Model-based design of energy-efficient applications for IoT systems

Alexios Lekidis
Department of Informatics,
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
alekidis@auth.gr

Panagiotis Katsaros
Department of Informatics,
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
katsaros@csd.auth.gr

A major challenge that is currently faced in the design of applications for the Internet of Things (IoT) concerns with the optimal use of available energy resources given the battery lifetime of the IoT devices. The challenge is derived from the heterogeneity of the devices, in terms of their hardware and the provided functionalities (e.g data processing/communication). In this paper, we propose a novel method for (i) characterizing the parameters that influence energy consumption and (ii) validating the energy consumption of IoT devices against the system's energy-efficiency requirements (e.g. lifetime). Our approach is based on energy-aware models of the IoT application's design in the BIP (Behavior, Interaction, Priority) component framework. This allows for a detailed formal representation of the system's behavior and its subsequent validation, thus providing feedback for enhancements in the pre-deployment or pre-production stages. We illustrate our approach through a Building Management System, using well-known IoT devices running the Contiki OS that communicate by diverse IoT protocols (e.g. CoAP, MQTT). The results allow to derive tight bounds for the energy consumption in various device functionalities, as well as to validate lifetime requirements through Statistical Model Checking.

1 Introduction

The evolution of the IP-based internet towards the Internet of Things (IoT) has introduced innovations in applications from various domains, such as the smart grid, the building and home automation, the health monitoring and has even boosted a new industrial area, the so-called Industry 4.0. IoT leverages miniature devices that can exchange data autonomously through wireless communication. These devices, are usually of small size and low cost, and are also supplied with battery power, in order to widen the applicable deployment possibilities. An important challenge related to the use of battery power is that the device lifetime depends solely on the resource demands imposed by the IoT application.

The main difficulty in addressing this challenge is the underlying nature of IoT applications, namely that they are based on web services designed for continuous and long-lived service delivery through IoT devices with limited lifetime [6]. Thus, researchers have focused on mechanisms and protocols for low-power wireless communication, as well as on energy-efficient device hardware design [2]. However, to the best of our knowledge, there is only limited work towards methods and techniques for real-time monitoring and characterization of the energy consumption in IoT devices [9]. The main reason behind this is that such methods usually require direct interaction with the device hardware, which in most cases is not supported by the devices [7]. Moreover, the existing analytical methods to estimate energy consumption in resource-constrained devices [19] use the device manufacturer characteristics, which may not be always accurate when compared with measurements taken in the system's operation environment [18].

To address these limitations of existing methods and techniques for monitoring the energy consumption of IoT devices, Dunkels et al. [7] introduced a software-based solution, which is available for IoT

Submitted to:
MeTRiD 2018

© Lekidis A., Katsaros P.
This work is licensed under the
Creative Commons Attribution License.

applications in the Contiki OS through the powertrace module. Compared to a hardware-oriented approach, powertrace allows deriving a generic and hardware-agnostic model that can be applied to various device types, and enables a fine-grained analysis of the energy consumption at the network-level. However, powertrace can only support energy monitoring for the individual IoT devices, as well as the entire system. A resulting limitation is that powertrace cannot measure nor estimate the energy consumption for the device communication with its connected peripherals (e.g. sensors/actuators). A possible solution to this end should aim towards a proper characterization and assessment of all the parameters and scenarios that are impacting the energy consumption on a system-level, as described in [16]. Specifically, that work focuses on characterizing and estimating energy evolution over time in Wireless Sensor Networks (WSN) through distribution fitting techniques. Distribution fitting is a result of energy profiling in different WSN scenarios; the authors also stress that the analysis and correlation of parameters influencing energy consumption is a fundamental step towards a system design flow.

In this paper, we propose a model-based characterization of energy consumption in IoT systems and in their constituent devices. This is achieved by the progressive development of faithful models at the system-level, which incorporate valid energy profiles for the various system operation scenarios. The models are implemented in the Behavior-Interaction-Priority (BIP) component framework [1] based on specifications for the functional behavior and energy constraints for the system under design. BIP enables effective semantics-preserving transformations for the system-level models, as well as the simulation and validation of IoT systems in every development stage. The approach supports the system's validation through Statistical Model Checking (SMC) against the related requirements provided as user input. This results in valuable feedback to the system designer for enhancements concerning: (i) the IoT device lifetime and (ii) the communication/computation energy cost in each IoT device, its sensors/actuators as well as in the overall system.

The approach is illustrated through a Building Management System (BMS) which features several well-known IoT devices, such as the Zolertia Z1 [1], the Sky mote [2], the OpenMote [3] and SimpleLink Sensortag [4]. To this respect, we provide detailed profiling and characterization of the energy consumption, as well as methods to build efficient IoT applications in BMS systems. Concretely, this paper has the following contributions:

- an energy-aware model that allows providing valid bounds for the energy consumption in different device functionalities (e.g. data processing/communication)
- a detailed analysis of the parameters that influence energy consumption in IoT systems, as well as the results from their integration into the energy-aware model
- a validation technique for energy-consumption and battery lifetime requirements of an application design through SMC

The analysis of parameters that affect energy consumption through our approach allows for obtaining trustworthy design decisions concerning the scenarios that have a predominant effect in the energy efficiency of the IoT devices and the overall system.

The rest of the paper is organized as follows. Section 2 provides a brief introduction in the Contiki IoT ecosystem and the techniques for software-based energy management through the powertrace, as well as in the BIP framework, which is used for rigorous system design. Section 3 illustrates the proposed method for energy-consumption management in IoT systems, which is later used in Section 4 to validate

[1] http://zolertia.com/sites/default/files/ Zolertia-Z1-Datasheet.pdf
[2] https://wirelesssensornetworks.weebly.com/blog/tmote-sky
[3] http://openmote.com/
[4] http://www.ti.com/ww/en/wireless_connectivity/sensortag/

IoT system requirements and provide valid bounds for the lifetime of IoT devices. Finally, Section 5 provides conclusions and perspectives for future work.

2 Background

2.1 Contiki powertrace

Powertrace [7] is a Contiki library that allows the annotation of Contiki programs with primitives, for monitoring the energy flow in IoT devices. It identifies four individual operating modes that contribute to a device's energy consumption:

- Low Power (LPM): the device is usually idle waiting for an event
- CPU: time duration that the microcontroller is used for calculations/data processing
- Radio transmission (Tx): indicating data transmission
- Radio reception (Rx): indicating data reception

The energy consumption varies according to the time that a device remains in each of the above modes. In order to measure this time, the library provides code primitives that can be used for every IoT device type. A data logger is used to store the data, which supports energy analytics. Characteristic examples of such analytics are the duty cycle or the device lifetime. The former refers to the percentage of time that a device remains in one operating mode, whereas the latter refers to the total time duration that a device operates autonomously. The period that powertrace uses to measure and log the data can be configured by the user and has an impact on the performance and accuracy of the mechanism. Finally, the energy calculation in powertrace also supports hardware-specific parameters, such as the real-time timer (RTIMER [5]) that is used to measure the hardware clock cycles of the device per second.

2.2 The BIP component framework

BIP (Behavior-Interaction-Priority) [1] 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 the interacting components. Priorities are used to filter amongst possible interactions and to steer system evolution so as to meet performance requirements, e.g. to express scheduling policies. A set of atomic components can be composed into a generic compound component by the successive application of connectors and priorities.

BIP is supported by a rich toolset including tools that are used to check stochastic systems, through the *Statistical Model Checking* (SMC) technique. SMC was proposed as a means to cope with the scalability issues in numerical methods for the analysis of stochastic systems. Consider a system model M and a set of requirements, where each requirement can be formalized by a stochastic temporal property ϕ written in the Probabilistic Bounded Linear Temporal Logic (PBLTL) [10]. SMC applies a series of simulation-based analyses to decide PBLTL properties of the following two types:

1. *Is the probability $Pr_M(\phi)$ for M to satisfy ϕ greater or equal to a threshold θ?* Existing approaches to answer this question are based on hypothesis testing [13]. When $p = Pr_M(\phi)$, to decide if $p \geq \theta$, we can test H: $p \geq \theta$ against K: $p < \theta$. Such a solution does not guarantee a correct result but it allows to bound the error probability. The strength of a test is determined by the parameters

[5] http://anrg.usc.edu/contiki/index.php/Timers#Step_5_-_Introduction_to_rtimer

(α, β), such that the probability of accepting K (resp. H) when H (resp. K) holds is less than or equal to α (resp. β). However, it is not possible for the two hypotheses to hold simultaneously and therefore the ideal performance of a test is not guaranteed. A solution to this problem is to relax the test by working with an indifference region (p_1, p_0) with $p_0 \geq p_1$ ($p_0 - p_1$ is the size of the region). In this context, we test the hypothesis $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$ instead of H against K. If the value of p is between p_1 and p_0 (the indifference region), then we say that the probability is sufficiently close to θ, so that we are indifferent with respect to which of the two hypotheses K or H is accepted.

2. *What is the probability for M to satisfy ϕ?* This analysis computes the value of $Pr_M(\phi)$ that depends on the existence of a counterexample to $\neg\phi$, for the threshold θ. This computation is of polynomial complexity and, depending on the model M and the property ϕ, it may or may not terminate within a finite number of steps. In [10], a procedure based on the Chernoff-Hoeffding bound [11] was proposed, to compute a value for p', such that $|p' - p| < \delta$ with confidence $1 - \alpha$, where δ denotes the precision.

The SMC of BIP models is automated by the SMC-BIP tool [17] that supports both types of PBLTL properties. The tool accepts as inputs the PBLTL property, a model in BIP and a couple of confidence parameters. The tool provides a verdict in the form of the 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 Rigorous design of energy-efficient IoT systems

The overall flow of our method is presented in Figure 1. The process depends on an XML parameter configuration input file that extends the WPAN network configuration given in [14] as well as on the application design expressed in a Domain Specific Language (DSL) [15]. The third input is the requirement specification, which contains user requirements regarding energy constraints for the IoT application. The method proceeds throughout the steps described below:

1. **Translation for the generation of the Contiki Simulation Configuration**: This step leverages the XML-based configuration with parameters that affect the energy consumption in an IoT application, as they are presented in Section 3.1. The configuration is systematically translated in order to produce all the necessary configuration files for the deployment of a Contiki IoT application as well as the Contiki simulation configuration (CSC) file, with the system architecture that is simulated within the native Contiki simulation environment (Cooja [8]). The DSL in [15] is also used to provide all the necessary application-specific parameters during the translation stage.
2. **Transformation for the System Model**: The actions comprising this step are two-fold. First, the DSL application description of the previous step is used to form an Application Model, which is later enhanced with the OS/kernel model, that is formed from the BIP IoT component library [15]. The combination of the two models is performed through the addition of the application mapping in the DSL [15], that specifies the deployment of application modules onto the systems nodes.
3. **Code generation**: The CSC file from step 1 along with the DSL application description are used to generate deployable code. The code is annotated with energy characteristics to allow the calculation of the energy consumed in each operating mode of powertrace. The code is accordingly simulated in Cooja with the addition of the powertrace library. The simulation result is used for the energy characterization in the next step.
4. **Energy characterization**: The analysis of the energy-oriented behavior and characteristics leads to the construction of an energy model representing the operating modes for each device, as well as

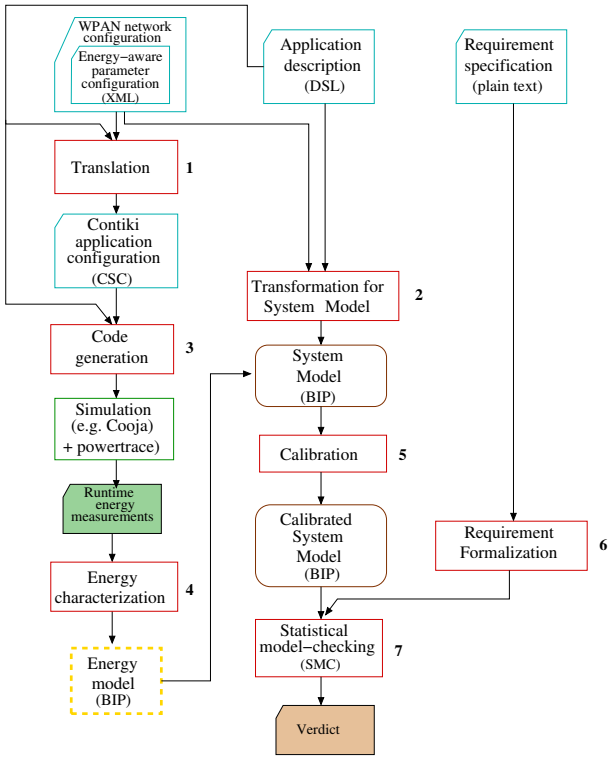

Figure 1: The proposed design method

the total amount of energy in the system. This model includes all the influential hardware/software energy constraints, derived from the simulation in the previous step. Those constraints are added in the energy model in the form of probabilistic distributions using a distribution fitting technique similar to the one presented in [14].

5. **Calibration for the construction of an energy-aware System Model**: This step concerns with the addition of parameters for the runtime characterization of the IoT application with respect to the BIP System Model as well as the generation of glueing code for the combination of the BIP System Model with the energy model obtained from the previous steps. The combination leads to the construction of the Calibrated BIP System Model, which allows to analyze energy aspects and to evaluate energy requirements.

6. **Requirement formalization:** This step concerns with the process of expressing a requirement with temporal logic properties. The properties are derived from the input requirement specification, where they are expressed in natural language. The resulting properties of this step are used as input in step 7.

7. **Statistical model checking (SMC):** In order to verify the model against the requirements we use SMC. The resulting verification verdict allows to find tight bounds for the energy consumed in the IoT application as well as in the individual devices.

3.1 Energy-aware parameter configuration

As a first step of the proposed method, the XML-based energy-aware parameter configuration is translated into the Contiki application configuration files. To better understand the importance of the selected

parameters for the model we hereby introduce the three categories of parameters that affect the overall energy consumption, namely (i) the application layer, (ii) the MAC layer and (ii) the physical layer parameters. Each category introduces parameters that depend on each other, but there are no interdependencies between the different categories. For instance, the choice of the IoT application protocol is independent from the choice of the duty-cycling mechanisms in the MAC layer.

MAC layer: The Contiki network stack uses a Radio Duty Cycling (RDC) mechanism, allowing a device radio to remain active and listen only for certain periods in time. The remaining time it moves in the LPM mode, where it cannot receive any transmitted packet. This mechanism reduces the consumed energy, as the radio remains in listening mode for significantly less time. RDC mechanisms are divided into 1) synchronous and 2) asynchronous, according to the technique that is used for awakening the IoT device. The former allow to maintain a Time-Division Multiple Access (TDMA) mechanism period, with which the device wakes up only in the beginning of the period to receive any incoming packets for a fixed time interval and afterwards switches to LPM mode. The latter allow a device to receive only the packets destined for it. This is possible through a Sender-initiated mode, where the sender transmits frequently a preamble packet, that is acknowledged only when the Receiver is awake, or a Receiver-initiated, where the Receiver transmits broadcast Probe Packet to all network nodes to indicate that it is awake. Contiki implements various mechanisms for RDC that are based on the two categories and have different characteristics. These characteristics are considered as the main parameters that influence the energy consumption in RDC and are:

1. *RDC protocol*: This parameter refers to the protocol that is used on top of the MAC protocol in the Contiki network stack, to implement the duty cycling for the IoT devices. The selection of this protocol affects significantly the overall consumed energy, as it allows the device to switch between the operating modes with different mechanisms. Therefore, the choice is also present as one of the parameters of the XML-based energy configuration. In Contiki, there are four allowed values for this choice, namely the ContikiMAC, the X-MAC, the Low Power Probing (LPP) [8] as well as the no use of RDC protocol named as nullRDC. ContikiMAC is a protocol based on the principles behind low-power listening but with better power efficiency. X-MAC is based on the original X-MAC protocol [5], but has been enhanced to reduce power consumption and maintain good network conditions. Contiki's LPP is based on asynchronous receiver-initiated transmission scheme protocol, where the devices wake up and send a probe to inform other devices on their receiving availability.
2. *RDC frequency*: This parameter specifies the frequency with which the IoT devices wake-up to check the channel for any packets whose transmission is pending through their RDC mechanism. Channel checking is accomplished by transmitting probe packets to verify any existing activity. In the presence of channel activity they remain in the Rx operating mode to receive any transmitted packets from the other network nodes, otherwise they switch to the LPM mode for another duty-cycling period. By remaining longer in Rx mode an IoT device has increased energy consumption, therefore we consider the RDC frequency as a significant parameter in the energy parameter XML configuration.
3. *Packet retransmissions*: IoT applications are often prone to transmission errors in the MAC layer that are leading to successive loss of packets, which are usually caused by extensive loss of bandwidth. For this reason wireless MAC-layer IoT protocols usually include a mechanism for the retransmission of non-acknowledged packets in the MAC layer. This mechanism increases the IoT system reliability, but also keeps the IoT device in the Tx operating mode for longer time durations. Therefore, the overall energy consumed by an IoT device is increased, making this parameter part of the energy parameter configuration in our design method.

Application layer: A Contiki IoT application can use a variety of protocols for data communication. Each protocol offers different mechanisms and is characterized by different energy consumption. It is also possible to introduce a different size for the packets to be transmitted. For this reason we define the application layer parameters as follows:

4. *Application protocol*: This parameter refers to the protocol that is chosen for data exchange. This choice depends strongly on the application requirements i.e. safety critical applications require reliable data transfer through MQTT communication, whereas sensor applications require energy efficiency and faster communications, therefore they rather rely on CoAP communication. Nevertheless, since the protocols offer different mechanisms, this choice has a significant impact on the lifetime of an IoT device.
5. *Header size*: IoT applications usually include compression algorithms to reduce the packet header, in order to reduce the overall size of the exchanged message. Such algorithms ensure the minimal radio usage period for data transmission, therefore minimizing the cumulative energy of consumption. However, in IoT devices this is a trade-off since for the computation time of compression/decompression of the packets, the IoT device remains in the CPU operating mode. For these reasons, the header size is considered as a parameter in our input configuration file.

Communication medium: IoT applications usually contain wireless devices that communicate over a communication medium that is prone to errors, such as collisions that impact the energy consumption. This category includes the following parameters:

6. *Radio interference*: This parameter is related to the presence of interference in the communication medium as a form of additive noise from simultaneous transmissions with the same radio frequency from proximity networks. Interference leads to increased packet collisions that can impact the energy consumption in the nodes, as they remain in the Tx operating mode for longer time durations. This parameter is added to the energy parameter configuration by introducing and properly configuring the disturber mote type as a part of the Contiki IoT application [3].

When the previous analysis is expanded to the entire IoT system architecture we ought to find additional parameters impacting the energy consumption. Such a characteristic parameter is the routing protocol that is used along with IPv4/IPv6 in the networking layer of the Contiki protocol stack to ensure the connection of Wide Area Network (WAN) IoT networks (e.g. meshed or multi-hop). In this scenario IoT devices have to be configured as a border router [12] to connect distant networks, which requires the extensive use of communication bandwidth. The use of the Routing Protocol for Low power and Lossy Networks (RPL) routing mechanisms increases computation/communication time durations in the IoT devices as well as in the overall system.

3.2 Energy model

As a part of step 4 of the design flow, an energy model is derived reflecting the true energy consumption in the BIP System Model. The main equations used to compute energy constraints in the Contiki IoT environment are as follows. Initially, the duty cycle (D) as a device ratio is computed by:

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

where y indicates the operating mode for the IoT device and N_y the relative number of occurrences that the device visits the respective operating mode y, given that it cannot be in two modes within the same time interval. Based on the duty cycle of a device in a given operating mode, we can also derive the overall energy consumption (in Joule) over every device operating mode:

$$E_{total} = \sum_{\forall i \in D_{LPM}}^{N_{LPM}} I_{LPM} * V_{LPM} * \Delta t_{LPM_i} + \sum_{\forall j \in D_{Tx}}^{N_{Tx}} I_{Tx} * V_{Tx} * \Delta t_{Tx_j} + \sum_{\forall k \in D_{Rx}}^{N_{Rx}} I_{Rx} * V_{Rx} * \Delta t_{Rx_k} + \sum_{\forall z \in D_{CPU}}^{N_{CPU}} I_{CPU} * V_{CPU} * \Delta t_{CPU_z} + E_P \quad (2)$$

where in every tuple $\{Iy, Vy, \Delta t_y\}$ y indicates the operating mode as in Equation 1 and I, V, and Δ indicate respectively the current (in Ampere), voltage (in Volts) and time intervals in which the device remains in each operating mode. The sum is over the number of occurrences $N_{LPM}, N_{Tx}, N_{Rx}, N_{CPU}$ of the device visiting the respective operating mode y and D_y indicates the duty cycle for each mode. E_P indicates the energy consumed by the device peripherals (in Joule). Finally, the device lifetime (lf) is computed by:

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

where C_{batt} indicates the overall capacity of the battery for autonomous operation (in Ampere hours), Vcc the operating voltage (in Volts) and E_{total} the total amount of energy.

In Figure 2 we illustrate a fragment of the BIP energy model that is derived in Step 4 of our method. We also present in Listing 1 the code with associated actions as in the model to facilitate its description.

```
1  #include "contiki.h"
2  #include "powertrace.h"
3  #include <stdio.h>
4  PROCESS(power, "powertrace example");
5  AUTOSTART_PROCESSES(&power);
6  PROCESS_THREAD(power, ev, data)
7  {
8    static struct etimer et;
9    PROCESS_BEGIN();
10   /* Start powertracing */
11   int RTIMER = 1; // 1 second reporting cycle
12   powertrace_start(CLOCK_SECOND * RTIMER);
13   etimer_set(&et, CLOCK_SECOND*t);
14   while(1) {
15     ....
16     if(etimer_expired(&et)) {
17       coap_init_message(request, COAP_TYPE_CON,
18         COAP_POST, 0);
19       coap_set_payload(request, (uint8_t *)msg,
20         sizeof(msg) - 1);
21       ...
22       COAP_BLOCKING_REQUEST(&server_ipaddr,
23         REMOTE_PORT, request, client_chunk_handler);
24   }
25 }
26 PROCESS_END();
```

Listing 1: Contiki powertrace client

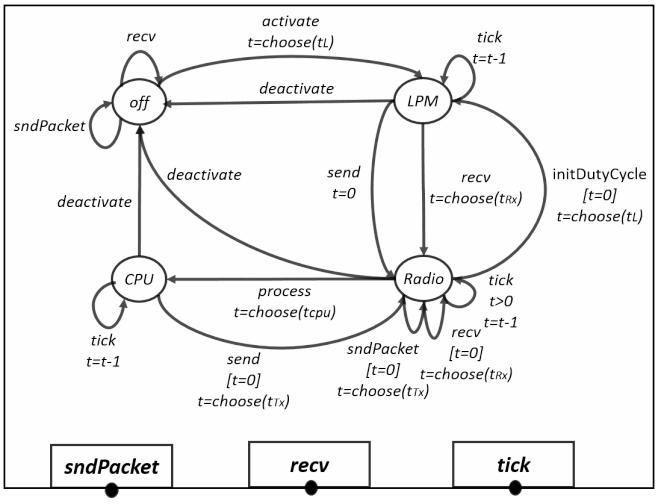

Figure 2: BIP energy model

By using the *powertrace_start* command the radio is activated and it immediately switches to the LPM operating mode. This is illustrated in Figure 2 through the *activate* transition. In lines 17-18 the CoAP message is initiated, which triggers the *sndPacket* transition in the model. Before a packet is transmitted the device has to initialize its header and payload as its shown in lines 17-20. This is represented with the *process* transition in the model, which switches it to the CPU state (i.e. indicating the CPU processing mode). When processing is finished the packet is transmitted through the *sndPacket* transition. When no further transmissions or receptions (through the transition *recv*) of packets take place, the model will return to the LPM state for another duty-cycle through the *initDutyCycle* transition. The

durations considered in every state or transition of the model are selected from probabilistic distributions that are gathered from profiling the generated code that is simulated in Cooja [14]. The gathered distributions correspond to each one of the powertrace operating modes. Finally, the model contains 3 exported transitions (i.e. *sndPacket*, *recv*, *tick*) to interact with the OS/Kernel components of the BIP system model [15] that is constructed in Step 2 of our method.

4 Case-study: Energy-aware building management system

In this section we present the case-study for validating our method in the context of a Building Management System (BMS). The application aims at automating the operation of the basic structures of a building installation, as well as at the remote control of buildings through a WAN network that consists of multiple WPAN networks, one for each building floor. Such an architectural setup is employed by new-generation building facilities. In particular, the case-study describes a company building, consisting of rooms to which workers have access during working-hours (8.00 till 18.00).

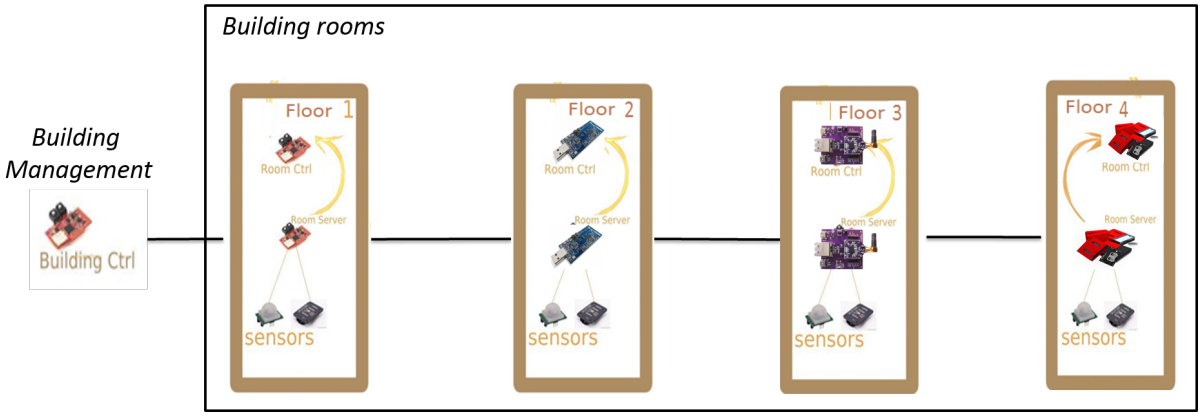

Figure 3: Node topology in the building management system

Figure 3 presents the heterogeneous IoT system architecture, where each floor has two unique IoT devices in the roles of a floor controller and floor server. The architectural heterogeneity allows to analyze energy consumption in a variety of IoT devices, such as Zolertia Z1 (level 1), Sky mote (level 2), OpenMote (level 3) and SimpleLink Sensortag (level 4). Each set of floor devices communicates its state to the lower-level and the lowest level (floor 1 in Figure 3) forwards all the floor states to the Building Management device located in level 1. This device forms the central system supervisor. It is notified by all room controllers for their current status and can be accessed or managed remotely as part of the WAN network. The state of each floor is determined by a set of sensor/actuator resources present in the floor server devices. In this case-study, we used the common resources amongst the four considered device types i.e. temperature sensor, humidity sensor, motion sensor, light sensor/actuator, alarm actuator, light actuator, thermostat actuator.

Concerning the system functionality, each floor server is a node in which every sensor/actuator is represented as a REST endpoint through which the floor controller can at any time know the current state of the room. For example, for the temperature sensor the floor controller monitors the temperature in the room and in case it exceeds the user-defined upper or lower limits, then it switches on the thermostat. Additionally, the building also employs an energy-saving mechanism, in order to automatically diminish the thermostat limits during non-working hours. Moreover, the floor controller is also subscribed to the motion sensor, as well as to the light sensor/actuator, in order to detect motion and open the lights if

the ambient light in the environment is not sufficient. The corresponding lights that open as a result of motion detection are linked to the location, where motion was detected.

4.1 Application of the proposed method

We focus now on the individual steps of the design flow in Figure 1 for the outlined BMS application.
Step 1: Translation of the energy parameter configuration
The parameters in Section 3.1 that influence the energy consumption obtain their actual values based on the application requirements. For the BMS application, we were based on the default values taken from the kernel libraries of the Contiki OS (Table 1). Moreover, to better demonstrate the impact of the parameters, we added as a part of our simulations a variation range. The following table introduces this variation range along with the default values in the Contiki OS.

Energy model parameter	Default value	Variation range
RDC protocol	X-MAC	[Contiki-MAC, X-MAC, LPP, nullRDC]
RDC frequency	8 Hz	[2-32] Hz (even number)
Packet retransmissions	4	[0-5] $\in \mathbb{Z}$
Service protocol	CoAP	[CoAP, MQTT, HTTP]
Header size	48 bytes	[32-64] bytes (even number)
Interference	0	[0-1] $\in \mathbb{R}$

Table 1: Energy parameters of the XML configuration

Accordingly, in steps 2 and 3 we used the described XML configuration along with the DSL to generate the Contiki application as well as the BIP System Model.
Step 4: Energy characterization
Due to the device heterogeneity for each floor in the BMS application, we could only apply the distribution fitting technique to energy data coming from the same device type. We then fitted the data into Poisson distributions for Tx and CPU modes, whereas the energy for Rx and LPM mode followed a normal distribution during the course of a day. The resulting distributions of this step are used in Step 5 to calibrate the model of Figure 2.
Step 6: Formalization of system requirements
We identified three requirements for the BMS system, from which only the first one concerns the IoT device lifetime, whereas the remaining two concern the IoT device duty-cycle in different operating modes. These requirements are:
Requirement 1. Device lifetime should be at least 1 week.
Requirement 2. The duty-cycle in the LPM mode should remain higher than 90% during working hours.
Requirement 3. The duty-cycle in the Rx mode should not exceed 20% during working hours.

4.2 Experiments

In this section we demonstrate the experiments for evaluating the aforementioned requirements. The current and voltage values for each operating mode that were used for the calculation of the total energy, duty cycle and device lifetime from the equations of Section 3.2 were obtained from the IoT devices' datasheet.

We use the parameters of Section 3.1 to demonstrate the impact of a certain parameter to the overall energy consumed in a device. To this end, we focused on a certain category parameter and experimented with all variations of the other parameters in the same category, while the parameters in different categories were set in their default values (Table 1). This is due to the independence between the categories.

The system requirements were validated through the continuous simulation of the BIP system model for a working week, where we used SMC to measure the probability for satisfying the requirements of Section 4.1.

Requirement 1. We evaluated the property $\phi_1 = lf \geq 168$, where 168 indicates the sum of hours during the week. For this experiment we used two scenarios the first being with the default values of Table 1 and the second with different sets of the parameters of the same table. For the first scenario we found that $P(\phi_1) = 0.9$, whereas for the second scenario we have conducted several sets of experiments. These experiments allowed us to demonstrate the contribution of the energy parameters to the device lifetime (Figure 4). The greatest lifetime impact is observed in experiments involving the RDC protocol parameter as the difference between maximum and minimum battery autonomy is 130 hours, whereas with radio interference the difference is 148. On the other hand, we also observe that the variation in the retransmissions parameter has no impact on the system, as the WAN network of our case-study is formed by several smaller WPAN networks with a small number of IoT devices. Furthermore, this Figure allows the system designer to know the values to be used for the parameters of Table 1 to satisfy this requirement.

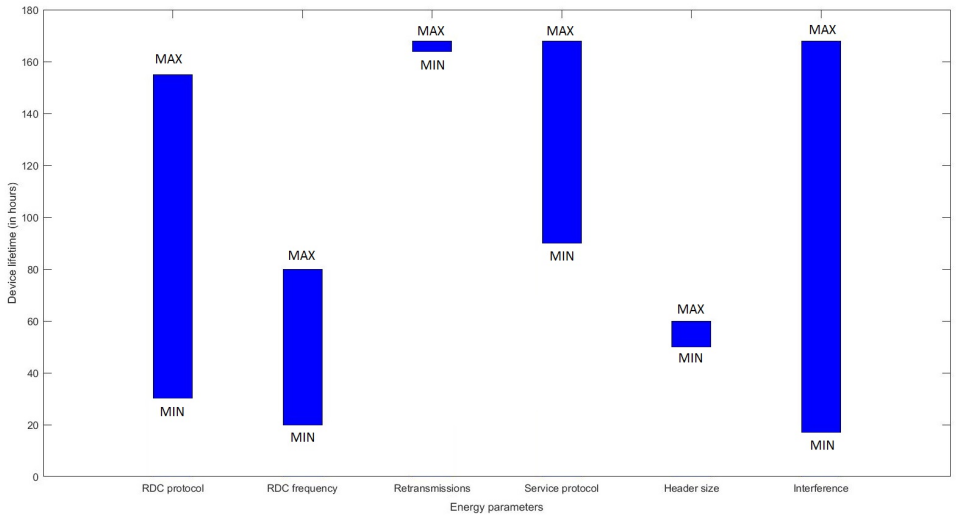

Figure 4: Device lifetime for different variations of energy parameters

Requirement 2. We verified the property $\phi_2 = D_{LPM} \geq 90\%$. The property is significantly influenced by the RDC protocol parameter. In particular, it holds for the LPP RDC protocol ($P(\phi_2) = 1$) as it is illustrated by Figure 5, which focuses on the duty cycle of each operating mode. In the same Figure, since the energy in certain modes (i.e LPM) during our experiments was significantly higher than in all the remaining operating modes we chose to present our results in logarithmic scale. The reasoning behind the energy saved with LPP in LPM is that the sum of packets is exchanged during small and continuous time intervals, while in all the remaining time intervals the device switches to a deep sleep mode. Concerning the other RDC protocols for ContikiMAC and XMAC, we found $P(\phi_2) = 0.5$ and $P(\phi_2) = 0.7$ respectively. The difference between the two RDC protocols is explained by the packet transmission in ContikiMAC rather than a pulse as in X-MAC, in order to activate the wake-up signal. Finally, for the nullRDC protocol $P(\phi_2) = 0$ as the radio remains in Rx for long time intervals.

Requirement 3. We verified the property $\phi_3 = D_{Rx} \leq 20\%$. The property holds when the chosen RDC protocol is LPP (Figure 5). However, with the two other protocols, the observed probability was $P(\phi_3) = 0.8$ for XMAC, $P(\phi_3) = 0.6$ for ContikiMAC and $P(\phi_3) = 0$ for the nullRDC. This is due to the difference in awakening frequency of the device to listen to incoming packets, especially during working hours when the data exchange is intense (Figure 5).

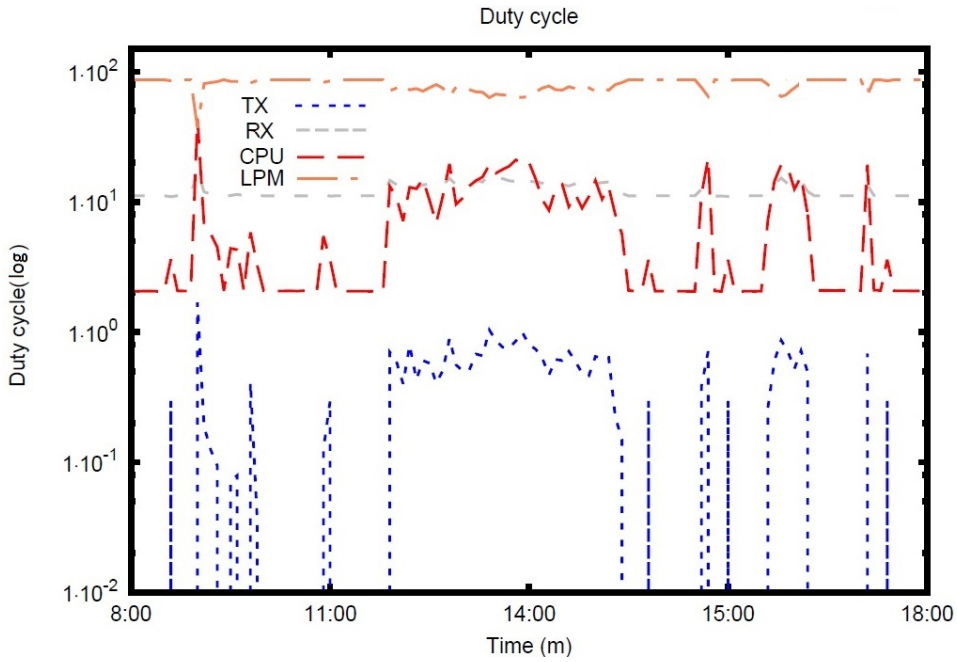

Figure 5: Duty cycle of the Zolertia floor controller during a working day (in logarithmic scale)

5 Conclusion

We presented a novel method for optimizing the energy configuration in IoT applications, as well as in the individual IoT devices. The method is based on the principles of rigorous system design by using the BIP component framework. Currently, the provided support concerns with the design of REST service-based applications which are deployed on nodes running the Contiki OS. The method takes as input the application design description in DSL and an XML-based set of energy parameters and generates a system model in BIP for validating requirements related to energy characteristics. The system model is calibrated with energy constraints that are obtained by the simulation of the code generated from the application description. The calibrated model is afterwards used to validate the requirements through Statistical Model Checking (SMC).

As a proof of concept, the described method has been applied to a building management system. The system consists of several subsystems deployed in multiple floors of a smart building facility using several well-known IoT devices. We have verified the energy-related requirements concerning the device lifetime, as well as the duty-cycle of the devices and the overall system. The results allow us to optimize the energy consumption on the system and increase the device lifetime.

As future work, it is worth to consider the aspect of remote control in the building, as well as its impact on the overall energy consumption. This can be accomplished through the presence of border routers and RPL routing mechanisms in the IoT application. In this direction, an important architectural characteristic of IoT systems is fog and cloud computing [4], where a significant part of the computation is no longer handled by the resource-constrained IoT devices. Hence, the overall energy consumption in the system is reduced. Additionally, the Building Management Controller that was presented in the case-study can also perform control actions based on the data it gathers, such as shutting down the heating and lighting system if there is no motion for certain hours during the day.

References

[1] Ananda Basu, Bensalem Bensalem, Marius Bozga, Jacques Combaz, Mohamad Jaber, Thanh-Hung Nguyen & Joseph Sifakis (2011): *Rigorous component-based system design using the BIP framework*. IEEE software 28(3), pp. 41–48.

[2] Luca Benini, Robin Hodgson & Polly Siegel (1998): *System-level power estimation and optimization*. In: Proceedings of the 1998 international symposium on Low power electronics and design, ACM, pp. 173–178.

[3] Carlo Alberto Boano, Kay Römer, Fredrik Österlind & Thiemo Voigt (2011): *Demo abstract: Realistic simulation of radio interference in COOJA*. In: European Conference on Wireless Sensor Networks (EWSN 2011), February 2011, Bonn, Germany.

[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu & Sateesh Addepalli (2012): *Fog computing and its role in the internet of things*. In: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, ACM, pp. 13–16.

[5] Michael Buettner, Gary V Yee, Eric Anderson & Richard Han (2006): *X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks*. In: Proceedings of the 4th international conference on Embedded networked sensor systems, ACM, pp. 307–320.

[6] Walter Colitti, Kris Steenhaut, Niccolo De Caro, Bogdan Buta & Virgil Dobrota (2011): *REST enabled wireless sensor networks for seamless integration with web applications*. In: Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on, IEEE, pp. 867–872.

[7] Adam Dunkels, Joakim Eriksson, Niclas Finne & Nicolas Tsiftes (2011): *Powertrace: Network-level power profiling for low-power wireless networks*.

[8] Joakim Eriksson, Fredrik Österlind, Niclas Finne, Nicolas Tsiftes, Adam Dunkels, Thiemo Voigt, Robert Sauter & Pedro José Marrón (2009): *COOJA/MSPSim: interoperability testing for wireless sensor networks*. In: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), p. 27.

[9] Kyriakos Georgiou, Samuel Xavier-de Souza & Kerstin Eder (2017): *The IoT energy challenge: A software perspective*. IEEE Embedded Systems Letters.

[10] T. Hérault, R. Lassaigne, F. Magniette & S. Peyronnet (2004): *Approximate probabilistic model checking*. In: Verification, Model Checking, and Abstract Interpretation, Springer, pp. 73–84.

[11] W. Hoeffding (1963): *Probability inequalities for sums of bounded random variables*. Journal of the American statistical association 58(301), pp. 13–30.

[12] Matthias Kovatsch, Simon Duquennoy & Adam Dunkels (2011): *A low-power CoAP for Contiki*. In: Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on, IEEE, pp. 855–860.

[13] Axel Legay, Benoît Delahaye & Saddek Bensalem (2010): *Statistical model checking: An overview*. In: Runtime Verification, Springer, pp. 122–135.

[14] Alexios Lekidis (2015): *Design flow for the rigorous development of networked embedded systems*. Ph.D. thesis, Université Grenoble Alpes.

[15] Alexios Lekidis, Emmanouela Stachtiari, Panagiotis Katsaros, Marius Bozga & Christos K Georgiadis (2018): *Model-based Design of IoT Systems with the BIP Component Framework*. Software Practice and Experience.

[16] Borja Martinez, Marius Monton, Ignasi Vilajosana & Joan Daniel Prades (2015): *The power of models: Modeling power consumption for IoT devices*. IEEE Sensors Journal 15(10), pp. 5777–5789.

[17] Ayoub Nouri, Saddek Bensalem, Marius Bozga, Benoit Delahaye, Cyrille Jegourel & Axel Legay (2015): *Statistical model checking QoS properties of systems with SBIP*. International Journal on Software Tools for Technology Transfer 17(2), pp. 171–185.

[18] Xavier Vilajosana, Qin Wang, Fabien Chraim, Thomas Watteyne, Tengfei Chang & Kristofer SJ Pister (2014): *A realistic energy consumption model for TSCH networks*. IEEE Sensors Journal 14(2), pp. 482–489.

[19] Hai-Ying Zhou, Dan-Yan Luo, Yan Gao & De-Cheng Zuo (2011): *Modeling of node energy consumption for wireless sensor networks*. Wireless Sensor Network 3(01), p. 18.