Paper published at the 4th Intl. Workshop on Experiences with the Design and Implementation of Smart Objects (SMARTOBJECTS). Los Angeles, CA, USA. June 2018.

Improving the Efficiency and Responsiveness of Smart Objects Using Adaptive BLE Device Discovery

Tobias Renzler
Institute of Technical Informatics
Graz University of Technology, Austria
t.renzler@student.tugraz.at

Michael Spörk
Institute of Technical Informatics
Graz University of Technology, Austria
michael.spoerk@tugraz.at

Carlo Alberto Boano
Institute of Technical Informatics
Graz University of Technology, Austria
cboano@tugraz.at

Kay Römer
Institute of Technical Informatics
Graz University of Technology, Austria
roemer@tugraz.at

ABSTRACT

The ability of fine-tuning the performance of Bluetooth Low Energy (BLE) communication is essential to create low-power wireless applications with heavy user interaction, such as smart thermostats or door locks. One of the key challenges when designing such applications is finding the right trade-off between a system's responsiveness and energy-efficiency. Although there exists research works that improve the performance of BLE communication, all these approaches focus on connection-based BLE. Most BLE-based applications, however, spend the majority of their time in connection-less device discovery, waiting for approaching users. The energy-efficiency and timeliness in this state are defined by parameters that are often statically set at compile time. Although supported by the BLE specifications, how to dynamically adapt these parameters to user behavior is still an open question. In this paper, we tackle this challenge and design a strategy to improve the energy-efficiency and responsiveness of BLE device discovery. Towards this goal, we model the device discovery process and identify its key parameters. We further design an adaptive advertising strategy that allows smart objects to adapt their device discovery parameters to the user behavior. We implement this adaptive strategy and measure its performance in a real-world application, the Nuki Smart Door Lock. Our experiments show that a smart lock using our strategy consumes 48% less energy while reducing the device discovery time by up to 63% compared to the use of static parameters. Furthermore, we discuss how nearby BLE devices can be used to inform the lock about approaching user devices and hence to improve its responsiveness in low-power phases even further.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Dependable and fault-tolerant systems and networks*; • **Networks** → *Network protocols*; • **General and reference** → Experimentation;

SMARTOBJECTS'18, June 25, 2018, Los Angeles, CA, USA

ACM ISBN 978-1-4503-5857-6/18/06…$15.00
https://doi.org/10.1145/3213299.3213306

KEYWORDS

Adaptive advertising, BLE, Device discovery, Energy efficiency, Parameter adaptation, Responsiveness, Smart lock.

ACM Reference Format:
Tobias Renzler, Michael Spörk, Carlo Alberto Boano, and Kay Römer. 2018. Improving the Efficiency and Responsiveness of Smart Objects Using Adaptive BLE Device Discovery. In *SMARTOBJECTS'18: 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects, June 25, 2018, Los Angeles, CA, USA*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3213299.3213306

1 INTRODUCTION

Since its release in 2010, Bluetooth Low Energy (BLE) has become an increasingly popular wireless technology to connect smart objects to the Internet of Things (IoT). Besides its low power consumption and reliable communication, especially its wide adoption in consumer electronic devices (such as smartphones and wearables) makes BLE the wireless technology of choice to build IoT applications requiring heavy interaction with the end-users. Smartphones, for example, are often used to interact with BLE-based smart objects such as health and fitness monitors [8, 28], pet trackers [27], smart thermostats [25], and smart door locks [12].

BLE efficiency vs. responsiveness. This increasing popularity of Bluetooth Low Energy has led to a large body of research works aiming to improve its communication performance, especially when it comes to minimizing energy consumption and end-to-end delays [16, 24, 29]. One of the key challenges in designing BLE-based IoT applications is, indeed, the omnipresent trade-off between system responsiveness and energy-efficiency. This trade-off makes protocol parametrization a *catch-22 dilemma*, whereby longer radio-on times are required to increase the responsiveness of the system, but extended radio-off times should be used to preserve the typically limited energy budget of battery-powered devices. For instance, some BLE-based applications such as health monitors [28] may need to minimize energy consumption for economic viability, while still ensuring a timely delivery of alarm messages as soon as the vital signs of the user start to deteriorate. In this regard, the BLE standard offers the possibility to fine-tune several communication parameters in order to design a system that is more energy-efficient or responsive depending on the application's needs.

BLE parameter adaptation. Spörk et al. have shown how to use connection parameters as tuning knobs to adapt BLE's connection-based communication to changes in the observed traffic load at runtime [29]. Mikhaylov has shown how to adjust the connection interval to increase the responsiveness of a system based on BLE during data transfer [24]. These and other BLE parameter adaptation approaches [19, 20] can significantly increase the performance of BLE systems, but only focus on adapting the communication parameters of an *already-established BLE connection*. Most BLE-based applications, however, do not spend a significant portion of time with an established connection. Instead, in several IoT applications, the smart objects remain mostly inactive and only sporadically need to timely react to user input. For example, a smart door lock only interacts with the user when the latter approaches the front door and wants to lock/unlock it. Similarly, a smart thermostat does not need to interact with a user when the latter is outside the house. In this class of IoT applications, the BLE smart objects spend most of their time performing *device discovery*, i.e., they either scan for other BLE devices in range, or advertise their presence so that other devices can establish a connection.

Need for adaptive BLE device discovery. How often a BLE device advertises its presence or scans for other devices in range is defined by a set of *device discovery parameters*. The latter are typically chosen during the development phase and *statically* set at compile time [21, 22]. However, static parameters are often not suitable for BLE-based systems that need to interact with an end-user. Most applications have indeed phases in which the smart objects need to be as responsive as possible (e.g., when users want to interact) and other phases where they can focus on conserving energy in order to preserve the capacity of batteries (e.g., when no user is nearby). The number and duration of these phases essentially depend on *user behavior*. A developer designing a smart lock could, for example, statically select the device discovery parameters of a smart door lock to be responsive in the early morning and late afternoon, when most people typically leave to work and arrive back home, respectively, and to privilege energy-efficiency in the other times of the day. This may work well in a typical family household, but not in a single household where a user works mostly at night.

This dependency on user behavior makes it hard for developers to select suitable static parameters for the BLE device discovery process that result in a satisfactory usability of the system [15]. Hence, there is a need to efficiently adapt the device discovery parameters at run-time based on the user behavior. Despite the BLE specifications [2] actually support a run-time adaptation of device discovery parameters, no prior work has – to the best of our knowledge – yet studied how to efficiently adapt these parameters to increase the performance of the device discovery process, nor proposed a practical implementation on a real-world IoT application.

Contributions. In this paper, we focus on a BLE-based smart door lock and design strategies to significantly improve its energy-efficiency and responsiveness. After reviewing related work (Sect. 2), we first model the BLE device discovery process and gain a deep understanding of which device discovery parameters significantly influence the energy consumption and responsiveness of a BLE-based IoT system (Sect. 3).

Using the derived model, we design and implement an *adaptive advertising* strategy allowing a generic smart object to learn from past activities, and adapt the BLE device discovery parameters at run-time based on the observed user behavior (Sect. 4). In particular, our adaptive advertising strategy learns the user habits and tries to automatically find a compromise between the energy consumption and responsiveness of the device discovery process.

We implement the proposed adaptive strategy on a Cypress CY8C4248LQI-BL483 platform [6] – the same one used by the *Nuki Smart Lock* [12] manufactured by Nuki Home Solutions (Sect. 5). We then evaluate the performance of our adaptive advertising strategy and show that it can successfully adapt the BLE device discovery parameters on the smart lock depending on the user behavior, with a decrease in energy consumption up to 48% and a reduction in the average device discovery time by up to 63% compared to the use of static BLE communication parameters (Sect. 6).

To improve the responsiveness of the smart object when it is using BLE device discovery parameters that privilege energy efficiency to responsiveness, we propose the use of a BLE *range extender* (Sect. 7). The latter is a second BLE device seamlessly informing a smart object about other BLE devices approaching its communication range. We show how the range extender can complement the proposed adaptive advertising strategy to increase the responsiveness and efficiency of a smart object even further.

We finally conclude the paper with a summary of our contributions and an outlook on future work (Sect. 8).

2 RELATED WORK

A large body of works has focused on BLE technology, especially on the provision of services such as neighbor discovery [16], indoor localization [7], group management [11], and locality-based authorization [10], as well as on the design of new platforms [3].

BLE modeling. A few works have modeled the energy-efficiency and performance of BLE communication. Liu et al. [22] present an analytical model showing the influence of the advertising interval, scanning interval, and scanning window on the energy consumption of BLE devices during device discovery. Similarily, Kamath et al. [17] accurately measure the energy consumption of a BLE device that exchanges data packets over a BLE connection. Jeon et al. [15] model the performance of device discovery in BLE and show the trade-off between discovery latency and energy consumption. They conclude that the behavior of the device discovery depends on the BLE parameters of the advertising and scanning device. Cho et al. [4] show that, by increasing the number of BLE devices that perform device discovery, the discovery latency of the devices increases exponentially. The authors highlight that this contention is mainly caused by parameters selected inappropriately. Likewise, Julien et al. [16] present a protocol that calculates suitable BLE communication parameters based on application metrics that minimize collisions in the BLE device discovery. To evaluate the suitability of parameters, Kindt et al. [18] have proposed an algorithm to compute the discovery latency of BLE and ANT depending on their selected parameters.

Unlike these works, in this paper we accurately model the energy consumption of an advertiser as a function of the selected advertising interval and the used advertising payload. Furthermore, we

outline the influence of the advertising and scanning parameters on the device discovery latency and show how to choose these parameters appropriately to minimize device discovery time.

BLE parameter adaptation. Based on their analysis of BLE device discovery process, Liu et al. [21] identify that standardized BLE parameters may be highly inefficient and propose to adaptively reduce the advertising interval when encountering a long discovery latency. Once device discovery has been successfully performed, the mechanism proposed by Mikhaylov [24] may be used to optimize the connection establishment process of BLE by adapting the connection interval. Lee et al. [20] perform experiments showing the influence of the connection interval on the packet delivery rate of BLE connections . Likewise, Gomez et al. [13] evaluate the performance of BLE connections and mention that the connection interval and slave latency parameters could be adapted at runtime to realize high-level application metrics. Spörk et al. [29] monitor the traffic load of a BLE connection in the BLE slave and adapt the connection parameters accordingly.

In contrast to these approaches, we adapt the advertising interval depending on user behavior and, therefore, optimize the device discovery process for each individual system.

3 BLE DEVICE DISCOVERY

To find BLE device discovery parameters that select a suitable trade-off between energy-efficiency and responsiveness, we investigate the BLE device discovery process. We first describe the latter in detail and identify the key parameters that influence its energy consumption and responsiveness in Sect. 3.1. Next, we discuss how to select suitable parameters on a BLE scanner and derive a lightweight mathematical model that allows us to estimate the energy consumption of a BLE advertiser as a function of the used advertising parameters in Sect. 3.2.

3.1 BLE Primer

BLE supports two modes of communication: a connection-less and a connection-based mode. When using the connection-based communication mode, two BLE devices bidirectionally exchange data over an established BLE connection. In this mode, the two devices use one of 37 BLE data channels selected by the Adaptive Frequency Hopping (AFH) algorithm. To use the connection-based mode, however, two BLE devices need to perform device discovery first to synchronize and establish a connection. During the device discovery process, a BLE device is either an *advertiser* that periodically broadcasts its BLE capabilities or a *scanner* that is listening to nearby BLE devices and initiates a connection if necessary.

Advertising. A BLE advertiser periodically broadcasts short data packets during so called *advertising events*, as shown in Fig. 1. These advertising events are non-overlapping time-slots that are equally spaced-out over time. The time between the start of two consecutive advertising events is defined by the *advertising interval* t_{ADV_INT}, which can be configured to a value between 20ms and 10.24 seconds. To avoid persisting collisions with other BLE devices, a random delay t_{DELAY} between 0 and 10ms is added at the end of each advertising event. During each advertising event, the advertiser transmits its advertising payload on up to three advertising channels, with

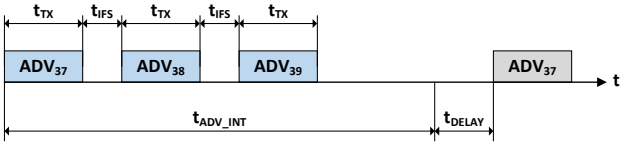

Figure 1: A BLE advertising event on channels 37, 38, and 39.

a mandatory Inter Frame Spacing (t_{IFS}) of 150μs [2]. These advertising channels (BLE channels 37, 38, and 39) are dedicated BLE radio channels that are only used for device discovery and data broadcasting and were choosen to have minimum overlap with other wireless technologies in the 2.4 GHz band, such as Wi-Fi.

The time spent on each of the advertising channels t_{TX} is determined by the advertising payload length. After sending the payload on a channel, the BLE device may receive a scan or connection request from a scanning device. A scan request is used to request further broadcast information, while a connection request is sent to establish a BLE connection. At the end of an advertising interval, the BLE radio is powered-off until the start of the next advertising event in order to conserve energy.

Scanning. A BLE scanner periodically listens for nearby advertisers to scan for further information or to initiate a connection. The timing of BLE scanning is defined by two parameters: the scan interval and the scan window, as shows in Fig. 2. The *scan interval* t_{SCAN_INT} defines the time between the start of two consecutive scanning phases; the *scan window* t_{SCAN_WIN} determines the time the radio scans during a scanning phase. When the scan window is equal to the scan interval, the scanner is never turned off and the device performs *continuous scanning*. During such a scanning phase, the device is only able to listen on one of the three advertising channels, using a different advertising channel for each event. Once a nearby advertiser broadcasts on the same channel on which the scanner is currently listening, an advertising packet can be successfully received, as shown in Fig. 2.

If the advertising packet has been successfully received, the scanner is aware of the advertiser and the device discovery process was successful. In this case, the scanner may choose one of three options: (i) continue listening, (ii) sending a scan request, or (iii) initiating a connection. By continuing to listen, the scanner is performing so called *passive scanning* and is passively observing its environment without disclosing its presence. In case the scanner wants to request more information from the advertiser, it performs *active scanning* by sending a scan request. After receiving a scan request (SCAN_REQ), an advertiser sends additional broadcast information using a scan response (SCAN_RSP), as shown in Fig. 2. To initiate a connection, the scanner sends a connection request carrying all the necessary connection parameters, such as the connection interval and the data channel map. After a connection has been successfully established, both advertiser and scanner stop device discovery operations and use the BLE connection to exchange data.

3.2 Lightweight Model

In this work, we are interested in how the device discovery parameters influence the responsiveness of device discovery and the energy consumption of a BLE advertiser. We focus on modeling the behavior of the BLE advertiser, because these devices are usually

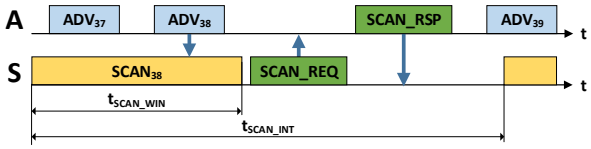

Figure 2: BLE device discovery: BLE advertising (A) and active scanning (S) on three advertising channels with successful discovery (scan request & response) on channel 38.

battery-powered and thus have a limited energy budget. These advertisers also allow a fine-grained control of the used advertising interval. The scanning devices, such as smartphones, instead, usually have a continuous power supply or are frequently recharged. Because these devices run a complex operating system (e.g., Android), a developer has limited control over the used scanning parameters.

Responsiveness. To decrease the discovery time of a system, a developer has the possibility to change the device discovery parameters on the BLE advertiser device and on the BLE scanner. As shown in [15, 22], a short advertising interval directly leads to a short device discovery latency, but also to a high power consumption on the advertiser. To decrease the discovery time, however, a developer could also adapt the scan parameters (scan window and scan interval). We therefore suggest the following scan parameters that minimize the device discovery latency of the application:

- A scanner should scan on all three advertising channels to minimize the effect of radio interference and avoid not detecting advertisers using only specific advertising channels.
- The scanner should use continuous scanning, where the scan window is equal to the scan interval and therefore the scanner's radio is always listening for nearby advertisers.
- The scan interval of the scanner should be set to $t_{SCAN\_INT} = t_{ADV\_INT} + t_{DELAY} + 3t_{TX} + 2t_{IFS}$. This ensures the shortest possible scan window that is still able to receive at least one advertising packet successfully.

Energy consumption. The overall energy consumption of a BLE advertiser mainly depends on the used advertising interval [15, 22]. A BLE-based smart object may use n different advertising intervals over time ($t_{ADV\_INT1}, \ldots, t_{ADV\_INTn}$), which result in n different power consumptions of the system ($P_1, \ldots, P_n$). The energy consumed by the smart object over any given period is

$$E = \sum_{i=1}^{n} P_i \cdot t_i \tag{1}$$

where P_i is the system's power consumption when using an advertising interval t_{ADV_INTi}, and t_i the amount of time during which t_{ADV_INTi} is used.

The power consumption of the system is calculated as:

$$P_i = P_{DS} + E_{TX} \cdot f_{ADVi} \tag{2}$$

where P_{DS} is the power consumption of the smart object when the system is in deep sleep mode (e.g., the radio is not used), and E_{TX} is the energy spent during a single advertising event for sending the advertising payload on all advertising channels. f_{ADVi} is the number of advertising events per second, calculated as:

$$f_{ADVi} = \frac{1}{t_{ADV\_INTi} + t_{DELAY}} \tag{3}$$

Table 1: Measured power consumption for different advertising payload length on the CY8C4248LQI-BL583 platform.

Adv. interval [ms]	n_{Bytes} [bytes]	Power [mW]
152.5	30	1.1615 ± 0.0030
152.5	3	0.8581 ± 0.0029
Deep sleep	0	0.0546 ± 0.0027

where the advertising interval is represented by t_{ADV_INTi} and t_{DELAY} defines the average delay that is added after every advertising event by the BLE radio. Because the individual t_{DELAY} values are randomly and equally distributed between 0 and 10 milliseconds, we use an average of $t_{DELAY} = 5\,ms$ for our model (see Fig. 1).

The energy E_{TX} spent during an advertising event depends on the application-specific advertising payload length (n_{Bytes}) and is:

$$E_{TX} = E_{TX,Base} + E_{TX,Byte} \cdot n_{Bytes} \tag{4}$$

where $E_{TX,Base}$ is the hardware-specific base energy consumed by the system during an advertising event (e.g., for powering the radio and processing BLE advertisement data). $E_{TX,Byte}$ is the hardware-specific energy cost for sending a single byte of advertising payload on all three advertising channels.

To calibrate our model for a specific platform, we only need to measure the system's power consumption during: (i) deep sleep, (ii) advertising with a fixed interval T_{ADV_INT} and a long payload, and (iii) advertising with the same interval T_{ADV_INT} and a short payload. We use an advertising inverval of 152.5ms, but the calibration can be done with an arbitrary T_{ADV_INT} value. Table 1 shows all necessary measurements that need to be performed in order to calibrate our model. Based on these measurements and using Equations 2 to 4, we derive $E_{TX,Base}$ to be 121.2472570 μJ and $E_{TX,Byte}$ to be 1.7696481 μJ on the CY8C4248LQI-BL583 platform.

4 ADAPTIVE ADVERTISING

Based on the investigation of BLE advertising and the resulting model presented in Sect. 3, we introduce next an adaptive advertising strategy. Using the latter, a smart object is able to learn from past user behavior, identify different interaction phases, and find an appropriate trade-off between the system's energy consumption and responsiveness for each phase by adapting the advertising interval at runtime. In phases where a user is likely to interact with the smart object, a short advertising interval is used to increase responsiveness. During phases where user interaction is unlikely, the strategy uses a long advertising interval to minimize energy consumption. Every smart object that uses our adaptive strategy calculates its own individual schedule based on its user behavior.

Our adaptive advertising strategy is split into three parts. First, the smart object needs to log information about its recent device discovery activities (Sect. 4.1). Second, our strategy identifies different user interaction phases in the logged data (Sect. 4.2). Third, the advertising interval of the system is adapted at runtime according to the requirements of each individual phase (Sect. 4.3).

4.1 Logging Device Discovery Activities

Our strategy adapts the advertising interval of a device based on its observed past user behavior. Therefore, the device needs to store the timestamps when it interacted with its users (e.g., when it has

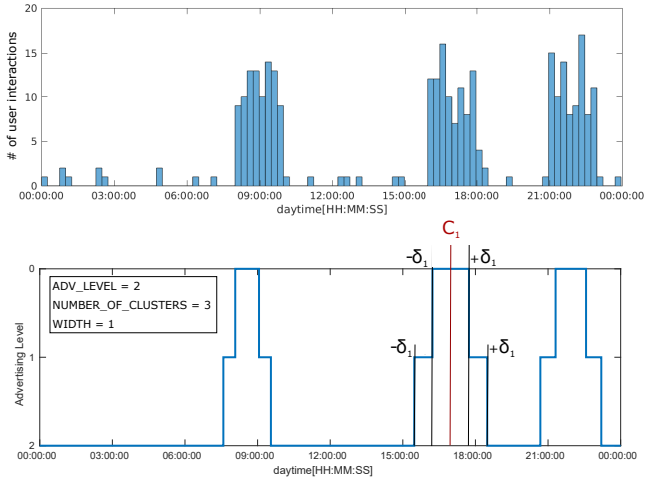

Figure 3: Progress of advertising levels during a single day (bottom) for three different advertising levels and clusters based on 300 recent user interactions (top).

received a scan or connection request). In principle, there are no limitations on the number of logged timestamps. However, large databases may be needed when the advertising should be adapted over long time periods, such as several months or a whole year. Because all calculations on the timestamps require little computational effort, they can be performed on the smart object and the data does not need to leave the system, which ensures user privacy.

4.2 Identifying User Interaction Phases

Using the stored timestamps, our adaptive advertising strategy analyzes the past user behavior by clustering time values and identifying individual interaction phases with different device discovery requirements. All logged timestamps are mapped to an adaptation period P on which basis the advertising schedule is calculated. For example, Fig. 3 shows a mapping of 300 timestamps to a period P of 24 hours, resulting in a schedule that adapts the advertising interval on a daily basis. We use a 24-hour adaptation period, but our strategy is able to calculate hourly or even yearly schedules.

To identify the different user interaction phases over time, we use the algorithm *k-means* [23]. Using this algorithm, the mapped timestamps are categorized in k different clusters on the one-dimensional time axis. Because the initial cluster center positions used by k-means have an influence on the algorithms performance and its result, we use a simplified version of the *k-means++ initialization* algorithm [1]. The first initial cluster center is set to a randomly selected timestamp from our log data. All other initial cluster centers are then placed to maximize the distance between all starting positions. Although this placement of the initial cluster centers requires more computation time than placing the centers at random positions, this approach ensures a fast convergence of k-means and therefore results in a lower overall energy consumption.

After the k-means algorithm has identified k different clusters with a high user interaction density, our adaptation strategy calculates the different interaction phases. First, it takes each cluster center C_i and calculates the individual standard deviation δ_i of the cluster's data points. Then, our adaptation algorithm identifies the time periods from one standard deviation δ_i before the cluster center to one standard deviation δ_i after the cluster center ($C_i \pm \delta_i$) as the phases where the fastest possible advertising level should be used, as shown in Fig. 3. The standard deviation δ_i of each cluster (and therefore its duration) depends on the cluster's individual data points. To calculate the next application phases, our strategy identifies the period of δ_i before and after the fastest interaction phase as the phase where the second fastest advertising level should be used. This calculation continues until the slowest possible advertising level is reached, resulting in a stair-shaped advertising schedule.

Our adaptive advertising strategy is configurable and supports a wide range of different smart object applications. By changing its parameters, it can be tailored to specific application needs.

Advertising levels. Instead of using concrete advertising interval values, our strategy calculates the advertising schedule based on different advertising levels. This makes our strategy independent from any specific hardware platform and its supported advertising interval values. Each advertising level can be mapped to a certain advertising interval on the used device. Higher adverting level values indicate longer advertising intervals and hence a low responsiveness of the system; advertising level 0 is the level with the highest possible responsiveness. The parameter `ADV_LEVEL` is used to configure the lowest supported advertising level used by our adaptation strategy, resulting in `ADV_LEVEL + 1` possible levels.

Number of clusters. The `NUMBER_OF_CLUSTERS` parameter defines the number of clusters that are identified by k-means. While choosing a high number of clusters generally leads to a finer adaptation schedule, it may also cause problems when the user interaction timestamps are widely spread over time. In such scenarios, our strategy may detect many different clusters during the whole adaptation period P and the resulting advertising schedule may never use the high advertising levels where energy would be conserved.

Phase durations. By changing the `WIDTH` parameter of our strategy, one can increase or decrease the duration spent in higher advertising levels. When `WIDTH` is set, it is used as a divisor of the individual phase lengths. Then, the fastest phases around each cluster range from $C_i - \frac{\delta_i}{WIDTH}$ to $C_i + \frac{\delta_i}{WIDTH}$. A `WIDTH` above 1 reduces the time spent in each level making it possible to enter low-power states faster, but may impact the system's responsiveness.

Cluster weighting. Depending on the user behavior, the k-means algorithm may identify clusters that only consist of very few timestamps. Such clusters are likely outliers and entering the fastest advertising level in such cases would reduce the battery lifetime of the system without any user benefit. To tackle this issue, we use a two stage cluster weighting that helps mitigating unnecessary advertising using the fastest advertising level due to data outliers. The weighting is defined by two parameters. The `IGNORED_THRESHOLD` defines the minimum percentage of timestamps a used cluster needs to hold. The `FASTEST_LEVEL_THRESHOLD` defines the minimum percentage of timestamps a cluster needs to hold to use the fastest advertising level. For example, with an `IGNORED_THRESHOLD` of 5, a cluster needs to hold as least 5% of the timestamps to not be ignored by our strategy. With a `FASTEST_LEVEL_THRESHOLD` of 10, a cluster needs to hold at least 10% of the timestamps to use the fastest advertising level. Using this cluster weighting, the number

of active clusters may be reduced and outlying data points may be filtered. As a consequence, however, the responsiveness of the system may be decreased for certain scenarios.

4.3 Adapting the Advertising Interval

Using the detected user interaction phases, we define an advertising schedule that defines when an adaptation of the advertising interval has to be performed.

From phases to a schedule. Based on the calculated phases, we extract points in time, so called *wake-up points*, where the system needs to wake-up and change its advertising interval. Here, we only consider the application phases that remain after cluster weighting has been performed. Using our strategy, the advertising level is a function that only increases or decreases by one level at a time. Depending on the actual distribution of user data, however, there may be two different types of inconsistencies that need to be resolved before creating the final advertising schedule. First, an inconsistency may occur when the phases of two different clusters overlap in time. For example, one cluster may suggest the fastest advertising level, while another cluster suggests advertising level 2. Second, the advertising level used at the start of the adaptation period P may be different than the level used at the end of the period. This causes a problem when the system enters a new period.

We tackle these problems by using a *scan line principle* that we apply to two consecutive adaptation periods of our schedule (two days in a row). Using this scan line, we scan over the whole schedule and consider all proposed changes in the advertising levels from the active clusters. If, at any point in time, multiple different levels are suggested, the highest advertising level is used.

Adaptation at runtime. In order to conserve energy, all unused system components are switched off whenever possible: including entering the low-power state of the CPU. With the calculated advertising schedule, it is possible to wake-up the CPU only when an adaptation of the advertising interval is necessary. We use the alarm functionality of the CPU, a timer interrupt that occurs at a configurable real-time clock value, to trigger a wake-up and adaptation when needed. Therefore, we sort all necessary wake-up points in ascending order and set the alarm to the earliest wake-up point. Once an alarm is triggered and the advertising interval is set, the subsequent alarm is set until the adaptation period end is reached.

Recalculation. We trigger a recalculation of the advertising schedule at the end of each adaptation period. When performing a recalculation, all the steps of our adaptation strategy (k-means clustering, phase calculation and weighting, calculating the advertising schedule, and setting the wake-up times) are performed.

5 IMPLEMENTATION

We use our adaptive advertising strategy in a real-world smart lock application, the Nuki Smart Lock [12]. This smart lock uses the CY8C4248LQI-BL583 BLE chip by Cypress Semiconductor [6], which features an ARM Cortex-M0 core with 32kB of memory and support for BLE v4.2 [2]. We implement our strategy using the CY8CKIT-042-BLE Development Kit (shown in Fig. 4) that features the CY8C4248LQI-BL583 chip and program the chip with the freely available PSoC Creator that provides a full BLE stack [5].

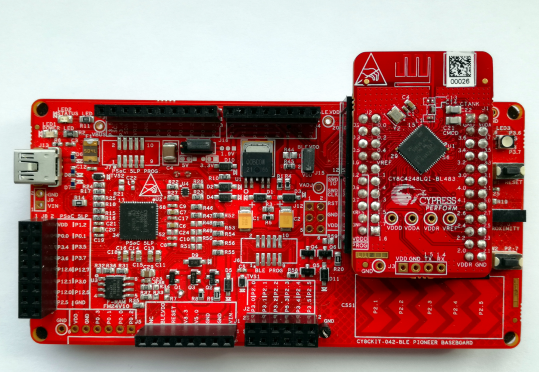

Figure 4: CY8CKIT-042-BLE development kit with the CY8C4248LQI-BL583 BLE chip used in our implementation.

In this application, our smart lock acts as BLE advertiser periodically broadcasting advertising packets in order to show its presence. A user of the smart lock can interact with the lock using a smartphone or a special Nuki Key Fob. Both devices act as a BLE scanner and initiate a connection with the lock when communication is needed. As mentioned in Sect. 3, the BLE advertiser consumes less energy than scanning devices as its radio is mostly off. This allows the smart lock to conserve energy, only communicating when a user wants to use the smart lock to open its door from outside.

Using this platform, we are able to store up to 300 user interactions on the lock before it runs out of memory. Each stored user interaction contains a timestamp measuring when it occurred and stores if the user issued a locking or an unlocking request. Once 300 interactions have been stored in the device memory, the system overrides the oldest entry, so that the lock always works with the 300 most recent values. Due to the limited number of user interaction entries, we make use of an adaptation period $P = 24$ hours, meaning that our strategy computes the advertising schedule for a whole day. In this work, we do not distinguish between the days of the week and recalculate the advertising schedule daily.

The Nuki Smart Lock supports a wide range of customers starting from single households with only one daily locking and unlocking action up to office buildings with many people doing different shifts. To support such a range of user behavior, we select the following parameters of our adaptation strategy. The smart lock is using an advertising payload length of 30 bytes. We configure our adaptation strategy to use five different advertising levels (`ADV_LEVEL = 4`). The mapping of the advertising levels to the actual advertising intervals is shown in Table 2. For identifying the different phases of user interactions, we use ten different clusters in our k-means algorithm (`NUMBER_OF_CLUSTERS = 10`). We use the unmodified standard deviation as our phase lengths (`WIDTH = 1`) and configure our cluster weighting to ignore clusters that contain less than 5% of the timestamps (`IGNORED_THRESHOLD = 5`), as well as to only use the fastest advertising level on clusters that contain at least 10 % of the timestamps (`FASTEST_LEVEL_THRESHOLD = 10`).

All user interaction phases are stored in fixed-sized arrays on the lock. The wake-up points are managed in a sorted array with the earliest wake-up point at the start. The first alarm is set to be issued when the first adaptation of the advertising interval needs

Table 2: Mapping of the advertising levels to the advertising intervals (T_{ADV_INT}) used by our smart lock application and the resulting measured average device discovery time.

Advertising level	T_{ADV_INT} [ms]	Discovery time [ms]
0	152.5	172.4
1	417.5	408.3
2	1022.5	1078.7
3	2000.0	1992.1
4	4000.0	4145.8

to be performed. Once an alarm goes off, the advertising interval is adapted and the next wakeup time is configured. At the end of the day, a full recalculation of the advertising schedule takes place.

6 EVALUATION

Our experimental evaluation uses three different smart lock use cases (Sect. 6.2) and answers the following questions:

- What is the latency of the used advertising intervals?(Sect. 6.1)
- Does our adaptive advertising strategy improve the average responsiveness of a real-world smart object? (Sect. 6.3)
- How does the energy consumption of a smart object differ when using our adaptive advertising strategy? (Sect. 6.4)
- Is our lightweight advertising model accurate so that it can be used to predict the energy consumption of a BLE advertiser using different advertising intervals? (Sect. 6.5).

6.1 Measuring the Device Discovery Latency

We start by measuring the average device discovery time when using the different advertising levels presented in Table 2. To evaluate the device discovery time, we use the CY8CKIT-042-BLE platform as an advertiser as described in Sect. 5. Additionally, we use an nRF52 [26] BLE device from Nordic Semiconductor as a scanner to accurately measure the time a scanner takes to detect the smart lock using different advertising intervals. The scanner runs the Zephyr OS [9] and performs passive continuous scanning with a scan interval of 5 seconds using all three advertising channels. These scan parameters are one of the possible scan settings on modern Android smartphones, therefore our setup provides measurements that are comparable to the real world use case in which a user interacts with the smart lock using a smartphone. We measure the device discovery time on the nRF52 by using a timer that starts when the device begins to scan. The timer stops when the scanner has successfully received an advertising packet from the smart lock. After a successful discovery, the scanner stops for a random value between 0 and 5 seconds before it starts scanning again.

Table 2 shows the average measured device discovery time for every used advertising interval. The measurements were performed 100 times for every used interval. As discussed in Sect. 3, we see that the advertising interval has a direct impact on the device discovery latency. While the discovery time is only 172.4 ms with advertising level 0, it increases by a factor of 24 when using advertising level 4.

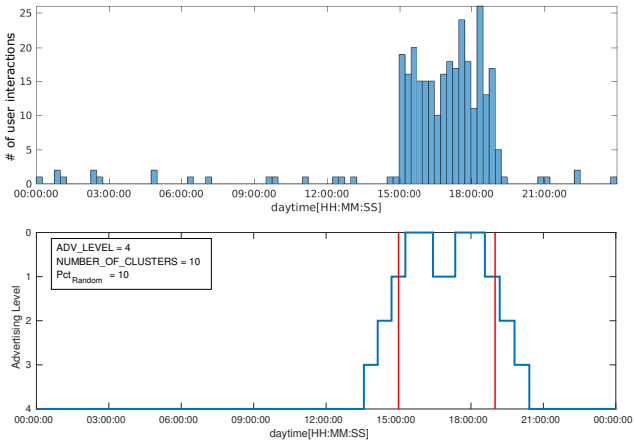

Figure 5: Timestamp distribution (top) and calculated advertising schedule (bottom) for a single household use case.

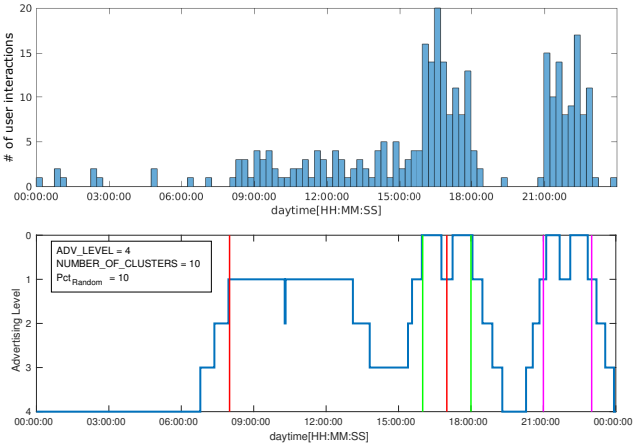

Figure 6: Timestamp distribution (top) and calculated advertising schedule (bottom) of an office building use case.

6.2 Evaluation Scenarios

To evaluate the performance of our adaptive advertising strategy, we simulate three different user scenarios: (i) a single household, (ii) an office building, and (iii) a completely random user behavior.

Single household. In this use case, we simulate a household with a single person with predictable behavior. We create timestamp logs where 90% of the values are spread over the interval between 15:00:00 and 19:00:00. This simulates the user returning from work every day in the afternoon. The remaining 10% of the timestamps are randomly spread over the 24 hour period. Fig. 5 shows the timestamp distribution and the resulting advertising schedule of this scenario calculated by our strategy.

Office building. As shown in Fig. 6, this use case simulates an office building with three user groups behaving differently. 30% of the timestamps are spread between 08:00:00 and 17:00:00 to simulate regular employees that arrive and leave from work (red). Another 30% of the timestamps are distributed between 16:00:00 and 18:00:00 and represent night shift workers entering the building (green). The office cleaning crew is simulated by another 30% of the timestamps

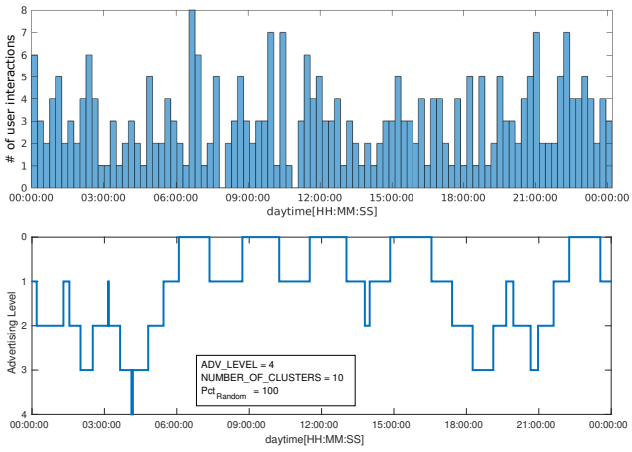

Figure 7: Timestamp distribution (top) and calculated advertising schedule (bottom) for a random user behavior.

spread between 21:00:00 and 23:00:00 (magenta). The remaining timestamps are randomly distributed over 24 hours.

Random behavior. With this use case, we simulate the worst case for our strategy, where no predictable user behavior can be extracted from the logged timestamps. All timestamp values are randomly distributed over the 24 hour period, as shows in Fig. 7.

6.3 Responsiveness

With the measured device discovery latencies shown in Table 2 and the advertising schedules calculated by our strategy in Sect. 6.2, we can evaluate the responsiveness of the smart lock in all three use cases. Without using our strategy, the smart lock uses a fixed advertising interval of 417.5 ms. To evaluate the responsiveness improvements for each scenario, we take the individual timestamp logs and identify the percentage of timestamp values that have a better, equal, or worse responsiveness compared to the fixed advertising interval.

Table 3 shows the resulting timestamp percentage (better, equal, or worse device discovery latency) for the three use cases. We see that the benefit of using our strategy differs for the individual use cases. The more predictable a user is, the better the resulting responsiveness of the smart lock in phases where a user wants to interact. For the single household and the office building, our strategy significantly reduces the average device discovery time in over 50% of the user interactions from 408.3 ms to 172.4 ms. Although these two scenarios had 10% interactions randomly distributed, our strategy leads to only 8.7% and 14.3% of user interactions that have a higher average device discovery. With a lower percentage of randomly distributed user interactions, the performance of our adaptive strategy would improve even more. Even when the user behavior is completely random, our strategy leads to a better or equal device discovery latency in 64.4% of user interactions.

6.4 Energy Consumption

Next, we measure the energy efficiency of our adaptive advertising strategy. Therefore, we use our smart lock application on the CY8CKIT-42-BLE platform as described in Sect. 5. To evaluate the energy efficiency of our strategy, we first measure the energy consumption of the smart lock using the default and fixed advertising interval of 417.5 ms over 24 hours. Next, we use the same device to run the advertising schedules calculated by our strategy and measure the energy consumed over 24 hours. The measurements were performed using the Monsoon Power Monitor [14].

Table 4 shows the energy consumption of the smart lock for all three use cases when using a fixed advertising interval compared to using our adaptive advertising strategy. We can see that a smart lock using our adaptation strategy is able to reduce its energy consumption by more than 48% when used in a single household use case. When used in an office building, our strategy is able to reduce the consumed energy by over 20% compared to a fixed advertising interval. Even in the worst case, when the user behavior is completely random over the 24 hour period, our strategy only consumes 23.33% more energy than the existing static solution.

Table 3: Improvements in responsiveness in the three different use cases when using our adaptive advertising strategy compared to a fixed advertising interval of 417.5 ms.

Scenario	Better[%]	Equal[%]	Worse[%]
Single household	57.0	34.3	8.7
Office building	51.0	34.7	14.3
Random behavior	35.7	28.7	35.6

Table 4: Measured energy consumption of a smart lock using a fixed advertising interval E_{Fixed} compared to using our adaptive advertising strategy E_{Adapt} over a 24 hour period.

Scenario	E_{Fixed} [J]	E_{Adapt} [J]	Savings [%]
Single household	40.513	20.857	48.52
Office building	40.513	31.804	21.50
Random behavior	40.513	50.933	-23.33

Table 5: Estimated energy consumption E_{Model} compared to the measured energy consumption $E_{Measured}$ of a smart lock using our adaptive advertising strategy over 24 hours.

Scenario	E_{Model} [J]	$E_{Measured}$ [J]	Δ_{Est} [%]
Single household	21.219	20.857	-1.90
Office building	31.355	31.804	+1.74
Random behavior	49.965	50.933	-1.41

6.5 Model Validation

Finally, we evaluate the accuracy of our energy consumption model of BLE advertising from Sect. 3. In particular, we use our model to estimate the energy consumption of the different advertising schedules from Sect. 6.2 over 24 hours using a payload of 30 bytes.

Table 5 shows the energy consumption estimated by our model compared to the actual measured energy consumption of the device for all three application scenarios over a whole day. We can see that our model is able to accurately estimate the energy consumption of the BLE device based on its daily advertising schedule. The maximum estimation inaccuracy of our model Δ_{Est} is below 2%.

6.6 Limitations

To conserve energy, our adaptive advertising strategy increases the advertising interval during phases with low user interaction (e.g., during night time). As mentioned above, our strategy re-calculates the advertising schedule on a daily basis. This introduces an issue when user behavior drastically changes from one day to the next. In such cases, users may experience a high latency and energy may be wasted in periods in which it would not be needed. Once interactions occur in periods where a high advertising level is used, the user may suffer from a long device discovery time. To mitigate this problem, we introduce the range extender concept in the next section.

7 RANGE EXTENDER

The range extender improves the performance of adaptive advertising when a user interacts with a smart object outside of his predicted behavior. We use other BLE devices to inform a smart object about approaching users and therefore extend its range, resulting in a better responsiveness. For example, a smart lock may use a BLE temperature sensor outside the house as a range extender to extend its communication range and detect approaching users in advance. Once a range extender device detects a new approaching BLE device, it informs the smart lock and the latter may intermittently enter a faster advertising state to limit the discovery time for the approaching user, therefore improving user experience. We shortly outline the main design principles of the range extender and illustrate the concept in Fig. 8.

Any standard compliant BLE device that is able to support at least two simultaneous BLE connections can be used as a range extender. The range extender is implemented using a custom BLE GATT service [2] and needs to be places in communication range to the smart object, whose range it needs to extend. A smart object may use multiple range extenders. Even a device implementing the range extender service can receive information about nearby devices from other range extenders.

7.1 Detecting Approaching Devices

To inform smart objects about approaching users, a range extender first needs to detect BLE devices, such as smartphones and wearables. A challenge in detecting these devices is that current smartphones and wearables only support BLE scanning, but not BLE advertising. Hence, instead of just scanning for nearby smartphones, a range extender needs to use BLE advertising to detect devices in its proximity. This is a very different, yet standard compliant, approach to the device discovery proposed by the BLE specification [2].

To detect nearby smartphones, the range extender performs undirected scannable BLE advertising (`ADV_SCAN_IND`) with a fixed advertising interval to broadcast its presence. Nearby smartphones that perform active scanning, are able to detect the range extender, and issue a scan request (`SCAN_REQ`) to get further information. According to the BLE specifications, these scan request messages contain the BLE device address of the scanning device. By extracting the BLE address from the received request, the range extender knows all the necessary information about the approaching user device. This device discovery process is shown in step II in Fig. 8.

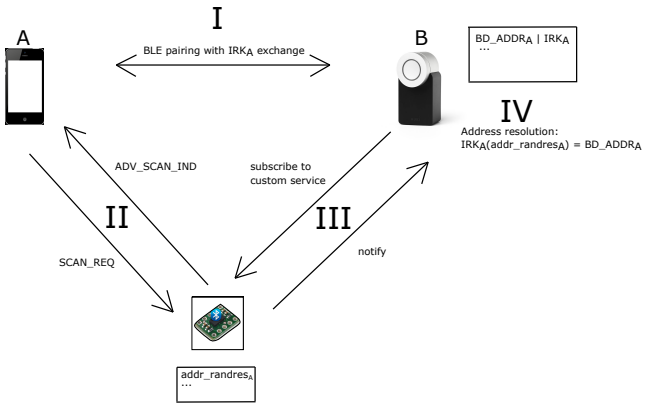

Figure 8: Based on a one-time BLE bonding procedure (I), the smart lock performs address resolution (IV) using the received private addresses of the range extender (III) to decide which devices are relevant. The range extender is using BLE advertising to detect nearby BLE scanning devices (II).

7.2 Notifying the Smart Object

The range extender stores a list of nearby BLE devices sorted by their most recent interaction. BLE devices that were longer inactive are likely to be out of range and therefore are removed from this list. When a new nearby device is detected, its BLE address is added to the range extender's list and any subscribed smart object is notified. This notification is performed using a custom BLE GATT service on the range extender. A smart object may subscribe to this service to get notified when the list of nearby devices changes. This subscription and notification process is shown in step III in Fig. 8.

Using BLE mesh, it is also possible that list update messages are transmitted over multiple hops to reach their destination.

7.3 Resolving Device Addresses

When a smart object is informed about an approaching BLE device, it needs to decide if this device is relevant for its application based on the BLE device address. Smartphones of nearby pedestrians, for example, are not relevant for a smart lock and adapting the BLE advertising due to these devices would only lead to wasted energy.

When identifying relevant devices, we need to distinguish between public and private BLE addresses. Public addresses uniquely identify a BLE device and one can directly check if they are relevant for the smart object. Private addresses, however, are frequently changed to avoid user tracking and can only be resolved by other trusted BLE devices. Almost all smartphones use private BLE addresses, thus the smart object needs to perform address resolution to retrieve the public BLE address. According to the BLE specification [2], a trusted BLE device uses the Identity Resolution Key (IRK) of its peer device (e.g., a smartphone) to translate the private into a public address. Therefore, every user device needs to initially share its IRK and public device address with the smart object using the standardized BLE bonding procedure during setup (see step I in Fig. 8). Once the IRK and public device address of a user device are known by the smart object, any private BLE address that the device uses can be resolved, as shown in step IV in Fig. 8.

Performing address resolution on the smart object makes it possible to use multiple simple range extender devices that do not need

to individually establish trust with every supported smartphone. Furthermore, with this approach, the private and sensitive user information never needs to leave the smart object.

7.4 Adapting the Advertising Interval

Once a relevant nearby BLE device is identified, the smart object can intermittently decrease the used advertising level overriding the advertising schedule of our adaptive advertising strategy. This allows the smart object to reduce the device discovery time when a user is nearby for a short period, hence improving user experience.

8 CONCLUSION AND FUTURE WORK

Our adaptive advertisement strategy for connection-less BLE allows BLE devices to learn from recent user behavior and adapt their application requirements accordingly. This strategy can be implemented on any smart object that supports BLE advertising with multiple advertising intervals. By using our adaptation strategy, BLE-based smart objects are able to reduce their power consumption by up to 50% while reducing user experienced latency by up to 95% in most cases. To handle unexpected user interactions, we introduce the concept of range extender, which improves a smart object's performance while allowing to preserve its energy budget.

Our strategy can be used to significantly improve the performance of other constrained BLE applications that require user interaction. Google Nearby beacons [1] could conserve energy when no potential users are nearby and could increase responsiveness in times of high user interaction. Other applications, such as Tile[2] and PitPatPet[3], could also use our strategy to significantly improve their battery lifetime without increasing the user experienced latency.

Depending on their requirements, smart objects may modify our adaptive advertising strategy to best suit their application needs:

Weekday Distinction. If device memory is not limited, the advertising schedule may be calculated on a weekly basis. This would allow to distinguish between individual days of the week.

Amount of users. A smart object may extract the individual users from its interaction logs. For example, a smart lock that has three users may detect that all of its users have already entered the house. Therefore, it can disregard the advertising schedule and enter a low-power mode because no user will open the door from outside.

Parameter adaptation. By applying extended data analysis, the parameters of our adaptive advertising strategy (such as the number of time clusters or advertising levels) could be dynamically selected at runtime in order to achieve the best possible performance.

ACKNOWLEDGMENTS

We would like to thank the Nuki Home Solutions GmbH for their cooperation and for supporting this research. This work was performed within the LEAD-Project "Dependable Internet of Things in Adverse Environments", funded by Graz University of Technology. This work was also partially funded by the SCOTT project. SCOTT (http://www.scott-project.eu) has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422. This joint undertaking receives support from the European Unions Horizon 2020 research and innovation programme and Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, Netherlands, Belgium, Norway. SCOTT is also funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2017 and April 2020. More information at https://iktderzukunft.at/en/.

[1] https://developers.google.com/nearby/
[2] https://www.thetileapp.com/
[3] https://www.pitpatpet.com/

REFERENCES

[1] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7), 2012.
[2] Bluetooth SIG. *Bluetooth Core Specification Version 4.2*, 12 2014. Rev. 4.2.
[3] B. Campbell, J. Adkins, and P. Dutta. Cinamin: A Perpetual and Nearly Invisible BLE Beacon. In *Proc. of the* 1^{st} *NextMote Workshop*, 2016.
[4] K. Cho, W. Park, M. Hong, G. Park, W. Cho, J. Seo, and K. Han. Analysis of Latency Performance of Bluetooth Low Energy (BLE) Networks. *Sensors*, 2014.
[5] Cypress Semiconductor Corporation. *PSoC Creator Component Datasheet - Bluetooth Low Energy (BLE)*, 12 2015.
[6] Cypress Semiconductor Corporation. *PSoC 4: PSoC 4XX8_BLE Family Datasheet*, 04 2017. Rev. *L.
[7] R. Faragher and R. Harle. Location Fingerprinting With Bluetooth Low Energy Beacons. *IEEE Journal on Selected Areas in Communications*, 33(11), 2015.
[8] Fitbit, Inc. Fitbit, 2018. https://www.fitbit.com/, accessed on 28.03.2018.
[9] Linux Foundation. Zephyr Project, 2017. https://www.zephyrproject.org/, accessed on 28.03.2018.
[10] J. Fürst, K. Chen, M. Aljarrah, and P. Bonnet. Leveraging Physical Locality to Integrate Smart Appliances in Non-Residential Buildings with Ultrasound and Bluetooth Low Energy. In *Proc. of the* 1^{st} *IEEE IoTDI Conference*, 2016.
[11] D. Giovanelli, B. Milosevic, C. Kiraly, A.L. Murphy, and E. Farella. Dynamic group management with Bluetooth Low Energy. In *Proc. of the* 2^{nd} *IEEE ISC2 Conference*, 2016.
[12] Nuki Home Solutions GmbH. Nuki, 2018. https://nuki.io, accessed on 28.03.2018.
[13] C. Gomez, J. Oller, and J. Paradells. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors*, 12(9), 2012.
[14] Monsoon Solutions Inc. Power Monitor Software, 2018. http://msoon.github.io/powermonitor, accessed on 28.03.2018.
[15] W.S. Jeon, M.H. Dwijaksara, and D.G. Jeong. Performance Analysis of Neighbor Discovery Process in Bluetooth Low-Energy Networks. *IEEE Transactions on Vehicular Technology*, 66(2), 2017.
[16] C. Julien, C. Liu, A.L. Murphy, and G.P. Picco. BLEnd: Practical Continuous Neighbor Discovery for Bluetooth Low Energy. In *Proc. of the* 16^{th} *ACM/IEEE IPSN Conference*, 2017.
[17] S. Kamath and J. Lindh. Measuring Bluetooth Low Energy Power Consumption. *Texas Instruments Application Note AN092, Dallas*, 2010.
[18] P. Kindt, M. Saur, and S. Chakraborty. Neighbor Discovery Latency in BLE-Like Duty-Cycled Protocols. *arXiv preprint arXiv:1509.04366*, 2015.
[19] P. Kindt, D. Yunge, M. Gopp, and S. Chakraborty. Adaptive Online Power-Management for Bluetooth Low Energy. In *Proc. of the IEEE INFOCOM Conference*, 2015.
[20] T. Lee, M. S. Lee, H. S. Kim, and S. Bahk. A Synergistic Architecture for RPL over BLE. In *Proc. of the* 13^{th} *IEEE SECON Conference*, 2016.
[21] J. Liu, C. Chen, and Y. Ma. Modeling and Performance Analysis of Device Discovery in Bluetooth Low Energy Networks. In *Global Communications Conference (GLOBECOM), 2012 IEEE*. IEEE, 2012.
[22] J. Liu, C. Chen, Y. Ma, and Y. Xu. Energy Analysis of Device Discovery for Bluetooth Low Energy. In *Proc. of the* 78^{th} *IEEE VTC Fall Conference*. IEEE, 2013.
[23] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. University of California Press, 1967.
[24] K. Mikhaylov. Accelerated Connection Establishment (ACE) Mechanism for Bluetooth Low Energy. In *Proc. of the IEEE PIMRC Conference*, 2014.
[25] Nest Labs. Nest Thermostat, 2018. https://www.nest.com/, accessed on 28.03.2018.
[26] Nordic Semiconductor. *nRF52832 Product Specification*, 10 2017. Rev. 1.4.
[27] PitPat. Dog Activity Monitor, 2018. https://www.pitpatpet.com/, accessed on 28.03.2018.
[28] Roche Media Release. Roche launches innovative Accu-Chek Guide blood glucose monitoring system, August 2016.
[29] M. Spörk, C. A. Boano, M. Zimmerling, and K. Römer. BLEach: Exploiting the Full Potential of IPv6 over BLE in Constrained Embedded IoT Devices. In *Proc. of the* 15^{th} *ACM SenSys Conference*, November 2017.