Open access • Journal Article • DOI:10.1145/3379339

Energy Modeling for the Bluetooth Low Energy Protocol — Source link

Philipp H. Kindt, Daniel Yunge, Robert Diemer, Samarjit Chakraborty

Institutions: Technische Universität München, Valparaiso University, University of North Carolina at Chapel Hill

Published on: 16 Mar 2020 - ACM Transactions in Embedded Computing Systems (ACMPUB27New York, NY, USA)

Topics: Energy consumption, Bluetooth, Wireless Application Protocol, Energy modeling and Wireless network

To appear in the ACM Trans. on Embedded Computing Systems
Author's version January 2020
Published version might be different

Energy Modeling for the Bluetooth Low Energy Protocol

PHILIPP H. KINDT, Technical University of Munich (TUM), Germany
DANIEL YUNGE, Ponticifia Universidad Católica de Valparaíso, Chile
ROBERT DIEMER, ASC GmbH, Germany
SAMARJIT CHAKRABORTY, University of North Carolina at Chapel Hill (UNC), USA

Bluetooth Low Energy (BLE) is a wireless protocol optimized for low power communication. To design energy-efficient devices, the protocol provides a number of parameters that need to be optimized within an energy, latency and throughput design space. Therefore, an energy-model that can predict the energy consumption of a BLE-based wireless device for different parameter value settings is needed. As BLE differs from the well-known Bluetooth Basic Rate (BR) significantly, models for Bluetooth BR cannot be easily applied to the BLE protocol. Within the last years, there have been a couple of proposals on energy models for BLE. However, none of them can model all the operating modes of the protocol. This paper presents an energy model of the BLE protocol, which allows the computation of a device's power consumption in all possible operating modes. To the best of our knowledge, our proposed model is not only one of the most accurate ones known so far (because it accounts for all protocol parameters), but it is also the only one that models all the operating modes of BLE. Based on this model, guidelines for system designers are presented, that help choosing the right parameters for optimizing the energy consumption. The model is publicly available as a software library for download.

ACM Reference Format:
Philipp H. Kindt, Daniel Yunge, Robert Diemer, and Samarjit Chakraborty. 2020. Energy Modeling for the Bluetooth Low Energy Protocol. *ACM Trans. Embedd. Comput. Syst.* 1, 1, Article 1 (January 2020), 31 pages. https://doi.org/10.1145/3379339

1 INTRODUCTION

Optimizing energy consumption is a crucial design requirement in many wireless sensor networks. Long battery lifetimes are especially important for body-worn medical sensors, mobile phones and interface devices, such as wireless mice. The Bluetooth Low Energy (BLE) protocol [23] was introduced in 2010 for providing low-power wireless connectivity in such applications. It has become very popular over the past 10 years and is widely used today in a variety of devices. BLE-based sensors are able to operate on a coin cell for several months to several years [11], depending on their processing and communication demands and the parametrization of the (BLE) protocol. The protocol leaves open a large degree of freedom in terms of choosing the parameter values. Such parametrizations have a significant impact on the power consumption of the device. As a result, appropriate parameter optimizations are an important part of the design process.

Authors' addresses: Philipp H. Kindt, philipp.kindt@tum.de, Technical University of Munich (TUM), Germany; Daniel Yunge, daniel.yunge@pucv.cl, Ponticifia Universidad Católica de Valparaíso, Chile; Robert Diemer, r.diemer@asc-sensors.de, ASC GmbH, Germany; Samarjit Chakraborty, samarjit@cs.unc.edu, University of North Carolina at Chapel Hill (UNC), USA.

1539-9087/2020/1-ART1 $15.00
https://doi.org/10.1145/3379339

In this paper, we present a comprehensive energy model for BLE in all its possible modes of operation. Our work integrates existing models for BLE and makes a number of new contributions:

(1) Our model takes into account *all* relevant operating modes (i.e., connected communication, non-connected communication and the connection establishment procedure) and parameters of the protocol, whereas all known models till date only capture a limited set of operating modes. Towards this, we combine, refine and extend a number of recently proposed energy models for BLE.
(2) We analyze the variations of parameter values and perform a sensitivity analysis to study their impact on power consumption.
(3) We validate the model through detailed experiments, viz., by comparing the results from our proposed model with those obtained from measurements. The maximum error between predicted and measured current was 3.4 %. Such comparisons have not been done for any of the previously proposed energy models, baring one exception in [18], where only one mode was studied.

Our model helps in finding the right parametrizations, since the impact of different parameter values on the energy consumption can be studied. Measuring the current of BLE using some configurations and simply extrapolating these results for deriving the best parametrization is often not possible, since some relations between parameters and energy consumption are non-linear.

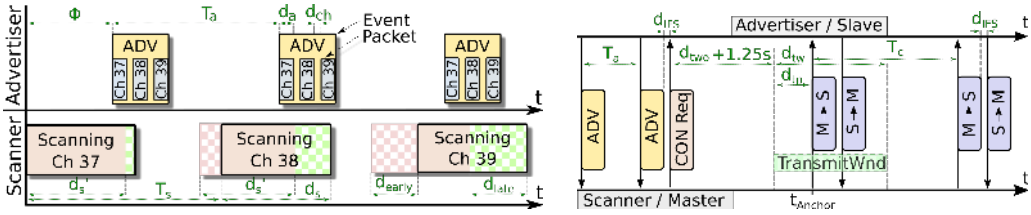
The rest of this paper is organized as follows. In Section 2, we give a short introduction to the BLE protocol. Related previous work and our contributions are described in Section 3. We then present our proposed energy model for BLE in Section 4. This model is compared against simulations and real world measurements in Section 5. In Section 6, the model is used to provide guidelines for appropriate parametrization of the protocol in different scenarios. To ease the understanding of the equations we present in this paper, a table of symbols is provided in the appendix.

2 THE BLE PROTOCOL

In this section, we summarize the main features of the BLE protocol that are relevant for our proposed energy model. More details may be found in the BLE specification [23]. BLE is designed to maximize the fraction of time during which a device can stay in a sleep mode. Consider two devices that attempt to communicate with each other. First, both devices need to discover each other. Towards this, one device periodically sends so-called advertising (ADV) packets, whereas the other device scans for these packets. As soon as the scanner has received at least one advertising packet, it can go to the *initiating* state for establishing a connection. In this state, handshaking packets are exchanged. Upon success, both devices go into the *connected* state, in which the former advertiser becomes the slave and the former scanner becomes the master. In a connection, the master controls the timing and both devices can exchange data in a time-sliced and hence energy-efficient manner.

For the purpose of our model, we distinguish between the connected mode, the non-connected mode and the establishment of a connection. For the sake of simplicity of exposition, we consider BLE piconets with only two participating devices. For multiple devices, the following two limitations apply. First, if a connection master maintains multiple slaves, whenever the connection events from two slaves overlap in time, only one of them can be served by the master, which might impact the energy consumption. Second, if multiple devices attempt to discover each other simultaneously (see below for details on this procedure), their beacons will collide with a certain probability. Hence, the connection setup will take longer than predicted by our model in such cases.

Non-Connected Communication. Non-connected communication is mainly used for neighbor discovery. In addition, small amounts of data can be exchanged between two devices without requiring a prior synchronization. Here, one device is in the advertising mode and the other one is in the



(a) Advertising and scanning in BLE. (b) Connection establishment packet flow.

Fig. 1. Connection Setup in BLE.

scanning mode. Advertising and scanning take place as shown in Figure 1(a): An advertising device periodically sends out advertising packets. A group of consecutive packets form an *advertising-* or *ADV-event*. In each of these advertising events, an advertising packet is sent on at least one of three dedicated adverting channels. Which subset of the three channels is to be used is determined by the application. Advertising events occur regularly with the *advertising interval* T_a. Independently from this, a scanner periodically switches on its receiver for a duration of d_s time units, called the *scan window*. This is repeated at every interval T_s, called the *scan interval*. In the next period, the scanner hops to the next advertising channel and again listens for advertising packets. The specification [23] requires the scanner to use all three advertising channels. If the scanner receives an advertising packet, it may send a response packet to the advertiser within the same ADV-event. The advertiser expects a response on the same advertising channel $d_{IFS} = 150\,\mu$s after the end of the advertising packet. d_{IFS} is called the *interframe-space*. The advertising interval is composed of a static interval $T_{a,0}$ and a random part ρ, i.e., $T_a = T_{a,0} + \rho$, where ρ is a random amount of time between 0 ms and 10 ms. Whenever we refer to a certain value of the advertising interval, this value actually corresponds to the static component $T_{a,0}$, which can be chosen by the host within certain boundaries. The random offset ρ is added to $T_{a,0}$ mainly to mitigate the effect of collisions. If two devices chose the same $T_{a,0}$ and one pair of packets collides, all subsequent ones would also collide without the offset ρ.

Connection Establishment. A connection can be established after a device has been detected by advertising/scanning. After having received at least one advertising packet, the scanner sends a connection-request packet d_{IFS} time units later. This packet contains two parameters called *transmitWindowOffset* d_{two} and *transmitWindow* d_{tw} that determine the timing of the connection establishment procedure. In doing so, the scanner is in the *initiating* state and is therefore called the *initiator*. The durations d_{tw} and d_{two} affect the connection establishment procedure, as depicted in Figure 1(b). 1.25 ms + d_{two} time units after the end of the connection-request packet, the transmit window starts, in which the initiator may schedule its first regular connection event. The constant value of 1.25 ms is defined by the BLE specification. The initiator may schedule the first connection event an arbitrary amount of time t_p after the beginning of the transmit window. The point in time of this first even is called t_{Anchor}. The advertiser has to listen during the entire transmit window until there is a successful reception. Afterwards, the connection is valid and data can be exchanged.

As described in the next section, in the connected mode, data is transmitted periodically with the *connection interval* T_c. The procedure described for connection establishment is also used by the master for changing the parameters of an existing connection, with the following minor modifications. In a regular connection event, the master sends the new values for T_c along with other updated parameters to the slave. The transmit window for the first packet being affected by the new parameter values starts $d_{two} + T_{c,o}$ time units after the transmission of the connection update packet has started. $T_{c,o}$ is the connection interval before the parameter update procedure.

Connected Communication. During the connection establishment phase, both devices have agreed on a connection interval T_c that determines the period with which the connection events occur. In addition, there is an anchor point t_{anchor} that both devices have been synchronized on. At $t_{anchor} + kT_c, 0 \leq k \leq \infty$, both devices have to be awake and M will send a packet. S will then acknowledge it by sending another packet 150 μs later, which might be either an empty response, a payload packet, or a packet performing control functions. If there is more data to be transmitted, more pairs of packets are exchanged in the same manner, separated from each other by d_{IFS} time units. Apart from pre- and postprocessing, the device may sleep during all other times. In addition, there is a parameter called the *slave latency* N_{sl}, which increases the sleep duration of S in case there is nothing to signal in a connection event. If M and S have agreed on a slave latency of N_{sl}, the slave might skip N_{sl} connection events without waking up. The connection interval T_c must range from 7.5 ms to 4.0 s and N_{sl} may be up to 500 events.

3 RELATED WORK

Performance Modeling of BLE: The performance of BLE, including its energy consumption, has been extensively studied in the literature. In [7], an energy-model for Bluetooth BR/EDR in the *sniff mode* was presented, that cannot easily be applied to BLE due to differences between the two protocols. For BLE, different models have been presented. The maximum throughput of BLE was modeled in [8], taking into account a given bit error rate. In [11], Texas Instruments provided guidelines on measuring the current consumption of its CC2540 BLE device in the connected mode by measuring the current consumption of the partial events and summing them up. Based on this, TI provided an estimate for the battery lifetime. By entering the measured event currents and the durations into an Excel sheet provided by TI, the average current consumption may be calculated. This has been the first step towards an energy model for BLE, which only took the connected mode into account. Other manufacturers of BLE SoCs have published comparable tools. Further, in [29], an energy model in the connected mode similar to TI's scheme was presented and evaluated for different protocol parameters. Another event-based model was presented for advertising events in [18]. Further previous studies on the performance evaluation of BLE, which included aspects like energy consumption, latency, memory requirements of the BLE stack, throughput and maximum piconet size [9], [21], [22] have been published. However, none of these models cover all modes of operation, including non-connected communication, connection establishment, and connected communication. All of these modes contribute to the overall energy consumption. For example, to the best of our knowledge, there has been no known model for the connection setup procedure and for connection parameter updates previously to this work. Combining different previously known models into a holistic one and complementing the missing parts is not feasible, since the assumptions underlying the different models are not consistent. For example, [11] subdivides a BLE connection event into 8 distinct phases, whereas [21] assumes only 5. In our proposed model, the methodology and assumptions are consistent for individual parts of the model and *all* modes of operation are covered in the same manner. To the best of our knowledge, results from any of the previously proposed models have not been compared to measured power curves, bearing one exception that is limited to neighbor discovery [18]. Consequently, the accuracy of these models is not clear. Our measurements reveal that our model is in close proximity with the measured current consumption. The error for the average current consumption per connection interval we measured in our experiments was 3.4 % (cf. Section 5.1). Compared to the models presented in the literature, our model has the following advantages:

(1) To the best of our knowledge, we present the first model that accounts for i) all possible modes of operation, ii) all relevant parameters, and iii) all possible parameter values of the BLE protocol. For example, our proposed model is the first one that can model the energy

consumption for the connection setup procedure, for updating the connection parameters and for neighbor discovery using all possible parameter values.
(2) Using multiple measurements, we identify the degree of variation in the values of different model parameters. We further analyze their impact on the overall power consumption.
(3) To the best of our knowledge, the outputs of none of the known energy models for BLE have been compared to measured data yet, bearing one exception presented in [18], where the energy consumption of an advertiser for neighbor discovery was modeled and compared to measured data. We have therefore validated the results from our proposed model with real experimental results.
(4) To the best of our knowledge, the energy model proposed in this paper is the only one for which the source-code has been made publicly available [13] in the form of a C-library, which can e.g., be included in commonly used network simulation environments.

Neighbor Discovery in BLE: Besides connected communication, neighbor discovery also contributes to the energy consumption of BLE. Energy models for this case are more complex than for the connected mode, as neighbor discovery in BLE is a probabilistic procedure. Nevertheless, this has been addressed for the *STEM-B* - protocol, in which neighbor discovery is done in a similar way as in BLE. A model for this was presented in [28] and applied to BLE in [18], [20] and [19], predicting the discovery latency and energy-consumption of the advertiser in a neighbor discovery procedure. However, as stated in [19], this model is only valid for a certain range of possible parameter values ($T_{a,0} < d_s - 10\,\text{ms}$ or $T_s = d_s$, as defined in Section 2). The first of these parametrizations implies that the time between any two adjacent advertising packets is always smaller than the length of a scan window. The second one implies that the receiving device is always scanning. In these two cases, the analysis is significantly simplified. However, they capture only a very small part of the whole design space (cf. Figure 8, where we have highlighted the parametrizations for which $T_{a,0} < d_s - 10\,\text{ms}$).

Another model for the neighbor discovery latency of BLE has been presented in [5], [4]. The main assumption underlying this model is that the reception probability of every packet remains constant within all advertising intervals. However, this assumption does not hold true in practice: E.g., let us consider a case in which $T_s = 2 \cdot T_a$. Here, if one packet overlaps with a scan window, the next one obviously cannot overlap with it. Since this model does not account for correlated reception probabilities among multiple packets, it cannot not predict valid discovery latencies. In [10], another model for the neighbor discovery latency of BLE has been presented. It is based on quantizing the range of initial offsets between the advertiser and the scanner into multiple, fixed-length slots. Because of the finite length of each slot, in combination with the necessity to round the advertising interval to the next integer number of slots that is coprimal to the scan interval, this model can only approximate the actual discovery latencies. Further, short slot lengths lead to very large computation times, since for every slot, the modular multiplicative inverse of two intervals needs to be computed. The random delay ρ that the BLE specification requires is not accounted for.

An analytical, closed-form model for the neighbor discovery latency of BLE-like protocols has been presented in [12]. However, this model only accounts for one channel and also neglects the random delay ρ. Therefore, as stated in [12], it approximates the mean latencies well for most values of $T_{a,0}$, T_s and d_s, but can lead to unbounded errors for some values. A common strength of both [12] and [10] is the capability to infer upper bounds on the discovery latencies, which however are not valid for BLE in practice, since the random delay ρ is not accounted for.

The energy consumption of a BLE advertiser has been analyzed in [27]. While [27] does not consider the neighbor discovery procedure of BLE and its associated energy consumption, it focuses on analyzing single advertising events of different BLE SoCs.

	Precision	$T_{a,0} \geq d_s - 10\,\text{ms}$	3 Chan	Rnd. Delay	Lat. Bound	Complexity
Ours	Approx	✓	✓	✓	✗	medium
Simulations	Approx	✓	✓	✓	✗	high
Liu [19]	Exact	✗	✓	✗	✗	negligible
Cho [5]	-	✓	✓	✗	✗	negligible
Jeon [10]	Approx	✓	✓	✗	✗	medium
Kindt [12]	Exact	✓	✗	✗	✓	low

Table 1. Precision, ability to model cases with $T_{a,0} \geq d_s - 10\,\text{ms}$, 3-channel discovery, the random delay ρ, ability to infer an upper latency bound, computational complexity of known BLE neighbor discovery models.

Table 1 summarizes the capabilities of different models. It is worth mentioning that the precisions presented in this table are only valid for the features the models support (i.e., the capability to model cases with $T_{a,0} > d_s - 10\,\text{ms}$, to account for all 3 channels, to account for the random delay ρ in BLE). E.g., the model in [12] can give exact estimates to problems on one channel without the random advertising delay, but cannot model the actual BLE neighbor discovery problem precisely. Despite most of these models having appeared after our proposed one, none of them can account for the random delay of the BLE protocol, which would require fundamentally different approaches. Hence, our model is the only one that can estimate the mean discovery latency of the unmodified BLE protocol, including the random delay, accurately for all parameter values.

Usage of our Model: A preliminary version [15] of this paper has been available as a technical report. In the meantime, it has gained considerable attention in the scientific community. Multiple works rely on the model presented in this paper and some of them have proposed extensions to it.

The following works extend our model for the BLE neighbor discovery procedure. An extension to account for different definitions of the discovery latency has been proposed in [16]. Since our proposed model contains a tunable parameter that is used for trading modeling accuracy against computational complexity, [16] has experimentally evaluated the accuracies obtained for different values of this parameter. Further, [6] presents a version of our proposed model that reduces the computational complexity, but achieves lower accuracies for most parameter values. In summary, while the form of the model presented in this paper covers the most common notion of the neighbor discovery procedure of BLE, these works have extended the model towards other use-cases.

Other work makes use of our model for energy estimation in the connected mode. An attempt to partially port our model into an Excel-sheet has been described in [2]. Other works, e.g., [26], [17] or [14], apply our energy model for estimating energy consumptions or make use of conclusions obtained from our model (e.g., [1]), but do not extend or refine the model itself.

4 BLE ENERGY MODEL

In this section, we present a comprehensive energy model for BLE that is capable of predicting the energy-consumption of the protocol in all modes of operation.

4.1 Overview

Figure 2 gives an overview on our approach. Exchanging packets in BLE takes place in so-called *events* and hence, the protocol can be modeled as a temporal sequence of events that take place periodically. Energy models of these events depend on their types (connection- , advertising- or scan events), as can be seen on the left in Figure 2. The dotted boxes in the figure depict protocol parameters that are fed into the model.

In Stage 1) of Figure 2, the energy of all events that occur is estimated. In Stage 2), the average duration $\overline{d_{adv}}$ of the BLE neighbor discovery procedure is modeled for the advertising/scanning

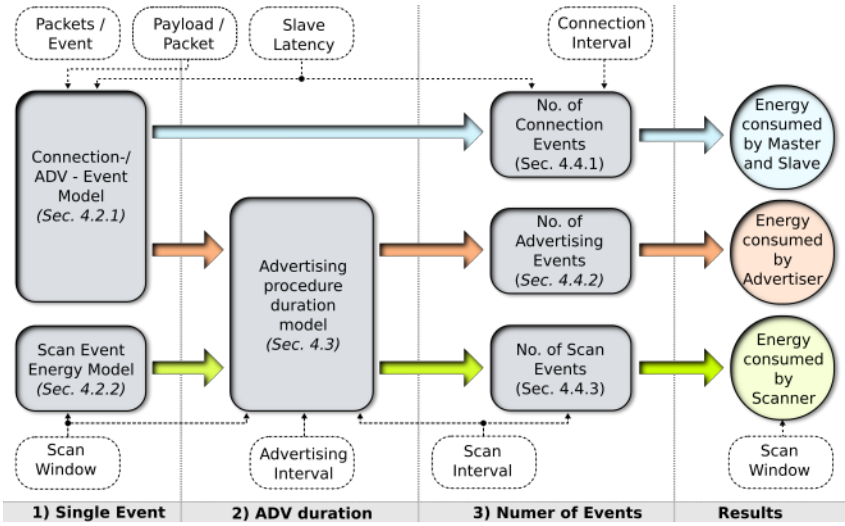


Fig. 2. High-level diagram of the BLE energy model. The dotted boxes denote BLE protocol parameters used by the model.

mode. This duration is of probabilistic nature. It determines how often an advertising event or scan event has to be repeated before the data (either payload or handshaking for the initialization of a connection establishment procedure) has successfully been received. In the third stage, the number of identical events that occur within a certain amount of time is calculated. Next, the energy consumed per event is multiplied by the number of times it occurs and the results for different events are then summed up. In the following, we describe every block in Figure 2. In addition, we describe an energy model for the connection establishment procedure, which is not shown in the figure. This procedure occurs whenever the scanner/initiator has discovered its remote device and establishes a connection or connection parameters are to be updated in an existing connection.

4.2 Event Model

4.2.1 Connection/Advertising Events. In this section, we present a model for connection events. As already mentioned, the energy consumption of an advertising event can be modeled similarly, as both events draw similar currents. Therefore, the model for connection events of a master, as described in this section, can also be used to model advertising events. A previous energy model for connection events in BLE has been presented in [29]. In our model, we extended this by taking into account window widening, non-ideal durations, a communication preamble and correction terms to account for distortions of the current curve caused by resistive and capacitive elements in the power supply line. This is done for achieving the maximally possible accuracy.

Most devices (e.g., Bluegiga BLE112/TI CC2540) make use of a linear voltage regulator, which keeps the current consumption independent from the supply voltage. For this reason, we present all parameter values of our model in terms of electric current I in Ampere and electric charge Q in Coulomb. From these values, the power- and energy-consumption can be easily obtained for a given supply voltage. The description in this section is made for a BLE slave. For a master, the *rx*- and *tx*-phases described below are interchanged and no window-widening occurs.

Within a connection event, ten distinguishable phases with nearly constant current consumption can be identified, as shown in Figure 3. These phases are as described next.

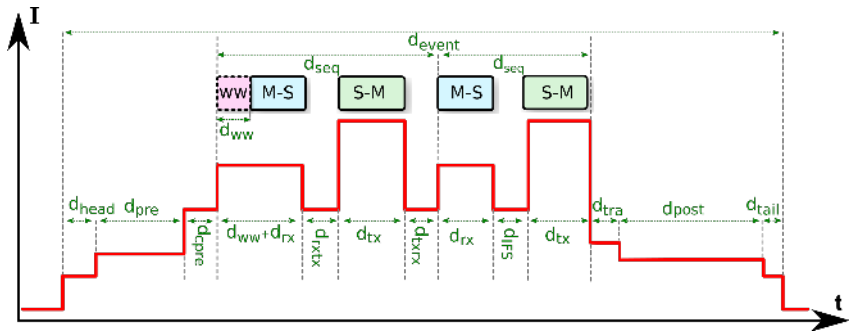


Fig. 3. Model for a BLE connection event of a slave device. The waveform is not true to scale and hence, no values are marked on the axes.

(1) **Header (*head*):** At the beginning of a connection event, the device wakes up. Thereby, a current I_{head} is consumed for the duration of d_{head}.
(2) **Preprocessing (*pre*):** Prior to the actual communication, the device wakes up and prepares the functions related to the BLE standard, which are used during communication, such as the logical link control and adaptation protocol (L2CAP), generic access profiles (GAP), generic attribute profile (GATT), security manager (SM), etc. While the average current is nearly constant over time during this phase, its duration might vary depending on the BLE device, the BLE stack and the BLE functionality that is used (e.g., sending attributes with or without confirmation). In addition, random variations occur (see Appendix A for details).
(3) **Communication preamble (*cpre*):** The wireless communication is initiated in this phase. The current and duration in this phase may change with different BLE device models.
(4) **Window widening (*ww*):** The slave has to start listening for a certain amount of time before the master starts sending its first packet to compensate for the time drift that might be generated due to clock skew. This additional time is called *window widening* d_{ww}. It depends on the nominal sleep clock accuracy (SCA) of each transceiver, the connection interval T_c and the average slave latency $\overline{N_{sl}}$, as shown in Equation (1). The sleep clock is the clock that determines the points in time for waking up a sleeping device. The current magnitude of this phase is the same as the magnitude of the *rx*-phase. The duration d_{ww} can be calculated according to Equation (1) [23], assuming that the average clock skew is equal to zero (positive and negative skew compensate for each other).

$$d_{ww} = \frac{(SCA_{Ma} + SCA_{Sl}) \cdot T_c \cdot \overline{N_{sl}}}{10^6}, \qquad I_{ww} = I_{rx} \tag{1}$$

(5) **Reception (*rx*):** The over-the-air-bitrate of BLE is specified to be 1 MBits/s, therefore one bit is transmitted within 1 µs. Consequently, the *rx*-phase should ideally take $N_{rx} \cdot 8\,\mu s$, with N_{rx} being the number of bytes received. Because the RF circuitry needs some time to initialize, the duration of the *rx*-phase is slightly longer than its ideal value. To account for this, a correction-offset d_{prerx} is added to d_{rx} as in the equation below.

$$d_{rx} = N_{rx} \cdot 8\,\mu s + d_{prerx}. \tag{2}$$

The current I_{rx} is nearly constant; some devices, such as the CC2540 [25], have multiple reception-gain settings that cause different current draws.
(6) **Interframe-Space (*rxtx*, *txrx*):** After the *rx*-phase and before the slave starts sending a packet to the master, there is a phase for switching from reception to transmission and vice-versa. Its duration is slightly shorter than the over-the-air gap d_{IFS} between two packets.

(7) **Transmission(tx):** In this phase, the slave transmits data to the master. Its duration can be modeled in a manner similar to the *rx*-phase using Equation (3).

$$d_{tx} = d_{pretx} + N_{tx} \cdot 8\,\mu s \tag{3}$$

N_{tx} is the number of bytes transmitted. The current consumption I_{tx} depends on the *tx*-power that is used. In addition, I_{tx} slightly varies with the channel a packet is sent on. As the channel is determined by a pseudo-random hopping sequence, these variations can be assumed to occur randomly with a given standard deviation. The BLE protocol allows the transmission of multiple pairs of packets in a single connection event. After another *txrx*-phase, the *rx*, *rxtx* and *tx*-phases might be repeated to account for multiple pairs of packets. We evaluate such events with more than one pair of packet in Section 6.3.

(8) **Tx transient (tra):** After the data transmission has ended, the current decreases from I_{tx} to the postprocessing-current I_{post}. Whereas the current consumption drops with a RC curve, an effective constant current I_{tra} and an effective duration d_{tra} can be chosen appropriately to take the charge consumed by this into account.

(9) **Postprocessing (*post*):** After the communication has ended, the device may execute additional tasks, e.g., wired communication to a host processor or data buffering for the next transmission. Therefore, the duration of this phase is strongly dependent on the BLE device and its firmware. In our experiments, we measured the post-processing duration caused by a firmware created with Bluegiga's BGScript [3] without accounting for additional tasks. Both on BLE112- and CC2540-devices, d_{post} is subjected to strong random variations [11]. When executing one BLE functionality (e.g., sending a packet), we measured varying postprocessing times, as presented in the tables in Appendix A.

(10) **Tail ($tail$):** After the BLE device has completed the tasks related to the post-processing, it goes to a sleep mode to reduce the energy consumption. This phase in the model accounts for the current consumed for initiating the low power mode.

The phase durations observed in the over-the-air traffic differ from those in a measured current waveform. Here, we consider the phase durations of the current draw, since they determine the energy consumption.

For each phase *ph*, the charge consumed can be calculated by $Q_{ph} = d_{ph} I_{ph}$. With the phases described, the charge consumed for a connection event, Q_{cE}, can be computed as in Equation (4).

$$Q_{cE} = Q_{head} + Q_{pre} + Q_{cpre} + Q_{ww} + Q_t + Q_{tra} + Q_{post} + Q_{tail} \tag{4}$$

Q_t accounts for the actual communication taking place. For a communication with N_{seq} pairs of packets, it is

$$Q_t = \sum_{i=1}^{N_{seq}} (Q_{rx}(i) + Q_{tx}(i) + Q_{rxtx} + Q_{txrx} + Q_{to}) - Q_{txrx} \tag{5}$$

Q_{to} is an offset to account for distortions in the current curve. Due to the distortion of the current curve caused by resistive and capacitive elements in the power supply line, the shape of the phases of the current curve are not perfectly rectangular. In phases with constant lengths, these distortions can be compensated for by using effective values. In contrast, for the communication sequence, an offset term needs to be added, as the varying durations make it impossible to find effective values. The duration of a connection event can be calculated similarly to Equation (4) by adding the partial durations of all phases.

4.2.2 Scan Event. The current waveform of a scan event depends on the actual scanning mode that is carried out (e.g., passive scanning or active scanning). We describe the energy consumption for an event with active scanning, which is the most complex waveform, and simplify it for events with no reception of data and for events which receive connection-request packets. In the equations in

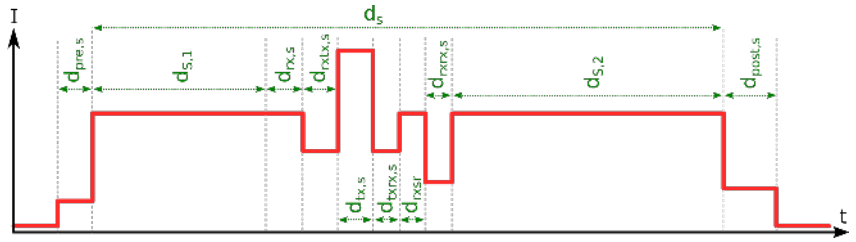


Fig. 4. Current waveform of a scan event for active scanning. The waveform is not true to scale.

this section, parameters denoted with an *s* indicate *scanning*, as the parameter values differ from the values in the connected mode. Figure 4 shows the current-waveform for active scanning, assuming that exactly one advertising event is received during one scan event. As for the connection event, the device wakes up, drawing a current for a duration of $d_{pre,s}$ time-units. Afterwards, it scans for the scanning time $d_{S,1}$ until an incoming advertising packet is received, consuming a high current $I_{scan} = I_{rx,s}$ due to the permanent usage of the receiver circuit. During the reception of the packet, the current consumption remains the same. The device then switches from receiving to transmission, which takes $d_{rxtx,s}$ amounts of time. Next, it sends a scan request packet lasting for $d_{tx,s} = N_{tx} \cdot 8\,\mu s + d_{pretx,s}$ time units. Here, N_{tx} is the number of bytes sent and $d_{pretx,s}$ is a correction offset accounting for charging and discharging capacitances. Afterwards, the devices switches from *tx* to *rx*, taking the time $d_{txrx,s}$ and subsequently, a scan response packet from the remote device is received. This takes $d_{rxsr} = N_{rx} \cdot 1\,\mu s + d_{prerx,s}$ time units. Before the device continues scanning, there is another turnaround phase with a constant current for $d_{rxrx,s}$ units of time. When the time $d_{S,2}$ has expired, the device stops scanning and goes to sleep, consuming a smaller current for the time $d_{post,s}$ before the sleep phase begins. The scan durations $d_{S,1}$ and $d_{S,2}$ depend on the time an advertising packet is received and are hence random. Fortunately, for calculating the energy consumption, only the sum of $d_{S,1}$ and $d_{S,2}$ is of interest, which is given by the scan window d_s. It is approximately $d_{scan} = d_s - d_{rxtx,s} - d_{tx,s} - d_{txrx,s} - d_{rxsr} - d_{rxrx,s}$. Therefore, the charge consumed by a scan event with active scanning is:

$$\begin{aligned} Q_{sEv} =& d_{pre,s} + d_{scan} I_{rx,s} + d_{rxtx,s} I_{rxtx,s} + d_{tx,s} I_{tx,s} + d_{txrx,s} I_{txrx,s} + d_{rxsr} I_{rx,s} \\ &+ d_{rxrx} I_{rxrx} + d_{post,s} I_{post,s} + Q_{crx,s} + Q_{ctx,s} \end{aligned} \tag{6}$$

$Q_{crx,s}$ and $Q_{ctx,s}$ are correction offsets to compensate for non-rectangular shapes in the reception and transmission phases. Parameter values for BLE112-devices are presented in Appendix A. In many cases, Equation (6) can be simplified. For sending a connection request packet, the waveform begins similarly to what is shown in Figure 4, but instead of a scan-request packet, a connection request packet is sent. In this case, the *tx*-phase is followed by the postprocessing phase without further sections in between. Our experiments showed that the pre- and post-processing phases for scanning with a connection request last longer than for active scanning on a BLE112-module. This behavior depends on the device and its BLE stack. For the sake of simplicity of explanation, we nevertheless assume these durations to be constant in the values presented in this paper, as this could easily be accounted for by adjusting the values to the given case. Because of the reduced number of phases in the case a connection is initiated, $d_{rxtx,s}$, $d_{rx,s}$, $Q_{crx,s}$ and $d_{rxrx,s}$ can be set to zero and d_{scan} is shortened to the time between the beginning of the idle scanning and the time the advertising packet has been received completely. If no advertising packet is received or only passive scanning is used, there is only idle-scanning (or the reception of advertising-packets, which consumes the same energy as idle-scanning) and the charge consumed can be computed in a simplified way:

$$Q_{sEv,Idle} = d_{pre,s,i} I_{pre,s,i} + d_{post,s,i} I_{post,s,i} + d_s I_{rx} \tag{7}$$

If d_s is long compared to $d_{pre} + d_{post}$, pre- and postprocessing can be neglected and the formula above can be simplified further. As pre- and postprocessing-durations and their effective currents differ significantly between active/idle scanning and scanning with establishing a connection, they are denoted with an index i for *idle*[1]. In many cases, it is hard to predict whether a scan event will receive an advertising packet or not and the point in time the reception takes place. For connection establishment, a solution for this problem is described in Section 4.3. For active scanning, discrete event simulations can be combined with the model described to get an estimate for the energy consumption. Another special case is continuous scanning ($d_s = T_s$): In this case, a scan event does not end with postprocessing, but the next *rx,s*-phase starts after d_s time-units have passed, with a short phase for channel-changing that consumes a charge of $d_{chch} \cdot I_{chch}$ in between.

4.2.3 Connection Procedure. In this section, we propose a model for the energy spent on establishing a connection and, as both procedures are similar, for updating the communication parameters of an existing connection. These procedures are described in Section 4.2.3 (cf. Figure 1(b)). To establish a connection, the master sends a connection request packet first, with the energy consumption associated with this event being $Q_{ev,cR,Ma}$. For updating the parameter values of an existing connection, the master consumes $Q_{ev,cU,Ma}$. Subsequently, there is no communication for i) $d_{sl,cR} = d_{two} + 1.25\,\text{ms}$ after the end of the connection request packet in the case of a connection establishment, or ii) $d_{sl,cU} = d_{two}$ after the end of the old connection interval in the case of a connection parameter update. After that, the transmit window begins, as shown in Figure 1(b). Within the size of the transmit-window d_{tw}, the master may schedule its first packet, thereby defining the anchor-point t_{Anchor} for future connection events. With some constraints, the master is free to choose the transmit window offset d_{two} between 1.25 ms and $T_{c,n}$, with $T_{c,n}$ being the future connection interval. For the transmit window size d_{tw}, the master can choose any value[2] in the following range.

$$1.25\,\text{ms} < d_{tw} < \min(10\,\text{ms}, T_{c,n} - 1.25\,\text{ms}) \tag{8}$$

With $d_p < d_{tw}$ being the duration from the beginning of the transmit window to the time the first packet sent by the master, the charge consumed by the master for establishing a connection can be modeled as:

$$Q_{cE,Ma} = Q_{ev,cR,Ma} + (1.25\,\text{ms} + d_{two} + d_p) I_{sl} \tag{9}$$

For a connection update, it is:

$$Q_{cU,Ma} = Q_{ev,cU,Ma} + (T_{c,o} + d_{two} + d_p - d_{ev,cU,Ma}) I_{sl} \tag{10}$$

I_{sl} is the sleep current of the BLE device. $T_{c,o}$ is the connection interval before the connection parameter update took place. $Q_{ev,cR,Ma}$ and $Q_{Ev,cU,Ma}$ are the charges consumed by the events the connection request/update packets are sent within. For an event with a connection request packet, these values can be modeled as described in Section 4.2.1 as an advertising event with 37 bytes sent by the advertiser and with a response of 44 bytes length sent by the scanner. Connection update requests can be modeled as a connection event with a 22-byte packet sent by the master, with packets received from the slave depending on the payload the slave has to send.

The master's energy-consumption mainly stems from sending the connection request/update packet to the slave. Opposed to that, the energy consumption of the slave is dominated by the

[1]For the BLE112-stack, our measurements had the following limitations: 1) for scanning with connection-requests, only continuous scanning ($d_s = T_s$) was possible. 2) the time the device was actually scanning was slightly below d_s. For scan windows between 12.5 ms and 250 ms, it was approx. 1.85 ms shorter on the average over 1725 analyzed events.

[2]As integers are used for all parameters in BLE, the values of d_{tw} and d_{two} are quantized and must be a multiple of 1.25 ms.

current of its receiver during idle-listening in the transmit window and during the reception of the packet. It can be modeled by the following equations:

$$Q_{cE,S} = Q_{ev,cR,Sl} + (d_{two} + 1.25\,\text{ms} - d_{ww,cE})I_{sl} + (d_p + d_{ww,cE})I_{rx} \tag{11}$$

and

$$Q_{cU,S} = Q_{ev,cU,Sl} + (T_{c,o} + d_{two} - d_{ww,Cu} - d_{ev,cU,Sl})I_{sl} + (d_p + d_{ww,cU})I_{rx} \tag{12}$$

$Q_{ev,cR,Sl}/d_{ev,cR,Sl}$ is the charge consumed/duration spent by the event in which the connection request is sent. Similarly, $Q_{ev,cU,Sl}/d_{ev,cU,Sl}$ account for the event in which the connection update-packet is received. As $Q_{ev,cR,Sl}$ is a scan event, it can be modeled according to Section 4.2.2, whereas $Q_{ev,cU,Sl}$ is a connection event and can therefore be modeled according to Section 4.2.1. I_{rx} is the current consumption when the receiver listens to the radio channel. The first connection event after this procedure has to be modeled without window widening, as the equations above already account for it. Nevertheless, window widening occurs in the procedure itself, which broadens the transmit window. $d_{ww,cE}$ and $d_{ww,cU}$ contain the times the transmit window is broadened by. With SCA_{Ma} and SCA_{Sl} being the clock accuracies of master and slave in parts per million, the window-widenings for connection establishment $d_{ww,cE}$ and for connection parameter updates $d_{ww,cU}$ are:

$$d_{ww,cE} = \frac{SCA_{Ma}+SCA_{Sl}}{1000000}(1.25\,\text{ms} + d_{two}) \qquad d_{ww,cU} = \frac{SCA_{Ma}+SCA_{Sl}}{1000000}(T_{c,o} + d_{two}) \tag{13}$$

In Equations (9) and (10), we neglected the fact that the sleep duration is slightly shorter than what has been modeled, as the first event after the connection establishment or update partially overlaps with the sleep duration. However, due to the small sleep current of BLE devices, the impact of this is negligible. The values of d_{tw}, d_p, d_{two} are chosen freely by the BLE stack and are therefore unknown, making the equations above hard to evaluate. However, a worst case-model can be defined. Both for the master and the slave, the maximum energy is consumed if both parameters have their maximum values, viz. $d_p = \min(10\,\text{ms}, T_{c,n} - 1.25\,\text{ms})$ and $d_{two} = T_{c,n}$. Moreover, even if not explicitly defined by the BLE specification, some assumptions on the values of these parameters can be made for real-world devices, that lead to an estimation of the average energy consumed for establishing a connection or updating the connection parameters. Both parameters are used to allow the master to schedule a new anchor point t_{Anchor}. At this point in time, the first connection event of the new/updated connection starts. If the master already knows this point in time when sending the connection request/update packet to the slave, the first connection event can be scheduled by setting d_{two} appropriately. In addition, d_{tw} leaves open a time interval for the master to schedule the anchor point without determining it when sending the connection request or update packet. As there is no reason for the master for not being able to schedule t_{Anchor} already when sending the connection parameters to the slave, we assume that it will always choose the smallest possible value of d_{tw} (i.e., $\min(10\,\text{ms}, T_{c,n} - 1.25\,\text{ms})$). We further assume that every point in time within the transmit window is taken with the same likelihood. Hence, on an average, the master will schedule t_{Anchor} at the middle of this interval, i.e. $d_p = \frac{d_{tw}}{2}$. If the master chooses small transmit windows t_{tw}, there is a clear benefit for its energy consumption and as a consequence, an optimized BLE stack is likely to do so. However, making reasonable assumptions for the value of d_{two} is more difficult. d_{two} only influences the charge spent by sleeping and by window widening. The energy consumption related to this is negligible against the energy consumed for idle-listening within d_{tw} and for receiving a packet. From an energy-perspective, the BLE stack can chose an arbitrary value. For energy modeling, d_{two} can be assumed to have its maximum value $T_{c,n}$, if the actual value of the device is unknown.

Whereas a coarse estimation of the energy consumed by the procedure described can be made using these assumptions, for more precise energy estimations, the parameters actually chosen by the BLE stack considered need to be analyzed. We analyzed the values chosen by the stack of a

Bluegiga BLE112-device using a Texas Instrument's packet sniffer software [24]. The corresponding setup is depicted in Figure 6(b). A passive BLE device was used to wiretap the BLE traffic between two different communicating devices. We developed a parser for the logs generated by the SmartRF Packet Sniffer software [24], allowing for the import of the sniffed data into MATLAB.

By manually experimenting different parameter values, we found that $d_{tw} = 3$ ms is constant for all parameter values both for connection establishments and parameter updates. We further analyzed 26 connection procedures, where d_p had an average value of 1.43 ms, which is close to $\frac{d_{tw}}{2}$, as we had assumed. d_{two} may vary for different values of T_C when establishing a new connection, while being constantly equal to zero for connection updates. We developed a software that repeatedly connected and disconnected two BLE nodes. The packet-sniffer-logs were analyzed to obtain d_{two} for different values of T_c. The sniffer was able to measure time with a resolution of 1 µs, and the measurement accuracy was mainly determined by the skew of the RC-oscillator on the BLE device. Assuming a clock accuracy of $50 \cdot 10^{-6}$, the maximum skew would be 50 µs within a measured time period of 1 s. An analysis of about 12,000 connection procedures revealed the following results for a BLE112-device: d_{two} grows with increasing values of T_c, but there are some random variations for the same value of T_c. The following approximations are based on a linear least squares fitting. For $T_c > 12.5$ ms, an estimation for d_{two} is given by $d_{two} = T_c - 6.454$ ms. The maximum deviation that occurred in all the measurements was 6.046 ms (i.e., 0.87 % of the corresponding measured value), the mean square error was 9 µs and the standard deviation σ was 2.959 ms. For $7.25\text{ ms} < T_c < 12.5$ ms, a good estimation for d_{two} is given by $d_{two} = 0.389T_c + 0.484$ ms. Here, the mean square error that occurred in our measurements is 6.3 µs, the standard deviation σ is 2.527 ms and the maximum deviation is 5.140 ms (i.e., 51.4 % of the corresponding measured value). These values are valid for BLE112-devices in piconets with one slave. For other situations or different hardware, these measurements have to be repeated.

4.3 Neighbor Discovery / Advertising and Scanning

Advertising and scanning in BLE were described in Section 2. During these phases, the advertiser (A) is not synchronized with the scanner (S) and the reception of a packet sent to the scanner by A is not guaranteed. Instead, the time after which a scanner receives an advertising packet needs to be described as a probabilistic process. In the following, we derive a model for the mean latency $\overline{d_{adv}}$ until discovery occurs. Here, the discovery latency $\overline{d_{adv}}$ is defined as the average time from the first advertising packet sent in range until there is a successful reception by the scanner.

4.3.1 Problem Definition. Figure 1(a) shows a typical advertising/scanning situation. A scanner S starts listening at its first scan event at time $t = 0$. An advertiser A begins advertising or comes into range after a random offset ϕ relative to the beginning of an arbitrary scan event. It periodically repeats its advertising event with period $T_a = T_{a,0} + \rho$, whereby ρ is a random offset between 0 ms and 10 ms. The question we address is: How much time will pass, until an advertising event "meets" a scan event for the first time?

An advertising event i is received successfully by the scanner, if its starting time t_a is contained within a set of suitable times $t_{i,suc}$. The hatched areas in Figure 1(a) on the left side of the scan events show the times d_{early} an advertising event can begin before the scan event starts to be received successfully. The hatched areas at the right sides of the scan events show the times d_{late} that do not lead to a successful reception while a scan event takes place. This is due to each advertising event consisting of three packets on three different channels. $t_{i,suc}$ is defined by the necessary and sufficient condition that the advertising packet on the channel the scanner is scanning on must overlap completely with a scan event. The interval $t_{i,suc}(t_{sE})$ for a given scan event at time t_{sE} begins d_{early} time units before the beginning of the scan event t_{sE} and ends d_{late} time units

Channel	d_{early}	d_{late}	d'_s
Ch 37	0	d_a	$d_s - d_a$
Ch 38	$d_a + d_{ch}$	$2d_a + d_{ch}$	$d_s - d_a$
Ch 39	$2d_a + 2d_{ch}$	$3d_a + 2d_{ch}$	$d_s - d_a$

Table 2. d_{early} and d_{late} and the effective scan window d'_s for the three advertising channels [18]. d_a is the duration of an advertising packet, d_{ch} the duration of hopping to the next channel, as shown in Figure 1(a).

before its end $t_{sE} + d_s$. If an advertising packet has the duration d_a and changing the channel takes d_{ch} amounts of time, d_{early} and d_{late} depend on the channel of the scan event and as given by Table 2 (cf. also [18]). As can be seen easily, the effective scan window $d'_s = d_s + d_{early} - d_{late}$ is constant for all channels. In the following, we present an algorithm that estimates $t_{i,suc}$ and, based thereupon, the discovery latency $\overline{d_{adv}}$ and the associated energy consumption.

4.3.2 Numeric Approximation. In general, the problem of calculating the expected discovery latency $\overline{d_{adv}}$ is a complex probabilistic problem. Except for trivial special cases (i.e., $T_{a,0} < d_s - 10\,\text{ms}$ and $T_s = d_s$), to the best of our knowledge, no closed-form solution is known.

Algorithm 1 provides an approximate value of the expected discovery latency $\overline{d_{adv}}$. It works as follows. Let us start with a given offset ϕ between a scan event and an advertising event. For this offset ϕ, we calculate the probability of a successful reception $p_{hit}(n)$ of each advertising packet n sent at $\phi + n \cdot T_{a,0} + \rho$, $n = 0, 1, ...$, given that all previous events have not been received by the scanner. Without the random advertising delay ρ, this probability would be either 0 or 1 for each event - if the advertising begins within $t_{i,suc}$, it is 1, for all other cases, it is 0. With a random offset ρ, the time at which an advertisement event begins, t_{advEvt}, can lie within a broad time interval, which widens for increasing event numbers n. Therefore, p_{hit} depends on n. ρ is modeled as a random variable ρ, having the distribution

$$f(\rho) = \begin{cases} \frac{1}{10\,\text{ms}} & \text{, if } 0 \leq \rho \leq 10\,\text{ms} \\ 0 & \text{, else.} \end{cases} \tag{14}$$

As the Bluetooth specification [23] only specifies the possible range of values for ρ without suggesting its distribution, we thereby assume a uniform distribution. The time t_{advEvt} an advertising event begins can be modeled as in Equation (15):

$$t_{advEvt}(n, \phi) = \underbrace{\phi + n \cdot T_{a,0}}_{A} + \underbrace{\sum_{1}^{n} \rho}_{B} \tag{15}$$

For calculating the probability p_{hit} of whether an advertising event is successfully received, the probability density function (PDF) $t_{advEvt}(n, \phi)$ for the starting time of an advertising event is required. For Term B in Equation (15), the shape of the distribution depends on n. For n = 1, this distribution is uniform. As n random offsets ρ occur, the resulting distribution can be described as the sum of n independent and identical random variables ρ. In general, the PDF of a sum of n random variables is the convolution product of their PDFs [30]. According to the central limit theorem for large n, the convolution product passes into a Gaussian distribution having the effective mean $\mu' = n\mu$ and the effective standard deviation $\sigma' = n\sigma$ [30]. Accordingly, for the distribution $f(B)$ of B, we assume a Gaussian distribution with the mean $\mu_\rho = n \cdot 5\,\text{ms}$ and standard deviation $\sigma_\rho = \sqrt{n/12} \cdot 10\,\text{ms}$ for an arbitrary $n > 2$. For n = 1, the uniform distribution from Equation (14) is used. For n = 2, we assume the distribution to be a symmetric triangular distribution with $f(B) = f(\rho) * f(\rho)$ (i.e., the exact solution of the convolution product). For the sake of simplicity of exposition, in this section, a Gaussian distribution is also assumed for $n = 1$ and $n = 2$. However, the values we present in this paper have been calculated without using this simplification. The

Term A in Equation (15) causes a shift of the mean of $f(p_{hit})$. The approximate distribution for the time $t_{advEvt}(n, \phi)$ is:

$$f(t_{advEvt}(n,\phi)) = \Phi\left(\frac{\phi - n \cdot T_{a,0} - n \cdot 5\,\text{ms}}{\sqrt{\frac{n}{12}} \cdot 10\,\text{ms}}\right) \tag{16}$$

$\Phi(t)$ is the cumulative distribution function of the standard normal distribution. Therefore, p_{hit} can be calculated as in Line 9 of Algorithm 1 by evaluating $f(t_{advEvt}(n, \phi))$ for all scan intervals k the beginning of the advertising event $t_{advEvt}(n)$ might lie within. The first scan event that might receive the advertising event n is given by k_{min} and the last possible one is scan event number k_{max}. The indices k_{min} and k_{max} can be calculated using the formulas in Line 5 of the algorithm. With t_{ai} being the ideal point in time an advertising-event starts (i.e. without the random advertising delay ρ), p_k in Line 9 can be calculated. $p_k(k, n, t_{ai}, d_{early}, d_{late}, T_s)$ is the probability for the advertising event n being received successfully by the scanner. It can be calculated as follows.

$$p_k = \Phi\left(\frac{kT_s + d_s - d_{late} - t_{ai} - n \cdot 5\,\text{ms}}{\sqrt{\frac{n}{12}} \cdot 5\,\text{ms}}\right) - \Phi\left(\frac{kT_s + d_{early} - t_{ai} - n \cdot 5\,\text{ms}}{\sqrt{\frac{n}{12}} \cdot 5\,\text{ms}}\right) \tag{17}$$

Thus, the expected latency for a given offset ϕ can be calculated as in Line 10 of the algorithm. $p_{cM}(n)$ is the probability that n advertising events do not lead to a successful reception (cumulative miss probability). In Line 12 of Algorithm 1, the probability for n cumulative misses is calculated for the current advertising event under consideration. d_{exp} in Line 10 is a good approximation of the exact expected discovery latency given a certain offset ϕ.

With increasing values of n, the probability that one of the advertising events considered until so far is received successfully grows. Consequently, p_{cM} shrinks with growing values of n. The algorithm finishes, if $(1 - p_{cM})$ is smaller than a lower bound ϵ. The higher ϵ is, the better the accuracy of the algorithm becomes, but the corresponding computational complexity increases.

Following the steps described above, the resulting values of d_{exp} must be integrated over all possible values of ϕ. We perform a simple numerical integration by evaluating the integral for $\phi \in [0, 3 \cdot T_s]$ in steps of Δ, multiplying the results with Δ and computing the sum of these values. The variable *ch* contains the value of the current channel the scanner listens to and is calculated in Line 7 of the algorithm. The function *getInterval(ch)* used in the algorithm looks up d_{early} and d_{late} from Table 2. The function $d_{advEvnt}(ch)$ calculates the duration of an advertising event that is received successfully by the scanner on the current channel, as described in Section 4.2.1. This duration is given by d_a for Channel 37. For Channel 38, it is $2d_a + d_{ch}$ and for Channel 39, it is $3d_a + 2d_{ch}$. In order to bound the computation time, the inner while-loop of Algorithm 1 should be aborted if d_{exp} exceeds an upper bound $d_{exp,max}$. In practice, this is not a real limitation, since parametrizations leading to latencies larger than, for example, $d_{exp,max} = 1000$ s are no reasonable choices for real world applications. The algorithm has two parameters ϵ and Δ that influence the accuracy of the results. The smaller Δ is and the closer ϵ gets to 1, the more accurate the results become. Compared to discrete event simulations, this algorithm has a reduced complexity when choosing these values appropriately, as not all scan events need to be examined, and as different values for ρ are accounted for implicitly by the Gaussian distribution.

Figure 5 shows the modeled discovery latencies for different values of $T_{a,0}$ and for different scan window lengths d_s. It leads to some interesting observations:

- Smaller values of d_s lead to higher advertising latencies $\overline{d_{adv}}$ for all values of $T_{a,0}$.
- For $T_{a,0} < d'_s - 10\,\text{ms}$, the function rises only slowly and there are no peaks.
- For some parameter values (for example, for $T_{a,0} + 5\,\text{ms} = n \cdot T_s$), the expected advertising latency d_{adv} becomes very large. These are coupling phenomena. If the first advertising event

Algorithm 1 Calculation of the discovery latency $\overline{d_{adv}}$ for case 2b

$d_{adv} \leftarrow 0$
for $\phi = 0$ to $3T_s$ step Δ **do**
  $n \leftarrow 0, d_{exp} \leftarrow 0, p_{hit} \leftarrow 0, p_{cM} \leftarrow 1, ch \leftarrow 37$
  while $1 - p_{cM} \leq \epsilon$ **do**
    $t_{ai} \leftarrow \phi + nT_{a,0}, k_{min} \leftarrow \lfloor \frac{t_{ai}}{T_s} \rfloor, k_{max} \leftarrow \lfloor \frac{t_{ai}+n\cdot 5\,\text{ms}}{T_s} \rfloor, p_{hit} \leftarrow 0$
    for $k = k_{min}$ to k_{max} **do**
      $ch \leftarrow mod(j, 3) + 37$
      $(d_{early}, d_{late}) \leftarrow getInterval(ch)$
      $p_{hit} \leftarrow p_{hit} + p_k(k, n, t_{ai}, d_{early}, d_{late}, T_s)$
      $d_{exp} \leftarrow d_{exp} + p_k\, p_{cM} \cdot (n \cdot (T_{a,0} + 5\,\text{ms}) + d_{advEvnt}(ch))$
    end for
    $p_{cM} \leftarrow p_{cM}(1 - p_{hit}), n \leftarrow n + 1$
  end while
  $d_{adv} \leftarrow d_{adv} + d_{exp}$
end for
$\overline{d_{adv}} \leftarrow \frac{d_{adv}}{\lfloor \frac{3T_s}{\Delta} \rfloor}$

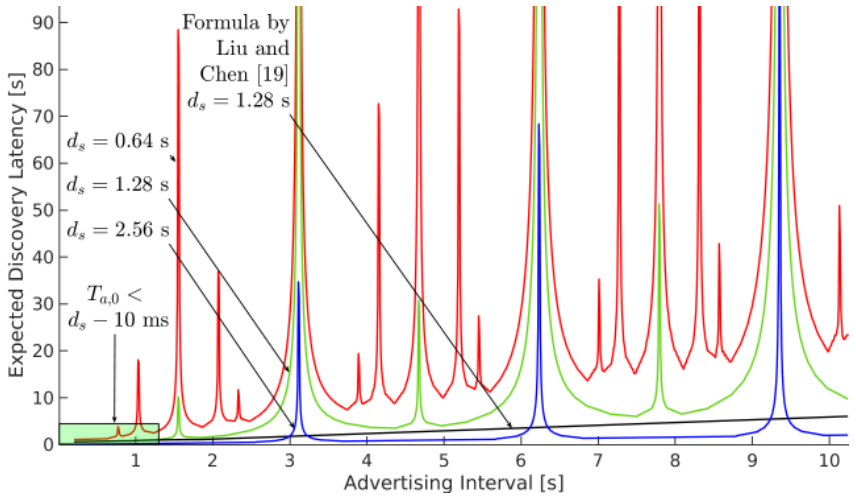


Fig. 5. Results of the algorithm for $T_s = 3.12$, $\epsilon = 0.9999$, $\Delta = 93.6$ ms, $d_a = 446\,\mu$s, $d_{ch} = 150\,\mu$s. The solid line in the bottom shows the results of the analytical model presented in [19] with $T_s = 2.56$ s and $d_s = 1.28$ s.

does not hit a scan event and its advertising interval is for example close to the multiple of the scan interval T_s, the next advertising events have a high probability of also missing the scanner. For small scan windows d_s, the number of coupling-peaks increases.

- In contrast, local minima exist, which lead to short discovery latencies $\overline{d_{adv}}$.

Note that if $T_{a,0}$ becomes smaller than the effective scan window length d'_s (d'_s is the scan window length accounting for d_{early} and d_{late}, as given by Table 2) minus the maximum random delay of 10 ms, the distance between two subsequent advertising packets falls below the length of a contiguous reception phase and hence, no scan event can be missed. As a result, the latency becomes bounded. This area is highlighted by the frame in the lower left of Figure 5. A closed-form model for this special case has been presented in [19]. As can be seen, this previously known model does not apply for larger values of $T_{a,0}$.

4.4 Computing the Number of Events

In the previous sections, we have computed the charges consumed for every type of event in BLE. Further, in Section 4.2.3, we have computed the discovery latencies in advertising/scanning mode. The energy consumption of a device is given by relating the charges consumed by these events to their frequencies of occurrence. In this section, we first compute the frequency of occurrence in the connected mode, and then the average number of events needed for neighbor discovery. This results in the overall current consumption of a device.

4.4.1 Connected Mode. For a master, the number of connection events $N_{c,ma}$ in a given amount of time T_g is constant. For the slave, the number of connection events $N_{c,sl}$ is different as it may skip some events due to the slave latency parameter N_{sl}. With an average number of $\overline{N_{sl}}$ skipped events, the number of events per interval are given by:

$$N_{c,sl} = \left\lfloor \frac{T_g}{\overline{N_{sl}} T_c} \right\rfloor, \qquad N_{c,ma} = \left\lfloor \frac{T_g}{T_c} \right\rfloor \tag{18}$$

The overall charge consumption of the master or the slave per T_g is the sum of the event energies and the sleeping energy:

$$\overline{Q_{con,Ma/Sl}} = \sum_{n=1}^{N_c} Q_{event}(n) + \left(T_g - \sum_{n=1}^{N_c} d_{event}(n) \right) I_{sl} \tag{19}$$

In Equation (19), N_c can be set to the number of connection events for the master ($N_c = N_{c,ma}$) or for the slave ($N_c = N_{c,sl}$). Q_{event} can be calculated according to Section 4.2. I_{sl} is the sleep current of the device and $d_{event}(n)$ the duration of event n.

4.4.2 Advertiser. The energy consumption of one advertising event $Q_{advEvent}$ can be modeled as described in Section 4.2. For calculating the expected energy consumption $\overline{Q_{adv}}$ for a discovery procedure of the advertiser, it must be taken into account that not all advertising events are identical. Events that are successfully received might be shorter than the other events, since packets on different advertising channels are skipped after a successful reception and since the *rx*-phase takes longer due to receiving the response. Algorithm 1 can easily be extended for an estimation of the advertiser's energy. With $\overline{Q_{adv}}$ being the expected energy of the advertiser, one could add the following assignment after Line 10 of Algorithm 1:

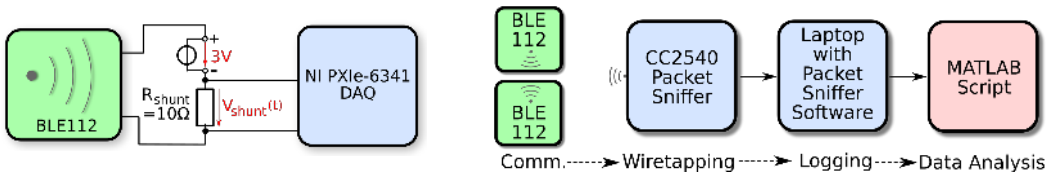
$\overline{Q_{adv}} \leftarrow \overline{Q_{adv}} + p_k\, p_{cM}((n-1)(Q_{full} + (T_{a,0} + 5\,\text{ms} - d_{full})I_{sl}) + Q_{last}(ch))$

Q_{full} is the charge consumed by a full advertising packet that is sent on all three channels without receiving a response and d_{full} is its duration. $Q_{last}(ch)$ is the charge of a packet that is successfully received on channel *ch*. It can be calculated according to Section 4.2.

4.4.3 Scanner. For assessing the charge spent during discovery by the scanner, one can modify Algorithm 1 as described above for the advertiser. An alternative is an approximation based on $\overline{d_{adv}}$, which works as follows. We assume that the advertiser starts advertising with a random offset ϕ with a maximum value of $3T_s$ from the beginning of the fist scan event. The power consumption of idle scanning before the advertiser starts advertising is not accounted for in the equations below. The expected energy consumption of the scanner $\overline{Q_s}$ is given by $\overline{Q_{active}} + \overline{Q_{sleep}}$, with

$$\overline{Q_{active}} = \overline{N_s} Q_{sEv,idle}(d_s), \qquad \overline{Q_{sleep}} = \overline{N_s}(T_s - d_s) I_{sl}, \qquad \overline{N_s} = \frac{\overline{d_{adv}}}{T_s} \tag{20}$$

$\overline{N_s}$ in Equation (20) is the expected number of scan events that occur within $\overline{d_{adv}}$. $\overline{Q_{active}}$ accounts for the time the scanner is actively scanning and $\overline{Q_{sleep}}$ accounts for the sleep current I_{sl}. $Q_{sEv,idle}$ is the energy consumed by a scan event without receiving an advertising packet, as described in Section 4.2.2, neglecting the energy consumed by sending the response.



(a) Current Measurements (b) Time Measurements for Connection Setup

Fig. 6. Measurement Setups

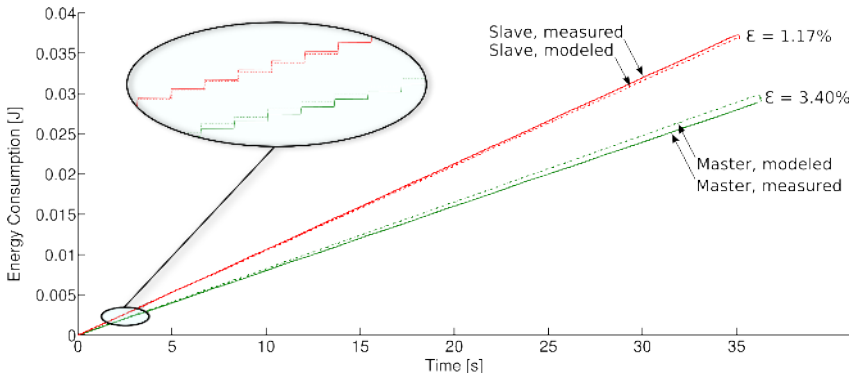


Fig. 7. Modeled and measured energy consumption for sending a test sequence. The test sequence consists of packets with varying payloads between 0 and 20 bytes. Here, $T_c = 100$ ms.

4.5 Parameter Extraction

To obtain the parameter values needed for the model equations described, a semi-automatic procedure has been used for generating current waveforms and extracting the values. On a BLE112 device with a custom firmware created using BGScript [3], we altered the values of important protocol parameters, such as the number of bytes sent in each packet, and measured the waveforms of more than 5000 connection events. The measurements were taken according to [11] using a National Instruments NI PXIe-6124 data acquisition board (DAQ) with a sampling rate of 100 kHz. We calculated the current consumption $I(t)$ of the chip by analyzing the voltage $V_{shunt}(t)$ measured by the DAQ across a $10\,\Omega \pm 1$ % shunt resistor in the power supply of the BLE device, as depicted in Figure 6(a). The measured waveforms were analyzed using MATLAB scripts we developed. The model parameter values we obtained using these scripts are presented in Appendix A and B.

5 EVALUATION OF THE MODEL

In this section, we discuss the validity of our model. Towards this, we conducted measurements both in the non-connected and connected mode and compare measured and modeled data.

5.1 Connected Mode

To compare the modeled and the measured energy consumption in the connected mode, we set up a test scenario. The main goal of this scenario is to continuously vary important parameter values for obtaining comparisons over a large fraction of the whole design space. The scenario worked as follows. Two devices exchanged data using a connection interval T_c of 100 ms. One device sent data with an attribute-write request, the other node acknowledged the data received. We did this comparison for both a master and a slave sending the payload. The payload sent varied continuously from 1 to 20 bytes, the number of packets sent per event was one per direction. The

number of total bytes received in a response was either 10 or 15, depending on the event. The sequence of connection events we had carried out repeatedly is as follows:

(1) Send a packet with a payload sweeping between 0 and 20 bytes plus 17 bytes overhead and receive an empty polling packet with 10 bytes total length.
(2) Send an empty polling packet with a total length of 10 bytes and wait for the acknowledgement packet with 15 bytes length.

As the BLE112 device showed an unexpectedly long duration of the first *rx*-phase of each connection event when acting as a slave, we did not use the value of d_{prerx} from the tables in the Appendix, but used the effective duration of $d_{prerx} = 388\,\mu s$[3] for the slave.

To achieve high measurement accuracies, we scheduled the preparation of the packets using a separate task that was run periodically on the BLE module. The current waveforms generated by this task were filtered out. Even though we used high-quality measurement equipment, the bias current of the measurement device was larger than the sleep current of the BLE device. Therefore, on the average, its measured value was slightly negative. A Matlab script detected the points in time at which each connection event occurred in the measured current waveform and triggered the model to compute the charge consumed for the same event at the same point in the simulated time.

Figure 7 shows the results of the experiment described. The curves depict the energy-consumption over time for a master and a slave, both computed with our proposed model and obtained by measurements. As can be seen, for the slave, the total error of the mean current predicted by our model $\varepsilon_m = \frac{|I_{measured} - I_{model}|}{I_{measured}}$ is 1.17 %. When only considering the connection-events without the sleep periods in between the events, the relative error ε_m of the current is 3.5 %. If only the parts of an event beginning with the communication preamble and ending with the *tx*-transient phase are considered, ε_m is 5.1 %. This error is caused mainly by a missprediction of the rx-phase duration. As the error for the whole event is lower than that, some of the error is compensated by the pre- and postprocessing phases. In our test-scenario, we sent the payload and waited for a confirmation from the remote node, whereas we obtained the model-values in Appendix A for communication without confirmation. The confirmation causes a rise of the average pre- and postprocessing duration. By adjusting these values for sending with confirmation ($d_{pre} = 413\,\mu s$, $d_{post} = 1077\,\mu s$), the error of the whole event including the sleep current increases to 6.0 %, as the error-compensation described does not occur anymore.

For the master, the overall error of the mean current in the test scenario ε_m is 3.4 %. Considering the connection events only, ε_m is 0.6 %. Therefore, the error for only the events is lower than for the slave. The parameter values in the tables of Appendix A have their maximum precisions for BLE slaves and differ slightly for a master. In particular, d_{Cpre} is longer for the master, but this error is compensated by the pre- and postprocessing phases. As a consequence, the higher overall error must be caused by the sleep current that we could not measure precisely, as described above. For the slave, this error further reduces the overall error by compensating overestimations of the events by the model. In conclusion, the error between the model and real-world measurements is low (less than 3.4%), which makes the model usable for energy estimations of practical applications.

5.2 Advertising/Scanning

To verify the accuracy of the discovery latencies estimated by Algorithm 1, we compare its predictions to measured latencies. The results from this comparison are depicted in Figure 8. The dashed curve shows the computed discovery latency $\overline{d_{adv}}$ for different advertising intervals $T_{a,0}$. In addition to this curve, which has been obtained by using Algorithm 1, each cross in the figure shows

[3]This value has been calculated based on our experiments and is valid on a BLE112 device for the first *rx*-phase of a connection event of a slave. A measurement with a CC2540-device with TI's BLE stack did not show such an abnormality.

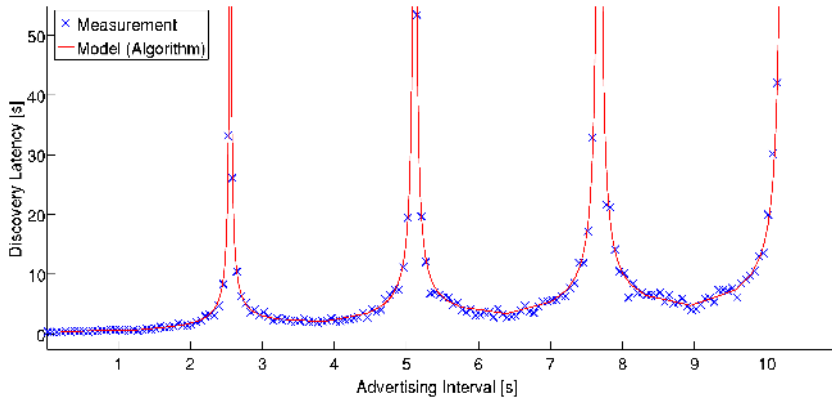


Fig. 8. Measured discovery latency compared to modeled values. Model parameters: $T_s = 2.56\,\text{s}, d_s = 1.28\,\text{s}, \epsilon_m = 0.9999, \Delta = 76.8\,\text{ms}, t_{max} = 1000\,\text{s}, d_a = 446\,\mu\text{s}, d_{ch} = 150\,\mu\text{s}$.

the mean value of 100 real-world measurements that have been repeated with the same parameter values. These measurements have been performed as follows. Two BLED112-modules have been connected to a PC and controlled by two separate processes on this PC. During all times, one module was scanning with a fixed set of parameters. As soon as a scan response was received, the process controlling this module sent the time of reception to another process controlling the second module. Inter-process communication was handled by Unix-Domain-Sockets. The process controlling the second module started the advertising procedures at random points in time between 0 s and $3T_s$ and calculated the duration between the start of each advertising procedure and the first reception by the scanner. After having repeated this 100 times for one advertising interval, the average duration was computed and $T_{a,0}$ was increased by 63.5 ms. This sequence was repeated until the maximum advertising interval was reached. Advertising intervals that lied within a coupling peak have been skipped to limit the wall-clock time of the measurement. As it can be seen in Figure 8, results from the measurements and results obtained from the model lie in close proximity. In addition to these real-world measurements, we have compared results obtained from the model to results from extensive discrete event simulations, which also confirmed the validity of our model.

6 RESULTS AND CONCLUSION

6.1 Sensitivity Analysis

Some of the durations and current magnitudes of the individual phases in a connection event (e.g., post-processing or transmission) vary randomly. For example, the duration of the post-processing phase d_{post} is subjected to strong random variations. Another varying value is the current magnitude I_{tx} in the transmission phase. If the standard deviations and the minimal/maximal values for the varying parameters are known, the impact of these variations on the overall current consumption within one connection interval can be determined. These variations are important for computing maximum, short-time current consumptions.

In this section, we analyze the impact of variations of d_{post} and I_{tx} on the energy consumption in one connection interval. For other parameters, this sensitivity analysis can be done similarly. Given a phase ph, $d_{min} \leq d_{ph} \leq d_{max}$, $I_{min} \leq I_{ph} \leq I_{max}$, the sensitivity S on the duration d_{ph} or current I_{ph} of a phase is:

$$S(d_{ph}) = {}^{\Delta Q_{total}}/_{\Delta d_{ph}} \qquad S(I_{ph}) = {}^{\Delta Q_{total}}/_{\Delta I_{ph}} \tag{21}$$

Q_{total} is the total charge consumed within one connection interval, as calculated in Equation (19) with $N_c = 1$. All parameter values can be assumed to be independent from each other. Therefore, the

sensitivity analysis can be done by accounting for the variations of each parameter individually, assuming that all other parameters have fixed values. With Q_{sl}/I_{sl} being the charge consumed/current drawn for sleeping, the sensitivity $S_{d_{ph}}$ can be written as:

$$S(d_{ph}) = \frac{\Delta Q_{ph}}{\Delta d_{ph}} + \frac{\Delta Q_{sl}}{\Delta d_{ph}} = \frac{Q_{ph}(d_{max}) - Q_{ph}(d_{min}) - (d_{max} - d_{min})I_{sl}}{(d_{max} - d_{min})} \tag{22}$$

The sensitivity on the current I_{ph} consumed during a phase ph is

$$S(I_{ph}) = \frac{\Delta Q_{ph}}{\Delta I_{ph}} = \frac{Q_{ph}(I_{max}) - Q_{ph}(I_{min})}{I_{max} - I_{min}} \tag{23}$$

By assuming that the current I_{ph} and the duration d_{ph} of a phase are independent from each other, one can further simplify:

$$S(I_{ph}) = d_{ph}, \qquad S(d_{ph}) = I_{ph} - I_{sl} \tag{24}$$

For the duration of the post-processing phase d_{post}, which is the parameter that was subjected to the strongest variations in our measurements, the change in the charge-consumption of a connection interval is (using $S(d_{post}) = I_{post} - I_{sl} \approx 8.0\,\text{mA}$, $\Delta d_{post} = 0.5\,\text{ms}$, $Q_{total} = 25\,\mu\text{C}$): $\Delta Q_{total}(d_{post}) = \Delta d_{post} \cdot S(d_{post}) = 0.5\,\text{ms} \cdot 8\,\text{mA} = 4.0\,\mu\text{C}$. The relative change $\frac{\Delta Q_{total}}{Q_{total}}$ is 16 %. This means, due to variations in the post-processing phase, the energy consumption of a whole connection event might vary by 16 %. The relative change in the energy consumption of a connection interval caused by variations in the transmission current I_{tx} is (using $S(I_{tx}) = d_{tx} \cdot N_{seq} = 350\,\mu\text{s}$, $\Delta I_{tx} = 0.9\,\text{mA}$, $Q_{total} = 29\,\mu\text{C}$): $\frac{\Delta Q_{total}}{Q_{total}} = \frac{0.9\,\text{mA} \cdot 350\,\mu\text{s}}{29\,\mu\text{C}} = 1\,\%$. Here, Δd_{post} and ΔI_{tx} roughly correspond to the differences between the minimum and maximum values given in Appendix A.

6.2 Using the Model

Along with this paper, an implementation of our proposed energy model has been made available [13]. It is written as a C-library that provides easy-to-use functions for estimating the energy consumption of a BLE device and comes with a complete documentation in the Doxygen format. It has a small resource demand and does not rely on any external code besides from standard libraries. Therefore, it can easily be ported to embedded platforms. The usage of the library is intuitive and illustrated with many examples. For example, to calculate the charge-consumption for a connection interval of $T_c = 100\,\text{ms}$, 5 pairs of packets per connection event with 10 bytes received and 20 bytes sent per packet and with 3 dBm of transmission power, the following example can be used.

```
double charge = ble_e_model_c_getChargeConnectionIntervalSamePayload(0,0.1,5,10,20,3);
```

6.3 Design Guidelines

In this section, we present possible uses of our model, leading to guidelines on optimizing the parameter values towards low power consumption.

6.3.1 Advertising/Scanning. Figure 9 shows the expected charges consumed by the advertiser and the scanner for different scan windows d_s. It reveals multiple interesting results. First, there are some beneficial values of $T_{a,0}$, for which the latency becomes lower than for neighboring values with higher energy consumption. In other words, by choosing values of $T_{a,0}$ that lie within local minima, both the latency and the duty-cycle can be minimized. Another interesting outcome is that for small values of $T_{a,0}$, the energy consumption is less sensitive on d_s than for higher values. From these results, we propose the following choices to energy-optimize neighbor discovery procedures:

(1) For devices with short expected idle-scanning durations (i.e., both devices begin the advertising/scanning procedure at approximately the same point in time), we suggest using continuous scanning for achieving the lowest possible discovery latency while still maintaining energy efficiency. If executed continuously, the energy consumption for the scanner will be maximal whereas the advertiser can chose arbitrarily large intervals. Therefore, this mode is also beneficial when the

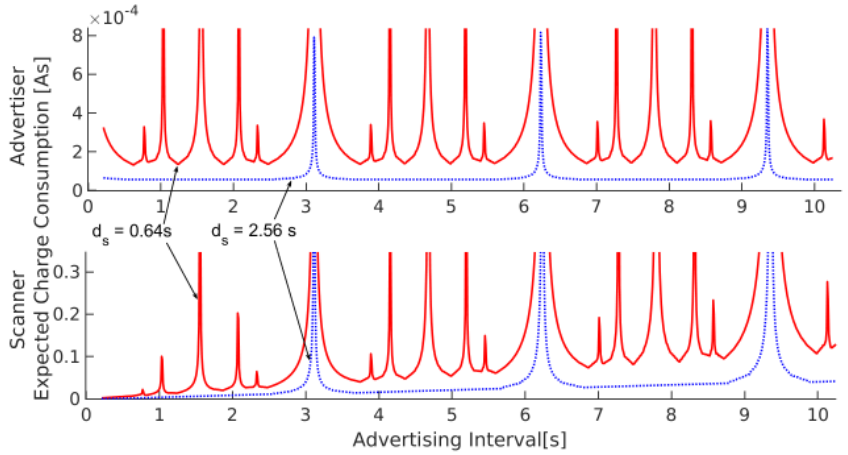


Fig. 9. Expected advertising/scanning charge for $T_s = 3.12\,\text{s}$, $d_a = 446\,\mu\text{s}$, $d_{ch} = 150\,\mu\text{s}$, $\Delta = 93.12\,\mu\text{s}$, $\epsilon = 0.9999$. The charge consumed by each scan event has been assumed to be $i_{rx,S} \cdot d_s$.

scanner has a significantly higher energy budget than the advertiser (e.g., because it is connected to the grid).

(2) For devices that usually spend the vast majority of their time in idle-scanning or idle-advertising, a reasonable trade-off between latency and energy-consumption needs to be achieved. Unlike in the scenario above, where the energy spent for discovery is large against the one for idle-scanning/advertising, here, the energy for idle-scanning/advertising needs to be optimized. We suggest first choosing T_s and d_s such that the desired energy-consumption of the scanner is met. Next, all values of $T_{a,0}$ should be computed using our model, and one which lies close to a local latency minimum and close to the desired energy-consumption of the advertiser needs to be chosen. Third, for further optimizing the discovery procedure, these steps can be repeated for different values of T_s and d_s that lead to the same energy-consumption.

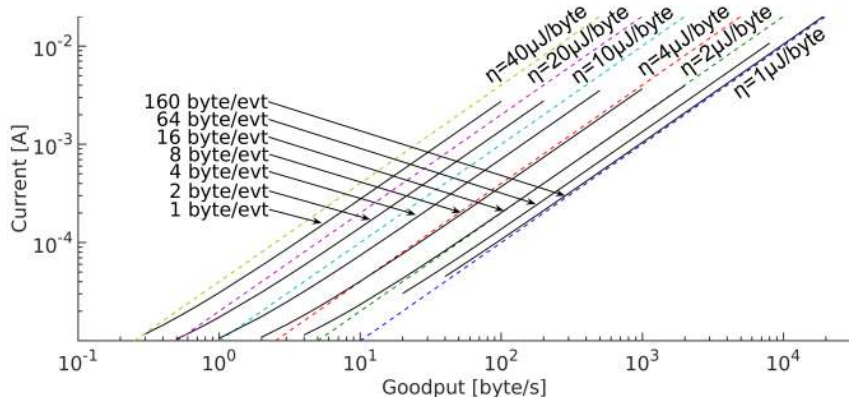


Fig. 10. Current consumption for different payload throughputs (goodputs) using different payload sizes per event for a slave. The protocol overhead is 17 bytes per packet. Payloads that exceed the maximum capacity per packet are distributed into multiple packets.

6.3.2 Connected mode. In the connected mode, a parametrization needs to be chosen that optimizes the energy consumption for given constraints on the minimum throughput and maximum latency.

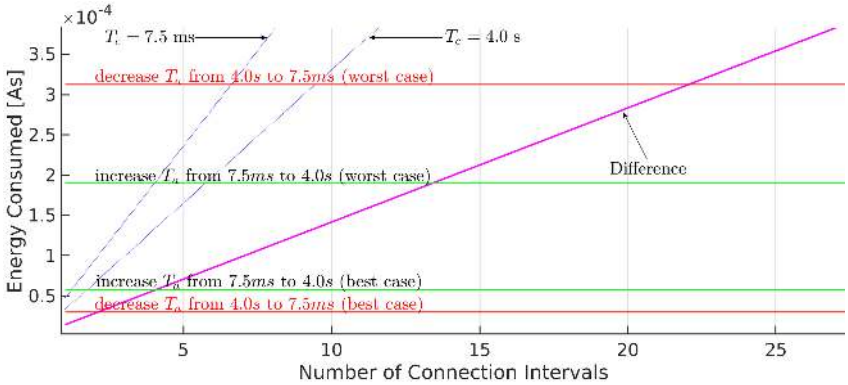


Fig. 11. Energy considerations when increasing or decreasing the connection interval for a slave. The horizontal lines depict the update costs. The bold diagonal line depicts the difference in the consumed charge.

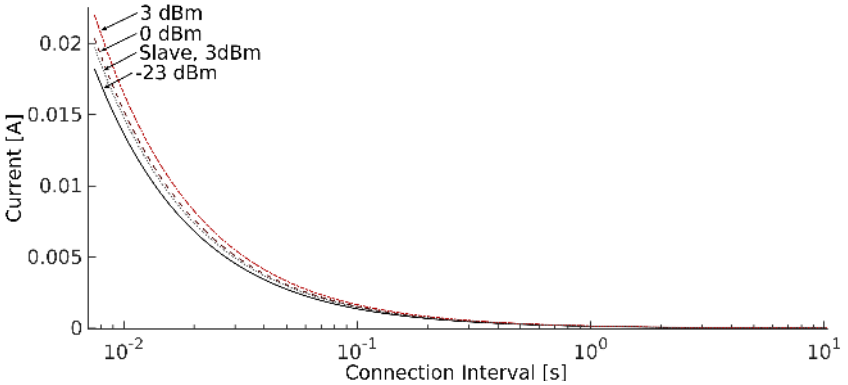


Fig. 12. Current consumption of a master for different connection intervals and *tx*-power levels for 8 packets per event with 20 bytes of payload + 17 bytes overhead in each packet. The line for the slave shows the current consumption of the slave to receive these packets and to respond with an empty packet.

Figure 10 shows the mean current of a BLE device with different amounts of payload bytes per connection event. Every packet contains 17 bytes of protocol overhead and a maximum of 20 bytes of payload. If there are more than 20 bytes of payload in an event, two or more (pairs of) packets can be transmitted within one event. Given a fixed amount of payload (and hence packets) per event, different throughputs of the payload (goodputs) can be achieved by modifying the connection interval. The dotted, diagonal lines depict different levels of energy-efficiency per byte. As can be seen, different numbers of payload per event are associated with different efficiencies $\eta = \frac{bytes}{charge}$. A parametrization drawing a higher current consumption can have a better efficiency per byte of payload than others drawing a lower current. For example, the point with the smallest current and smallest goodput at the lower left of the figure has a far lower efficiency than achieving the same goodput with 16 bytes per event. Another fact that can be observed is that for any of the curves with a fixed payload per event, the increase of the goodput leads to a rise of the efficiency but at the same time to an increased current consumption. To achive a high efficiency η for a given goodput, our suggestions are as follows.

(1) Add as much payload in a packet as possible [29]. In an ATT_HANDLE_NOTIFY-event, which is typically used for transmitting payload via the attribute protocol of BLE, the protocol overhead is 17 bytes, whereas the maximum payload per event is 20 bytes. Not utilizing the whole 20 bytes of payload leads to a larger overhead per byte of payload.

(2) Maximize the number of (filled) packets per event, [29] while increasing the connection interval to the maximum value that still meets the required throughput- and latency constraints.

The efficiency per byte is important in scenarios in which a certain, large amount of payload needs to be transmitted in a burst. Below, we consider a scenario in which the required data rate varies over time, e.g., for a sensor that opportunistically transmits changes in the sensed data.

(3) When idle, the connection interval T_c should be as large as possible to avoid empty polling packets. Figure 12 shows the power consumption for transmitting eight 20-byte-payload-packets per event for different values of T_c. Clearly, the energy consumption is highly sensitive on T_c.

(4) In some cases, it is beneficial to update the connection interval. Due to the necessary idle-listening in the course of the resynchronization procedure described in Section 4.2.3, such updates involve a significant energy overhead, especially for the slave. Updates should only be performed if the expected savings are higher than the update overheads. The two dotted lines in Figure 11 depict the charges consumed over time by a slave for receiving 20 bytes per connection interval from the master, with $T_c = 0.0075$ s and $T_c = 4.0$ s. The bold line in this figure represents the difference in the charge consumption between the two connection intervals and is therefore equal to the savings per time if one would change T_c from 7.5 ms to 4.0 s. The horizontal lines depict the cost in terms of charge for updating $T_c = 7.5$ ms to $T_c = 4.0$ s and vice versa. As described in Section 4.2.3, the master might chose different values for d_{tw} and d_{two}, which lead to different energy consumptions. Therefore, both the charge for updating T_c in the best- and worst-case are depicted. For example, in the worst-case, an update from $T_c = 7.5$ ms to 4.0 s would pay of after 23 connection intervals.

(5) If the variations in the required throughput cannot be predicted with sufficient accuracies, the slave latency feature of BLE should be used instead of an update of T_c. In general, there is a trade-off: Updating T_c yields higher savings than skipping events using slave latency (since also the master can save energy), but using slave latency comes with no additional overhead. In contrast, updating T_c is energy-expensive.

(6) Figure 12 reveals that energy can be saved if the *tx*-power-level is adjusted to a value that is sufficient to transmit the data across the distance between sender and receiver without any losses.

7 CONCLUDING REMARKS

In this paper, a comprehensive energy model of the BLE protocol was presented. The model parameter values for BLE112-devices were given. Modeled results have been compared to measured ones for the connected mode and for the neighbor discovery procedure. In a test-scenario, an error of the predicted current of no more than 3.4 % was measured for the connected mode. The proposed model can be used for battery lifetime calculations, for predictions on worst case peak currents and for deriving power management strategies. With the results presented, power-management-algorithms for updating the connection parameters adaptively to the current situation can be developed. Further research on an error estimation for the parameters Δ and ϵ of Algorithm 1 seems to be desirable for reducing the runtime of the algorithm. An open research topic is the reception probability in continuous broadcasting scenarios, where data is exchanged using advertising/scanning without ever establishing a connection.

REFERENCES

[1] I. Agadakos, J. Polakis, and G. Portokalidis. 2017. Techu: Open and Privacy-Preserving Crowdsourced GPS for the Masses. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*.

[2] D. Auguste. 2015. *Model for Predicting Bluetooth Low Energy Micro-Location Beacon Coin Cell Battery Lifetime.* Master's thesis. Regis University, Dayton Memorial Library.

[3] Bluegiga. 2011. BLE112 Data Sheet. (2011). Version 1.21.

[4] K. Cho, G. Park, W. Cho, J. Seo, and K. Han. 2016. Performance analysis of device discovery of Bluetooth Low Energy (BLE) networks. *Computer Communications* 81 (2016), 72 – 85.

[5] K. Cho, W. Park, M. Hong, G. Park, W. Cho, J. Seo, and K. Han. 2015. Analysis of Latency Performance of Bluetooth Low Energy (BLE) Networks. *Sensors* 15, 1 (2015), 59–78.

[6] A. Del Campo, L. Cintioni, S. Spinsante, and E. Gambi. 2017. Analysis and Tools for Improved Management of Connectionless and Connection-Oriented BLE Devices Coexistence. *Sensors* 17, 4 (2017), 792.

[7] M.C. Ekstrom, M. Bergblomma, M. Linden, M. Bjorkman, and M. Ekstrom. 2012. A Bluetooth Radio Energy Consumption Model for Low-Duty-Cycle Applications. *IEEE Transactions on Instrumentation and Measurement (TIM)* 61, 3 (2012), 609–617.

[8] C. Gomez, I. Demirkol, and J. Paradells. 2011. Modeling the Maximum Throughput of Bluetooth Low Energy in an Error-Prone Link. *IEEE Communications Letters* 15, 11 (2011), 1187–1189.

[9] C. Gomez, J. Oller, and J. Paradells. 2012. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors* 12, 9 (2012), 11734–11753.

[10] W. S. Jeon, M. H. Dwijaksara, and D. G. Jeong. 2017. Performance Analysis of Neighbor Discovery Process in Bluetooth Low-Energy Networks. *IEEE Transactions on Vehicular Technology* 66, 2 (2017), 1865–1871.

[11] S. Kamath and J. Lindh. 2012. Application Note AN092: Measuring Bluetooth Low Energy Power Consumption. (2012). Revision SWRA347a, URL: http://www.ti.com/lit/an/swra347a/swra347a.pdf.

[12] P.H. Kindt, M. Saur, and S. Chakraborty. 2018. Neighbor discovery latency in BLE-like protocols. *IEEE Transactions on Mobile Computing (TMC)* 17, 3 (2018), 617–631.

[13] P. Kindt, D. Yunge, R. Diemer, and S. Chakraborty. 2014. C-implementation of the BLE energy model. http://www.rcs.ei.tum.de/en/research/wireless-sensor-networks/bleemod. (2014).

[14] P. Kindt, D. Yunge, M. Gopp, and S. Chakraborty. 2015. Adaptive Online Power-Management for Bluetooth Low Energy. In *IEEE International Conference on Computer Communications (INFOCOM)*.

[15] P.H. Kindt, D. Yunge, Diemer R., and S. Chakraborty. 2014. Precise Energy Modeling for the Bluetooth Low Energy Protocol. *CoRR* abs/1403.2919 (2014).

[16] A. Liendo, D. Morche, R. Guizzetti, and F. Rousseau. 2018. BLE Parameter Optimization for IoT Applications. In *IEEE International Conference on Communications (ICC)*. Kansas City, MO, United States.

[17] A. Liendo, D. Morche, R. Guizzetti, and F. Rousseau. 2018. Efficient Bluetooth Low Energy Operation for Low Duty Cycle Applications. In *IEEE International Conference on Communications (ICC)*.

[18] J. Liu and C. Chen. 2012. *Energy Analysis of Neighbor Discovery in Bluetooth Low Energy Networks*. Technical Report. URL: research.nokia.com/files/public/TR-EnergyAnalysis-BLE.pdf.

[19] J. Liu, C. Chen, and Y. Ma. 2012. Modeling and performance analysis of device discovery in Bluetooth Low Energy networks. In *2012 IEEE Global Communications Conference (GLOBECOM)*.

[20] J. Liu, C. Chen, and Y. Ma. 2012. Modeling Neighbor Discovery in Bluetooth Low Energy Networks. *IEEE Communications Letters* 16, 9 (2012), 1439–1441.

[21] E. Mackensen, M. Lai, and T.M. Wendt. 2012. Bluetooth Low Energy (BLE) based wireless sensors. In *IEEE Sensors*. 1–4.

[22] E. Mackensen, M. Lai, and T.M. Wendt. 2012. Performance analysis of an Bluetooth Low Energy sensor system. In *IEEE International Symposium on Wireless Systems (IDAACS-SWS)*.

[23] Bluetooth SIG. 2010. Specification of the Bluetooth System 4.0. (June 2010). Volume 0, available via bluetooth.org.

[24] Texas Instruments. 2010. SmartRF Packet Sniffer User Manual. (2010). Revision F, URL: www.ti.com/lit/pdf/swru187.

[25] Texas Instruments. 2014. CC2540F128, CC2540F256: 2.4-GHz Bluetooth Low Energy System-on-Chip. (2014). SWRS084E, Version revised in june 2013, URL: http://www.ti.com/lit/gpn/cc2540.

[26] F. Samie, S. Paul, L. Bauer, and J. Henkel. 2018. Highly efficient and accurate seizure prediction on constrained IoT devices. In *Design, Automation Test in Europe Conference Exhibition (DATE)*.

[27] R. Schrader, T. Ax, C. Röhrig, and C. Fühner. 2016. Advertising power consumption of Bluetooth Low Energy systems. In *International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS)*.

[28] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. 2002. Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions on Mobile Computing (TMC)* 1, 1 (2002), 70–80.

[29] M. Siekkinen, M. Hiienkari, J.K. Nurminen, and J. Nieminen. 2012. How low energy is Bluetooth Low Energy? Comparative measurements with ZigBee/802.15.4. In *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*.

[30] R. Yates and D. J. Goodman. 2005. *Probability and Stochastic Processes*. John Wiley & Sons, Inc. Second Edition.

A PARAMETER VALUES FOR BLE112 IN CONNECTED MODE

In the two tables of this section, we present values for all model parameters of the connected mode for a BLE112-device. The slave clock latency is given by a certain factor, using which a certain amount of wall clock time needs to be multiplied by. For the BLE112-device, it is reported to lie between $31 \cdot 10^{-6}$ and $50 \cdot 10^{-6}$ during the handshaking for the connection establishment process. In this paper, we therefore assume a sleep clock accuracy of $50 \cdot 10^{-6}$. The duration d_{prerx} is longer than its table value for the first *rx*-phase within a connection event of a slave. For the latter case, d_{prerx} is 388 μs. The values presented have been measured by analyzing about 4000 connection events of a BLE-attribute write procedure without confirmation of the remote node for different payload lengths and different numbers of events per packet. The connection interval T_c was 100 ms, the *tx*-power was 3 dBm. We also performed the measurement for different *tx*-power levels. These *tx*-power levels mainly result in different currents for the *tx*-section. Other parameters such as Q_{cso} and d_{tra} are influenced too, but the impact on their values and hence on the event energy consumption is much lower than the impact of the *tx*-current. For a good approximation, we assume all parameters except the *tx*-current to be constant for different tx-power levels. Further, we assume a sleep current I_{sl} of 0.9 μA, as described in [3].

Phase	d_{avg} [ms]	d_{min}, [ms]	d_{max} [ms]	d_{std} [ms]	I_{avg} [mA]	I_{min} [mA]	I_{max} [mA]	I_{std} [mA]
head	0.578	0.500	0.640	0.012	5.924	5.558	6.165	0.085
pre	0.305	0.010	0.450	0.109	7.691	5.570	7.997	0.153
rx	-	-	-	-	26.505	25.967	27.676	0.302
rxtx	0.080	0.060	0.100	0.004	14.128	13.793	14.653	0.115
tx	-	-	-	-	36.445	35.571	38.763	0.559
pretx	0.053	0.014	0.084	0.018	-	-	-	-
txrx	0.057	0.040	0.070	0.005	15.125	14.605	16.048	0.198
cpre	0.073	0.050	0.080	0.004	12.238	11.633	13.006	0.200
prerx	0.123	0.110	0.140	0.005	-	-	-	-
tra	0.066	0.040	0.090	0.011	11.636	8.964	14.721	1.416
post	0.860	0.610	1.110	0.101	7.980	7.919	8.221	0.065
tail	0.080	0.060	0.340	0.013	4.129	3.088	6.995	0.380
Phase	Q_{avg} [uC]	Q_{min} [uC]	Q_{max} [uC]	Q_{std} [uC]	-	-	-	-
to	-1.2	-1.8	-0.8	0.2	-	-	-	-

Table 3. Model parameters for the connected mode with a tx-power of 3 dBm for a BLE112 transceiver. Average (X_{avg}), minimal(X_{min}) and maximal (X_{max}) values are given along with the standard deviation X_{std}.

tx-power	$I_{tx}[mA]$
3 dBm	36.5
2 dBm	33.5
0 dBm	32.1
−1 dBm	31.5
−2 dBm	30.6
−3 dBm	30.1
−5 dBm	29.1
−6 dBm	28.8
−8 dBm	28.4
−10 dBm	28.1
−12 dBm	27.9
−15 dBm	27.7
−17 dBm	27.6
−19 dBm	27.5
−21 dBm	27.5
−23 dBm	26.3

Table 4. Current consumption in the transmission-phase for all supported tx-power-levels of a BLE112-transceiverver.

B PARAMETER VALUES FOR BLE112 IN SCANNING MODE

The table below presents values for all model parameters for the scanning mode. They were obtained by analyzing 1257 scan events for a tx-power of 3 dBm.

Part	d_{avg} [ms]	d_{min}, [ms]	d_{max} [ms]	d_{std} [μs]	I_{avg} [mA]	I_{min} [mA]	I_{max} [mA]	I_{std} [mA]
pre,s	0.700	0.680	0.730	10	7.087	6.924	7.253	0.065
rx,s	-	-	-	-	26.399	26.042	26.480	0.043
rxtx,s	0.115	0.110	0.120	0.498	15.011	14.617	15.519	0.288
tx,s	-	-	-	-	35.999	35.650	36.488	0.247
pretx,s	0.014	4e-3	0.024	1.184	-	-	-	-
txrx,s	0.089	0.080	0.090	2.332	16.670	15.875	17.224	0.244
rxsr	-	-	-	-	26.426	26.279	26.563	0.058
prerx,s	0.074	0.068	0.088	2.45	-	-	-	-
rxrx,s	0.377	0.370	0.380	4.488	9.633	9.426	9.768	0.11
post,s	0.816	0.710	1.820	246	8.012	7.820	8.138	0.060
chch,s	1.325	1.320	1.330	4.983	8.550	8.470	8.624	0.042
Part	Q_{avg} [uC]	Q_{min} [uC]	Q_{max} [uC]	Q_{std} [uC]	-	-	-	-
ctx,s	-0.2264	-0.3244	-0.1456	0.0143	-	-	-	-
crx,s	-0.1350	-0.1900	-0.0851	0.0123	-	-	-	-

Table 5. Model parameter values for scan events. Average (X_{avg}), minimal(X_{min}) and maximal (X_{max}) values are given along with the standard deviation X_{std}.

C TABLE OF SYMBOLS

The symbol X stands either for the charge Q, the current I or the duration d of a phase of a connection event.

BLE Protocol		
Symbol	**Unit**	**Description**
T_a	[s]	Advertising interval
T_s	[s]	Scan interval
d_s	[s]	Scan window
d_{IFS}	[s]	Interframe-space
$\rho(t)$	[s]	Random advertising delay
d_{tw}	[s]	Transmit window
d_{two}	[s]	Transmit window offset
t_{anchor}	[s]	Anchor point: Reference point for connection interval
T_c	[s]	Connection interval
$T_{c,o}$	[s]	Connection interval before a connection parameter update
$T_{c,n}$	[s]	Connection interval after a connection parameter update
d_c	[s]	Duration of a connection event
N_{sl}	-	Slave latency
$\overline{d_{adv}}$	[s]	Expected discovery latency in the advertising/scanning mode
Event Model for the Connected Mode		
Symbol	**Unit**	**Description**
X_{head}	[C/A/s]	Device wakeup phase
X_{pre}	[C/A/s]	Preprocessing phase
X_{cpre}	[C/A/s]	Communication preamble phase
X_{ww}	[C/A/s]	Window widening phase
X_{prerx}	[C/A/s]	Constant part of the reception phase
X_{rx}	[C/A/s]	Reception phase
N_{rx}	-	Number of bytes received
X_{IFS}	[C/A/s]	Interframe space
X_{rxtx}	[C/A/s]	Reception-to-transmission transition phase
X_{txrx}	[C/A/s]	Transmission-to-reception transition phase
X_{pretx}	[C/A/s]	Constant part of the transmission phase
X_{tx}	[C/A/s]	Transmission phase
N_{tx}	-	Number of bytes sent
X_{tra}	[C/A/s]	Transient phase
X_{post}	[C/A/s]	Postprocessing phase
X_{tail}	[C/A/s]	Tail phase (i.e., phase that initiates the sleep mode)
d_{seq}	[s]	Duration of a communications sequence (i.e., exchange of one pair of packets)
d_{event}	[s]	Duration of a whole connection event (from head to tail)
X_t	[C/A/s]	Communication sequence (reception, transmission, interframe-spaces and correction term for capacitances/resistances)

Q_{to}	[C]	Correction term for capacitances/resistances in power supply line
$SCA_{Ma/Sl}$	[ppm]	Sleep clock accuracy of master/slave
I_{sl}	[A]	Sleep current of the BLE device

Event Model for Scan Events		
Symbol	**Unit**	**Description**
X_{pre}	[C/A/s]	Wakeup and preprocessing phase
$X_{S,1}$	[C/A/s]	Phase of scanning until the first reception takes place
$X_{rx,s}$	[C/A/s]	Reception phase (advertising packet is received)
$X_{rxtx,s}$	[C/A/s]	Reception-to-transmission transition phase
$X_{tx,s}$	[C/A/s]	Transmission of scan-request (or connection request) phase
$X_{txrx,s}$	[C/A/s]	Transmission-to-reception transition phase
X_{rxs}	[C/A/s]	Phase of the reception of a scan response
$X_{rxrx,s}$	[C/A/s]	Reception-to-scanning transition phase
$X_{S,2}$	[C/A/s]	Phase of scanning until the end of the scan event
$X_{post,s}$	[C/A/s]	Postprocessing phase
I_{scan}	[A]	Current consumption of the device while scanning
Q_{sEv}	[C]	Charge consumed by a scan event (general case)
$Q_{sEv,Idle}$	[C]	Charge consumed by a scan event (idle scanning)
$Q_{crx,s}$	[C]	Correction term for non-rectangular shapes of the reception-phase(s)
$Q_{ctx,s}$	[C]	Correction term for non-rectangular shapes of the transmission-phase(s)

Connection Procedure Model		
Symbol	**Unit**	**Description**
$Q_{ev,cR,Ma/Sl}$	[C]	Charge consumed by the master/slave for an event a connection request event is sent within
$Q_{ev,cU,Ma/Sl}$	[C]	Charge consumed by the master/slave for an event a connection update packet is sent within
$d_{sl,cR/cU}$	[s]	Sleep duration after a connection request/update event
$Q_{cE,Ma/Sl}$	[C]	Charge consumed for a connection establishment by the master/slave
$Q_{cU,Ma/Sl}$	[C]	Charge consumed for a connection parameter update by master/slave
d_p	[s]	Time from the beginning of the transmit window until the master sends the first packet using the new T_c
$d_{ww,cE/cU}$	[s]	Window widening for connection establishments/parameter updates

Discovery Latency Model		
Symbol	**Unit**	**Description**
d_{advPkg}	[s]	Duration of an advertising packet
ϕ	[s]	Offset between the beginning of the first scan event and the first advertising event
$T_{a,0}$	[s]	Constant part of the advertising interval

ρ	[s]	Random delay for advertising events
$t_{i,suc}$	[s]	Set of points in time an advertising packet which is sent within is received successfully
d_{early}	[s]	Time an advertising event can be sent before the beginning of the corresponding scan event to be received successfully
d_{late}	[s]	Time an advertising event must be sent before the end of the corresponding scan event to be received successfully
d'_s	[s]	Effective scan window ($d'_s = d_s - d_a$)
d_a	[s]	Duration of an advertising packet
d_{ch}	[s]	Time the advertiser needs for changing the channel it sends its packets on
t_{sE}	[s]	Time a scan event starts at
t_{advEvt}	[s]	Time an advertising event is sent at
$p_{hit}(n)$	-	Probability of an advertising event n for hitting a scan event
$f(\rho)$	-	Probability density function for the random advertising delay
μ	-	Expected value
σ	-	Standard deviation
$f(t_{advEvt})$	-	Probability density function for the beginning of an advertising event
p_k	-	Probability for an advertising event being received in the scan event k
$\Phi(t)$	-	Standard normal cumulative distribution
t_{ai}	[s]	Point in time an advertising event would take place without taking the random delay ρ into account
$k_{min}(n)$, $k_{max}(n)$	-	Lowest and highest index of a scan event the advertising event n could overlap with
$p_{cM}(n)$	-	Cumulative miss probability (probability that n advertising events in a row miss the scan event)
d_{exp}	[s]	Expected discovery latency for a given pair of values Φ and k
ϵ	-	Parameter of Algorithm 1 that determines the accuracy and computational complexity
ch	-	Channel number (37/38/39)
$d_{exp,max}$	[s]	Maximum expected discovery latency for a given pair of values ϕ and k before the calculation is aborted
Δ	[s]	Interval between two offsets ϕ that are examined
$\overline{d_{adv}}$	[s]	Expected discovery latency

Estimating the Number of Events		
Symbol	**Unit**	**Description**
$N_{c,ma/sl}$	-	Number of connection events of the master/slave within a given amount of time

$\overline{Q_{con}}$	[C]	Overall charge consumed by the master/slave in the connected mode
T_g	[s]	Amount of time under consideration
$\overline{N_{sl}}$	-	Average slave latency
X_{event}	[C/A/s]	Charge/current/duration of an arbitrary event
X_{full}	[C/A/s]	Full advertising event
X_{last}	[C/A/s]	Last advertising event
$\overline{Q_{adv}}$	[C]	Expected charge consumed by an advertiser (whole discovery process)
$Q_{advEvent}$	[C]	Charge consumed by an advertising event
$X_{37(38/39)}$	[C/A/s]	Advertising event on channel 37/38/39
$\overline{N_a}$	-	Expected number of advertising events
$\overline{Q_s}$	[C]	Expected energy of scanner (whole discovery process)
$\overline{Q_{active}}$	[C]	Expected charge of scanner for all times it is not sleeping
$\overline{Q_{sleep}}$	[C]	Expected charge of scanner for all times it is sleeping
$Q_{sEv,idle}$	[C]	Charge consumed by an idle scan event (i.e., a scan event without reception)
$\overline{N_a}, \overline{N_s}$	[C]	Expected number of advertising/scan events during neighbor discovery

Model Validation and Sensitivity Analysis		
Symbol	**Unit**	**Description**
ε_m	-	Relative error between measured and modeled values
S	var.	Sensitivity
Q_{ph}	[C]	Charge consumed by a phase/state the device is in (e.g., preprocessing, reception,..)
d_{min}/d_{max}	[s]	Minimum/maximum phase duration
I_{min}/I_{max}	[A]	Minimum/maximum current of a phase
ΔQ_{total}	[C]	Variation of the total charge consumed in a given phase
η	[Bytes/C]	Efficiency