© 2017 IEEE.
Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting /republishing this materialfor advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

DOI: 10.1109/TMC.2017.2737008

Neighbor Discovery Latency in BLE-Like Protocols

Philipp H. Kindt, Marco Saur, Michael Balszun and Samarjit Chakraborty

***Abstract*—Neighbor discovery is the procedure using which two wireless devices initiate a first contact. In low power ad-hoc networks, radios are duty-cycled and the latency until a packet meets a reception phase of another device is determined by a random process. Most research considers slotted protocols, in which the points in time for reception are temporally coupled to beacon transmissions. In contrast, many recent protocols, such as ANT/ANT+ and Bluetooth Low Energy (BLE) use a slotless, periodic-interval based scheme for neighbor discovery. Here, one device periodically broadcasts packets, whereas the other device periodically listens to the channel. Both periods are independent from each other and drawn over continuous time. Such protocols provide 3 degrees of freedom (viz., the intervals for advertising and scanning and the duration of each scan phase). Though billions of existing BLE devices rely on these protocols, neither their expected latencies nor beneficial configurations with good latency-duty-cycle relations are known. Parametrizations for the participating devices are usually determined based on a "good guess". In this paper, we for the first time present a mathematical theory which can compute the neighbor discovery latencies for all possible parametrizations. Further, our theory shows that upper bounds on the latency can be guaranteed for all parametrizations, except for a finite number of singularities. Therefore, slotless, periodic interval-based protocols can be used in applications with deterministic latency demands, which have been reserved for slotted protocols until now. Our proposed theory can be used for analyzing the neighbor discovery latencies, for tweaking protocol parameters and for developing new protocols.**

***Index Terms*—Bluetooth Low Energy, ANT, Neighbor Discovery, MANETS.**

I. Introduction

With the increasing pervasiveness of smartphones and wireless sensors, mobile ad-hoc networks (MANETs), in which all participating devices are battery-powered, have become widely-used. MANETs nowadays are applied e.g., for connecting gadgets to smartphones, for localization beacons, object tracking and mobile hotspots. In such applications, energy saving is a crucial requirement.

The procedure to establish a first contact between two devices in a MANET is usually referred to as *neighbor discovery*. To save energy, the devices usually apply duty-cycling, which means that they repeatedly switch on their radios and sleep in the meantime. A successful discovery can take place if and only if one radio listens on the channel at the same point in time at which another one sends a packet. As the clocks of these devices are completely unsynchronized, the point in time when two devices meet for the first time is random. A popular example of a protocol that applies such a duty-cycled scheme is Bluetooth Low Energy (BLE) [1]. Neighbor discovery is an active area of research and many approaches have been proposed. Mainly, they fall in one of the following categories:

All authors are with the Chair of Real-Time Computer Systems (RCS), Technische Universität München (TUM), Munich, Germany. Email: kindt/saur/balszun/chakraborty@rcs.ei.tum.de

Wakeup-Receivers: Approaches such as [2], [3] assume that there is an additional, low power receiver that continuously senses the channel for wakeup beacons. Many wireless systems do not have such an additional receiver. Therefore, they make use of one of the following approaches.

Birthday Protocols: These protocols require every node to switch between a low-power-state, a sending-state and a receiving-state at random points in time with given transition probabilities [4], [5], [6]. Whereas good average latencies can be achieved, the main drawback is the lack of guarantees on upper latency bounds.

Slotted Protocols: In this protocol family, time is subdivided into fixed-length slots. The most important concept of slotted protocols is that the points in time at which packets are sent (and hence also the number of packets) are always associated to listening phases. In each slot, a device can either sleep or be active. Typically, each device needs to send a packet at the beginning and the end of each active slot and must listen during the entire duration in between [7]. Multiple patterns of active and sleep-slots have been proposed for providing bounded worst-case latencies. For example, coprimality-based approaches [8], [7], [9] assume that active slots occur with a certain repetition period. The periods of two devices are chosen such that they are mutually co-prime and hence, the Chinese Remainder Theorem (CRT) guarantees an overlap within a limited number of slots. As an extension, [10] proposes adding additional active slots to protocols like Disco [7] or U-Connect [9] for exchanging information on devices discovered by the already known neighbors. An alternative scheme is used by quorum-based protocols [11], in which a period of m slots is assumed. The slots of a hyper-period of length m^2 are arranged in a $m \times m$-matrix. One node choses a row for determining its active slots, whereas the other one choses a column, such that discovery is guaranteed within a limited number of slots. Recently, a protocol called *Searchlight* has been proposed in [12], where it has been claimed that it outperformed all other quorum- and CRT-based protocols at the time of its publication. In each period of Searchlight, there are two active slots. The first one has a fixed position in each period, whereas the other one repeatedly changes its position in a cyclic manner, thereby deterministically meeting the remote node's fixed slot within a finite number of slots. Another quorum-based protocol called *Hedis* [13] is claimed to outperform Searchlight in asymmetric cases, in which both devices operate with different duty-cycles. Moreover, the parametrizable protocol *Hello* [14] generalizes the already mentioned protocols *Searchlight* [12], *Disco* [7] and *U-Connect* [9], which can be described as special cases of it. The main drawbacks of slotted strategies are a) the restricted set of duty-cycles that can be realized, b) redundant slots [14], and c) the requirement to send beacons always directly before or after scanning phases, which constrains the design space.

Slotless, periodic interval-based solutions: While slotted protocols require that beacon transmissions are temporally coupled to reception phases, slotless, periodic interval (PI)-based solutions break away from this constraint. Packet transmission occurs with an interval T_a, while receptions take place for a certain duration d_s once per period T_s. The parameters T_a, T_s and d_s are drawn over continuous time and can be chosen freely by the protocol designer. Though such schemes can also be used for symmetric two-way discovery, most existing protocols (e.g., BLE and ANT/ANT+) assign different roles to different devices. In particular, they assume that one device only broadcasts advertising packets, whereas the other one only scans. One of the first periodic interval-based protocols, STEM-B, has been proposed in [15]. It works as follows.

- One device, called the *initiator* or *advertiser*, broadcasts advertising packets with a fixed interval T_a.
- Another device that is called the *target* or *scanner* continuously listens for packets during d_s amounts of time once per interval T_s. We refer to d_s as the *scan window*.
- If the scanner receives a packet, it sends a response on a different wireless channel. Once the initiator receives this packet, e.g. by listening after each transmission [1], both devices are synchronized.

Popular slotless protocols: Though most recent research has considered slotted discovery, adaptations of the slotless scheme described above are widely used in practice. The ANT/ANT+[16] protocol, which is used in over 100 million devices [17], implements this scheme, with the only modification that it performs the complete discovery procedure on one channel [18]. Further, it has been adopted and slightly modified for BLE, in which an advertiser periodically sends up to three packets in a row on different channels [1]. This period consists of the sum of an interval T_a and a random delay of up to $10\,\mathrm{ms}$. On the remote side, the scanner listens periodically on one of these three channels for a duration of d_s, thereby toggling the channel once per interval T_s.

As we will describe in detail in Section II, even though such protocols are widely used e.g., in billions of BLE devices, their behavior currently cannot be fully analyzed. Except for the trivial case of $T_a < d_s$, no theory for computing their latencies exists. In particular, it is not known whether they can guarantee any upper latency bounds. Moreover, the impact of different parameter valuations for T_a, T_s and d_s cannot be studied in a systematical manner. As a result, there is currently no feasible way to choose optimal protocol parameters, e.g., for BLE. However, parameter optimization is extremely important since, as we will show, unfavorable parametrizations risk lying in a hyperbolic peak and therefore cause long mean latencies and high energy consumptions for the connection setup.

Difference from slotted protocols: The main reason for this lack of understanding is the complex analysis of PI-based, slotless protocols. Whenever two slots in a slotted protocol overlap in time, discovery is guaranteed, regardless of the actual temporal length of the overlap. Therefore, only a finite number of different temporal offsets between the active slots of two devices need to be considered, which makes the analysis simple. In contrast, in slotless, PI-based protocols, there could be an infinite number of initial offsets with different associated latencies. This makes existing models for slotted protocols unsuitable for slotless solutions. The theory presented in this paper shows that the range of possible initial offsets can be subdivided into multiple, variable-length partitions with constant discovery latencies and hence the number of possibilities becomes finite for PI-based protocols, too. Another challenge is finding the borders of these partitions, which is not straightforward.

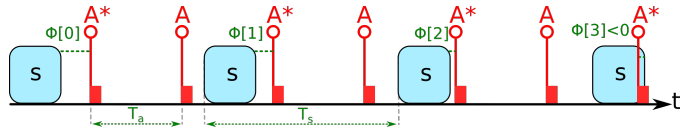

Fig. 1. Shrinking distances $\Phi[k]$ between neighboring advertising packets (*A*) and scan windows (*S*). The index k is a counting number for the scan windows.

In this paper, we for the first time present a mathematical model that can compute the mean and worst-case neighbor discovery latencies of periodic interval-based, slotless protocols for all possible parameter values.

Modeling PI-based protocols: The key idea of our proposed model is to track the change in the temporal distance between neighboring advertising packets and scan windows over time, rather than considering the corresponding absolute points in time. This temporal distance $\Phi[k]$ is depicted in Figure 1 for $k = 0$ to $k = 3$. In this example, an advertising packet is the *right neighbor* of a scan window k, if it is the closest packet temporally on the right of it. In Figure 1, such neighboring advertising packets are tagged with *A**, whereas the remaining ones are tagged with *A*. In this example, after each instance k of the scan interval T_s, the temporal distance $\Phi[k]$ shrinks by a constant value. This shrinkage occurs repeatedly for multiple subsequent intervals, until the temporal distance falls below the length of one scan window d_s. Therefore, a match between the advertiser and the scanner occurs, as for the last scan window of Figure 1. In other examples, $\Phi[k]$ grows instead of shrinking, or the distance $\Phi[k]$ shrinks or grows when considering suitable multiples of T_a and T_s. Nevertheless, this concept can be generalized to all possible constellations. These generalizations require a more elaborate procedure, which is described in Section IV. By taking into account all possible initial offsets between the first advertising packet and the first scan window, the exact expected value of the discovery latency and its maximum value can be calculated using analytical methods with low computational complexity.

Comparison with related models: This technique differs significantly from all known models for PI-based protocols. All existing models are restricted to the trivial case of $T_a < d_s$, and rely on counting the number of T_a-intervals until reaching the time-instance at which the scanner switches on its receiver for the first time. Parametrizations with $T_a > d_s$ are very relevant in practice, since they enable significantly lower duty-cycles of the advertiser and are widely used in the practice, e.g., as a recommended parametrization in the BLE find me profile [19]. To the best of our knowledge, the technique of tracking the change in the temporal distance among neighboring periodic events, as proposed in this paper, has not

been known before, neither for neighbor discovery nor in other domains. This generic technique is suitable for modeling possible overlaps of all kinds of periodically repeated actions, and can therefore be applied in other applications, e.g., for accelerating discrete-event simulations too.

Implications of our results: With our proposed theory, we can for the first time show that a maximum discovery latency can be guaranteed for all parametrizations except for a finite number of singularities, at which the latency converges towards infinity. This insight allows interval-based protocols even with $T_a > d_s$ to be applied in scenarios where deterministic maximum latencies are required, in which currently only slotted protocols are used. In addition, our proposed theory can be used to find optimal parametrizations of protocols like ANT/ANT+ or BLE. Recent work based on this paper[1] has shown that optimized PI-based protocols can significantly outperform all previously known discovery protocols, e.g., by achieving 10x shorter discovery-latencies than *Searchlight* [20] [21] [22].

Organization of the paper: The rest of this paper is organized as follows. In Section II, we present related work. We formally define the problem of neighbor discovery in Section III. In Section IV, we present our proposed theory for solving the neighbor discovery problem. In Section V, we present an efficient implementation of our proposed technique. Readers who are only interested in understanding our proposed theory may only focus on Section IV. Only those wo are interested in an implementation may continue to Section V. Next, we evaluate our theory with comprehensive discrete event simulations and real-world measurements in Section VI. In Section VII, we describe how the parameter values of interval-based protocols can be chosen in an optimal fashion. Finally, we summarize our results and discuss their implications in Section VIII. To ease the readability of this paper, a complete table of symbols used is given in Appendix G.

II. Related Work

In this section, we provide an overview on existing models for periodic interval-based, slotless protocols, such as BLE. As already mentioned, exact models only exist for the trivial case of $T_a < d_s$. This implies that the duration of one scan window must be larger than the time between two advertising packets. Therefore, every scan attempt is successful and the discovery latency is limited to roughly $1 \cdot T_s$. The first known model (for $T_a < d_s$) has been presented along with the introduction thew STEM-B protocol [15], which is one of the first PI-based protocols that have been proposed. In [23], [24] and [25], this solution has been adopted to the BLE protocol to account for multiple channels, but the limitation to $T_a < d_s$ remains.

For the general case, as [18] states, there is no known model to compute the discovery latency of PI-based protocols. Simulations are computationally very complex, since the discovery process needs to be simulated repeatedly for a large number of initial time offsets in order to assess the mean discovery latencies. An attempt to reduce this complexity has been proposed in [26]. However, the resulting complexity is still impractically high, and only an estimation of the mean latency can be given. No estimate on the maximum discovery latency can be provided. An attempt to derive a probabilistic model for $T_a \geq d_s$ has been made in [27], [28]. Here, the probability that an advertising packet meets a scan window has been assumed to be constant for all packets on a particular channel. In particular, if p_m is the probability of a miss of the first advertising packet on a particular channel, then it is assumed that the miss-probability of the k'th advertising packet is p_m^k. However, the periodic nature of BLE implies that a strong correlation between the probability and the interval instance exists and hence $p_m(k)$ is a function of the interval count k. Hence, as we will also show in Section VI-D, the results obtained from such models do not correlate well with the actual discovery latencies. Also, under the assumption of constant matching probabilities for all packets, a deterministic discovery is not guaranteed and therefore no upper bound can be inferred. In [29], the discovery latency of BLE has been computed by applying the probabilistic model checker PRISM [30] for some discrete values of $T_a > d_s$, T_s and d_s. However, in our opinion, the necessary details to understand this work have not been provided in [29]. In addition, the initial offset between the first advertising packet and scan window has been assumed to be fixed. As a result, this approach is unable to determine the mean- and worst-case discovery latencies among all possible initial offsets.

Usually, PI-based protocols are considered to be non-deterministic, since no upper bound could be determined until now. To overcome the lack of determinism, it has been proposed to configure the BLE protocol such that the parameter values fulfill the CRT [31]. However, such configurations restrict the range of possible valuations unnecessarily. In summary, to the best of our knowledge, no valid model for PI-based protocols for the general case ($T_a \geq d_s$) has been presented until now.

Our contributions: In this paper, we solve this problem and present the first mathematical theory of PI-based protocols. In particular, we make the following contributions:

1) We for the first time propose a model which can compute the exact mean and maximum discovery latencies for the complete range of possible parametrizations, including $T_a > d_s$. This theory leads to important new insights into such protocols and provides a full understanding of the probabilistic processes involved.

2) By using our model, we demonstrate that the discovery-latency is bounded for almost all parametrizations. The popular belief until now was that parametrizations with $T_a > d_s$ lead to unbounded discovery latencies. This new finding has important implications on the design of wireless protocols and the parametrization of protocols like BLE.

3) We evaluate our proposed solution both with comprehensive discrete-event simulations and real-world measurements, which are based on an implementation on two wireless radios. Thereby, we demonstrate that our predicted latency bound is safe (i.e., the latency never exceeds this bound) and tight (i.e., the bound can be reached in practice), and that our estimated mean latencies are precise.

[1] This paper has been available as a preprint on arXiv.org since sept. 2015.

III. Problem Formulation

In this section, we formally define the problem that is addressed in this paper. One device, referred to as the *advertiser* (A), periodically sends out *advertising packets*, starting from an uniformly distributed[2] random point in time t_{a0}. Transmitting a packet takes d_a time-units. The packets are sent periodically with the *advertising interval* T_a. The remote device is called the *scanner* (S). It periodically switches on its receiver once per *scan interval* T_s and listens for advertising packets for a duration called the *scan window* d_s. The neighbor discovery process is successful once an advertising packet has been received successfully by the scanner. Therefore, both devices need to be awake simultaneously for at least one packet transmission duration d_a. In Figure 2, the first advertising packet (labeled with (A_1)) is not received successfully, as it only partially overlaps with the scan window S_2. The second (A_2) and third (A_3) advertising packets are not received either, because they do not overlap with any scan window at all. The fourth packet (A_4) lies entirely within the scan window S_4 and is therefore received successfully. We assume no packet loss and only one channel. Then, the necessary and sufficient condition for a successful reception is given by the existence of some $i, j \in \mathbb{N}_0$, such that

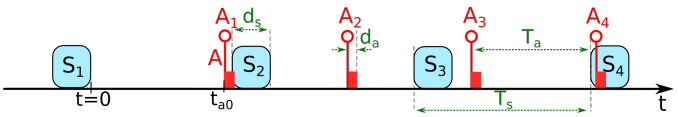

Fig. 2. Sequence of advertising packets $(A_1 - A_4)$ and scan windows $(S_1 - S_4)$ for neighbor discovery.

$$j \cdot T_s - d_s \leq t_{a0} + i \cdot T_a \leq j \cdot T_s - d_a, \tag{1}$$

where t_{a0} is the random point in time at which the first advertising packet is sent and i and j are the indices or the counting numbers of all advertising packets/scan windows. The questions answered in this paper are the following. What are the maximum and average times for an advertising packet to lie within a scan window for the first time (as given by Inequality 1), i.e., what are the max and mean discovery latencies?

IV. Protocol Analysis

In this section, we describe the key concepts of our proposed solution to compute the mean- and maximum neighbor discovery latencies. To develop an intuitive understanding of this, we first present four different examples. Next, we formalize and generalize these examples. Towards this, without any loss of generality, we first set the duration of the advertising packets d_a to 0. Because a successful reception occurs only if the advertising packet is received entirely by the scan window, this measure can be compensated for by shortening the scan window d_s by d_a time units. Further, we have to add d_a time units to all computed latencies.

A. Examples

a) $\mathbf{T_a \leq d_s}$: Situations with $T_a \leq d_s$ form the most simple cases. Consider the situation depicted in Figure 3 a). Again, the straight lines with circles on their heads depict the starting times of the advertising packets, whereas the boxes depict the scan windows. Here, $T_a \leq d_s$ and hence a successful match is guaranteed within one scan interval. The temporal distance between the second scan window and each advertising packet shrinks by T_a time units with every packet sent. Given an initial offset $\Phi[0]$ between the end of the first scan window and the first advertising packet, the discovery-latency is defined by the number of advertising intervals that need to pass until the scan window is reached. It follows that the neighbor discovery latency d_{nd} is:

[2]The assumption of uniformly distributed initial time-offsets influences the computed mean discovery latencies, but not the worst-case latencies.

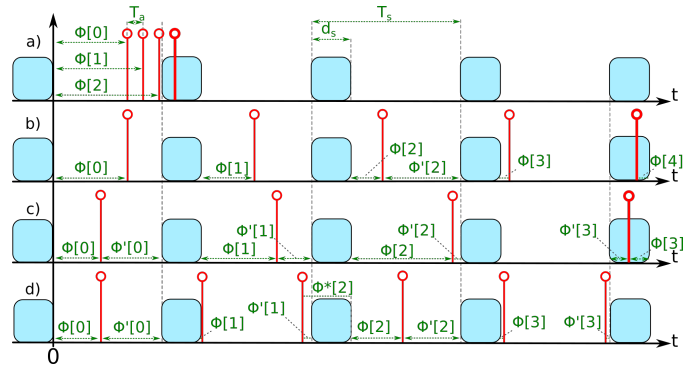

Fig. 3. Examples for different situations of neighbor discovery processes. The rounded boxes depict the scan windows, whereas the lines with circles at their heads depict the advertising packets.

$$d_{nd}(\Phi[0]) = \begin{cases} \left\lceil \frac{T_s - \Phi[0] - d_s}{T_a} \right\rceil \cdot T_a, & \text{if} \quad \Phi[0] \leq T_s - d_s, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

As can be seen easily from the figure, the maximum discovery latency for this example is $d_{nd,m} = \lceil \frac{T_s - d_s}{T_a} \rceil \cdot T_a$.

b) $\mathbf{T_a > d_s}$: Whenever $T_a > d_s$, as depicted in Figure 3 b), a successful discovery within one scan interval is not guaranteed anymore. Here, it is beneficial to describe the problem using a different representation. Instead of considering the absolute points in time the advertising packets and scan windows begin at, we examine the relative time differences $\Phi[k]$ between each scan window $k = 0, 1, 2, ...$ and its neighboring advertising packets. As we will introduce later, there are "shrinking" and "growing" constellations of scan windows and advertising packets. The example in Figure 3 b) is shrinking, and here only each closest neighboring advertising packet that is temporally on the right of a scan window (i.e., the first advertising packet on the right of each scan window) is relevant.

From Figure 3 b), one can observe that in this example, the temporal distance $\Phi[k]$ is monotonically shrinking for increasing indices k, and the amount of shrinkage is constant per scan interval k. This shrinkage is denoted as γ_0, and it is $\gamma_0 = \Phi[k] - \Phi[k+1]$. Here, one can observe that a match occurs after $\Phi[k]$ becomes smaller than 0 (cf. $\Phi[4] < 0$ in Figure 3 b). This holds true for any initial offset, as long as the shrinkage γ_0 is smaller or equal than d_s .

In the previous example with $T_a \leq d_s$, (cf. Figure 3 a)), we had counted the number of advertising intervals that fit into the temporal distance to the next scan-window (i.e., $T_s - d_s - \Phi[0]$ time units) for computing the discovery latency. The example from Figure 3 b) can be handled in the same way - with the only difference being that we have to

count the number of γ_0-intervals that fit into the initial offset, instead of the number of advertising intervals. Hence, the discovery latency is defined by Equation (3).

$$d_{nd}(\Phi[0]) = \begin{cases} \left\lceil \frac{\Phi[0]-d_s}{\gamma_0} \right\rceil \cdot T_a & \text{, if} \quad \Phi[0] \geq d_s \\ 0 & \text{, else.} \end{cases} \tag{3}$$

c) $\mathbf{T_a > d_s}$, growing: In the previously examined situation (depicted in Figure 3 b)), the offset $\Phi[k]$ shrank for increasing values of k. However, there are cases in which $\Phi[k]$ becomes larger with increasing values of k, as depicted in Figure 3 c). Whereas the distance from a scan window k to its right neighboring advertising packet $\Phi[k]$ grows with increasing values of k, the distance to its left neighboring advertising packet $\Phi'[k]$ shrinks. Growing constellations can therefore be handled similarly to shrinking ones.

d) $\gamma_0 > \mathbf{d_s}$: The examples b) and c) both had $\gamma_0 \leq d_s$ in common. However, for $\gamma_0 > d_s$, the computation becomes more involved. In the example from Figure 3 d), the distance $\Phi[0]$ shrinks from $\Phi[0]$ to $\Phi[1]$ by γ_0 time units. After $\Phi[1]$, this distance Φ_2^* (we label this distance with *, because it is no longer the distance to the closest advertising packet on the right of the scan window) becomes smaller than $-d_s$, since the advertising packet has moved to the left side of the scan window. Because advertising packets on the left have no chance of hitting the scan window considered (given the situation is shrinking), the advertising packet that needs to be considered changes (e.g., from $\Phi[2]^*$ to $\Phi[2]$ in the figure). One advertising packet is skipped and the offset $\Phi[k]$ becomes larger again for this packet (i.e. $\Phi[2] > \Phi[1]$).

We say that after $\Phi[1]$, the advertising packet "overtakes" the scan window. After that, from $\Phi[2]$ to $\Phi[3]$, the distance shrinks again by γ_0, until the relevant packet changes the next time. Shrinking the initial offset successively by γ_0 time units cannot guarantee discovery, since the "overtaking" might repeat. However, let us consider two consecutive skipped advertising packets, e.g., the packets with the offsets $\Phi'[1]$ and $\Phi'[3]$ in Figure 3 d). Both packets fulfill the condition that they are the first ones after an advertising packet has "overtaken" the corresponding scan window, and it is $\Phi'[1] - \Phi'[3] = const. = \gamma_1 < \gamma_0$. In other words, when considering appropriate multiples of the advertising interval (here: $i_2 = 3 \cdot T_a$) and scan interval (here: $j_2 = 2 \cdot T_s$), given an offset $\Phi[k]$ between a scan window k and its neighboring advertising packet, the offset between a scan window that is i_n scan intervals and j_n advertising intervals later will be $\Phi[k] - \gamma_1$. In case 3 d), $\gamma_1 < d_s$ and therefore the advertising packets and scan windows will match guaranteed after the initial offset $\Phi[0]$ has shrunk by a certain number of steps of length γ_1.

B. γ-Sequences

We now generalize the above examples. Towards this, let us first define the concept of γ-sequences and state multiple properties, which we claim to be true. We will then use these properties to compute the neighbor discovery latencies for arbitrary parametrizations. We will focus our attention on proving these claims in Appendix C.

Definition 1. γ-sequences: Let there be a set of parameters (T_a, T_s, d_s) and pairs of integers $(i_n,\ j_n)$, $n = 0, 1, 2, ...n_m$, with $i_n \geq i_{n-1}$ and $j_n \geq j_{n-1}$. We call a sequence of scan windows and advertising packets defined by $(1 \cdot i_n \cdot T_a,\ 1 \cdot j_n \cdot T_s),\ (2 \cdot i_n \cdot T_a,\ 2 \cdot j_n \cdot T_s), ...$ a *γ-sequence* of order n, if the following properties are fulfilled.

Property 1. If the time offset between any scan window and advertising packet is Φ, the time offset between the scan window that is j_n scan-intervals later and the advertising packet that is i_n advertising intervals later will be $\Phi \pm \gamma_n$. Sequences of tuples of scan windows and advertising packets, in which such offsets always decrease, are called *shrinking* γ-sequences, whereas constellations, in which these offsets increase, are called *growing* γ-sequences.

Property 2. $0 \leq \gamma_n \leq \min(T_a, T_s),\ \forall n$

Definition 2. Mode $\mathbf{m}$: We define $m_n \in (g, s, c)$ as the *mode* of a γ-sequence of order n. If the offset Φ between any scan window and any advertising packet becomes larger every i_n advertising packets and j_n scan-windows, then $m_n = g$ (growing). If it becomes smaller, then $m_n = s$ (shrinking). If $\gamma_n = 0$, then there is no shrinkage or growth, and $m_n = c$ (coupling).

Property 3. $\gamma_n = |T_s - |\sum_{k=0}^{n-1} a_k \cdot \gamma_k||$, $a_k \in \mathbb{Z} \setminus 0$. Here, the only valid choices of a_k are those that satisfy the following: i) $\gamma_n \leq \frac{1}{2}\gamma_{n-1}$, ii) the sum $\sum_{k=0}^{n-1} |a_k| \cdot (i_k T_a)$ is minimized, and iii) for a shrinking sequence, $a_k < 0$, otherwise, $a_k > 0$.

Property 4. For any γ_n, $i_n \cdot T_a$ has always at least the value of the smallest possible sum $\sum_{k=0}^{n-1} |a_k| \cdot (i_k T_a)$, for which $|\sum_{k=0}^{n-1} a_k \gamma_k| \geq T_s - \gamma_n \geq T_s - \frac{1}{2}\gamma_{n-1}$.

Property 5. The smallest n for which $\gamma_n \leq d_s$ is called the *maximum order* n_m for the given parametrization. It is

$$n_m = \begin{cases} \left\lceil \frac{\ln(\min(T_a,T_s)) - \ln(d_s)}{\ln(2)} \right\rceil, & \text{if} \quad \min(T_a, T_s) > d_s, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Definition 3. Penalty σ: Let there be a γ-sequence with shrinkage/growth γ_n. Then $\sigma_n = i_n \cdot T_a$ is called the *penalty* for increasing or decreasing any offset between a scan window and an advertising packet by γ_n time units.

Lemma 1. Property 3 implies that $\sigma_n = \sum_{k=0}^{n-1} |a_k| \cdot \sigma_k = i_n \cdot T_a$, where a_k are equal to the coefficients in Property 3. Further, $\sigma_n > \sigma_{n-1}$, since $i_n > i_{n-1}$.

Let us assume that the properties above hold true. We can then make use of them to solve the neighbor discovery problem, as defined by Inequality (1), as follows.

1) First, $\gamma_0, \gamma_1, ..., \gamma_{n_m}$ need to be computed. This computation is described in detail in Section IV-C.

2) Next, the entire range of possible initial offsets is subdivided into partitions with equal latencies. For each partition, the latency can be derived by identifying a set of γ-sequences

that leads to a coincidence of a packet and a corresponding scan window. This procedure is described in Section IV-D.

3) From the set of partitions with corresponding latencies, one can extract the global worst-case and mean latencies.

C. Computation of the γ - Parameter

This section is dedicated to the computation of γ_n. Property 3 implies that γ_n is defined by the absolute difference between T_s and the absolute value of a linear combination of lower-order γ-values, such that i_n (and hence σ_n) is minimized, under the constraint that $\gamma_n < \frac{1}{2}\gamma_{n-1}$. We in the following present a recursive scheme to compute γ_n given $\gamma_k,\ k \in 0..n-1$. First, we present the computation of the initial value γ_0 of this recursion.

1) Initial Values: For realizing a shrinkage or growth of γ_n time units every i_n advertising packets and j_n scan windows, the amount of time defined by $i_n \cdot T_a$ must be by γ_n time units shorter (for a shrinkage) or longer (for a growth) than $j_n \cdot T_s$. Hence, we have to identify the smallest i_n and a suitable j_n, such that $|i_n T_a - j_n T_s| = \gamma_n < \min(T_a, T_s)$ (cf. Property 2). If $T_a < T_s$, every single advertising packet is part of a growing γ-sequence with $i_0 = 1, j_0 = 0$, and it is $\gamma_0 = T_a$. If $T_a > T_s$, the γ-sequence of order 0 implies $i_n = 1$, since this is the lowest possible multiple of T_a. There are two possible values for j_n, such that Property 2 (i.e., $\gamma_0 < T_s$) is fulfilled. The first one is defined by the smallest multiple of T_s that exceeds T_a, which forms a shrinking sequence with $\gamma_{s,0}$. The other one is defined by the largest multiple of T_s that does not exceed T_a, which forms a growing Sequence with $\gamma_{g,0}$. It is

$$\gamma_{s,0} = \left\lceil \frac{T_a}{T_s} \right\rceil \cdot T_s - T_a,\ \ \gamma_{g,0} = T_a - \left\lfloor \frac{T_a}{T_s} \right\rfloor \cdot T_s. \qquad (5)$$

Though both $\gamma_{s,0}$ and $\gamma_{g,0}$ are valid values for γ_0, we choose $\gamma_0 = \min(\gamma_{g,0}, \gamma_{s,0})$, and set $m_0 = s$ or g, accordingly, because this will result in a lower highest order n_m. Next, we describe how to compute γ_n with $n > 0$.

2) Recursion Scheme: We in the following assume two sequences with $m_{n-1} = m_n = g$, as depicted in Figure 4, and restrict our description to this. We generalize it afterwards.

Let γ_{n-1} describe a certain offset-shrinkage or -growth (cf. Property 1). Since this shrinkage or growth applies for the offset between any pair of scan windows and advertising packets, also offsets that result from a previous shrinkage/growth are again shrunk or grown by γ_{n-1} time units, whenever the additional advertising and scan intervals pass. Hence, linear combinations $L_{\gamma_n} = \sum_{k=0}^{n} a_k \gamma_k$, with $a_k \geq 0$, if $m_k = g$ and $a_k < 0$, if $m_k = s$, will result in an effective growth or shrinkage of L_{γ_n} time units. Further, if $L_{\gamma_n} > 0$, we can form differences $L_{\gamma_n} - T_s$ and obtain an effective shrinkage

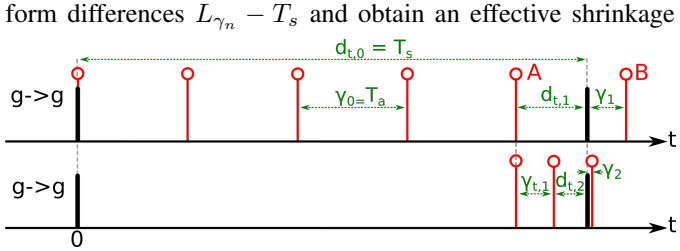

Fig. 4. Computation of γ_2 given γ_1 for growing sequences.

or growth $\gamma'_n = |L_{\gamma_n} - T_s|$. If L_{γ_n} involves i_n advertising intervals and j_n scan intervals, γ'_n can be regarded as the shrinkage (if $L_{\gamma_n} - T_s < 0$) or growth (if $L_{\gamma_n} - T_s > 0$) after every i_n advertising intervals and $j_n + 1$ scan intervals. Similarly, if $L_{\gamma_n} < 0$, $\gamma'_n = L_{\gamma_n} + T_s$ is the shrinkage (if $L_{\gamma_n} < -T_s$) or growth (if $L_{\gamma_n} > -T_s$) after i_n advertising and $j_n + 1$ scan intervals.

We can use this to construct $\gamma_n = |T_s - |\sum_{k=0}^{n-1} a_k \cdot \gamma_k||$, $a_k \in \mathbb{Z} \setminus 0$, by choosing a_k such that $|L_{\gamma_n}| = |\sum_{k=0}^{n-1} a_k \cdot \gamma_k|$ differs by less than $\frac{1}{2} \cdot \gamma_{n-1}$ time-units from T_s and hence, $\gamma_n < \frac{1}{2}\gamma_{n-1}$. There are different choices of a_k that fulfill $\gamma_n < \frac{1}{2} \cdot \gamma_{n-1}$, and we have to identify the one that leads to the lowest penalty $\sigma_n = \sum_{k=0}^{n-1} |a_k| \cdot \sigma_k$. Property 4 implies that shrinking or growing any offset by γ_n time-units always incurs at least the penalty σ_n of the linear combination $|\sum_{k=0}^{n-1} a_k \gamma_k|$ that i) reaches or exceeds $T_s - \frac{1}{2}\gamma_n$, and ii) has the lowest possible corresponding sum $\sum_{k=0}^{n-1} |a_k| \sigma_k$. Since we attempt to find the "best" (in terms of penalty) linear combination $|\sum_{k=0}^{n-1} a_k \gamma_k| > T_s - \frac{1}{2}\gamma_{n-1}$, Property 4 implies that one instance of γ_{n-1} in such a linear combination incurs already the penalty of the "best" linear combination that contains no instance of γ_{n-1}, if such a combination exists. As a result, coefficients a_k for higher sequence orders k always need to be minimized at the cost of larger coefficients a_k for lower orders k.

In the example depicted in Figure 4, γ_1 and γ_2 are computed given γ_0. Here, $m_1 = m_2 = g$, and therefore $L_{\gamma_{n-1}}$ must always exceed T_s. Hence, it is $\gamma_n = L_{\gamma_{n-1}} - T_s$. For computing γ_1, let us find the lowest multiple Q_0 of γ_0 time units, such that this multiple exceeds T_s. We can regard this as finding the number of γ_0 time-intervals that "fit" into a certain *distance to travel* $d_{t,0} = T_s$, i.e., $Q_0 = \lfloor \frac{d_{t,0}}{\gamma_0} \rfloor$. As can be seen in Figure 4, $\gamma_1 = (Q_0 + 1)\gamma_0 - T_s$, since this is the linear combination with the lowest sum of penalties (here: $(Q_0 + 1) \cdot \sigma_0$) that fulfills $\gamma_1 \leq \frac{1}{2}\gamma_0$.

For computing γ_2, we consider the largest multiple Q_0 of γ_0-intervals that does not exceed T_s. The remaining difference from T_s is $d_{t,1} = d_{t,0} - Q_0\gamma_0$. We now identify the largest multiple of γ_1 time units that "fit" into $d_{t,1}$, i.e., $Q_1 = \lfloor \frac{d_{t,1}}{\gamma_1} \rfloor$. Using this, $\gamma_2 = Q_0\gamma_0 + (Q_1 + 1)\gamma_1 - T_s$. Any linear combination that involves a higher number of γ_1 time units than Q_1 would violate Property 4. This scheme can be generalized for arbitrary orders n. With $Q_n = \lfloor \frac{d_{t,n}}{\gamma_n} \rfloor$, it is

$$\begin{aligned} \gamma_n &= (Q_{n-1} + 1) \cdot \gamma_{n-1} - d_{t,n-1}, \\ d_{t,n} &= d_{t,n-1} - Q_{n-1} \cdot \gamma_{n-1}, \\ \sigma_n &= \sigma_{s,n-1} + (Q_{n-1} + 1) \cdot \sigma_{n-1}, \\ \sigma_{s,n} &= \sigma_{s,n-1} + Q_{n-1} \cdot \sigma_{n-1}. \end{aligned} \qquad (6)$$

The first distance to travel $d_{t,0}$ is initialized by T_s, $\sigma_{s,0}$ is initialized by 0 and σ_0 by T_a. Here, $\sigma_{s,n}$ is the penalty (i.e., sum of advertising intervals) for any linear combination $\sum_{k=0}^{n-1} a_k \gamma_k$ that is by $d_{t,n}$ smaller than T_s.

In fact, one can show that Equation 6 holds true whenever $m_n = m_{n-1}$ (i.e., also for $m_n = s = m_{n-1}$). Similar considerations also exist for $m_n \neq m_{n-1}$, which lead to slightly different equations. The interested reader may refer to Appendix A. Depending on m_n and m_{n-1}, one out of two

TABLE I
RECURSIVE DEFINITION OF THE SEQUENCE MODE m_n GIVEN m_{n-1}

$d_{t,n-1} - \lfloor \frac{d_{t,n-1}}{\gamma_{n-1}} \rfloor$	$< 1/2 \cdot \gamma_{n-1}$	$> 1/2 \cdot \gamma_{n-1}$	$= 1/2 \cdot \gamma_{n-1}$
$m_{n-1} = s$	g	s	c
$m_{n-1} = g$	s	g	c
$m_{n-1} = c$	-	-	-

sets of equations has to be applied. This requires the mode m_n, which can be computed given γ_{n-1}, m_{n-1} and d_{t,n_1}, as defined in Table I.

D. Latency Computation

The first packet sent by a device that comes into range of a second, scanning device might fall into any part of its scan interval with an equal probability. Hence, the valid range of initial offsets between a scan window and an advertising packet is restricted to $\Phi[0] \in [0, T_s]$. For computing the discovery latency for a given $\Phi[0]$, we have to identify the linear combination $L' = \sum_{k=0}^{n_m} a_k \cdot \gamma_k$ that either fulfills ($-d_s \leq \Phi[0] + L' \leq 0$, or $T_s - d_s \leq \Phi[0] + L' \leq T_s$, and at the same time minimizes the corresponding sum of penalties $\sum_{k=0}^{n_m} a_k \cdot \sigma_n$. Minimizing the sum of penalties is required, because a linear combination with a higher sum of penalties would relate to a later coincidence of an advertising packet and a scan window, whereas the discovery latency is given by the first rendezvous. The intuition behind the latency computation procedure, along with an example, is given here. It is followed by the full algorithm in Section V.

As already explained, a γ-sequence incurs a shrinkage or growth of the temporal offset ϕ between any scan window and any advertising packet after every i_n advertising and j_n scan intervals. Equivalently, one could describe this as *one* advertising packet being "shifted" by multiples of γ_n towards the left (if $m_n = s$) or the right (if $m_n = g$) within *one* scan-interval. By summing up the penalties σ_n of every "shift" of length γ_n, one can account for the fact that each shrinkage or growth actually occurs for later pairs of scan windows and advertising packets. With this representation, the problem of finding such linear combinations L can be solved by finding the "best" (in terms of penalties) linear combination of "shifts" of length γ_n, until the advertising packet reaches either the scan window on the left or on the right of it. Again, coefficients a_k for high orders k always have to be minimized at the cost of lower-order ones, as proven in Appendix C-F.

1) First, we create a set of partitions, such that the corresponding scan window (i.e., the left one for $m_n = s$ or the right one for $m_n = g$) is either reached or "overtaken" after the same multiple of γ_0 time units from within each partition.

2) Each such partition is then subdivided in two other partitions. One of them contains all initial offsets from which the scan window can be reached (without "overtaking") after a multiple of γ_n time units. We call them *complete* partitions, since they do not need to be processed further, because discovery occurs after the corresponding multiple of γ_0 time units. The other ones contain the remaining offsets.

3) The partitions with the remaining offsets need to be subpartitioned again, such that after the same number of γ_1 time units, either the closest scan window or complete partition on the left (if $m_1 = s$) or right (if $m_1 = g$ is reached) or "overtaken".

4) Steps 2) and 3) are repeated for higher orders n, until $\gamma_n < d_s$.

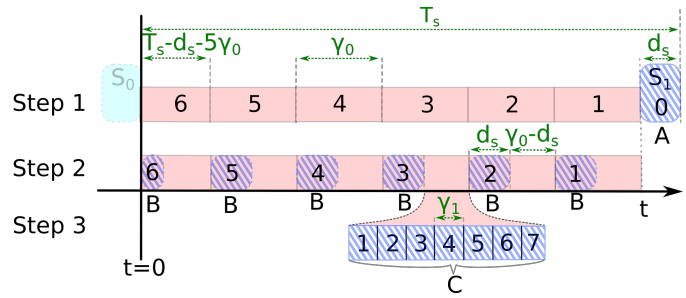

Fig. 5. Computing the mean discovery latency for a higher order sequence.

In the following, we illustrate this scheme using an example. Figure 5 exemplifies a partitioning of the entire range of initial offsets for a growing sequence with parameter γ_0 and a shrinking sequence with γ_1. In Figure 5, the hatched rounded rectangles depict the complete partitions, whereas the remaining rectangles depict temporary partitioning steps that need to be further subdivided later. In **Step 1** (cf. Figure 5), we subdivide the entire range $[0, T_s]$ into partitions of possible valuations of initial offsets, from which the scan window S_1 can be reached or "overtaken" using same multiple of γ_0 time units (i.e., the number of γ_0-intervals represented by the digits in each rectangle in Figure 5). A special case of this partitioning scheme is the scan window S_1 itself (cf. **A** in Figure 5)), since an an advertising packet falling into this scan window has zero latency.

In **Step 2**, each of these partitions is further subdivided into parts from which a multiple of γ_1 time units can reach the scan window S_1 (i.e., the hatched boxes in Figure 5), and the remaining parts, from which S_1 would be "overtaken". For the hatched parts denoted with **B** in the figure, no further partitioning is needed, since the discovery latency for an initial offset falling into such a part is defined by the number of γ_0-intervals until reaching the scan window S_1. These partitions are called *complete*.

However, the non-hatched parts from Step 2, which would "overtake" S_1, need to be further subdivided in **Step 3**. Each of them is subdivided into multiple partitions of length γ_1 (except for the leftmost one, which is shortened to fit within the $\gamma_0 - d_s$ time units of the containing partition obtained from Step 2). These partitions are marked with **C** in the figure. In each of these partitions, the number of γ_1 - steps a packet has to be "shifted" by to reach its closest complete partition (or S_0 from the leftmost partition, respectively) is equal. Once a complete partition has been reached (or S_0 from the leftmost partition), S_1 is approached in steps of γ_0, as described before. Since $\gamma_1 < d_s$, no parts of these partitions can "overtake" the corresponding partition of type **B** (or the scan-window S_0, respectively). Therefore, the partitioning process is complete.

The discovery latency of any partition can be computed by summing up the appropriate penalties. For every partition, we can count the number of γ_n - intervals until reaching the scan-window (S_1 in Figure 5). For example, from a partition of type

C, we count the number of γ_1-intervals n_1 until reaching a partition of type **B**. From there, we count the number of γ_0-intervals n_0 until reaching the scan window S_1. The digits in each partition in Figure 5 correspond to n_0 and n_1. The discovery latency d_{nd} is then $n_2 \cdot \sigma_2 + n_1 \cdot \sigma_1$.

This scheme, together with Property 5, also implies that a scan window can always be reached within a finite number of steps. Therefore, the worst-case latency is always bounded, if $m_n \neq c \; \forall n < n_m$.

V. Implementation of the Model

A straight-forward but computationally complex approach for computing the mean and worst-case neighbor discovery latencies would be iterating over every single partition and computing the latencies for each of them, as outlined in the previous section. However, the computation can be done more efficiently, without carrying out the entire partitioning explicitly. In what follows, we present an efficient algorithm for computing the discovery latency, which is based on two main concepts: 1) Discrete summations: Instead of iterating over every single partition of a given sequence order, we account for multiple of them at once by appropriate discrete summations. 2) Separation of sequence orders: Rather than carrying out the entire partitioning for all γ-sequences, we first only perform the partitioning for a single sequence order. After each order, the results are merged, since the resulting latency distributions are identical for many of the processed partitions. Therefore, the computational complexity of the next iteration is reduced. We provide a ready-to-use MATLAB-implementation of this algorithm for download (cf. Appendix F).

The algorithm we propose handles probability distributions that consist of multiple probability values over piecewise constant time intervals. For this purpose, we define a data structure called a *probability buffer* Ξ. It is described next.

A. Probability Buffers

A *probability buffer* consists of a set of k probability densities $\Xi[k]$, which are called *segments*. Each of them is defined over a unique, disjoint time-interval, which consists of a start time $t_s[k]$ and an end time $t_e[k]$. The elements are sorted by their starting times $t_s[k]$ in an ascending order. We define $\|\Xi\|$ as the number of segments in the probability buffer Ξ. For each probability buffer, we define an operation *add*(t_{ss}, t_{ee}, p), which creates a new segment from t_{ss} to t_{ee} with probability p. Existing probability segments that lie at least partially within $[t_{ss}, t_{ee}]$ are split and their probabilities are added for the overlapping time duration. To add a segment from t_{ss} to t_{ee} with probability p, we introduce the following notation: $[t_{ss}, t_{ee}] \leftarrow p$.

B. Example

An example for a neighbor discovery problem that is solved by our proposed algorithm is shown in Figure 6. As in the previous example from Figure 5, there is one growing sequence of order 0 and a shrinking sequence of order 1.

In **Iteration 1** of our proposed Algorithm (upper part of Figure 6), the partitioning for the order-0 sequence, as already

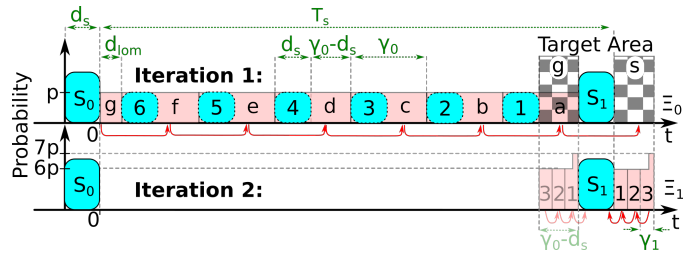

Fig. 6. Solution scheme of our proposed algorithm.

described, is carried out. Again, there are *hitting* partitions (1-6 in the figure), from which the scan window on the right (S_1) is reached after 1 to 6 steps of length γ_0. In addition, there are *missing* partitions (a-g), from which the scan window S_1 is "overtaken" after 1 to 7 steps of length γ_0. In this example, each hitting partition has a length of d_s, whereas each missing partition has length $\gamma_0 - d_s$, except for partition g, which is shortened to fit within $T_s - d_s$.

Since the first advertising packet will fall into every part of $[0, T_s]$ with an equal probability, we assign a probability density of $p = \frac{1}{T_s}$ to every partition that is created in Iteration 1. We can compose the mean discovery latency $\overline{d_{nd}}$ as a sum of multiple probability-weighted latencies $\sum_k d_{p,k} \cdot p_k \cdot (t_e[k] - t_s[k]) = \sum_k \overline{d_{p,k}}$. Here, $d_{p,k}$ is the discovery latency if the initial offset $\Phi[0]$ falls into a certain partition k. We call such a probability-weighted latency a *partial latency* $\overline{d_p}$. Each partial latency corresponds to a certain partition type (e.g., to all hitting partitions from Iteration 1, etc.)

In the Example from Figure 6, all hitting partitions (1-6) in Iteration 1 will incur a partial latency of $\overline{d_{p,0}} = d_s \cdot p \cdot \sum_{k=1}^{6} k \cdot \sigma_0$, with $p = \frac{1}{T_s}$, and the worst-case latency among all hitting parts is $6 \cdot \sigma_0$. An advertising packet that is sent within a missing partition (a-g) will "overtake" the scan window S_1 after the corresponding number of advertising intervals. After this overtaking has taken place, the subsequent advertising packet will be sent from one of the two *target areas* depicted in the figure, depending on the mode m_{n+1} of the next higher-order process. In our example, $m_{n+1} = s$, and the subsequent packet will be sent from target area s. This can be represented by "shifting" every missing partition by the appropriate multiple of γ_0 time-units, such that they lie within the target are s. Since multiple partitions are shifted into the same area, their probabilities have to be added. For the superposition of the partitions (a, b, c, d, e, f) in Figure 6, the resulting probability density will be $6 \cdot p$, since each of these partitions has a probability density of $p = \frac{1}{T_s}$. In addition, partition g will increase this merged probability density by $1 \cdot p$ within its corresponding part of the target area, as shown in the figure. The shifting procedure of all missing partitions into the target area s incurs a partial latency $\overline{d_{p,1}} = p \cdot \sigma_0 \cdot ((\gamma_0 - d_s) \cdot (1+2+3+5+6) + d_{lom} \cdot 7)$. Here, d_{lom} is the length of partition g. Similarly, the worst-case latency for a packet being shifted to target area s is $7\sigma_0$.

The resulting probability distribution within target area s is shown in the lower part of Figure 6, which is represented by the probability buffer Ξ_1. In **Iteration 2**, Ξ_1 is subdivided into multiple partitions, such that S_1 is reached (from target area s) after the same multiple of γ_1 time-units. Again, a

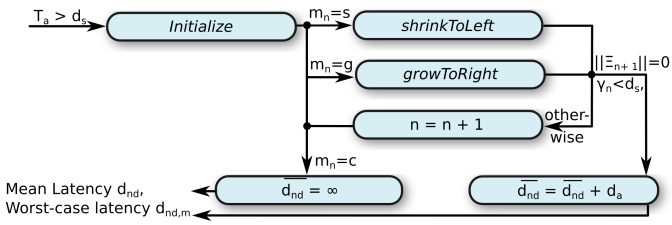

Fig. 7. Algorithmic solution to assess the discovery latency.

partial latency $\overline{d_{p,2}}$ for reaching the scan window from from the subpartitions (1,2,3) of Ξ_1 (cf. lower part of Figure 6) can be computed. Here, $\gamma_1 \leq d_s$, and the computation is complete. In other cases, Ξ_1 needs to be subpartitioned into hitting and missing parts again, and the procedure repeats.

As can be seen from the Figure, Ξ_1 can represent the entire probability distribution after Iteration 1 using only two segments (i.e., one within $[T_s, T_s + \gamma_0 - d_s - d_{lom}]$, and one within $[T_s + \gamma_0 - d_s - d_{lom}, T_s + \gamma_0 - d_s]$). In contrast, in the partitioning scheme described in Section IV-D, *every single* missing partition (e.g, all non-hatched partitions in Step 2 in Figure 5) need to be processed individually. Hence, this algorithm greatly reduces the computational complexity.

C. Algorithm Overview

A generic algorithm that corresponds to the described example is shown in Figure 7. A formal definition is given in Appendix E. The algorithm works as follows.

First, the *Initialize()*-function is called, which initializes the mean neighbor discovery latency $\overline{d_{nd}}$ by 0. Further, an initial probability buffer Ξ_0 is created. It contains one segment in $[0, T_s - d_s]$ with a probability of $\frac{1}{T_s}$. The case of the first packet being sent within $[T_s - d_s, d_s]$ is accounted for implicitly by setting the initial probability density to $\frac{1}{T_s}$.

Next, an iterative scheme begins. Depending on the mode m_n, either the *growToRight()*- or the *shrinkToLeft()*-function is called. If $m_n = c$, the algorithm is aborted with $\overline{d_{nd}} \leftarrow \infty$, instead. *growToRight()* performs the partitioning of its input probability distribution and the shifting to the appropriate target area, which results into the new probability buffer Ξ_{n+1} and multiple probability-weighted partial latencies $\overline{d_{p,k}}$. Further, it computes the largest sum of penalties d_m for every resulting partition as follows. If a partition to be processed has the largest sum of penalties of $d_{m,1}$, and the highest number of γ-intervals until reaching the next target area from this partition is n_i, then the resulting partition in Ξ_{n+1} will have the largest sum of penalties $d_{m,2} = d_{m,1} + n_i \cdot \sigma_n$ assigned to it. When multiple partitions are merged within one target area, the maximum value of d_m among the merged partitions is always assigned to the resulting partition. The *shrinkToLeft()*-function provides the equivalent functionality for shrinking sequences. After that, one out of two different steps is performed:

1) If $\gamma_n < d_s$ or Ξ_{m+1} is empty, then all existing partitions have been examined and $\overline{d_{nd}}$ has been computed. The algorithm terminates. The number of iterations until this occurs is always bounded (cf. Property 5).

2) In all other cases, n is increased and the scheme is repeated by another iteration.

The mean discovery latency is the sum of weighted partial latencies $\overline{d_{nd}} = E(d)$, and the worst-case latency $d_{nd,m}$ is the largest sum of penalties d_m in the last iteration of the algorithm. The values of γ_n, m_n and σ_n are computed at the beginning of every iteration. Finally, since we have assumed the packet length d_a to be 0, we add d_a to $\overline{d_{nd}}$ to account for the last, successful advertising packet sent. Next, we describe the *growToRight()*-function detail.

D. growToRight()

Based on a given probability buffer Ξ_n, which either results from the *Initialize()* - function or from a previous iteration of the algorithm, the *growToRight()*-function iterates through all segments of Ξ_n and processes each of them separately. This results into multiple partial latencies $\overline{d_p}$ and one resulting probability buffer Ξ_{n+1}.

We first split up each segment of the probability buffer Ξ_n into multiple partitions with constant values of N_{sh}. N_{sh} denotes the number of steps of length γ_n until reaching the right scan window (S_1 in Figure 6) or the appropriate target area (target area s for $m_{n+1} = s$ or target area g for $m_{n+1} = g$), respectively. Next, every such partition is then subdivided into a hitting and a missing part. For every hitting and missing part, a partial latency $\overline{d_p}$ for reaching the scan window is computed. In addition, every missing part contributes to a new segment in Ξ_{n+1}, which is obtained by shifting this missing part by $N_{sh} \cdot \gamma_n$ time-units.

If $m_n + 1 = g$, for any given point in time $t \in \Xi_n$, N_{sh} is given by $N_{sh} = \lfloor \frac{t_s - d_s - t}{\gamma_n} \rfloor$. For $m_{n+1} = s$, the next higher-order sequence shrinks and therefore the advertising packets will approach the temporally right scan window (S_1 in Figure 6) from the right target area (i.e., target area s in Figure 6). Further, the temporally right scan window in Ξ_n (i.e., S_1 in Figure 6) is regarded as the temporally left one in Ξ_{n+1}. This implies that the coordinate system of Ξ_{n+1} has its origin at the end of the temporally right scan window (S_1) of Ξ_n. Therefore, in addition to shifting every missing partition by $N_{sh} = \lceil \frac{Ts - d_s - t}{\gamma_n} \rceil$ steps of γ_n time-units towards the right, the partition has to be transformed into the new coordinate system. The shifting and the coordinate transformation of every point in time t within Ξ_n results into a point in time t' in $\Xi_n + 1$ of $t' = t - T_s + N_{sh}(t) \cdot \gamma_n$.

In what follows, we formally define the procedure of processing a segment $k \in \Xi_n$. In particular, based on multiple cases, we define the segments that need to be added to Ξ_{n+1} and the partial latencies $\overline{d_p}$ need to be summed up to obtain the mean discovery latency. We first compute the minimum (N_l) and maximum number (N_u) of γ-intervals that fit into the distance from every point of this segment to the right scan window, as defined by Equation (7).

$$N_l = \left\lceil \frac{T_s - d_s - t_e[k]}{\gamma_n} \right\rceil, \; N_u = \left\lfloor \frac{T_s - d_s - t_s[k]}{\gamma_n} \right\rfloor. \quad (7)$$

Figure 8 depicts a segment of Ξ_n and three possible cases for different values of N_l and N_u. The boxes with dotted rounded borders depict the hitting parts of this segment, whereas the remaining parts represent the missing ones. The

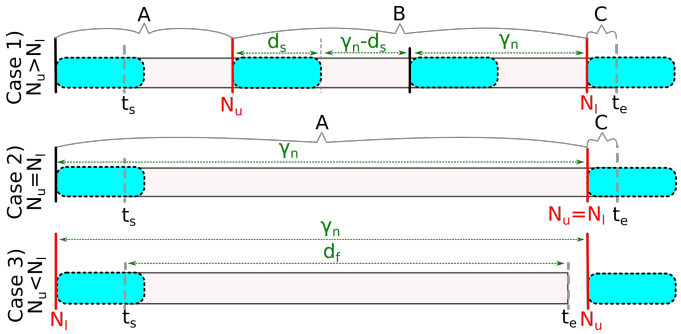

Fig. 8. Processing of one probability buffer segment $\Xi[k]$ for growing γ-sequences

figure also shows N_u and N_l and the borders t_s and t_e of the segment. We distinguish between 3 regions in each case: **A** (probability left of N_u), **B** (probability in between N_l and N_u) and **C** (probability right of N_l). The partial latency $\overline{d_p}$ of the considered segment $\Xi[k]$ is the sum of all three partial latencies of these regions, viz., $\overline{d_A} + \overline{d_B} + \overline{d_C}$. Further, d_{Nu} is the temporal length of the part left of N_u, if there is any. Similarly, d_{Nl} is the part right of N_l and d_f is the part in between. These values can be computed as follows.

$$\begin{aligned} d_{Nu} &= (T_s - d_s - N_u\gamma_n) - t_s[k]. \\ d_f &= t_e[k] - t_s[k]. \\ d_{Nl} &= t_e[k] - (T_s - d_s - N_l\gamma_n). \end{aligned} \tag{8}$$

Due to space constraints, we use a shorter notation and write p instead of $\Xi_n[k]$, l instead of $t_s[k]$ and r instead of $t_e[k]$. Then, the segments added to Ξ_{n+1} and the partial latencies are as follows.

Case 1) and Case 2): $\mathbf{N_u \geq N_l}$

For $m_{n+1} = s$ one has the following partial latencies:

$$\begin{aligned} \overline{d_A} &= p\sigma_n d_{Nu}(N_u + 1). \\ \overline{d_B} &= \begin{cases} 0, & \text{if } N_u = N_l, \\ p\sigma_n\gamma_n\frac{1}{2}\cdot \\ \cdot(N_u - N_l)(N_u + N_l + 1), & \text{else.} \end{cases} \\ \overline{d_C} &= p\sigma_n d_{Nl} N_l. \end{aligned} \tag{9}$$

For $m_{n+1} = s$, the following segments are added to Ξ_{n+1}:

$$\begin{array}{ll} [l - T_s + N_u\gamma_n + \gamma_n, \gamma_n - d_s] \leftarrow p, & \text{if } d_{Nu} < \gamma_n - d_s, \\ [0, \gamma_n - d_s] \leftarrow p, & \text{else.} \\ -, & \text{if } N_u = N_l, \\ [0, \gamma_n - d_s] \leftarrow p \cdot (N_u - N_l), & \text{else.} \\ -, & \text{if } d_{Nl} < d_s, \\ [0, r - T_s + N_l\gamma_n] \leftarrow p, & \text{else.} \end{array} \tag{10}$$

For $m_{n+1} = g$, the partial latencies are:

$$\begin{aligned} \overline{d_A} &= \begin{cases} p\sigma_n d_{Nu} N_u, & \text{if} \quad d_{Nu} < \gamma_n - d_s, \\ p\sigma_n((N_u + 1)\cdot \\ \cdot(d_{Nu} - \gamma_n + d_s))+ \\ +(\gamma_n - d_s)N_u), & \text{else.} \end{cases} \\ \overline{d_B} &= \begin{cases} 0, & \text{if} \quad N_u = N_l, \\ \frac{1}{2}p\sigma_n(N_u - N_l)\cdot(2d_s + \\ +\gamma_n \cdot (N_u + N_l - 1)), & \text{else.} \end{cases} \\ \overline{d_C} &= \begin{cases} p\sigma_n d_{Nl} N_l, & \text{if} \quad d_{Nl} < d_s, \\ p\sigma_n(d_s + d_{Nl}\cdot(N_l - 1)), & \text{else.} \end{cases} \end{aligned} \tag{11}$$

For $m_{n+1} = g$, the following probabilities are added to Ξ_{n+1}:

$$\begin{array}{ll} [l + N_u\gamma_n, T_s - d_s] \leftarrow p, & \text{if} \quad d_{Nu} < \gamma_n - d_s, \\ [T_s - \gamma_n, T_s - d_s] \leftarrow p, & \text{else.} \\ -, & \text{if} \quad N_u = N_l, \\ [T_s - \gamma_n, T_s - d_s] \leftarrow p \cdot (N_u - N_l), & \text{else.} \\ -, & \text{if} \quad d_{Nl} < d_s, \\ [T_s - \gamma_n, r + (N_l - 1)\gamma_n] \leftarrow p, & \text{else.} \end{array} \tag{12}$$

If $\gamma_n < d_s$, no segments are added to Ξ_{n+1} and the partial contribution $\overline{d_p}$ can be simplified (both for $m_{n+1} = g$ and $m_{n+1} = s$) to $\overline{d_p} = p\sigma_n \cdot (d_{Nl}N_l\frac{1}{2}\gamma_n(N_u - N_l)\cdot(N_u + N_l + 1) + d_{Nu}(N_u + 1))$.

Case 3): $\mathbf{N_u > N_l}$

From Figure 8 follows that this case has three subcases, depending on whether l, r or both lie within an area for which the current sequence matches, or not. For $l - (T_s - d_s - N_l\gamma_N) \leq d_s \wedge d_{Nl} \leq d_s$, both l and r lie within a matching area and therefore no probability is added to Ξ_{n+1}. The latency $\overline{d_p}$ (both for $m_{n+1} = s$ and g) is $pN_l\sigma_n(r - l)$. For $l - (T_s - d_s - N_l\gamma_N) \leq d_s \wedge d_{Nl} > d_s$, it is

$$\overline{d_p} = \begin{cases} p\sigma_n N_l(r - l), & \text{if} \quad m_{n+1} = s, \\ p\sigma_n(N_l(d_s - \gamma_n + d_{Nu})+ \\ +(N_l - 1)(d_{Nl} - d_s)), & \text{else,} \end{cases} \tag{13}$$

and the following segments are added to Ξ_{n+1}:

$$\begin{array}{ll} [0, r - T_s - N_l\gamma_n] \leftarrow p, & \text{if} \quad m_{n+1} = s, \\ [T_s - \gamma_n, r + (N_l - 1)\gamma_n] \leftarrow p, & \text{if} \quad m_{n+1} = g. \end{array} \tag{14}$$

Finally, for $l - (T_s - d_s - N_l\gamma_N) > d_s$, it is:

$$\begin{array}{ll} \overline{d_p} = p\sigma_n N_l(r - l). & \\ [l - T_s + N_l\gamma_n, r - T_s + N_l\gamma_n] \leftarrow p, & \text{if} \quad m_{n+1} = s, \\ [l + N_u\gamma_n, r + (N_l - 1)\gamma_n] \leftarrow p, & \text{if} \quad m_{n+1} = g. \end{array} \tag{15}$$

If $\gamma_n < d_s$, no segments are added to Ξ_{n+1} and Case 3) can be simplified to $\overline{d_p} = p(r - l)N_l\sigma_n$.

The worst-case latency is computed similarly to the mean latency by identifying the largest number of multiples of γ_n time-units until reaching either the appropriate target area (for missing partitions) or scan window (for hitting partitions) in each iteration. Sequences with $m_n = s$ are processed by the *shrinkToLeft()*-function, as described in Appendix B.

VI. Analysis and Evaluation

To evaluate the validity of our proposed theory, we compare its predicted latencies to measured latencies of $29,000$ discovery procedures, which have been carried out by two wireless radios. In addition, to cover a larger fraction of the whole parameter space, we have simulated $81,760,000$ discovery procedures and compare them to the values predicted by our theory. Towards this, we have carried out 7 different experiments. In Experiment a), we have measured latencies for sweeping advertising intervals and the fixed parameter values $T_s = 2.42\,\text{s}, d_s = 0.59\,\text{s}$. In addition, we have simulated the latencies for sweeping values

of T_a, while T_s and d_s have been set to the following values: b) $T_s = 2.56\,\mathrm{s}, d_s = 0.32\,\mathrm{s}$; c) $T_s = 2.56\,\mathrm{s}, d_s = 0.64\,\mathrm{s}$; d) $T_s = 7.68\,\mathrm{s}, d_s = 0.32\,\mathrm{s}$; e) $T_s = 2.56\,\mathrm{s}, d_s = 0.2015\,\mathrm{s}$, f) $T_s = 5.12\,\mathrm{s}, d_s = 0.32\,\mathrm{s}$. Further, to study the impact of different scan intervals, we present results obtained from our model for sweeping values of T_s with fixed values $T_a = 5.12\,\mathrm{s}, d_s = 0.64\,\mathrm{s}$ in Experiment g) (without comparing them to simulation results). The parameter values for these experiments have been chosen such that they study the impact of different values for T_a, T_s and d_s and such that the main characteristics of the latency curves can be examined easily. In addition, to allow for a quantitative comparison between ground-truth and computed values, we have calculated the maximum and root mean square deviations for these experiments. In particular, the following values are presented, which are based on the simulated or measured mean/max latencies d_{sim}/d_{mes} and the computed latencies d_{comp}.

- The Root Mean Square Error $\emptyset$) with

$$\emptyset = \frac{\sqrt{\sum_{T_{a,min}}^{T_{a,max}} (d_{comp}(T_a) - d_{sim}(T_a))^2}}{\|T_{a,max-} T_{a,min}\|}, \qquad (16)$$

  where the denominator is the number of data points with unique advertising intervals.
- The Normalized Root Mean Square Error κ with

$$\kappa = \frac{RMSE^2}{\max(d_{sim}) - \min(d_{sim})}. \qquad (17)$$

- The Maximum deviation $MD = \max(|d_{comp} - d_{sim}|)$.

A. Testbed Measurements

To compare the results of our proposed theory against real-world measurements, we conducted the following experiment. Based on the open-source BLE stack *BLESSED* [32], we have implemented the protocol studied in this paper. In particular, *BLESSED* implements the BLE protocol, but does not make use of additional random delays added to T_a. Advertising and scanning are carried out on one channel. This is exactly the problem described in this paper. We have created two different firmwares for two NRF51822 USB dongles. One device acts as the advertiser. First, it waits for a random amount of time between $[0, T_s]$ and then starts advertising with a certain interval T_a. The other firmware scans the channel with an interval $T_s = 2.42\,\mathrm{s}$ and a scan window $d_s = 0.59\,\mathrm{s}$ during all times. For every discovery procedure, the advertiser reports the point in time the advertising is started to a host PC via an USB connection. In the same way, the scanner reports the reception of a packet. With this experiment, the discovery latency can be measured directly. We have conducted the experiment in an anechoic chamber to avoid any interference from external devices (e.g. smartphones). To evaluate the accuracy of this setup, we have carried out $10,000$ additional measurements, in which we have measured a maximum loopback latency of $8\,\mathrm{ms}$ between the PC and the wireless node. This means that every measured discovery latency is increased by up to this amount of time. The advertising interval has been swept between $0.1\,\mathrm{s}$ and $3.0\,\mathrm{s}$ in steps of $0.01\,\mathrm{s}$. Each discovery procedure has been aborted after $60\,\mathrm{s}$, if no packet has been received within this period. For each advertising interval, the experiment has been repeated 100 times. The total number of discovery-procedures carried out has been $29,100$, which took about 43 hours of wall-clock time. Figure 9 depicts the upper latency bound $d_{nd,m}$ predicted by our proposed theory (solid line). In addition, the $29,100$ points in the figure show the measured discovery latencies of the wireless radios. As can be seen, our predicted bound is always reached, but never exceeded. Within latency peaks, most measurements have been aborted before successful discovery was achieved. The abortion took place after $60\,\mathrm{s}$ of wall-clock time, which explains the accumulation of latencies around $60\,\mathrm{s}$ within such areas.

TABLE II
COMPARISON OF COMPUTATION RESULTS AGAINST REAL-WORLD MEASUREMENTS (A) AND SIMULATIONS (B-F).

	$\overline{\emptyset}$	$\overline{\kappa}$	$\overline{MD}$	$\emptyset_m$	κ_m	MD_m
a)	0.55	0.933 %	3.62	1.12	1.908 %	7.32
b)	0.49	0.06 %	6.57	0.72	0.07 %	26.24
c)	0.35	0.05 %	6.35	0.41	0.04 %	25.62
d)	0.86	0.089 %	7.21	2.61	0.26 %	114.52
e)	0.61	0.066 %	9.02	0.66	0.067 %	38.60
f)	0.71	0.076 %	6.90	1.88	0.19 %	83.6

To further quantify the accuracy of our theory, among the 100 repetitions for each advertising interval, the mean and maximum discovery latencies have been determined. Figure 10a) shows the computed mean and worst-case latencies obtained from our theory, together with the measured values for a subset of the advertising intervals that have been realized. Each circle depicts the measured maximum latency out of 100 discovery procedures, and each cross the mean value. As can be seen, both for the mean and the worst-case latencies, the measurements lie in close proximity with our theory. Mean and maximum errors are given in Row a) of Table II. As can be seen, these errors are very small. Values with computed latencies higher than $60\,\mathrm{s}$ have been excluded from the computation of the errors, since they denote an aborted experiment, as already described. The computed upper latency bound has never been exceeded by more than $8.7\,\mathrm{ms}$ for any of the measured latencies. This error is caused mainly by the latency of the USB connection. The measurements fully confirm the validity of our proposed theory.

B. Discrete-Event Simulations

The high amount of wall-clock time required for the measurements limit both the number of repetitions for each advertising interval, as well as the number of examined parameter values. Therefore, we complemented the measurements by comprehensive simulation results. For this purpose, we have developed a custom discrete-event simulator in C++,

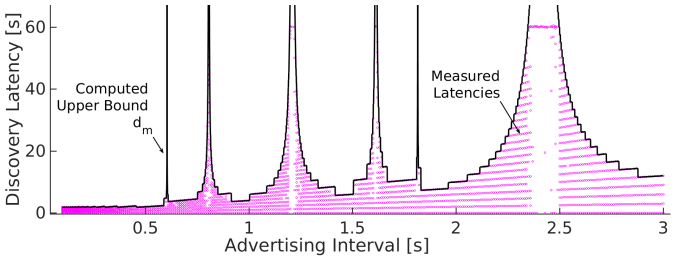

Fig. 9. Measured latencies (circles) and computed upper bound (solid line) for Experiment a) ($T_s = 2.42s$ and $d_s = 0.59s$).

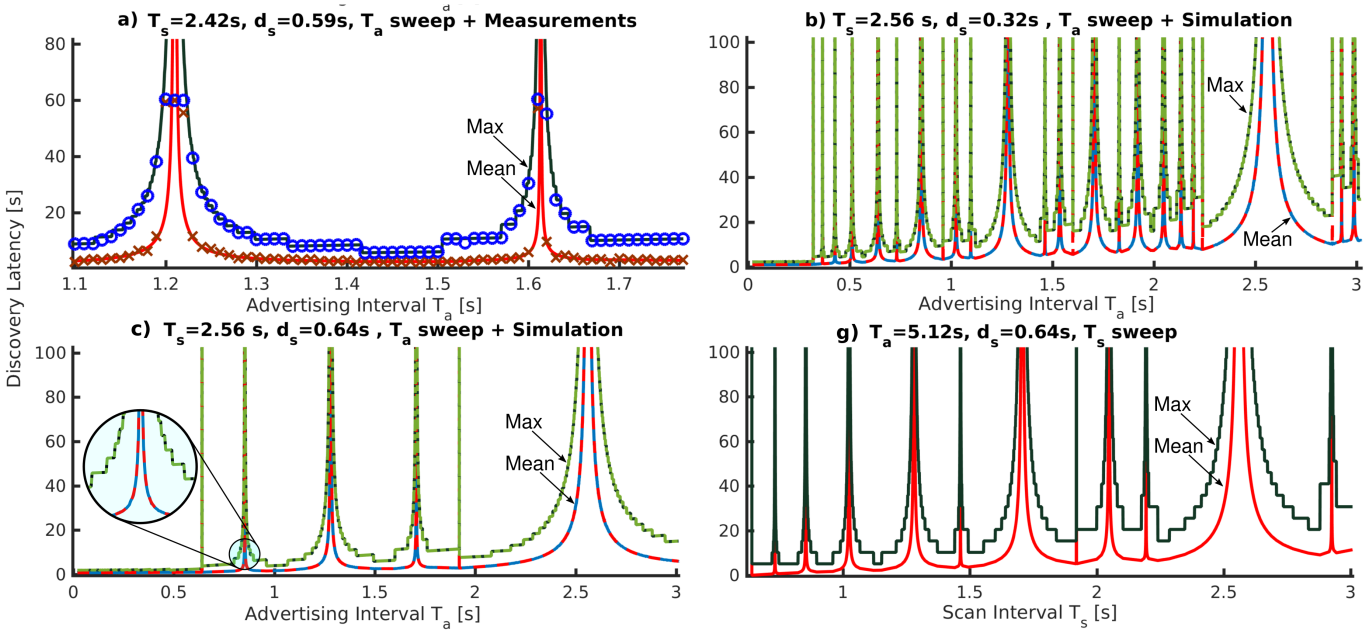

Fig. 10. a): Measured (solid line) and computed (dashed line) mean and maximum discovery latencies. b), c): Simulated (solid lines) and computed (dashed lines) mean and maximum discovery latencies. Dashed and solid lines lie in such a close proximity that no difference is visible, indicating a low error between simulated and computed results. Figure g) shows the computed latencies for a fixed value of T_a and varying values of T_s.

which carries out the PI-based discovery-process considered in this paper. The initial offset $\Phi[0]$ has been determined by an uniformly distributed random number generator in each simulation. The values of T_s and d_s have remained constant in each of our simulation experiments, while T_a has been varied in steps of $1.25\,\mathrm{ms}$ within $[10\,\mathrm{ms}, 10.24\,\mathrm{s} - 1.25\,\mathrm{ms}]$. The packet length d_a has been set to $248\,\mu\mathrm{s}$, which is a realistic value for protocols such as BLE. To determine the mean latency $\overline{d_{nd}}$ and the maximum latency $d_{nd,m}$, each simulation has been repeated $10,000$ times for every value of T_a and the mean- and maximum latencies have been computed out of the $10,000$ instances. The total number of simulations executed for each experiment has been $16,352,000$, which took more than a month of CPU time per experiment. Each single simulation has been aborted after $1000\,\mathrm{s}$ of simulated time, even if the computed latencies have exceeded this limit.

The Figures 10 b) and c) show the simulated latencies for experiments b) and c) together with the latencies computed by our proposed algorithm for a subset of the advertising intervals considered. In addition, the latencies predicted by our theory for Parametrization g), in which T_s instead of T_a is varied, are shown. The simulated curves are drawn using solid lines, whereas the curves obtained by our algorithm are drawn on top of them using dashed lines. As can be seen, both curves lie in such close proximity that no difference is visible in the figure. The only significant deviations from the simulation results exist at values of T_a at which the maximum latency has exceeded the maximum simulated time. Since each simulation has been aborted at $1000\,\mathrm{s}$ of simulated time, this behavior is expected.

To carry out a fair numerical comparison, we have excluded all values where the computed maximum latencies lie above $90\,\% \cdot 1000\,\mathrm{s}$ ($1000\,\mathrm{s}$ is the maximum simulated time). The numerical results of this comparison are given in Rows b)-f) of Table II. It can be observed that the root mean square errors $\emptyset$ are negligibly low. This indicates a good match of our proposed theory. Since the maximum predicted latency might not have been reached by any of the $10,000$ simulation repetitions carried out for each value of T_a, the maximum errors are higher, in particular for the worst-case latency MD_m. Our predicted upper bound has never been exceeded by any simulated value by more than $10 \times 10^{-10}\,\mathrm{s}$, which we considered as the floating point accuracy. Overall, these results fully confirm our proposed theory.

C. Discussion of the Results

The Figures 10 b) and c) show that the mean discovery latency has a nearly linear relation to the advertising interval for $T_a \leq d_s$. For $T_a > d_s$, both $\overline{d_{nd}}(T_a)$ and $d_{nd,m}(T_a)$ are composed of multiple peaks and minima. If T_s is increased or d_s is decreased, the number of peaks for different values of T_a is increased and these peaks tend to become narrower and steeper. Whereas the mean discovery latency is - except for some singularities - a smooth curve, the figures show (i.e., in the magnifying glass in Figure 10c)) that the maximum discovery latency is composed of multiple straight lines with points of discontinuity separating them from each other. The situation shown in Figure 10 g), in which T_s is varied, has similar properties. Figure 10 reveals interesting results on the behavior of the neighbor discovery latency for different values of T_a. First, we can observe that, except for the already mentioned singularities with $m_n = c$, the maximum latency is bounded for all values of T_a, T_s and d_s. To the best of our knowledge, this fact was previously unknown. It is now clear that interval-based protocols can guarantee deterministic latencies in general. Moreover, it shows that the advertising interval (and hence the duty-cycle) can be greatly increased without increasing the discovery latency by more than a nearly linear relation, if beneficial parametrizations are selected. The

highly irregular curves also underpin the need for systematic parameter optimizations in PI-based networks.

Further, new protocols can be developed based on our theory. Such protocols can rely on one or more pairs of optimized advertising and scan intervals. These intervals do not necessarily have to be constant. As an example, for BLE, strategies that successively reduce the advertising interval (e.g., by half after each interval) have been proposed [24] and can now be optimized systematically using our theory.

Using our proposed solution, unsynchronized continuous data transmission with duty-cycled, PI-based protocols (also referred to as continuous broadcasting, e.g. in the context of BLE beacons) becomes feasible, since the broadcaster and the receivers can be configured such that the number of packet repetitions is optimized and a deterministic reception is guaranteed. Since it is now known after how many packets a reception takes place in the worst-case, finding optimal configurations is straight-forward.

D. Comparison to Existing Models

In this section, we compare our proposed theory to existing models for BLE. We assume the same parametrization as for case b). For a fair comparison, we have implemented modified versions of these models, such that they describe the ideal PI-based problem, rather than the 3-channel discovery procedure of BLE. We have considered the model by Liu et al. [23], [24], [25], which is identical to the model from [15] when considering single-channel discovery. Besides our proposed theory, this model is the only one which predicts upper bounds (for $T_a < d_s$, only). Further, we have considered the predicted mean latencies of the model by Cho et al. [27], [28] and the accelerated simulation model by Kindt et al. [26], with $\Delta = 93.12\,\mathrm{s}$ and $\epsilon = 0.99$ into this comparison.

To compare to which extent existing models approximate the BLE protocol, we have conducted discrete-event simulations of the complete BLE discovery procedure, with 1000 repetitions per value of T_a. For this purpose, the simulator used for the evaluation of our model has been extended to account for 1) three channel discovery and 2) a random delay $\rho \in [0\,\mathrm{s}, 10\,\mathrm{ms}]$ being added to each advertising interval. As already described, the outputs of the simulator for the ideal PI discovery procedure are in line with the results obtained by our model, which are verified against real-world measurements (cf. Section VI-A). We have assumed an effective advertising interval $T_a' = T_a + 0.005\,\mathrm{s}$ to account for the mean shift of BLE's random delay. This increases the accuracy of all models considered.

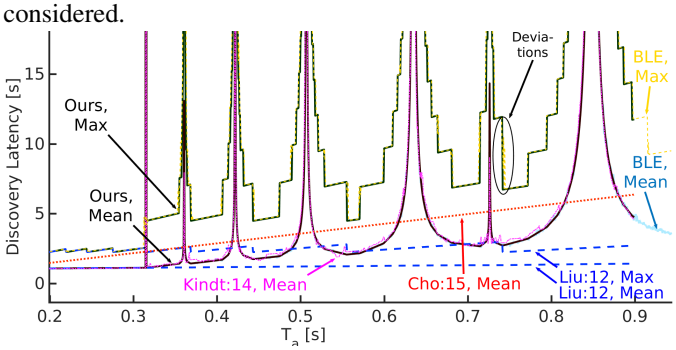

Fig. 11. Comparison to the models Liu:12 [23], [24], [25], [15], Cho:15 ([27], [28]), Kindt:13 [26], our proposed model and a simulation of BLE.

Comparison with other models: As shown in Figure 11, the model by Liu [23], [24], [25], [15] matches well for $T_a < d_s$, but is not valid for $T_a > d_s$. It can be seen from the figure that the model by Cho [27], [28] is unable to predict any latency peaks, due to the assumption of independent overlap probabilities. The simulation model by Kindt [26] models the overall trend, but is subjected to significant statistical variations.

Comparison with BLE: The results of this comparison are shown in Figure 11. As can be seen, our proposed model approximates the properties of BLE well. Small deviations of the predicted upper latency bound at border values can be observed e.g. around $T_a \approx 0.74s$, which are caused by the random delay and the multichannel discovery. While our model provides, to the best of or knowledge, the most accurate latency estimations for the BLE protocol that are available, it needs to be mentioned that for some parametrizations, the random delay could cause significant deviations from our predicted latencies. Studying the random delay and three-channel discovery is left open for future research.

E. Computational Complexity

In this section, we evaluate the computational complexity of Algorithm 1. Our aim is to establish that it has reasonable computational overhead and can therefore be used to compute a large number of latency values in short amounts of time.

The algorithm processes each segment of a probability buffer Ξ_n, evaluates the corresponding equations and adds new probability segments to the next probability buffer Ξ_{n+1}. The computational complexity for processing any order n is therefore determined by the number of segments in Ξ_n. Whereas the initial probability buffer contains only one segment, a conservative assumption is that each additional sequence-order $n \leftarrow n+1$ potentially triples the number of entries in Ξ_{n+1}, since each segment of Ξ_n might be subdivided twice in the worst-case. Therefore, the complexity grows exponentially with larger maximum orders n_m. However, the maximum order is bounded, and the bound grows only logarithmically with the smaller interval of T_a and T_s, as defined by Equation (4). Because practical problems (e.g., BLE) have limited maximum orders, the expected computational demands are low.

For a detailed evaluation, we have measured the computation time of a MATLAB-implementation using the tic/toc profiling mechanism of MATLAB [33]. The timing measurements were taken on a Lenovo Thinkpad X240 laptop with MATLAB R2015b under Linux Mint 17.2, using a single CPU core. For different values of d_s and T_s, we ran our proposed algorithm for all advertising intervals between $[20\,\mathrm{ms}, 10.24\,\mathrm{s}]$ in steps of $650\,\mu\mathrm{s}$, which is equivalent to the whole range of advertising intervals allowed in BLE. For each of these $16,353$ executions of Algorithm 1, the execution time was measured. The mean time the algorithm took until termination among the $16,535$ executions is shown in Figure 12 for multiple scan intervals and windows. The plot also includes the extreme case for BLE, with $d_s = 650\,\mu\mathrm{s}$ and $T_s = 10.24\,\mathrm{s}$. This parametrization leads to the maximum theoretically possible order of $n_m = 14$ for BLE[3]. The highest computation time reached has been

[3]During this experiment, the maximum problem order that actually appeared was 10.

1.8 ms for $d_s = 650\,\mu s$ and $T_s = 10.24\,s$. The results show that our proposed algorithm is computationally cheap in real-world situations and multiple orders of magnitude faster than discrete-event simulations. Recall that for the curves in Figure 10, more than a month of computation time has been spent for each simulation.

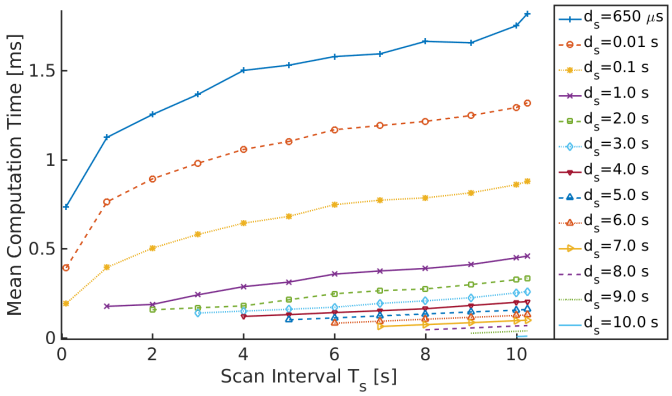

Fig. 12. Mean computation time per advertising interval for computing the mean- and maximum latencies for all advertising intervals $T_a \in [0.2\,s, 10.24\,s]$ in steps of $650\,\mu s$.

VII. Parameter Selection

In this section, we briefly describe how our proposed theory can be used to optimize the parametrization of PI-based discovery approaches. As shown in Section VI-E, Algorithm 1 is computationally cheap and therefore well suited to be executed repeatedly for a large set of parameters. For example, for a fixed tuple of values (T_s, d_s), all possible values of T_a allowed by the BLE specification [1] can be computed within a few seconds. With simulations, this is not possible due to the long simulation times and the limited accuracies. Therefore, for optimizing the parameters, we propose performing a design-space exploration based on our model. The optimization goal is specific to the application and is typically one out of the following possibilities:

1) Minimize the mean/maximum discovery latency as the first optimization goal. Since the same latencies can be achieved with multiple parametrizations, a second optimization criterion can be chosen.

2) Minimize the expected mean/maximum energy-consumption spent for the discovery. Given the energy E_a for an advertising packet and E_s for one scan window, a good approximation of this energy is $\overline{E_{nd,a}} = E_a \cdot \frac{\overline{d_{nd}}}{T_a} d_a$ for the advertiser and $\overline{E_{nd,s}} = E_s \cdot \frac{\overline{d_{nd}}}{T_s} d_s$ for the scanner. When considering the maximum discovery latency $d_{nd,m}$ instead of $\overline{d_{nd}}$ and by setting $E_a = E_s = 1$, this equation approximates the power-latency-product defined in [9].

3) Minimize the joint energy consumption of both the advertiser and the scanner, such that their sum becomes minimal.

A design space exploration for minimizing the mean-power-latency product is shown in Figure 13. For a fixed scan window $d_s = 2.5s$, T_a and T_s have been varied in steps of $62.5ms$ to compute the latency-duty-cycle product $\frac{d_a}{T_a} \cdot \overline{d_{nd}}$ of the advertiser. In the figure, all latencies exceeding $1 \times 10^{10}\,s$ have been truncated to this value. As can be seen, there are multiple optimal points for a given scan interval, such that the advertising interval can be chosen within a large set of values without affecting the energy efficiency.

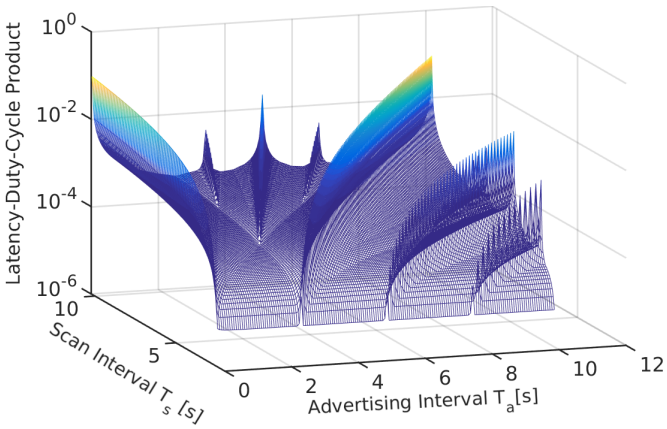

Fig. 13. Latency-duty-cycle product for a fixed scan window $d_s = 2.5s$.

VIII. Concluding Remarks

We have presented a new theory for computing the neighbor discovery latencies of periodic interval-based, slotless protocols. Our results imply two major novel insights, which have important implications.

First, the discovery latency of such protocols can now for the first time be computed accurately. Since PI-based concepts are widely used (e.g., for the BLE protocol and for ANT/ANT+), these protocols can now be configured in an optimized manner. Until now, the specifications of some wireless services recommend certain parameter ranges in which latency-peaks can occur. For example, the parametrization suggested by the BLE find-me profile [19] allows interval lengths that lie within such a peak, leading to long discovery latencies [26]. Therefore, such recommended parametrizations should be derived using our theory in the future.

Second, our results have revealed that the latency is bounded for almost all parameter values. This makes PI-based solutions applicable to networks with deterministic latency demands, where currently slotted protocols are being used. Periodic interval-based solutions provide the largest flexibility among all deterministic possibilities. Recent work based on this paper [20], [21] has revealed that with the same duty-cycle, PI-based protocols can achieve significantly shorter worst-case latencies. Also, novel protocols can be developed based on our theory, as described in Section VI-C. Our next step will be adopting the theory to commonly used protocols (such as BLE) by accounting for their unique peculiarities, such as multiple channels and random delays. Finally, with our results, we hope to motivate other researchers to work on slotless neighbor discovery protocols.

Acknowledgments

The authors would like to thank the anonymous reviewers and the editors for their constructive feedback on the earlier versions of this paper. This work was partially supported by *HE2mT - High-Level Development Methods for Energy-Saving, Mobile Telemonitoring Systems*, a project funded by the federal ministry of education and research of Germany (BMBF).

References

[1] Bluetooth SIG, "Specification of the Bluetooth system 4.0," June 2010, volume 0, available via bluetooth.org.

[2] D. Spenza, M. Magno, S. Basagni, M. Paoli, and C. Petroli, "Beyond duty cycling: Wake-up radio with selective awakening for long-lived wireless sensing systems," in *IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[3] M. Magno, V. Jelicic, B. Srbinovski, V. Bilas, E. Popovici, and L. Benini, "Design, implementation, and performance evaluation of a flexible low-latency nanowatt wake-up radio receiver," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 2, pp. 633–644, 2016.

[4] M. J. McGlynn and S. A. Borbash, "Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks," in *ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)*, 2001.

[5] R. Margolies, G. Grebla, T. Chen, D. Rubenstein, and G. Zussman, "Panda: Neighbor discovery on a power harvesting budget," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 34, no. 12, pp. 3606–3619, 2016.

[6] T. Chen, J. Ghaderi, D. Rubenstein, and G. Zussman, "Maximizing broadcast throughput under ultra-low-power constraints," in *ACM International on Conference on emerging Networking EXperiments and Technologies (CONEXT)*, 2016.

[7] P. Dutta and D. Culler, "Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications," in *ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2008.

[8] T. Herman, S. Pemmaraju, L. Pilard, and M. Mjelde, "Temporal partition in sensor networks," in *Stabilization, Safety, and Security of Distributed Systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, vol. 4838, pp. 325–339.

[9] A. Kandhalu, K. Lakshmanan, and R. Rajkumar, "U-Connect: A low-latency energy-efficient asynchronous neighbor discovery protocol," in *International Conference on Information Processingin Sensor Networks (IPSN)*, 2010.

[10] D. Zhang, T. He, Y. Liu, Y. Gu, F. Ye, R. K. Ganti, and H. Lei, "*Acc*: generic on-demand accelerations for neighbor discovery in mobile applications," in *ACM Conference on Embedded Network Sensor Systems, (SenSys)*, 2012.

[11] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2002.

[12] M. Bakht, M. Trower, and R. Kravets, "Searchlight: won't you be my neighbor?" in *Annual International Conference on Mobile Computing and Networking (MOBICOM)*, 2012.

[13] L. Chen, R. Fan, L. Chen, M. Gerla, T. Wang, and X. Li, "On heterogeneous neighbor discovery in wireless sensor networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[14] W. Sun, Z. Yang, W. Keyu, and L. Yunhao, "Hello: A generic flexible protocol for neighbor discovery," in *IEEE Conference on Computer Communications (INFOCOM)*, 2014.

[15] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Optimizing sensor networks in the energy-latency-density design space," *IEEE Transactions on Mobile Computing (TMC)*, vol. 1, no. 1, pp. 70–80, 2002.

[16] Dynastream Innovations Inc., "ANT message protocol and usage," 2014, revision 5.1, available via thisisant.com.

[17] Dynastream Innovations Inc., "Why ANT+: The facts," 2015, www.thisisant.com/business/why-ant/facts.

[18] Dynastream Innovations Inc., "ANT AN11: ANT channel search and background scanning channel," 2009, revision 2.1, available via thisisant.com.

[19] Bluetooth SIG, "Find me profile specificiation," June 2011, revision V10r00, available via bluetooth.org.

[20] P. Kindt, M. Saur, and S. Chakraborty, "Slotless protocols for fast and energy-efficient neighbor discovery," *CoRR*, vol. abs/1605.05614, 2016.

[21] P. Kindt, D. Yunge, G. Reinerth, and S. Chakraborty, "Griassdi: Mutually assisted slotless neighbor discovery," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2017.

[22] P. Kindt, N. Heitmann, G. Reinerth, and S. Chakraborty, "Understanding slotless neighbor discovery: Demo abstract," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2017.

[23] J. Liu, C. Chen, and Y. Ma, "Modeling neighbor discovery in bluetooth low energy networks," *IEEE Communications Letters*, vol. 16, no. 9, pp. 1439–1441, 2012.

[24] J. Liu, C. Chen, and Y. Ma, "Modeling and performance analysis of device discovery in bluetooth low energy networks," in *2012 IEEE Global Communications Conference (GLOBECOM)*, 2012.

[25] J. Liu and C. Chen, "Energy analysis of neighbor discovery in bluetooth low energy networks," Tech. Rep., 07/2012 2012, URL: research.nokia.com/files/public/TR-EnergyAnalysis-BLE.pdf.

[26] P. Kindt, D. Yunge, R. Diemer, and S. Chakraborty, "Precise energy modeling for the bluetooth low energy protocol," *arxiv.org*, 2013. [Online]. Available: http://arxiv.org/abs/1403.2919

[27] 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*, vol. 15, no. 1, pp. 59–78, 2015.

[28] K. Cho, G. Park, W. Cho, J. Seo, and K. Han, "Performance analysis of device discovery of bluetooth low energy (BLE) networks," *Computer Communications*, 2015.

[29] Y. Matsuo, D. Umehara, H. Murata, and S. Denno, "Performance evaluation of device discovery in bluetooth LE with probabilistic model checking," in *Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT)*, 2015.

[30] M. Kwiatkowska, N. G., and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *International Conference on Computer Aided Verification (CAV)*, ser. LNCS, vol. 6806. Springer, 2011, pp. 585–591.

[31] A. Kandhalu, A. Xhafa, and S. Hosur, "Towards bounded-latency bluetooth low energy for in-vehicle network cable replacement," in *International Conference on Connected Vehicles and Expo (ICCVE)*, 2013.

[32] Borges, P. et al., "Bluetooth low energy software stack for embedded devices (BLESSED)," 2015, https://github.com/pauloborges/blessed.

[33] Mathworks, Inc., "Documentation of the tic function," 2015, available via http://de.mathworks.com/help/matlab/ref/tic.html.

[34] O. Oystein, *Number Theory and its History*. McGraw-Hill, 1948.

Philipp H. Kindt received his Diploma in Electrical Engineering from Technical University of Munich (TUM) in 2012. He is currently working towards the Ph.D. degree in the area of low-power communication at Technical University of Munich (TUM). He has authored multiple peer-reviewed papers at conferences such as IEEE INFOCOM or ACM/IEEE IPSN. His research interests are wireless communication, microprocessors and mobile computing.

Marco Saur Marco Saur received the MS degree in Electrical Engineering and Information Technology from the Technical University of Munich (TUM) in 2016. He is currently working as a Software Engineer at Kinexon. His research interests include low-power wireless communication, mobile computing and embedded systems.

Michael Balszun received his Diploma in Electrical Engineering from Technical University of Munich (TUM) in 2011 and is currently a research assistant at the Chair of Real-Time Computer Systems. There, he is working towards a Ph.D. degree in the area of wireless controller design. His research interests include wireless real-time communication, robust control strategies and resource allocation schemes on embedded systems.

Samarjit Chakraborty is a Professor of Electrical Engineering at the Technical University of Munich (TUM), where the holds the Chair of Real-Time Computer Systems. Prior to joining TUM in 2008, he was an assistant professor of Computer Science at the National University of Singapore since 2003. He obtained his Ph.D. from ETH Zurich in 2003. His research interests include system-level design of embedded and cyber-physical systems, with applications in automotive, healthcare and sensor networks.

Appendix A
Recursive Computation of γ_n

In Section IV-C, we have described how to compute the value of the parameter γ_n. For sequences with orders $n > 1$, a recursive scheme to compute γ_n given γ_{n-1} has been presented for two consecutive growing sequences. Though it has been claimed that the equations presented also hold true for two consecutive shrinking processes, this case has not been explained. In this appendix, we first explain two consecutive shrinking sequences. Then, we describe recursion schemes for sequences with $m_n \neq m_{n+1}$. Again, the multiple Q_n is given by $Q_n = \lfloor \frac{d_{t,n}}{\gamma_n} \rfloor$.

Shrinking → Shrinking:

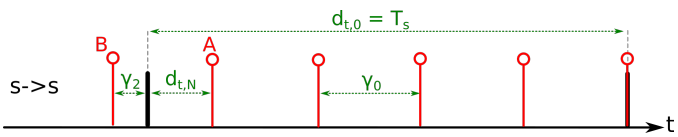

Fig. 14. Computation of γ_1 given γ_0 for $m_0 = m_1 = s$.

A situation with $m_0 = m_1 = s$ is shown in Figure 15. For $m_1 = m_2 = g$, we have formed the linear combination with the smallest possible sum $\sum_{k=0}^{n-1} a_k \cdot \sigma_k$ that exceeds T_s by no more than $\frac{1}{2}\gamma_0$. Here, since $m_0 = s$, the coefficient a_0 of the linear combination is negative and hence we form the negative combination of γ_0 time-units for which the absolute value exceeds T_s by no more than $\frac{1}{2}\gamma_0$ time-units. This also leads to Equation 6.

Growing → Shrinking and Shrinking → Growing:

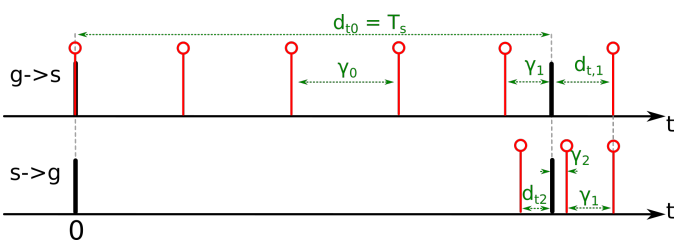

Fig. 15. Computation of γ_1 given γ_0 for $m_0 \neq m_1 \neq m_2$.

Figure 15 depicts a situation in which $m_0 = g$, $m_1 = s$ and $m_2 = g$. When comparing this to the situation in Figures 4 and 14, it becomes clear that when computing γ_n, a mode-change $m_{n-1} \neq m_n$ incurs one instance of γ_{n-1} fewer than for $m_n = m_{n-1}$. This leads to the following Equations, which hold true for $m_n = s$, $m_{n+1} = g$ and $m_n = g$, $m_{n+1} = s$.

$$\begin{aligned} \gamma_n &= d_{t,n-1} - Q_{n-1} \cdot \gamma_{n-1}. \\ d_{t,n} &= (Q_{n-1} + 1) \cdot \gamma_{n-1} - d_{t,n-1}. \\ \sigma_n &= \sigma_{s,n-1} + Q_{n-1} \cdot \sigma_{n-1}. \\ \sigma_{s,n} &= \sigma_{s,n-1} + (Q_{n-1} + 1) \cdot \sigma_{n-1}. \end{aligned} \tag{18}$$

Appendix B
shrinkToLeft() - Function

In Section V-D, we have presented the *growToRight()*-function, which grows a given distance Φ towards the temporally right scan window. Whereas *growToRight()* only works for growing sequences (i.e., $m_n = g$), *shrinkToLeft()* does the equivalent computation for shrinking sequences with $m_n = s$ of any order n, thereby reducing Φ ins steps of γ_n towards the temporally left scan window.

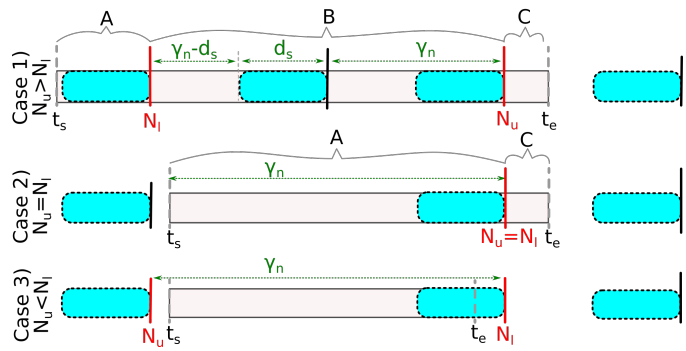

Fig. 16. Processing of one probability buffer segment $\Xi[k]$ for shrinking γ-sequences.

The partitioning of a segment for a shrinking sequence is depicted in Figure 16. The derivation of the equations is similar to the *growToRight()*-function and hence we only present the final results without further explanations. N_u and N_l are defined as

$$N_l = \left\lceil \frac{t_s[k]}{\gamma_n} \right\rceil, \quad N_u = \left\lfloor \frac{t_e[k]}{\gamma_n} \right\rfloor. \tag{19}$$

Due to space constraints, we use a shorter notation and write $p = \Xi_n[k]$, l instead of $t_s[k]$ and r instead of $t_e[k]$. The variables d_{Nl}, d_{Nu} and d_f are, in analogy to *growToRight()*:

$$d_{Nl} = N_l \gamma_n - l, \quad d_{Nu} = r - N_u \gamma_n, \quad d_f = r - l. \tag{20}$$

Like in *growToRight()*, there are three different cases.

Case 1) and Case 2): $\mathbf{N_u \geq N_l}$

For $m_{n+1} = s$, the partial latencies of the three parts **A**, **B** and **C** as depicted in Figure 16 are defined as follows.

$$\begin{aligned} \overline{d_A} &= \begin{cases} p\sigma_n N_l d_{Nl}, & \text{if} \quad d_{Nl} < d_s, \\ p\sigma_n(d_s N_l + \\ +(d_{Nl} - d_s)(N_l - 1)), & \text{else.} \end{cases} \\ \overline{d_B} &= \begin{cases} 0, & \text{if} \quad N_u = N_l, \\ \frac{1}{2} p\sigma_n (N_u - N_l) \cdot \\ \cdot (2d_s + \gamma_n(N_u + N_l - 1)), & \text{else.} \end{cases} \\ \overline{d_C} &= \begin{cases} p\sigma_n N_u d_{Nu}, & \text{if} \quad N_l\gamma_n - r \geq d_s, \\ p\sigma_n((d_{Nl} - d_s)(N_l - 1) + \\ +(d_s - \gamma_n + d_{Nu})N_l) \quad , & \text{else.} \end{cases} \end{aligned} \tag{21}$$

The following probability segments are added to Ξ_{n+1} for $m_{n+1} = s$.

$$\begin{aligned} &-, && \text{if} \quad d_{Nl} < d_s, \\ &[\gamma_n - d_{Nl}, \gamma_n - d_s] \leftarrow p, && \text{else.} \\ &-, && \text{if} \quad N_u = N_l, \\ &[0, \gamma_n - d_s] \leftarrow p \cdot (N_u - N_l), && \text{else.} \\ &[0, d_{Nu}] \leftarrow p, && \text{if} \quad d_{Nu} < \gamma_n - d_s, \\ &[0, \gamma_n - d_s] \leftarrow p, && \text{else.} \end{aligned} \tag{22}$$

For $m_{n+1} = g$, the following partial latencies $\overline{d_p}$ exist:

$$\begin{aligned}
\overline{d_A} =& p\sigma_n N_l d_{Nl}. \\
\overline{d_B} =& \begin{cases} 0, & \text{if} \quad N_u = N_l, \\ \frac{1}{2} p\sigma_n \gamma_n (N_u - N_l)\cdot & \\ \cdot(N_l + N_u + 1), & \text{else.} \end{cases} \\
\overline{d_C} =& p\sigma_n (N_u + 1) d_{Nu}.
\end{aligned} \tag{23}$$

The following probability segments are added to Ξ_{n+1} for $m_{n+1} = g$:

$$\begin{array}{ll}
-, & \text{if} \;\; d_{Nl} < d_s, \\
[T_s - d_{Nl}, T_s - d_s] \leftarrow p, & \text{else.} \\
-, & \text{if} \;\; N_u = N_l, \\
[T_s - \gamma_n, T_s - d_s] \leftarrow p \cdot (N_u - N_l), & \text{else.} \\
[T_s - \gamma_n, T_s - \gamma_n + d_{Nu}] \leftarrow p, & \text{if} \;\; d_{Nu} < \gamma_n - d_s, \\
[T_s - \gamma_n, T_s - d_s] \leftarrow p, & \text{else.}
\end{array} \tag{24}$$

If $\gamma_n < d_s$, then no probability is added to the resulting probability buffer and the partial latency contribution added to $\overline{d_{nd}}$ can be computed by

$$\begin{aligned}
\overline{d_p} \;\; = \;\; & p\sigma_n \cdot (d_{Nl} N_l \tfrac{1}{2} \gamma_n (N_u - N_l) \cdot (N_u + N_l + 1) + \\
& d_{Nu}(N_u + 1)).
\end{aligned} \tag{25}$$

Case 3): $\mathbf{N_u < N_l}$

Like in *growToRight()*, this case has three subcases. If $N_l \gamma_n - r \geq d_s$ holds true, the partial latency is (for both $m_{n+1} = s$ and $m_{n+1} = g$)

$$\overline{d_p} = p N_l \sigma_n d_f. \tag{26}$$

The following segments are added to Ξ_{n+1}:

$$\begin{array}{ll}
[\gamma_n - d_{Nl}, d_{Nu}] \leftarrow p, & \text{if} \quad m_{n+1} = s, \\
[T_s - d_{Nl}, T_s + d_{Nu} - \gamma_n] \leftarrow p, & \text{if} \quad m_{n+1} = g.
\end{array} \tag{27}$$

For $N_l \gamma_n - r < d_s \wedge d_{Nl} \geq d_s$, it is

$$\overline{d_p} = \begin{cases} p\sigma_n((d_{Nl} - d_s)(N_l - 1) + & \\ +N_l(d_s - \gamma_n + d_{Nu})), & \text{if} \quad m_{n+1} = s, \\ p\sigma_n N_l (r - l), & \text{if} \quad m_{n+1} = g. \end{cases} \tag{28}$$

The following segments are added to Ξ_{n+1}:

$$\begin{array}{ll}
[\gamma_n - d_{Nl}, \gamma_n - d_s] \leftarrow p, & \text{if} \quad m_{n+1} = s, \\
[T_s - d_{Nl}, T_s - d_s] \leftarrow p, & \text{if} \quad m_{n+1} = g.
\end{array} \tag{29}$$

If $N_l \gamma_n - r < d_s \wedge d_{Nl} < d_s$ holds true or if $\gamma_n \leq d_s$, no probability is added to the resulting buffer and the partial latency is $\overline{d_p} = p\sigma_n N_l d_f$.

Appendix C
Proofs of the Properties

In Section IV-B, we have defined γ-sequences and have introduced multiple properties, for which we claimed that they hold true for some (i_n, j_n). In this section, we proof that these properties can be fulfilled.

A. Property 1: Constant Shrinkage/Growth

Given an offset Φ between a scan window and an advertising packet, Property 1 implies that this offset will be by $\gamma_n \geq 0$ time-units larger or smaller for a scan window that is j_n scan intervals and an advertising packet that is i_n advertising intervals later.

Let us consider $i_n \cdot T_a$ and $j_n \cdot T_s$ time-units. Their difference is $|j_n \cdot T_s - i_n \cdot T_a| = \gamma_n$. Let t_s be an arbitrary point in time a scan window starts at, and t_a an arbitrary point in time an advertising packet is sent at, and $\Phi = t_s - t_a$. Then, for the offset Φ' that is i_n advertising intervals and j_n scan intervals later, it is

$$\begin{aligned}
\Phi' \;\; = \;\; & t_s + j_n \cdot T_s - (t_a + i_n \cdot T_a) = \\
= \;\; & t_s - t_a + (j_n \cdot T_s - i_n \cdot T_a) = \\
= \;\; & \Phi \pm \gamma \;\; \forall (t_a, t_s).
\end{aligned} \tag{30}$$

□

B. Property 2: Range of γ

Property 2 implies that there exist pairs (i_n, j_n), such that $0 \leq \gamma_n \leq \min(T_s, T_a)$. Let us first assume $T_s < T_a$. It is $\gamma = |j_n \cdot T_s - i_n \cdot T_a|$. We choose $j_n = \lceil \frac{i_n T_a}{T_s} \rceil$, and it is

$$\begin{aligned}
\gamma_n \;\; = \;\; & \left| \left\lceil \frac{i_n T_a}{T_s} \right\rceil T_s - i_n T_a \right| \\
\leq \;\; & \left| \left(\frac{i_n \cdot T_a}{T_s} + 1 \right) T_s - i_n \cdot T_a \right| \\
= \;\; & |i_n T_a + T_s - i_n T_a| \\
= \;\; & T_s
\end{aligned} \tag{31}$$

Similarly, if $T_s > T_a$, we can choose $j_n = \lceil \frac{i_n T_s}{T_a} \rceil$, which will result in $\gamma_n \leq T_a$. □

C. Property 3: Construction of γ

Property 3 claims that there are linear combinations $\gamma_n = |T_s - |\sum_{k=0}^{n-1} a_k \cdot \gamma_k||$, $a_k \in \mathbb{Z} \setminus 0$, which fulfill the following three properties.

- $\gamma_n \leq \frac{1}{2}\gamma_{n-1}$
- the sum $\sum_{k=0}^{n-1} |a_k| \cdot (i_k T_a)$ is minimized, and
- for a shrinking sequence, $a_k < 0$, otherwise, $a_k > 0$

A valid constructive algorithm for linear combinations that fulfill these properties has been presented in Section V, which proofs the existence of such linear combinations. □

D. Property 4: Priority of Lower Orders

This property claims that for any γ_n, $\sigma_n = i_n \cdot T_a$ is always at least as large as the smallest possible linear combination $\sum_{k=0}^{n-1} |a_k| \cdot \sigma_k$, for which $|\sum_{k=0}^{n-1} a_k \gamma_k| \geq T_s - \gamma_n \geq T_s - \frac{1}{2}\gamma_{n-1}$. The construction of γ_n is defined by $\gamma_n = |T_s - |\sum_{k=0}^{n-1} a_k \cdot \gamma_k||$, $a_k \in \mathbb{Z} \setminus 0$, such that i) $\gamma_n < \frac{1}{2}\gamma_{n-1}$ and ii) the corresponding penalty σ_n is minimized. The necessary condition $\gamma_n = |T_s - |\sum_{k=0}^{n-1} a_k \cdot \gamma_k|| < \frac{1}{2}\gamma_{n-1}$ implies that $|\sum_{k=0}^{n-1} a_k \cdot \gamma_k| \geq T_s - \frac{1}{2} \cdot \gamma_{n-1}$. This sum has a corresponding sum of penalties $\sum_{k=0}^{n-1} |a_k| \cdot \sigma_k$. Since $\gamma_n = |T_s - |\sum_{k=0}^{n-1} a_k \cdot \gamma_k||$, it is $|\sum_{k=0}^{n-1} a_k \cdot \gamma_k| \geq T_s - \gamma_n$. □

E. Property 5: Highest Possible Order

The maximum number of iterations of Algorithm 1 is identical to the maximum order n_m. In the following, we proof that the maximum order is bounded, as described by Property 5.

Property 3 implies that $\gamma_n \leq \frac{1}{2}\gamma_{n-1}$. Hence, in the worst case, $\gamma_n = \frac{1}{2}\gamma_{n-1}\ \forall n$. Using this, we can define a worst-case γ-parameter for the order n, $\gamma_{wc,n}$ by assuming a maximum initial value γ_0 of $\min(T_a, T_s)$ (cf. Property 2). It is:

$$\gamma_{n,wc} = \left(\frac{1}{2}\right)^n \cdot \min(T_a, T_s). \tag{32}$$

In the worst-case, the highest-order sequence is the first one for which $\gamma_{n,wc} \leq d_s$ holds true. Hence, it is

$$\frac{1}{2}^{n_m} \cdot \min(T_a, T_s) \leq d_s. \tag{33}$$

Solving this inequality leads to a maximum order n_m, as given by Equation (34).

$$n_m = \left\lceil \frac{\ln(\min(T_a, T_s)) - \ln(d_s)}{\ln(2)} \right\rceil \tag{34}$$

In addition, it is intuitively clear that if $\min(T_a, T_s) \leq d_s$, only order-0-sequences can occur (the advertiser can never "overtake" the scanner) and therefore, $n_m = 0$. Further, we have assumed d_s to be shortened by d_a time-units, as described in Section IV, and the equation above is only valid for shortened values of d_s.

The limited number of sequence orders also implies that whenever $\gamma_n \neq 0\ \forall n \leq n_m$, the worst-case discovery latency is bounded. This is because the discovery latency is defined by the sum of γ-intervals that fit into the scan interval, which is always finite if $\gamma \neq 0$.

□

F. Proof of Latency Computation

In Section V, we form linear combinations $L' = \sum_{k=0}^{n} a_k \cdot \gamma_k$, such that a packet that is sent with a certain initial offset $\Phi[0]$ from the first scan window is shifted by L' towards the next scan window. We have claimed that coefficients a_k for high values of k always need to be minimized, no matter how by how much coefficients a_k for lower values of k need to be increased to compensate for this minimization. In what follows, we proof this claim.

Let us consider two linear combinations L'_1 and L'_2 that both fulfill $-d_s \leq \Phi[0] + L' \leq 0$, or $T_s - d_s \leq \Phi[0] + L' \leq T_s$, respectively, and hence lead to a coincidence of the advertising packet and a scan window. Further, $L'_1 = \sum_{k=0}^{n} a_{k,1} \cdot \gamma_k$, whereas $L'_2 = \sum_{k=0}^{n-1} a_{k,2} \cdot \gamma_k$. Further, $a_{k,1}$ and $a_{k,2}$ are chosen such that the corresponding sum of penalties is minimized. We now study whether L'_1 or L'_2 has a lower sum of penalties.

If $m_{n-1} = m_n$, then $\sum_{k=0}^{n-1} a_{k,2} \cdot \gamma_k - \sum_{k=0}^{n-1} a_{k,1} \cdot \gamma_k \leq \gamma_{n-1}$, since $\gamma_n \leq \frac{1}{2}\gamma_{n-1}$. This means that by having $1 \cdot \gamma_n$ in L'_1, at most one instance of γ_{n-1} can be saved. Because $\sigma_n \geq \sigma_{n-1}$, L'_2 always has a lower sum of penalties than L'_1.

If $m_{n-1} \neq m_n$, the initial offset $\Phi[0]$ could potentially form a situation in which L'_1 is defined by $1 \cdot \gamma_n$, and $L'_2 = \sum_{k=0}^{n-1} a_k \gamma_k \geq T_s - d_s$ in the worst-case. In other words, there might be situations in which either $1 \cdot \gamma_n$, or an appropriate number of γ_{n-1} time-units of length $T_s - d_s$ or longer, will lead to a match. This can be seen as the worst case. Such a situation with $\gamma_{n-1} = g$ and $\gamma_n = s$ is depicted in Figure 17. We now study which of both possibilities L'_1, L'_2 has a lower sum of corresponding penalties.

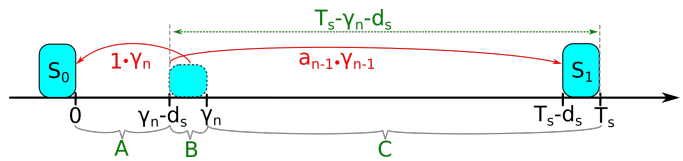

Fig. 17. Proof for Latency Computation

Here, the range of initial offsets $\Phi[0]$ between $\gamma_n - d_s$ and γ_n (Area **B** in Figure 17) forms the only case in which the left scan window S_0 can be reached by $1 \cdot \gamma_n$. From Property 4 follows that the corresponding penalty $1 \cdot \sigma_n$ will be at least the lowest possible sum of penalties that corresponds to a linear combination $|\sum_{k=0}^{n-1} a_k \cdot \gamma_k| \geq T_s - \gamma_n$. However, as can be seen from the figure, the temporal distance from the right scan window S_1 of the entire area **B** is no more than $T_s - d_s - \gamma_n < T_s - \gamma_n$ time-units. Hence, if there is a linear combination such that $\Phi[0] + \sum_{k=0}^{n-1} a_k \cdot \gamma_k$ lies within S_1, the penalty of σ_n will be at least as high as the sum of penalties $\sum_{k=0}^{n-1} a_k \cdot \sigma_k$. For the same reason, in area **C**, the latency of a suitable linear combination of $\sum_{k=0}^{n-1} a_k \cdot \gamma_k$ will be always lower than any linear combination that contains at least $1 \cdot \gamma_n$. Initial offsets $\Phi[0] < \gamma_n - d_s$ in area **A** can either be formed by $a_n \cdot \gamma_n + \sum_{k=0}^{n-1} a_k \gamma_n$, with $a_n \geq 1$ and $a_k > 0$ for at least some k, or by linear combinations without γ_n, i.e., $\sum_{k=0}^{n-1} a_k \cdot \gamma_k$. Clearly, $\sum_{k=0}^{n-1} a_k \cdot \gamma_k$ is the possibility with the lowest sum of penalties. □

Appendix D
Relation to the Euclidean Algorithm

Property 3 defines γ_n as $\gamma_n = |T_s - L_{\gamma_{n-1}}||$, where $L_{\gamma_{n-1}}$ is a linear combination $\sum_{k=0}^{n-1} a_k \cdot \gamma_k$. Let us consider two possibilities for $L_{\gamma_{n-1}}$, both assuming $m_0 = g$. The one, L_s, is the linear combination $\sum_{k=0}^{n-1} a_k \gamma_k$ that approximates T_s as close as possible, without exceeding it. The other one, $L_g = L_1 + \gamma_{n-1}$ slightly exceeds T_s, and $\min(|T_s - L_1|, |T_s - L_2|) < \frac{1}{2}\gamma_{n-1}$. If $L_s < L_g$, $m_n = s$ and $\gamma_n = T_s - L_s$. Otherwise, $m_n = g$ and $\gamma_n = L_g - T_s$. This shares similarities with the method of least absolute remainders, which the mathematician Leopold Kronecker has shown to be the most efficient version of the Euclidean algorithm [34]. For determining the next remainder (here: $|T_s - L_{\gamma_{n-1}}|$), rather than dividing the smallest previous reminder by its predecessor (as in the Euclidean algorithm), we divide the larger previous reminder (i.e., $d_{t,n-1}$) by the smaller one (i.e. γ_{n-1}.).

Appendix E
Definition of the Latency Computation Algorithm

The algorithm described in Section V is formally defined by Algorithm 1.

Algorithm 1 High-level algorithm for computing d_{nd}

Input: T_a, d_a, T_s, d_s
Output: $\overline{d_{nd}}$

$(\overline{d_{nd}}, \Xi_0, \vec{\gamma}, \vec{\sigma}, \vec{m}) \leftarrow Initialize(T_a, T_s, d_s)$
if $T_a \leq T_s$ **then**
  $n \leftarrow 0$
else
  $n \leftarrow 1$
end if
while $\gamma_n \geq d_s$ **do**
  if $m_n = g$ **then**
    $(\overline{d_{nd}}, \Xi_{n+1}) \leftarrow growToRight(\overline{d_{nd}}, \Xi_n, \gamma_n, \sigma_n, m_{n+1})$
  else if $m_n = s$ **then**
    $(\overline{d_{nd}}, \Xi_{n+1}) \leftarrow shrinkToLeft(\overline{d_{nd}}, \Xi_n, \gamma_n, \sigma_n, m_{n+1})$
  else
    $\overline{d_{nd}} \leftarrow \infty$; return
  end if
  if $(\|\Xi_n\| = \emptyset) \vee (\gamma_n < d_s)$ **then**
    break loop
  else
    $n \leftarrow n + 1$
  end if
end while
$\overline{d_{nd}} = \overline{d_{nd}} + d_a$

Appendix F
Software Library

Our proposed model can be downloaded as a MATLAB library from www.rcs.ei.tum.de/wireless_systems. The function

```
function [dAvg, dMin, dMax, o] =
getDiscoveryLatency(Ta,Ts,dA,ds,iL)
```

computes the mean- (*dAvg*), minimum- (*dMin*) and maximum- (*dMax*) discovery latency for a given parametrization. The parameters of this function are the advertising interval (*Ta*), the scan interval (*Ts*), the scan window (*ds*), the advertising packet-length (*dA*) and an iteration limit (*iL*) of the algorithm. The algorithm also returns the maximum sequence order (*o*) for the given parameters. An example for computing entire latency curves with sweeping parameters is available for download, too.

Appendix G
Table of Symbols

T_a	Advertising Interval
T_s	Scan Interval
d_s	Scan Window
d_a	Duration of an advertising packet
k	Counting index
n	Index for the order of a sequence
n_m	Maximum sequence order
h, i, j	Generic indices / integer multiples
$\mathbb{N}_0$	Set of all non-negative integers
Φ	Offset between an advertising packet and the closest neighboring scan window on its left side
Φ'	Offset between an advertising packet and the closest neighboring scan window on its right side
d_{nd}	Neighbor discovery latency
$\overline{d_{nd}}$	Mean neighbor discovery latency
$d_{nd,m}$	Max. neighbor discovery latency
γ	Growth/Shrinkage per interval
γ_s,γ_g	γ-parameter given the sequence is shrinking/growing
γ_{wc}	Worst-case value of γ
N_i	Number of intervals until reaching the next target area
m	Mode of a sequence (s $\leftarrow$ shrinking, g $\leftarrow$ growing, c $\leftarrow$ coupling)
d_t	Distance left to travel
σ, σ_s	Penalty related to γ_n / Penalty related to $d_{t,n}$
i_n, j_n	Multiplier of advertising (i_n) and scan intervals (j_n)
L_{γ_n}	Linear combination $\sum_{k=0}^{n} a_k \cdot \gamma_k$
a_k	Coefficient of a linear combination
Ξ	Probability buffer
$\|\Xi\|$	Number of segments in the probability buffer Ξ
$\Xi[k]$	Probability density of segment $k \in \Xi$
$t_s[k]$	Starting time of a segment $k \in \Xi$
$t_e[k]$	Ending time of a segment $k \in \Xi$
l,r	Abbreviations for $t_s[k]$ and $t_e[k]$
d_{lo}	Leftover duration (Duration of a partition which does not fit entirely in the partition of its previous-order sequence)
d_{loh}	Hitting duration of a leftover partition
d_{lom}	Missing duration of a leftover partition
$\overline{d_p}$	Weighted partial duration of a partition
p_i, d_i	Probability/Latency of a segment i of a probability buffer
N_{sh}	Number of intervals to shrink a certain distance
t/t'	Original/Transformed point in time
N_l, N_u	Minimum/Maximum number of γ-intervals until reaching a target area for a segment of Ξ
d_{Nl}, d_f, d_{Nu}	Length of the part of a segment that has $N_l - 1$/N_l...N_u/$N_u + 1$ intervals to travel until reaching a target area
$\zeta[k]$	Largest sum of penalties for a segment $k \in \Xi$
$d_{p,m}$	Maximum partial latency for a part of a segment
$\emptyset$, $\emptyset_m$	Root mean square error (RMSE) for mean/maximum latencies
κ, κ_m	Normalized root mean square error for mean/maximum latencies
MD, MD_m	Maximum deviation ($\max(\|d_{comp} - d_{sim}\|)$) for mean/maximum latencies
d_{comp}, d_{sim}	Computed/Simulated latency