework. The notable method phases include: I) model construction and II) energy estimation. The former takes as input the application design description in a DSL and an XML-based set of energy parameters, and generates a system model in BIP. During the second phase, the BIP model is calibrated with energy constraints, obtained by applying energy characterization techniques based on the execution traces of the deployed IoT application. The calibrated model can be afterwards fed to an aspect monitoring technique for rapid estimation of the scenarios under which the system requirements are met. More accurate estimations with the exact probabilities can be obtained using SMC. Both techniques provide their results as feedback to the IoT system designer.

As a proof of concept, the described method has been applied to a smart city application. The system consists of subsystems for building management and intelligent transportation. Data exchange between the subsystems is accomplished through an Orion IPv6 router, including additional security mechanisms (i.e. encryption, authentication). We have verified conditions related to the IoT device lifetime and the CPU duty-cycle for the security mechanisms using both aspect monitoring and SMC techniques. The results allow the reduction of the IoT system design and development time by rapidly identifying the feasible and deployable scenarios. Additionally, we have compared the two techniques with evaluation metrics and we reasoned on their use for IoT system requirement and deployment scenario validation.

Both the aspect monitoring and SMC techniques require extensive tests for all combinations of energy parameters in an IoT application. This can be further improved through the use of machine learning techniques, where only the system will be progressively learned and only scenarios affected by the system requirements will be tested. In our approach to realize this perspective, we plan to use the TensorFlow lite framework[9].

References

1. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., et al.: Understanding the mirai botnet. In: USENIX Security Symposium, pp. 1092–1110 (2017)
2. Basu, A., Bensalem, B., Bozga, M., Combaz, J., Jaber, M., Nguyen, T.H., Sifakis, J.: Rigorous component-based system design using the BIP framework. IEEE software **28**(3), 41–48 (2011). DOI 10.1109/MS.2011.27
3. Bozga, M., Jaber, M., Maris, N., Sifakis, J.: Modeling dynamic architectures using dy-bip. In: International Conference on Software Composition, pp. 1–16. Springer (2012)
4. Da Xu, L., He, W., Li, S.: Internet of things in industries: A survey. IEEE Transactions on industrial informatics **10**(4), 2233–2243 (2014)
5. Distefano, S., Merlino, G., Puliafito, A.: A utility paradigm for iot: The sensing cloud. Pervasive and mobile computing **20**, 127–144 (2015)
6. Dunkels, A., Gronvall, B., Voigt, T.: Contiki-a lightweight and flexible operating system for tiny networked sensors. In: Local Computer Networks, 2004. 29th Annual IEEE International Conference on, pp. 455–462. IEEE (2004)
7. Dunkels, A., Osterlind, F., Tsiftes, N., He, Z.: Software-based on-line energy estimation for sensor nodes. In: Proceedings of the 4th workshop on Embedded networked sensors, pp. 28–32. ACM (2007). DOI 10.1145/1278972.1278979
8. ETSI, E.: 302 637-3 V1. 2.2 (2014-11) Intelligent Transport Systems (ITS). Vehicular Communications
9. ETSI, T.: Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service. Draft ETSI TS **20**, 448–451 (2011)
10. Georgiou, K., Xavier-de Souza, S., Eder, K.: The iot energy challenge: A software perspective. IEEE Embedded Systems Letters (2017). DOI 10.1109/LES.2017.2741419
11. Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate probabilistic model checking. In: Verification, Model Checking, and Abstract Interpretation, pp. 73–84. Springer (2004)
12. Jiang, X., Dutta, P., Culler, D., Stoica, I.: Micro power meter for energy monitoring of wireless sensor networks at scale. In: 2007 6th International Symposium on Information Processing in Sensor Networks, pp. 186–195. IEEE (2007)
13. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J.: Ddos in the iot: Mirai and other botnets. Computer **50**(7), 80–84 (2017)
14. Lekidis, A., Katsaros, P.: Model-Based Design of Energy-Efficient Applications for IoT Systems. In: Proceedings of the 1st International Workshop on Methods and Tools for Rigorous System Design, MeTRiD@ETAPS 2018, pp. 24–38 (2018). DOI 10.4204/EPTCS.272.3
15. Lekidis, A., Katsaros, P.: Model-based energy characterization of iot system design aspects. In: From Reactive Systems to Cyber-Physical Systems: Essays Dedicated to Scott A. Smolka on the Occasion of His 65th Birthday, pp. 165–180 (2019)
16. Lekidis, A., Stachtiari, E., Katsaros, P., Bozga, M., Georgiadis, C.K.: Model-based design of iot systems with the bip component framework. Software Practice and Experience (2018). DOI 10.1002/spe.2568
17. Martinez, B., Monton, M., Vilajosana, I., Prades, J.D.: The power of models: Modeling power consumption for iot devices. IEEE Sensors Journal **15**(10), 5777–5789 (2015). DOI 10.1109/JSEN.2015.2445094
18. Mavromatis, I., Tassi, A., Piechocki, R.J.: Operating its-g5 dsrc over unlicensed bands: A city-scale performance evaluation. In: 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 1–7. IEEE (2019)
19. Nastic, S., Sehic, S., Le, D.H., Truong, H.L., Dustdar, S.: Provisioning software-defined IoT cloud systems. In: 2014 2nd International Conference on Future Internet of Things and Cloud (FiCloud), pp. 288–295. IEEE (2014)
20. Nouri, A., Bensalem, S., Bozga, M., Delahaye, B., Jegourel, C., Legay, A.: Statistical model checking qos properties of systems with sbip. International Journal on Software Tools for Technology Transfer **17**(2), 171–185 (2015). DOI 10.1007/s10009-014-0313-6
21. Pacheco, J., Hariri, S.: Anomaly behavior analysis for iot sensors. Transactions on Emerging Telecommunications Technologies **29**(4), e3188 (2018)
22. Patel, P., Cassou, D.: Enabling high-level application development for the internet of things. Journal of Systems and Software **103**, 62–84 (2015)
23. Shelby, Z., Hartke, K., Bormann, C.: The constrained application protocol (CoAP). Tech. rep. (2014)
24. Ye, W., Vijaykrishnan, N., Kandemir, M., Irwin, M.J.: The design and use of simplepower: a cycle-accurate energy estimation tool. In: Proceedings of the 37th Annual Design Automation Conference, pp. 340–345 (2000)

[9] https://www.tensorflow.org/lite

Appendices

A *Energy cost for IoT aspects*

This Appendix refers to the computation of the energy cost for energy terms presented in Section 2.2. Initially, the cost for each aspect E_{ASP_i} is computed as follows.

For the connectivity aspect:

$$E_{ASP_{CONN}} = \sum_{j=1}^{N_{Tx}} I_{Tx} * V_{Tx} * \Delta t_{Tx_j} + \sum_{k=1}^{N_{Rx}} I_{Rx} * V_{Rx} * \Delta t_{Rx_k} \quad (5)$$

For the processing aspect:

$$E_{ASP_{PROC}} = \sum_{z=1}^{N_{CPU}} I_{CPU} * V_{CPU} * \Delta t_{CPU_z} \quad (6)$$

Energy consumption for the security aspect is linked to both connectivity and data processing aspects, however the contribution percentage for each one varies and depends on the energy parameters of the IoT application. Hence:

$$E_{ASP_{SEC}} = \Delta E_{ASP_{CONN}} + \Delta E_{ASP_{PROC}} \quad (7)$$

An additional energy term that should be considered on top of these aspects is the energy consumed for data exchange or control actions that are handled by the IoT device peripherals, where:

$$E_{PER} = \sum_{w=1}^{N_{PER}} I_{PER} * V_{PER} * \Delta t_{PER_w} \quad (8)$$

where N_{PER} indicates the relative number of occurrences that the IoT device has interacted with its peripherals either for data exchange or control actions. The energy consumed in the energy saving (i.e. LPM) mode is computed as:

$$E_{LPM} = \sum_{h=1}^{N_{LPM}} I_{LPM} * V_{LPM} * \Delta t_{LPM_h} \quad (9)$$

where N_{LPM} indicates the relative number of occurrences that the IoT device switches off its radio to save energy. N_{LPM} depends on the RDC protocol that the IoT device is using.

For IoT applications that are frequently exchanging data, such as ITS applications that involve continuous broadcasting and listening for traffic awareness data, the time duration that a device remains in an operating mode cannot be easily distinguished. Hence, the calculation of energy cost for the connectivity aspects is based on an alternative form of Equation 5, focused on number of bytes that are transmitted or received. This equation is:

$$E'_{ASP_{CONN}} = \frac{(I_{T_x} * V_{T_x})}{bit} * \sum Tx\ bits + \frac{(I_{R_x} * V_{R_x})}{bit} * \sum Rx\ bits \quad (10)$$

, where $\frac{(I_{T_x} * V_{T_x})}{bit}$ and $\frac{(I_{R_x} * V_{R_x})}{bit}$ indicate respectively the energy consumed for the transmission/reception of one bit from the device of the IoT application.

For the same type of IoT applications, $E_{ASP_{CPU}}$ is calculated by replacing the term Δt_{CPU_z} of Equation 6 with the maximum CPU load time measurements that are obtained using the Linux dstat library. Moreover, $E_{ASP_{LPM}}$ is obtained by replacing the term Δt_{LPM_h} of Equation 9 with idle CPU time measurements from the dstat library. Finally, the computation of $E_{ASP_{SEC}}$ remains unchanged. The updated equations for energy cost computation were used to calculate the energy cost for the experiments of the case-study ITS subsystem in Section 4.2.

B *Distribution fitting for energy model calibration*

In this Appendix we describe the distribution fitting technique that is used in our method to calibrate the energy model (Fig. 3) with the energy measurements obtained from the application execution on the IoT architecture (Step 4 in Fig. 4). In our method distribution fitting is used under the consideration that the target model is a probability distribution. Through this technique the energy model includes the energy-oriented behavior of the real system-under-study in the form of variables that are characterized by a probability law. This constitutes the model faithful and allows us to use it for energy consumption estimation in place of the actual system. In the following part we describe the employed distribution fitting technique.

The distribution technique itself is based on the randomness of input data and thus cannot be applied to

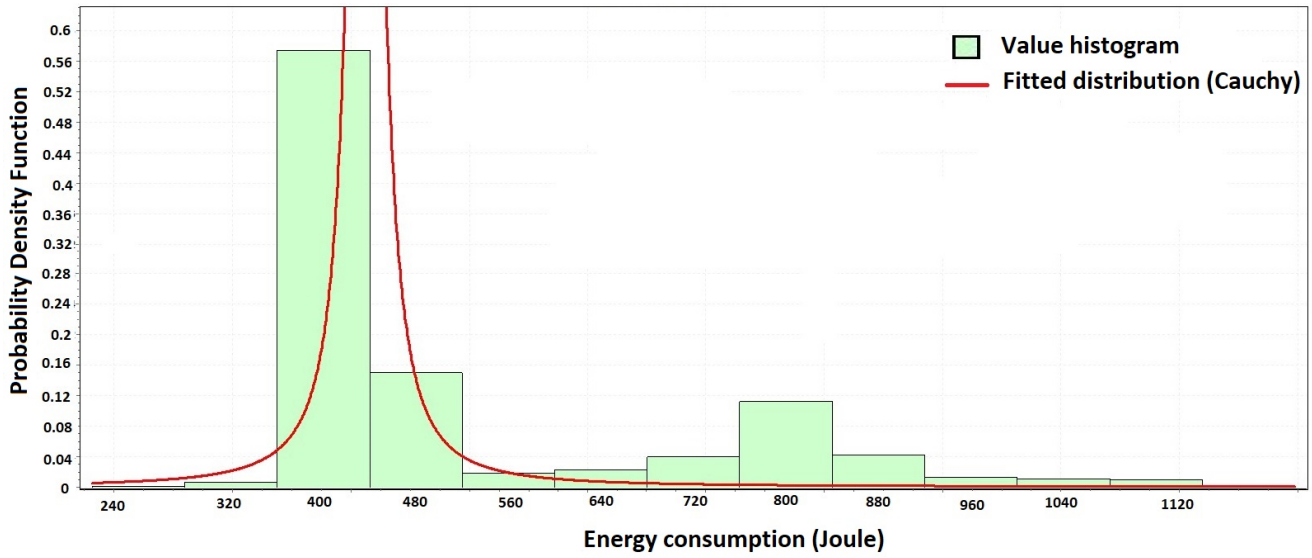

Fig. 13 Fitted energy distribution for the transmission (Tx) mode

deterministic or statistically correlated data. Instead of this, the data should be independent, such that one outcome of a random sample does not affect the outcome of another. This holds for energy data as IoT devices have asynchronous and not correlated changes, which is a consequence of relying in event-driven operating systems as the Contiki OS [7].

The fitting process is using well-known methods, such as moments matching and maximum likelihood. The moments matching method estimates the model parameters by using as many moments as the number of missing parameters of the candidate distribution. These moments depend on the probability law that the chosen candidate distribution follows. On the other hand, maximum likelihood finds the parameters that maximize the likelihood function. Then, the fitted distributions are validated against the input energy data using goodness-of-fit tests, such as the Kolmogorov-Smirnov (K-S).

An example fitted distribution characterizing the energy consumed while a device is in Tx mode for the ITS subsystem of Section 4.1 is illustrated in Fig. 13. Horizontal axis reflects the range in which energy values can vary, whereas the vertical illustrates the Probability Density Function (PDF). In this example, the distribution that is selected as a best fit is Cauchy with $\sigma = 8.8014, \mu = 409.99$ moments. The large energy consumption values are due to the overall size of the CAM and DENM messages (167 bytes) that include internal containers in comparison with the CoAP temperature and light messages of the BMS subsystem with maximum length of 30 bytes. If we consider the energy samples of the Cauchy distribution as $X = [x_1, x_2, ..., x_n]$, the distribution parameters θ_1 and θ_2 that maximize the likelihood function are computed as follows:

$$L(x_1, \ldots, x_n) = \prod_{i=1}^{n} \frac{1}{\pi(1 + {x_i}^2)} \tag{11}$$

During the validation phase, the goodness-of-fit tests have given 0.23228 error for Kolmogorov-Smirnov (K-S).

The fitted distributions are calibrating the energy model in the form of probabilistic variable $t\lambda$ in Fig. 3 (marked with blue color). This variable take values based on a non-deterministic selection that is following the probability law of four distinct fitted distributions for the time duration that the IoT device remains in each operating mode. Based on the chosen time duration the energy model afterwards uses the equations of Appendix A to compute the energy cost.