Modelling and Analysis of a Sigfox based IoT Network using UPPAAL SMC

Naeem, Muhammad; Albano, Michele; Larsen, Kim G.; Nielsen, Brian; Høedholt, Anders; Laursen, Christian Ø.

Published in:
IEEE Sensors Journal

DOI (link to publication from Publisher):
10.1109/JSEN.2023.3261667

Publication date:
2023

Document Version
Accepted author manuscript, peer reviewed version

Modelling and Analysis of a Sigfox based IoT Network using UPPAAL SMC

Muhammad Naeem, Michele Albano, Kim G. Larsen,
Brian Nielsen, Anders Høedholt and Christian Ø. Laursen

***Abstract*—Wireless sensor nodes are usually powered by batteries that have limited energy capacity. In many applications, the nodes are installed in inaccessible locations where they are problematic to replace or recharge. Therefore, energy optimisation is crucial for increasing the node's lifetime. This study presents a method for the analysis and prediction of the energy consumption of Sigfox-based Wireless Sensor Nodes. The method is illustrated in a use case where the nodes monitor the water level in drainage lines in cities to improve surface and wastewater management. We propose a formal model-based technique using the UPPAAL Statistical model checker tool to model and analyse the node's lifetime. Statistical model checking provides a highly scalable technique for performance analysis of complex cyber-physical systems. The model captures the energy-related behaviour of the node, the Sigfox radio specification, and the sensor, each parameterised with values from the device's datasheets. Furthermore, we calibrate the model using measurements obtained from real-world hardware. Finally, we evaluate a collection of strategies to optimise the battery lifetime of the node. We simulate the model with a 10,000 mAh battery, and the results indicate that we can extend the node's lifetime from 202 days to 2.71 years using our most optimised transmission strategy.**

***Index Terms*—Battery lifetime, Wireless sensor nodes, LPWAN, Design space exploration, Statistical model checking, Optimised transmission strategies**

I. INTRODUCTION

RAPID developments in the Internet of Things (IoT) are increasing its applications and deployments. The IoT includes various wireless sensor nodes deployed in different environments to collect data. The number of deployed IoT devices has grown dramatically, and a research study expected it to be in the range of tens of billions by the end of 2021 [1].

The sensor nodes are often battery-operated and typically deployed in remote and harsh environments where charging or replacing their batteries is inconvenient and expensive. Therefore, it is crucial to optimise the node's lifetime by developing efficient data processing and transmission strategies. Low-Power Wide-Area Network (LPWAN) protocols such as Sigfox, NB-IoT and LoRaWAN were developed to fulfil the criteria of IoT networks [2], [3], [4]. These protocols provide low data rate long-range communication while guaranteeing a very low energy consumption.

The Donut project (Distributed Online monitoring of the Urban water cycle) [5] developed a Sigfox-based sensor network and deployed it in drainage pipelines for monitoring the water level. The increased understanding of the water flow enabled by this sensor network may improve the sustainability of cities by, e.g., preventing overflow scenarios, planning the development and maintenance of the drainage system, and locating sources of floods. The sensors were powered by batteries, and replacing them often was not an option since it represented a time-consuming (and expensive) effort. Using energy harvesting techniques was unfeasible in most deployment sites: solar panels on sewerage manhole covers must be rigid enough to bear the potential automotive and pedestrian traffic load; separate solar panels require wiring and are vulnerable to vandalism; energy harvesting from the water flow will cause the generator to clog; vibration can not produce sufficient energy for this use case. Thus, optimising the sensor's energy usage became a priority to extend the lifetime of the sensor network.

In [6], we developed energy models of the designed prototype of the Sigfox-based sensor node to be used with Statistical Model Checking (SMC) [7], which allows evaluating the performance of probabilistic systems. We used the UPPAAL SMC model checking tool [8] to estimate the sensor node's operational lifetime with a given battery and provide an insightful breakdown of energy consumption in different system states. Furthermore, we investigated different transmission strategies and their impact on the sensor node's

The manuscript is submitted on Monday 20th March, 2023. Research funded in part by the Danish Innovationsfonden agency (DoNUT project, grant nr. 7076-00123B), and it is part of the Validation WP of the ERC Advanced Grant LASSO (Learning, Analysis, Synthesis and Optimisation of Cyber-Physical Systems).

Muhammad Naeem, Michele Albano, Kim G. Larsen and Brian Nielsen are affiliated with Aalborg University, Denmark (e-mails: mnaeem, mialb, kgl, bnielsen ;@cs.aau.dk). Anders Høedholt and Christian Ø. Laursen are affiliated with MONTEM A/S, Aarhus, Denmark (e-mails: a.hoedholt, c.laursen ;@montem.io).

lifetime.

This paper presents the revised and extended version of our prior work, overcoming a threshold problem due to weaving in water level and data error rate in the data compression strategy. We overcome the threshold problem by increasing the threshold level and applying Low Pass Filter(LPF) on measured data. Further, we simulate the model for multiple data sets obtained from sensor nodes deployed geographically in different locations to investigate the data compression strategy's effectiveness and calculate a more realistic average lifetime of a sensor node. We propose a new strategy that combines the effects of the previous ones. It also includes the detailed model of the case study as well as an example in UPPAAL SMC for a better demonstration. When evaluating the designed model with a $10,000\ mAh$ battery, our results indicate that we can extend the node's lifetime from 202 days to 2.71 years using our proposed transmission strategies.

The rest of the paper is structured as follows: Section II reviews previous work and introduces the Sigfox protocol and UPPAAL SMC. Section III describes the Donut case study. Section IV presents our modelling in UPPAAL SMC. Section V thoroughly examines our investigations and their outcomes. Finally, Section VI draws conclusions on the topic at hand.

II. BACKGROUND AND RELATED WORK

This section presents the background for the results presented in the paper. In particular, it discusses related work and describes the Sigfox protocol and the UPPAAL toolset.

A. Related work

Some studies that address energy analysis of the Sigfox protocol exist, and they present simulation results or analytical models to predict the battery lifetime of a Sigfox-based network [3], [9]–[14], either using power characteristics described in datasheets, or measurements from the hardware implementation. Simple analytical models are often too abstract and risk missing essential nuances, resulting in unduly conservative or pessimistic projections.

In model-based development, abstract models are used to analyse the system's behaviour. One goal is to make this analysis early in the development cycle prior to actual implementation to reduce the cost of finding and correcting defects. A further goal is to explore different designs and strategies without needing costly development, coding and tests. The present work follows these ideas.

In [15], David et al. used UPPAAL SMC for energy-aware analyses of the Lightweight Media Access Control protocol (LMAC). Panigrahi et al. [16] used probabilistic algorithms to predict the battery life of remote embedded devices. Using the IEEE 802.16e standard, Xue et al. created an optimised energy protocol for mobile stations and used the SMC technique for verification and performance analysis of the protocol [17].

Model-checking is a class of formal techniques focused on designing, comparing and evaluating the correctness of systems [18]. Model-checking has been successfully used in many application areas [19]–[21] and has proven particularly useful in the early design stages when only a model or a

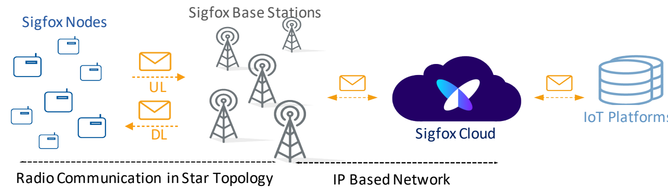


Fig. 1: Basic Architecture of the Sigfox network [6]

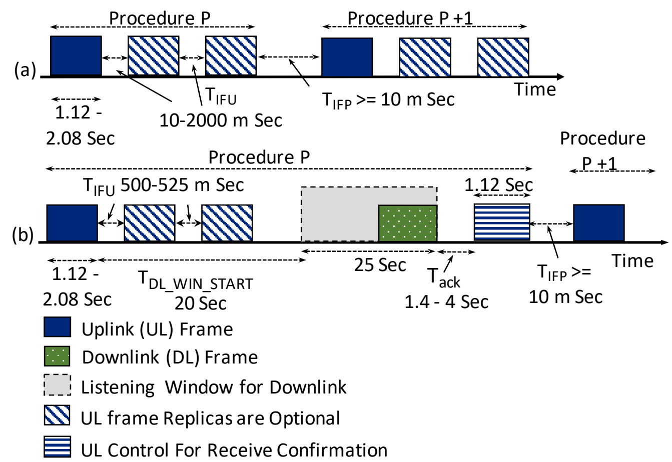


Fig. 2: Sigfox Procedures. (a) Uni-directional, (b) Bi-directional. [13]

blueprint of the product is available. SMC is an extension of model checking that focus on performance analysis of complex probabilistic systems [7].

Our model is based on the system's detailed realistic behaviour as described by a stochastic state-based formalism. The results are generated using the SMC technique, which produces statistically correct model sampling. The designed model includes all relevant energy-consuming states to reflect the system's behaviour realistically; we also include the battery self-discharge, which was not included in previous studies.

This literature review concludes that our study [6] is the first that uses model-checking approaches to analyse the Sigfox protocol. This requires a model but supports automated correctness checking thereof and especially avoids the programming cost and effort of a dedicated simulator.

B. Sigfox

Sigfox is a LPWAN protocol [2]. It is a patented network developed and operated by the Sigfox company. Figure 1 depicts the basic architecture of the Sigfox network, which is similar to the cellular network; each end node device directly communicates with the base station through a radio frequency link. From there, it is forwarded via IP based network to the IoT platform. Sigfox is primarily designed for M2M IoT applications with low data rates communication [22].

The Sigfox specifications vary by region. This case study is focused on the European region [23]. In Europe, it offers frequency bands from 868-868.6 MHz and $869.4 - 869.65$ MHz for transmission. The transmission baud rate is 100 bps for uplink (UL) and 600 bps for downlink (DL). Sigfox uses Ultra NarrowBand radio communication to achieve a long-link transmission with less power consumption.

A Sigfox node can send up to 6 UL messages per hour (144 messages/day) with up to 12 bytes of payload each and receive up to 4 8-byte DL messages/day. The Sigfox protocol supports two types of communication procedures: Unidirectional Procedure (UDP) and Bidirectional Procedure (BDP), see Figure 2. Using a UDP, a node sends a UL frame with max two replica frames. These replica frames are optional and may be sent to reduce packet loss probability. A small inter-packet delay T_{IFU} is used between the UL frames, and the node can start the next procedure after a minimum delay of T_{IFP}. In a BDP, a node sends UL frames similar to the UDP, and it also opens a listening window for 25 sec with a delay of $T_{DL\_WIN\_START}$ from the end of the first UL of the procedure. The node sends a control UL frame with a delay of T_{ack} for a receive confirmation. A node uses a control bit in the UL frame to select a procedure.

In Sigfox, only the node can initiate the communication, and it listens only after transmitting a message through a bidirectional procedure. So, the node can stay in sleep mode for up to 90% of the time to minimise energy consumption [22].

C. Statistical Model Checking (SMC) using UPPAAL SMC

SMC is an innovative technique introduced initially by Younes et al. [24] and Koushik et al. [7]. SMC is being used to estimate the qualitative and quantitative properties of probabilistic systems. It avoids the exhaustive state-space exploration problem of classic model checking, and it is less memory and time intensive comparatively. The basic idea of SMC is to create sufficient sample execution paths (runs) for a model and verify the model's formally specified requirements using statistical hypothesis testing and efficient probability estimation techniques. The *runs* are typically generated from a model with precise stochastic semantics. The runs may be generated using discrete-event simulation *techniques*, and they may be generated by existing discrete-event simulator *tools* [25].

UPPAAL SMC is an extension of the UPPAAL model checker [8]. It represents a system using networks of priced timed automata, and the automata model's behaviour depends on stochastic and non-linear dynamical properties.

1) Modelling in UPPAAL SMC : The modelling formalism of classic UPPAAL is timed automata [26], [27]. The UPPAAL tool offers a graphical user interface to describe a system as a network of concurrent timed automata. Each automaton instance is referred to as a process. A timed automaton is an extended state machine that is equipped with special variables called clocks, that advance automatically with a rate of 1. As shown in Figure 3, there are two basic model elements, a location and an edge (so-called transition) to specify states and actions, respectively. A location is represented by a circle; an initial state is indicated by a double lined circle. A location can be committed as well (denoted by a "C" in the location) which means that transitions from this state are executed without interleaving the action from other (non-committed) states. In other words actions from a committed state get priority w.r.t. non-committed states. This allows the easy description of atomic sequences. The purpose of an edge is to take some action. An edge may be guarded, i.e., executable if the edge condition evaluates to true. Executing an action may update variables and clocks and communicate with other process(es). Processes use synchronous channels for communication. A sender and a receiver processes use '`!`' and '`?`' signs, respectively with the relevant channel name.

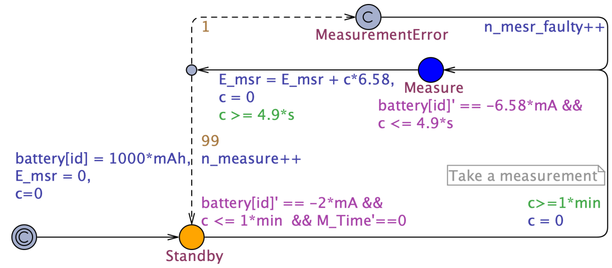


Fig. 3: Example model (Measurement action part of the main model)

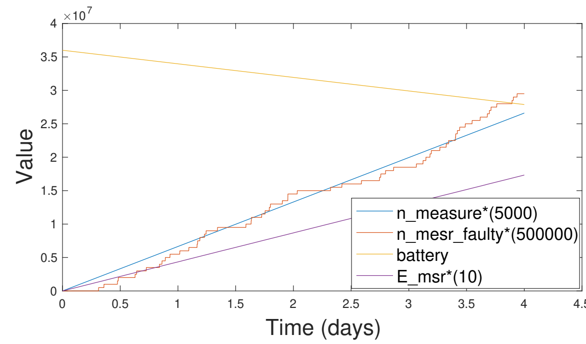


Fig. 4: Simulation results of example model. The value on y-axis represents counts for *n_measure* and *n_mesr_faulty*, and charge in mAh for *battery* and *E_msr*

Timed automata have formal semantics, and UPPAAL can verify formally whether functional requirements are satisfied by the model. The UPPAAL SMC is an extension of UPPAAL classic, and it uses stochastic hybrid automata [8] to model a system. In addition to the UPPAAL classic, it allows probabilistic weights to model the probabilistic choice between transitions (dotted lines). Furthermore, the clock rate can be controlled by means of ordinary differential equations.

Figure 3 shows a part of the node model automaton described in section IV-A to exemplify how systems are modelled within the UPPAAL SMC framework. The example model has three states and some transition actions between them. It models measuring water levels with a frequency of one minute. The `initial` state is committed, so the model immediately moves to the `standby` state after executing the variable assignments in the transition. The system remains in the `standby` state and moves to the `Measure` state when it needs to measure a value. Here is a probabilistic transition to the `Measure` state,

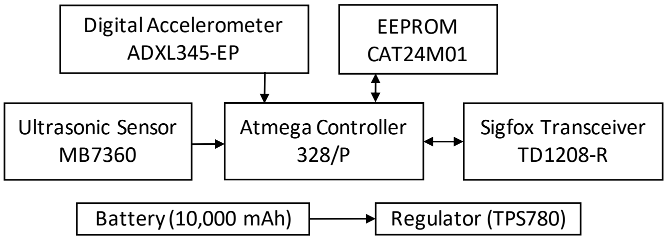


Fig. 5: Basic Hardware Architecture of the Sensor Node [6]

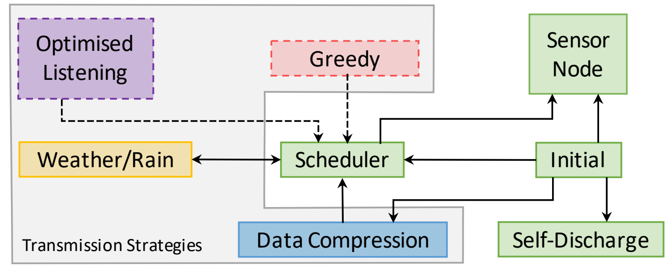


Fig. 6: Basic structure of the designed model in UPPAAL SMC

expressing that a sensor measurement is faulty with weight one and correct with weight 99. `c` is a local clock variable that controls the time spent in each state. `Battery[id]` is a clock variable indicating the remaining energy level, and its value decreases with each passing time unit with the rate of current consumption value in the particular state. `n-measure` represents the number of taken measurements, and `E-msr` represents the energy cost in the measuring state.

The query "*Simulate Query*: `simulate N [<=bound]{E1, .., Ek}`" was executed in UPPAAL SMC on the model shown in Figure 3 for a bound of four days duration to visualise the behaviour of the `n_mesr_faulty`, `n_measure`, `E_msr` and `battery[id]` variables, and it led to generating the plot shown in Figure 4.

III. CASE STUDY

A project of the Donut project partner, the Montem company, has built a prototype of a DWSN. Figure 5 depicts the basic hardware architecture of the prototype. As traditional water flow meters tend to clog when applied to sewage water, an ultrasonic sensor [28] is used to measure the water level. The Atmega controller [29] processes the burst of measurements, filters them to compute a water-level value for every measurement cycle and stores it in the EEPROM. These measurements are transmitted to the base station through the Sigfox transceiver (TD1207R/08R) [30]. The purpose of the digital accelerometer [31] is to facilitate positioning the sensor node in the correct position. Table I presents the DC power characteristics of all hardware components obtained from their datasheets. We considered the latest Sigfox radio specification EP-SPECS-1.5 presented in Feb 2020 [23].

IV. IMPLEMENTATION IN UPPAAL SMC

The energy-aware analysis of a Sigfox Sensor Node (SSN) is the main objective of this case study. The SSN controls its communication with the Sigfox base station, and the rest of the network does not affect the battery consumption of a sensor node. So, the designed model only considers all major energy-consuming activities associated with the SSN instead of the complete network.

To optimise the node operational lifetime on a given battery, we later in the section model four different transmission strategies: **Greedy**, **Optimised Listening**, **Weather-driven**, **Data Compression**, and **Combined Data Compression and Weather-driven**.

Our model consists of several sub-models interconnected using channels and shared global variables. The high-level structure is shown in Figure 6.

Four basic process automatons (`Initial, Sensor Node , Scheduler, and Self-Discharge`) model the behaviour of the SSN. Two further strategy control process automatons (`Weather Driven and Data compression`) represent the transmission strategies. The `Optimised Listening` and `Worst-Case Strategies` are part of the `Scheduler` automaton; we simulate these strategies by modifying the values of the variables that regulate the sending and measuring frequency.

The `Initial` automaton aims to trigger all other processes to an active state by using the `begin!` synchronous channel and initialising all variables and clocks.

A. Sensor-Node process automaton

Figure 7 depicts the `Node` automaton. It represents the major states (`Sleep`, `Standby`, `Sending`, `Listening` and `Measure`) and activities of the WSN. Normally, a node stays in `Sleep` state and moves to the `Standby` state when it needs to perform an action, be it either `Sending`, `Listening` or `Measure`. After completing an action, it moves to the `Standby` state if any other activity is scheduled; otherwise, it moves to the `Sleep` state. Lastly, it reaches the `Depleted` state when the battery level approaches zero.

In the node automaton, the `battery[id]` is a clock indicating the remaining energy level. `id` enables the simulation of multiple nodes. Its value decreases with each passing time unit with the rate of current consumption value depending on the particular state. Table I summarises the current consumption of each state. In our model, the base time unit is deci-second (ds) to avoid floating point values from the conditions on clocks because the UPPAAL tool else becomes unnecessarily slow. The value of `battery[id]` decreases by $4750mAds$ ($1.32mAh$) when a node spends $250ds$ in the listening state and consumes $19mA$ for each time unit. Table II describes the variables used in the *Node* automaton.

An incorrect measurement by the sensor has a 1% chance of occurring. With the `Measure` state, we add a probabilistic transition to reflect this behaviour. When the measured value is incorrect, the node measures it again.

TABLE I: DC Power Supply Characteristics of hardware

Sym	Description	Sigfox Min	Sigfox Max	Ultrasonic sensor	Atmega	Digital Accelerometer	Overall Min	Overall Max	Measured values
V_{DD}	Supply Voltages						$3.3V$	-	$3.3V$
I_{Sleep}	Sleep Current	$1.5\mu A$	$3.5\mu A$	0	$0.75\mu A$	$0.1\mu A$	$2.35\mu A$	$4.35\mu A$	$27.84\mu A$
I_{RX}	Receive Mode Current	$13mA$	$16mA$	0	$0.2mA$	$0.1\mu A$	$13.2mA$	$16.2mA$	$19.05mA$
I_{TX}	Transmission Mode Current	$50mA$	$50mA$	0	$0.2mA$	$0.1\mu A$	$50.2mA$	-	$47.03mA$
I_{App}	Continues Application Current	$10\mu A$	$10\mu A$	0	$0.2mA$	$0.1\mu A$	$210.01\mu A$	-	$4300\mu A$
I_{Msr}	Data Measuring Current	$2.55mA$	$3.45mA$	$3.1mA$	$0.2mA$	$23\mu A$	$5.873mA$	$6.773mA$	$6.58mA$

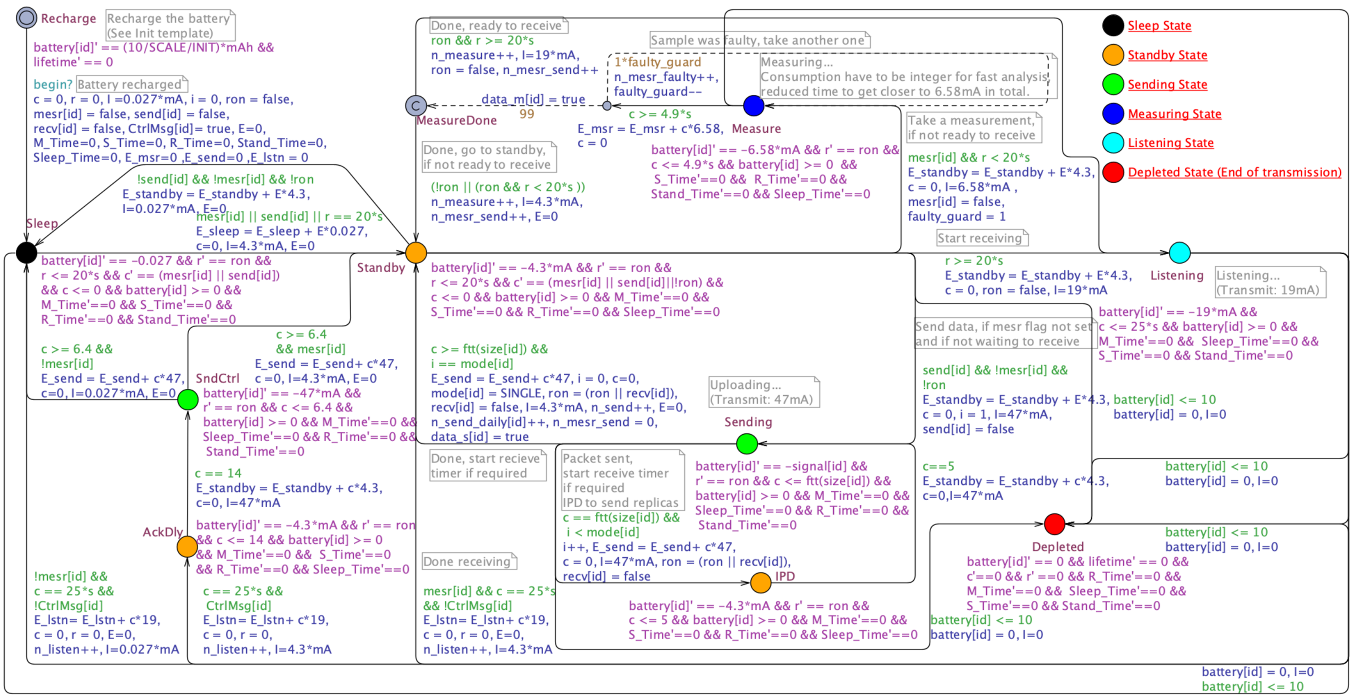


Fig. 7: Node automaton model [6]

TABLE II: Variables used in *Node* automaton. C, D, and B represents clock, double and boolean variable, respectively

Variable & Type		Description	
`Battery`	C	Represents the battery level of sensor node	
`c`	C	It is used to record the time spent in a state	
`r`	C	It adds a 20s delay for the listening.	
`M_Time`	D	It represents overall time spent in	measuring state
`S_Time`	D		sending state
`R_Time`	D		listening state
`Stand_Time`	D		standby state
`Sleep_Time`	D		sleep state
`E_msr`	D	It represents overall energy spent in	measuring state
`E_send`	D		Sending state
`E_lstn`	D		listening state
`E_standby`	D		standby state
`E_sleep`	D		sleep state
`IPD`	D	It is used as inter-packet delay in UL frames replicas	
`AckDly`	D	It's a time delay between DL and confirmation UL	
`mesr[id]`	B	It is used to control	measuring action
`send[id]`	B		sending action
`recv[id]`	B		listening action

B. Self-Discharge process automaton

This automaton models the behaviour of the battery self-discharge. In our case, the battery consumes around 3% of its electrical charge every month [32]. The model reduces approximately 0.1% of its energy as self-discharge every 24 hour based on Equation 1, corresponding to 3% in a month.

$$battery[id] = battery[id] - sdc\_rate[id] * (battery[id]/sdc\_time[id]) * 1 * day \quad (1)$$

$$sdc\_rate[id] = Self - DischargeRate = 3\% = 0.03$$
$$sdc\_time[id] = Self - DischargeTime = 1 * month$$

UPPAAL SMC can model more advanced battery models such as a Kinetic Battery Model (KiBaM) [33], but we do not use them, as our use-case does not have any intense current draw.

C. Scheduler process automaton

Figure 8 presents the `scheduler` automaton; it controls all activities of the node using boolean variables (`recv[id]`, `send[id]` and *mesr[id]*) depending on the transmission strategy. The `node` actions are guarded with these variables, and it uses `r,s` and `m` local clocks to schedule listening, sending and measuring, respectively, and uses `t` clock to reset the UL and DL frames counters after every 24 hour. The counters are used to abide by the Sigfox transmission restrictions.

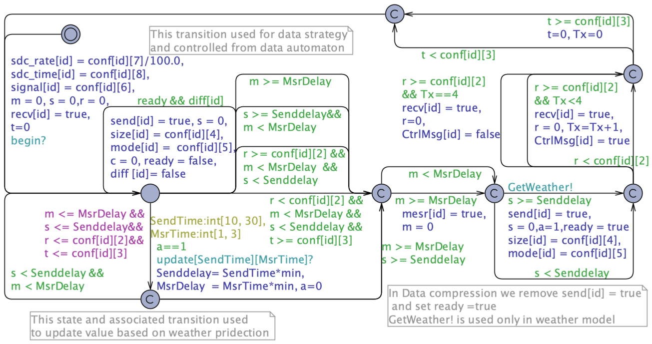


Fig. 8: Scheduler automaton model

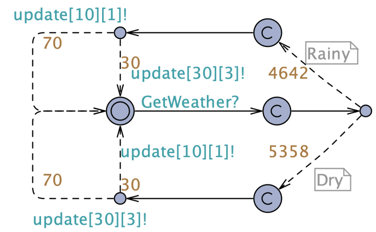


Fig. 9: Weather-Driven strategy process automaton model

To implement the Data compression strategy, we have modified the `scheduler` automaton to add two checks on the sending action shown in Figure 8. The first check is the minimum delay between two ULs because there is a restricted count/day. The `scheduler` enables `ready` (a boolean variable) $10min$ after sending a UL frame, which indicates that the node is ready to send. The second check is `diff` (a boolean variable) which is enabled by the Data compression process when it measures a significant variation in water level compared to the last sent value.

A transition is added with the ready state in the scheduler automaton to implement the Weather-Driven Strategy. Using this transition, the scheduler automaton is communicating with the Weather-Driven automaton using a channel `update[SendTime][MsrTime]?` to update the sending and measuring frequency based on the weather prediction.

D. Greedy strategy

The **Greedy** is a worst-case strategy; it simulates the SSN model for the maximum functional capacity of the Sigfox protocol. It schedules 144 UL (followed by BDP) and 4 DL frames per day at a max transmission power of $16dBm$, measuring water level every minute. All UL frames follow BDP; the gateway can respond to any UL frame within the restricted number of DL/Day.

E. Optimised listening strategy

Our results show that the Sigfox node spends 38% of its energy in the listening state while using the Greedy strategy. We propose that it can be optimised by avoiding BDP as much as feasible. The **Optimised Listening** strategy suggests scheduling only four UL frames following BDP per day.

F. Weather-Driven strategy process automaton

The amount of rain that falls in a given area significantly influences the water level in the pipes. We propose a **Weather-driven** strategy based on weather prediction information for the study region (Odense, Denmark), namely the chance of forecasted rainy and dry weather over the year. The strategy controls the number of UL frames and measurements based on weather prediction. It schedules sending and measuring operations every $10min$ and $1min$ for rainy and every $30min$ and $3min$ for dry weather. In the weather-driven model, a Sigfox node schedules up to four DL messages in a day and receives the weather forecast, which can be either rainy or dry. The weather-driven strategy has an advantage over the data compression strategy, where a node reduces its measuring frequency for dry days.

Figure 9 depicts the automaton model for weather prediction, using the probability of 0.4642 and 0.5358 for rainy and dry weather, respectively, as per [34]. The weather forecast has about 30% probability of being incorrect [35]. We model this behaviour using a probabilistic choice of rainy or dry weather. After both options, there is again a probabilistic choice; the prediction could be correct or wrong. The automaton updates the sending and measuring frequency for rainy and dry weather to the scheduler using the `update!` channel.

G. Data-Compression strategy process automaton

The aim of implementing a *data* compression strategy is to avoid sending unnecessary information. The base model collects a measurement every minute and sends all these measurements to the gateway every $10min$. Most of the time, the variation in water height is not that frequent, and we have the same value in multiple measurements.

A simple (lossy) compression approach is to send a UL frame only when measuring a significant variation in water level compared to the last measurement sent to the gateway. Even though its name can be deceiving, this strategy does not apply a traditional compression algorithm. Instead, it sends measured value(s) only when they are significantly different from the last previously sent value. The rest of the measurements are not sent. Using this strategy, we are reducing the number of UL frames and the payload size in each frame.

To evaluate the effect of this strategy, we use data sets of measurements generated from WSN hardware operating in a real environment. Figure 10 depicts an automaton for data processing. It measures a value every min from the data set, and it enables a `diff` variable when it notices a significant variation in the measured value. The `scheduler` enables sending a message only when both `ready` and `diff` variables are true.

H. Combined Data Compression and Weather-driven Strategy

The data compression strategy described in the previous section optimises the sending frequency by avoiding sending unnecessary information to the base station, but it measures

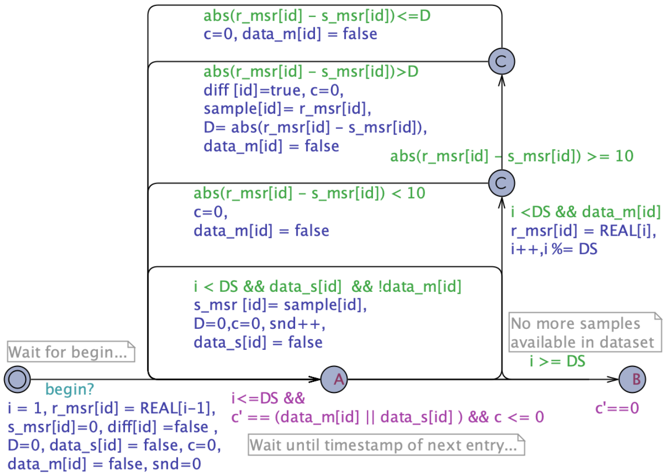


Fig. 10: Data automaton model

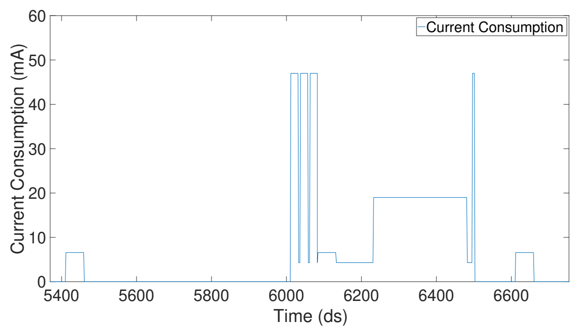


Fig. 11: Current consumption graph of the Sensor Node with Greedy Strategy

every minute. As shown in Table III, measuring state also consumes significant energy. We can limit measuring frequency by adding weather information in the data compression strategy model.

V. Simulation Results

This section presents results obtained from using the SSN model within UPPAAL SMC. We compare our results with data from other sources to check if the model agrees with them. Furthermore, the different transmission strategies (described in section IV) are evaluated.

A. Basic transmission cycle behaviour

Equation 2 simulates the basic model for $25min$ to monitor the behaviour of the current consumption (I) variable. As described in section IV-A, the value of I is updated for each state based on current consumption on that state.

```
simulate [<=25*min]{Sensor(0).I}        (2)
```

Figure 11 presents the model's current consumption behaviour for a few Sigfox cycles. The x-axis and y-axis of the graph present the time in ds and current consumption in mA, respectively. In the figure, small pulses show the measuring activities; three consecutive pulses consuming around $50mA$ are UL and its two replicas. Next follows the listening window, followed by a receive confirmation UL. Hence, the basic duty cycle behaviour is visualised.

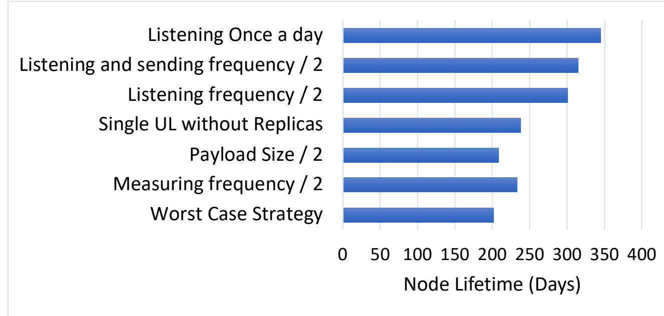


Fig. 12: Battery lifetime comparison by changing different protocol's parameters [6]

B. Energy consumption by node's states

Equation 3 in UPPAAL SMC calculates the overall energy consumed in each model state. The formula uses the energy variables described in table II.

```
simulate[<=250](E_msr, E_send, E_lstn
                        ,E_sleep, E_standby   (3)
```

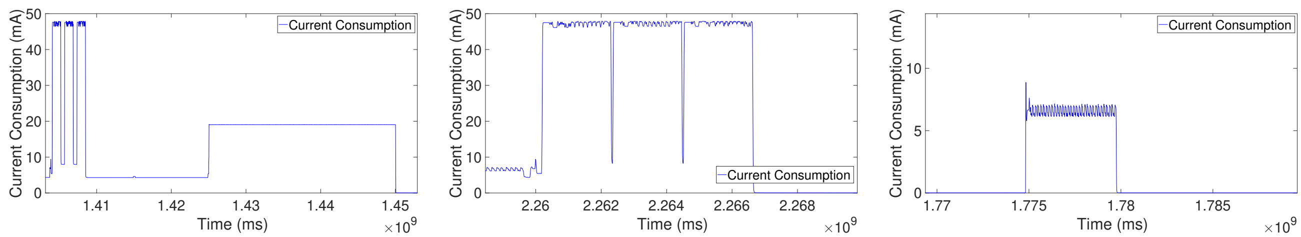
Table III shows the simulation results of energy consumption by the different states for all strategies. In Table III, we can observe that the most energy-consuming state within the Greedy strategy is the listening state, where the node needs to have its radio turned on for a long period of time. Measuring is the second most energy-consuming state; in fact, using an ultrasonic sensor requires activating an ultrasonic transducer and issuing numerous radio waves — a relatively energy-expensive operation. Finally, sending is also costly, but here the radio is on for shorter bursts compared to listening. The sleep and standby states require little energy, as one would expect from a well-designed sensor node with a clear active/inactive duty cycle.

Hence, three optimisations are possible: reduce listening, measure less, and transmit less. In particular, it may be inoptimal to spend around 38% of energy in a listening state when only 4 DLs are possible in a day. Hence, an obvious optimisation can revolve around coordinating the gateway and node regarding *when* the DL messages may occur.

C. Effect of reducing different transmission parameters

Figure 12 investigates the node's lifetime for the *Greedy* strategy compared to the effect on lifetime by changing the following parameters:

- Reduce listening, sending, and measuring frequencies to half.
- Avoid sending replicas of ULs.
- Reduce the payload size 50%.



(a) UL's and Listening Window (b) UL with 2 replicas (c) Measuring action

Fig. 13: Current consumption graph generated from a hardware module of WSN operating in a real environment

We observe that reducing listening and sending have more potential to increase the lifetime than limiting payload size and limiting replication. Reducing measuring frequency is also an important parameter, but there is a chance of losing important information in that scenario. We can possibly optimise measuring frequency by predicting water height based on weather and water use patterns. The battery discharge curve is non-linear because of battery self-discharge.

D. Comparison with actual power traces

In cooperation with the Montem company, we measured the current consumption values of the WSN from the hardware implementation. Figure 13 depicts the current consumption graphs generated from the real hardware platform, where graph (a) depicts three UL frames followed by a Listening window for DL, graph (b) portrays a zoomed view of UL frames, and graph (c) focuses on the measuring state. We calculate the average current consumption values and the average time required to perform different activities by analysing these measurements.

Table I describes the min and max current consumption statistics obtained from the hardware datasheets and average measured values from real hardware. We observe that these values differ a little from the values obtained from the datasheet. The real node consumes less energy in sending, but it consumes more energy in all other states compared to the datasheet values. In the listening state, it consumes even more than the max energy consumption mentioned in the datasheet.

E. Calibrated model

We calibrate the designed model using hardware measurements. Figure 14 presents the lifetime results from simulating the Greedy strategy, using in the model the minimum, maximum (obtained from the datasheets) and measured power characteristics. The Figure confirms that the model calibrated with the measured values is consuming even more energy than the model with max datasheet power ratings. The calibrated model with the Greedy strategy differs the most because it uses the BDP that involves more listening, and this is the state where the calibrated data differs most significantly.

Comparing the results using the datasheet (a lifetime of ca. 235 days) with using the calibrated values (lifetime ca. 215 days) gives a difference of 10%. So the results obtained from using data sheet values only (not requiring hardware construction) would still give usable results, particularly for

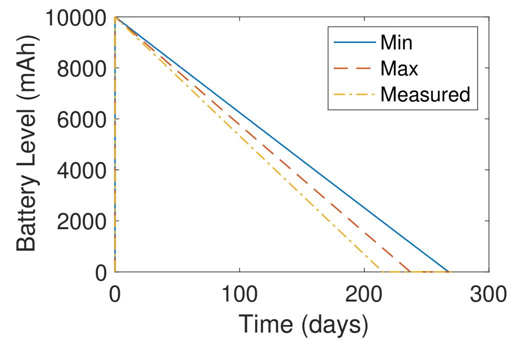


Fig. 14: Battery Consumption Graph For minimum, maximum and measured power characteristics [6]

the *relative effects* of the different communication strategies. All the following results are produced using the *calibrated* models, i.e. using the values measured on the real hardware.

F. Effect of optimised listening strategy

In our case study, the gateway rarely needs to communicate with the node. So we can schedule fixed time slots for bi-directional communication, and the rest follows the uni-direction procedure.

Figure 19(4th) depicts the model's simulation results with the optimised listening strategy. The figure shows that the battery lifetime increases more than 100%. Table III shows that the optimised listening strategy reduces the energy consumed in the `Listening` state from 38% to 2% during the entire lifetime.

G. Effect of weather-driven strategy

Here, the transmission is based on rainy or dry weather probability, so a range of battery lifetime is possible. Equation 4 simulates the `lifetime` of the model 200 times and predicts an average value with an achieved confidence level of 0.95.

```
simulate [200<=900*day]{Sensor(0).lifetime}   (4)
```

Figure 15 depicts the frequency histogram chart for the expected lifetime of the model (737 to 756 days). The battery discharge graph is shown in Figure 19(4th). The graph shows that the battery lifetime increases up to 3 times from the Greedy strategy. The disadvantage is of course insufficient measurements taken in the situation where a wrong weather

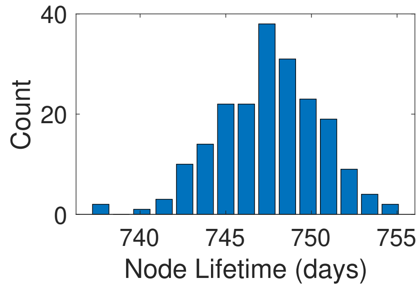


Fig. 15: Frequency Histogram of battery lifetime for weather-driven strategy

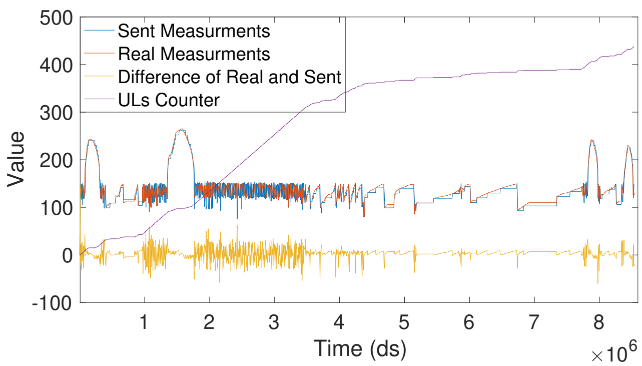


Fig. 16: Comparison of measured and sent water height values and effect of variation in water height on the frequency of UL frames counter [6]

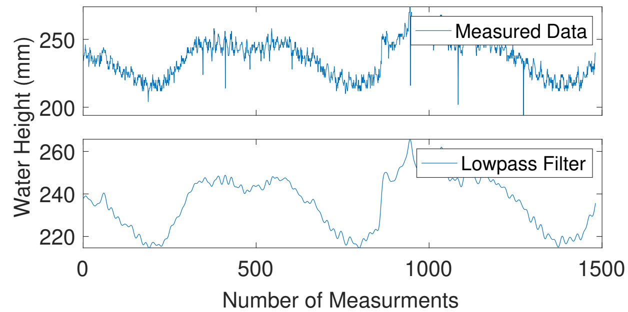


Fig. 17: Apply low-pass filter on measured dataset

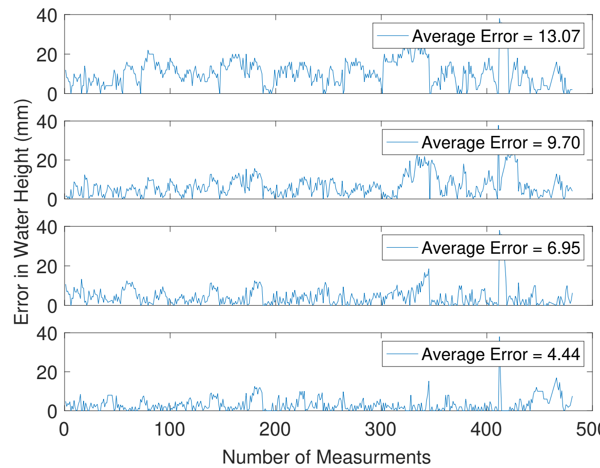


Fig. 18: Difference error between measured and sent data

forecast predicts dry weather on a rainy day. This can be alleviated by conducting some amount of extra measurements and checking in the node that the water level is as expected on a dry day, and then switching to more frequent sending if a discrepancy is observed. The latter approach is a candidate for future evaluation.

H. Effects of data compression strategy

We simulate the model with the data compression strategy for a duration of 20 days to investigate the effect of variation in water height on the sending frequency since the data compression strategy controls the sending frequency based on the measurements. For this experiment, we have used data streams generated from nodes operating in a real environment. Each node measures the water height value on a given location and sends data based on the data compression algorithm. As expected, Figure 16 shows that the sending frequency is increasing when measuring frequent variations in water height. The figure also presents the error between measured and sent values since, in this strategy, the model sends only the measured values having a variation of at least $10mm$ from the last sent value.

As shown in figure 17, some outlier data (erroneous ultrasonic measurements with very low water level) require sending data even without significant variation in the real water level. In order to prevent this inaccuracy, we remove the unrealistic variations from the data, and we apply a low pass filter to reduce the waving effect of the water from the data. The modified data are shown in Figure 17.

Figure 19(1st) presents that applying a Low Pass Filter on measured data improves the battery lifetime from 820 to 990 days by avoiding error variation in water level.

Because compression is lossy, there is a difference between measured and sent data. This error is shown in the first graph of Figure 18. We calculate an average error of 13.08mm in water height. We could not employ other (traditional) data compression techniques since the number of measurements in a frame is unpredictable because it could be possible that the model found a variation in water every 10 min, or there could be no variation in a whole day. By applying a low pass filter to the data, we can reduce the average error to 9.7mm. Furthermore, we can reduce the error by sending multiple data samples in a frame having the most significant difference in the last duration. We have analysed a dataset to calculate the effect of this workaround, which can differ for other scenarios. The figure shows that sending two measured values from each frame reduces the error to 6.95mm and can reduce up to 4.44mm by sending four values per frame.

We simulate the model to analyse the data compression strategy, which is parameterised with the threshold that triggers a UL transmission. A slight increase in the threshold level reduces communication of the water waving behaviour which

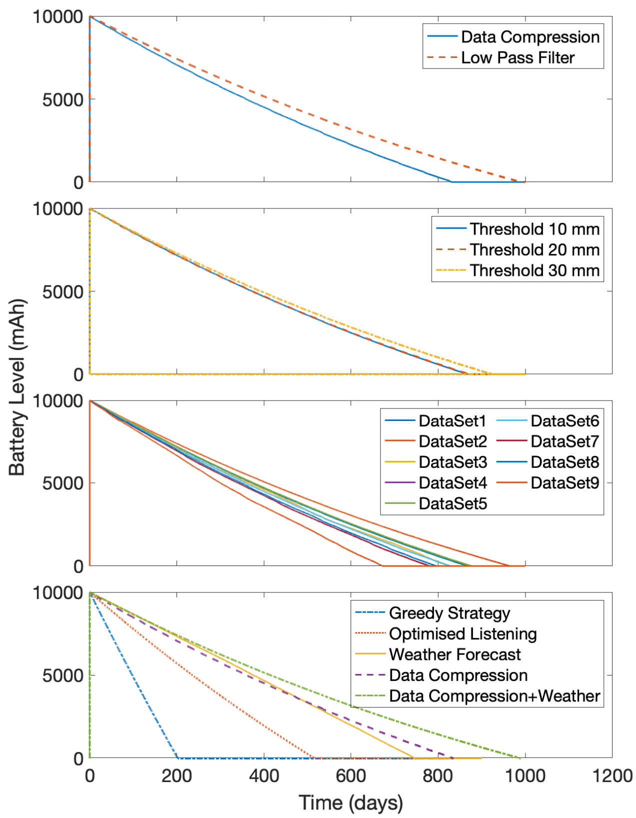


Fig. 19: (1st)Battery Lifetime for data compression strategy after applying a Low Pass Filter; (2nd)Battery Lifetime for data compression strategy with different threshold levels; (3rd)Battery Lifetime for data compression strategy for multiple datasets; (4th)Battery Lifetime for all Strategies

we suspect is of low interest for the domain analysts. Figure 19(2nd) shows that the battery life slightly increases by increasing the threshold level in the data compression strategy.

Furthermore, figure 19(3rd) depicts the model's simulation results with the data compression strategy for multiple datasets obtained from sensor nodes for the periods 2021-01-01 to 2022-01-26, deployed geographically in different locations. We investigate the data compression strategy's effectiveness and calculate the average lifetime. Figure 19(4th) illustrates that the strategy has the potential to increase the battery lifetime up to 4 times from the Greedy strategy. This simple method for data compression is obviously less effective for data sources with high variation in water level, as we have observed in a few instances. For other simulations of data compression strategy, we use a data set which gives an average lifetime by comparing all datasets.

I. Effects of Combined Data Compression and Weather-driven Strategy

It receives weather information from the base station, similar to the weather-driven model, and selects measuring

TABLE III: Energy Consumption in Each State (mAh)

	Greedy	Optimised Listening	Data Compression	Rain Model
Battery Self Discharge	655.9	1690	2524.6	2072.2
Sleep	110.9	317.1	517.4	367.4
Standby	380.1	91.2	69.6	91.4
Sending	2378.4	4340.5	1049	3400.8
Listening	3841.1	272.3	399	314.3
Measuring	2633.3	3359.7	5440	3753.7

frequency based on weather conditions. It uses a data compression strategy to enable sending. Figure 19(4th) presents that the strategy can improve the battery lifetime from 820 to 985 days by optimising sending and measuring frequency.

J. Comparison of all strategies

This subsection compares the effects of all the strategies. Table III presents the overall energy consumed by the battery self-discharge and each sensor node state. Figure 19(4th) depicts the battery lifetime graphs.

The graph reveals that the Greedy strategy model has a lifetime of about 202 days, the optimised listening model has a lifetime of about 388 days, the weather-driven model has a lifetime of about 750 days, and the data compression model has a lifetime of up to 990 days. The measurement data determines the node's lifetime while using a data compression strategy. Table III shows that the optimised listening model decreases energy usage in the listening state from 50% to 3%. It is also clear that the data compression strategy spends much more energy in the measurement state than in the sending state. This is expected as this is a direct consequence of being able to measure more with the goal of sending less. Finally, we observe that the curves decrease non-linearly. This is the self-discharge effect showing up when the lifetime gets long.

VI. Conclusions and Future work

This study presents an energy-aware analysis of a Sigfox-based DWSN using Uppaal SMC. The simulation results show the sensor node's lifetime and energy and time cost for the major system states. The comparison results demonstrate the model's behaviour with datasheet statistics and real hardware measurements. The results also show how transmission frequency, payload size, unidirectional and bidirectional communication and data compression technique dramatically affect the node's lifetime. The results reveal that we can extend the node's lifetime from 202 days to 2.71 years using efficient transmission strategies.

As future work, we will try to optimise the measuring algorithm and related hardware because it consumes up to 57% of the energy. We also identified a limitation of our tool, since the computation in Uppaal SMC becomes slower when using decimal value as `invariant` or **`guard`** in the model. We are exploring different modelling tricks to reduce this problem. Finally, we plan to compare our methodology to the well-known tool Network Simulator 3, `ns-3`, where we have already contributed with a Sigfox radio model [13].

REFERENCES

[1] H. Korala, D. Georgakopoulos, P. P. Jayaraman, and A. Yavari, "A survey of techniques for fulfilling the time-bound requirements of time-sensitive iot applications," *ACM Computing Surveys*, 2022.

[2] K. A. Aldahdouh, K. A. Darabkh, W. Al-Sit *et al.*, "A survey of 5g emerging wireless technologies featuring lorawan, sigfox, nb-iot and lte-m," in *2019 International conference on wireless communications signal processing and networking (WiSPNET)*. IEEE, 2019, pp. 561–566.

[3] M. E. Yuksel and H. Fidan, "Energy-aware system design for batteryless lpwan devices in iot applications," *Ad Hoc Networks*, vol. 122, p. 102625, 2021.

[4] S. Aguilar, A. Platis, R. Vidal, and C. Gomez, "Energy consumption model of schc packet fragmentation over sigfox lpwan," *Sensors*, vol. 22, no. 6, p. 2120, 2022.

[5] M. Ahm, L. Duus, P. Rasch, P. Laursen, A. Høedholt, K. Larsen, L. Sørensen, J. Nielsen, and M. Rasmussen, "Distributed online monitoring of the urban water cycle (donut): vision and initial results," *10th IWA Symposium on Modelling and Integrated Assessment*, 1–4 Sep 2019.

[6] M. Naeem, M. Albano, K. G. Larsen, B. Nielsen, A. Høedholt, and C. Østergaard Laursen, "Battery aware analysis of sensor networks in uppaal smc," in *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, IEEE. Budva, Montenegro: IEEE, 2021, pp. 1–6.

[7] K. Sen, M. Viswanathan, and G. Agha, "Statistical model checking of black-box probabilistic systems," in *International Conference on Computer Aided Verification*. Springer, 2004, pp. 202–215.

[8] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, "Uppaal smc tutorial," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 397–415, 2015.

[9] C. Gomez, J. C. Veras, R. Vidal, L. Casals, and J. Paradells, "A sigfox energy consumption model," *Sensors*, vol. 19, no. 3, p. 681, 2019.

[10] J. Finnegan and S. Brown, "An analysis of the energy consumption of lpwa-based iot devices," in *2018 Int. Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2018, pp. 1–6.

[11] D. Hernandez, G. Peralta, L. Manero, R. Gomez, J. Bilbao, and C. Zubia, "Energy and coverage study of lpwan schemes for industry 4.0," in *IEEE Int. Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*. IEEE, 2017, pp. 1–6.

[12] B. Nielsen, M. Albano, M. Naeem, T. Pedersen, A. Hødholdt, and C. Ø. Laursen, "Battery lifetime estimation and optimization for an urban water system wsn," in *IWA World Water Congress*, 2021.

[13] M. Naeem, M. Albano, D. Magrin, B. Nielsen, and K. Guldstrand, "A sigfox module for the network simulator 3," in *Proceedings of the WNS3 2022*, 2022, pp. 81–88.

[14] R. K. Singh, P. P. Puluckul, R. Berkvens, and M. Weyn, "Energy consumption analysis of lpwan technologies and lifetime estimation for iot application," *Sensors*, vol. 20, no. 17, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/17/4794

[15] A. David, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, J. Van Vliet, and Z. Wang, "Statistical model checking for networks of priced timed automata," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2011, pp. 80–96.

[16] D. Panigrahi, C. Chiasserini, S. Dey, R. Rao, A. Raghunathan, and K. Lahiri, "Battery life estimation of mobile embedded systems," in *VLSI Design 2001. Fourteenth International Conference on VLSI Design*. IEEE, 2001, pp. 57–63.

[17] J. Xue, Z. Yuan, and Q. Zhang, "Traffic load-aware power-saving mechanism for ieee 802.16 e sleep mode," in *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE, 2008, pp. 1–4.

[18] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[19] A. Rashid, O. Hasan, and K. Saghar, "Formal analysis of a zigbee-based routing protocol for smart grids using uppaal," in *2015 12th International Conference on High-capacity Optical Networks and Enabling/Emerging Technologies (HONET)*. IEEE, 2015, pp. 1–5.

[20] P. L. Laursen, V. A. T. Trinh, and A. E. Haxthausen, "Formal modelling and verification of a distributed railway interlocking system using uppaal," in *International Symposium on Leveraging Applications of Formal Methods*. Springer, 2020, pp. 415–433.

[21] Y. Fei, H. Zhu, and J. Yin, "Modeling and verifying nlsr protocol of ndn for cps using uppaal," *Journal of Software: Evolution and Process*, p. e2384, 2021.

[22] J. C. Zuniga and B. Ponsard, "Sigfox system description," *LPWAN@ IETF97, Nov. 14th*, vol. 25, 2016.

[23] *Sigfox Radio specifications*, Sigfox, Feb 2020. [Online]. Available: https://storage.googleapis.com/public-assets-xd-sigfox-production-338901379285/abaedf62-56de-402e-93c3-3a9c10a1cb49.pdf

[24] H. L. Younes and R. G. Simmons, "Probabilistic verification of discrete event systems using acceptance sampling," in *International Conference on Computer Aided Verification*. Springer, 2002, pp. 223–235.

[25] S. Sebastio and A. Vandin, "Multivesta: Statistical model checking for discrete event simulators," in *7th Int. Conf. on Performance Evaluation Methodologies and Tools*, ser. ValueTools '13. ICST, 2013, p. 310–315.

[26] R. Alur and D. Dill, "Automata for modeling real-time systems," in *International colloquium on automata, languages, and programming*. Springer, 1990, pp. 322–335.

[27] G. Behrmann, A. David, and K. Larsen, "A tutorial on UPPAAL," in *Formal methods for the design of real-time systems*. Springer, 2004.

[28] *MB7380, Ultrasonic sensor*, HRXL-MaxSonar, [Accessed 11 Oct 2022]. [Online]. Available: https://www.maxbotix.com/documents/HRXL-MaxSonar-WR_Datasheet.pdf

[29] *ATmega328/P*, Microchip, [Accessed 11 Oct 2022]. [Online]. Available: https://www.mouser.com/pdfdocs/gravitech_atmega328_datasheet.pdf

[30] *TD1207R/08R*, Sigfox, 2016. [Online]. Available: https://datasheetspdf.com/pdf-file/1338325/Telecom/TD1207R/1

[31] *Digital Accelerometer ADXL345-EP*, AnalogDevices, [Accessed 11 Oct 2022]. [Online]. Available: https://www.mouser.dk/datasheet/2/609/ADXL345-EP-1503767.pdf

[32] H. Ryu, H. Ahn, K. Kim, J. Ahn, K. Cho, and T. Nam, "Self-discharge characteristics of lithium/sulfur batteries using tegdme liquid electrolyte," *Electrochimica acta*, vol. 52, no. 4, pp. 1563–1566, 2006.

[33] D. Ivanov, K. G. Larsen, S. Schupp, and J. Srba, "Analytical solution for long battery lifetime prediction in nonadaptive systems," in *Int. Conf. on Quantitative Evaluation of Systems*. Springer, 2018, pp. 173–189.

[34] "Average monthly rainy days in odense (funen), denmark," 2022. [Online]. Available: https://weather-and-climate.com/average-monthly-Rainy-days,odense,Denmark

[35] M. Schleiss, J. Olsson, P. Berg, T. Niemi, T. Kokkonen, S. Thorndahl, R. Nielsen, J. Ellerbæk Nielsen, D. Bozhinova, and S. Pulkkinen, "The accuracy of weather radar in heavy rain: a comparative study for denmark, the netherlands, finland and sweden," *Hydrology and Earth System Sciences*, vol. 24, no. 6, pp. 3157–3188, 2020.