BLEnd: Practical Continuous Neighbor Discovery for Bluetooth Low Energy

Christine Julien
The University of Texas at Austin, USA
c.julien@utexas.edu

Chenguang Liu
The University of Texas at Austin, USA
liuchg@utexas.edu

Amy L. Murphy
Fondazione Bruno Kessler, Trento, Italy
murphy@fbk.eu

Gian Pietro Picco
University of Trento, Italy
gianpietro.picco@unitn.it

ABSTRACT

Identifying “who is around” is key in a plethora of *smart* scenarios. While many solutions exist, they often take a theoretical approach, reasoning about protocol behavior with an abstract model that makes simplifying assumptions about the environment. This approach creates a gap between protocol implementations and the models used during design and analysis. In this paper, we take a system approach to continuous neighbor discovery: starting with the concrete technology of Bluetooth Low Energy (BLE) we build a protocol, called BLEnd, tailored to its constraints. Moreover, we also consider the very real effects of packet collisions, to our knowledge a first in this domain. Our ultimate goal is to directly empower developers with the ability to determine the optimal protocol configuration for their applications; in this respect, the slotless operation of BLEnd offers richer alternatives than state-of-the-art protocols. Developers specify the minimum discovery probability, the target discovery latency, and the maximum expected node density; these are used by an optimizer tool to parameterize the BLEnd implementation towards maximum lifetime. This paper shows that BLEnd not only achieves the user-specified goals, but does so more efficiently than analogous configurations of competing protocols.

CCS CONCEPTS

•Networks →Network protocol design; •Human-centered computing →Ubiquitous and mobile computing systems and tools;

KEYWORDS

Continuous neighbor discovery, Bluetooth Low Energy

ACM Reference format:
Christine Julien, Chenguang Liu, Amy L. Murphy, and Gian Pietro Picco. 2017. BLEnd: Practical Continuous Neighbor Discovery for Bluetooth Low Energy. In *Proceedings of The 16th ACM/IEEE International Conference on Information Processing in Sensor Networks, Pittsburgh, PA USA, April 2017 (IPSN 2017),* 12 pages.
DOI: http://dx.doi.org/10.1145/3055031.3055086

1 INTRODUCTION

The ability to continuously discover neighboring devices in range of wireless communication is a key building block of many Internet of Things (IoT) scenarios. Smart spaces require the ability to detect the proximity of users to devices deployed in the space, to trigger interactions. Smart retail systems need to detect the time a person spends in each area of the store; information acquired via device discovery serves both to orchestrate interaction with the user and to analyze long-term shopping behavior to improve the retail experience. Smart cities take these capabilities to a larger geographical scale, enabling the automatic triggering of personalized services on a smartphone based on physical proximity to designated places (e.g., monuments or exhibits [9]) hosting fixed nodes.

Neighbor discovery can also be used as a stepping stone for services concerned with proximity among people, e.g., to ensure that tourists do not get separated from their tour group, to enable proximity-based authentication [11], or to ensure that children on the way to school are always close enough to at least one responsible adult [5, 14]. Studies on behavioral analytics are fueled by the ability to non-invasively detect proximity among humans [3] or animals [12]. In general, the *continuous* and unsupervised (i.e., without explicit user interaction, such as pairing) ability to detect proximity to other nearby, potentially mobile, devices enables new interaction patterns, unlocking novel application domains.

Continuous Neighbor Discovery: State of the Art. Most state-of-the-art continuous neighbor discovery protocols divide time into equal-sized slots, during which a node is either active or inactive. When the active slots of two nodes overlap, discovery occurs. The protocols are evaluated by assessing the *discovery latency*, defined as the number of slots until a neighbor is detected; a protocol’s *duty cycle*, defined as the number of active slots over a unit of time, serves as an indirect measure of energy consumption. Protocols are described relative to the mechanisms they use to determine which slots are active, with the result being either probabilistic or deterministic discovery. In the Birthday protocol [10], nodes randomly make a slot active with a given probability, offering good average case performance but not providing guarantees on discovery latency. Instead, Disco [4] and U-Connect [7] space active slots according to prime numbers, relying on the properties of the Chinese Remainder Theorem to guarantee discovery within a tight time bound. Searchlight [1] and BlindDate [17] offer hybrid approaches, placing some active slots for deterministic discovery, then adding more in a pseudo-random manner to improve performance.

Nihao [13] departs from these protocols by specifying that a slot can either be for listening or transmitting, where the latter behavior is defined by a single, short beacon at the beginning of the slot. By observing that a single short beacon costs much less than listening for the entire slot, Nihao proposes to "talk more and listen less", resulting in a protocol with more active slots, but with competitive consumption as most slots are cheaper, transmission-only slots.

Theory vs. Practice. Research on continuous neighbor discovery has been hitherto characterized by a strong slant towards theory, with most protocols taking the slotted approach described earlier and assuming slots of arbitrary length. Much work has focused on relating protocols to one another at the model level, based entirely on a fixed size slot. As a consequence, details concerning the actual behavior within a slot are often abstracted away.

Unfortunately, these assumptions overlook system-level constraints that significantly change tradeoffs and may even prevent the use of a given protocol with a given network technology. For instance, an average discovery latency of 10,000 slots or more is common [13]. This is acceptable when slots are small; a common slot length is 10 ms, yielding a latency of minutes. However, the Bluetooth Low Energy (BLE) standard, available on many commodity devices, prescribes that advertisements are separated by at least 20ms (and even 100ms for some advertisement types). This places a hard lower bound on slot duration and can increase discovery latencies by up to an order of magnitude, rendering them unacceptable *in practice*. Further, existing protocols' models ignore the density of nearby nodes, a factor that can increase the potential for beacon collisions that hinder discovery, as discussed next.

Our Perspective. We take a view motivated by a desire for a *practical* approach to continuous neighbor discovery.

First, we do not neglect system-level concerns; instead they are the starting point of our endeavor. We choose BLE as our reference platform as it is pervasive on many consumer electronics. On top of BLE, we devise a continuous neighbor discovery protocol, called BLEnd (BLE neighbor discovery), that is compatible with BLE's technological constraints and features, as outlined in Section 2.

Moreover, we emphasize metrics that impact the use of BLEnd in real environments. Specifically, we recognize that packet collisions reduce discovery rates, making it difficult if not impossible to reach 100% discovery. State-of-the-art protocols ignore this aspect, simultaneously aiming to reach this unattainable goal and failing to provide application designers with information about the concrete, negative effects of collisions. Instead, BLEnd enables designers to express requirements as a *service level agreement* that includes the target discovery latency plus two parameters related to collisions: expected node density and target discovery probability. An optimizer tool automatically derives the BLEnd parameters that meet these requirements with the lowest energy costs, allowing application designers to both tune BLEnd to their specific needs and to use it with a precise understanding of its actual performance.

Our Protocol: BLEnd. In a system without energy constraints, discovery can be achieved with an always-on receiver and periodic broadcasts to announce presence. As long as messages do not collide, discovery is guaranteed. Alternatively, if nodes are synchronized, they can exchange discovery messages at predetermined times, allowing the radio to be otherwise turned off. Unfortunately, most real systems have tight power budgets and providing perfect synchronization is expensive. Therefore our goal is to make guarantees about discovery latency while minimizing power consumption, allowing neighbor discovery to be continuous. As with other protocols, the key is scheduling transmitting and listening, whose spatio-temporal overlaps enable discovery. However, our design departs from the state of the art in many respects.

Existing approaches focus exclusively on *bi-directional* discovery (i.e., node A discovers node B and vice versa). While this is intuitive, we observe that, for many applications, *uni-directional* discovery, in which only one node in a pair discovers the other, is sufficient. For example, when discovery serves as the core mechanism for recording proximity in the human or animal social studies above, uni-directional detection is sufficient to demonstrate that A and B were in range at some point in time. Offline analysis can later infer that a bi-directional contact has taken place from a single uni-directional discovery. Unidirectional detection can also serve as a cornerstone for *triggering communication*, with one device detecting the other then initiating a separate bi-directional communication.

Based on these motivations, and in contrast with the state of the art, we design the core of BLEnd to support uni-directional discovery. This application-level choice has deep system-level implications, as it unlocks opportunities for energy optimization currently missed by the state of the art. As we discuss in Section 3, the focus on uni-directional discovery allows us to define a periodic schedule where each node can be duty-cycled during slightly more than half of the period and be *completely inactive* for the remaining portion, significantly reducing energy consumption. This is achieved without any assumptions about time synchronization and yet provides deterministic discovery latency guarantees at low duty cycles. This design decision does not affect BLEnd's generality or applicability; while uni-directional discovery is the fundamental building-block, it can be efficiently extended to bi-directional discovery.

We depart from dominant trends in the state of the art in two other respects. First, we *completely remove the concept of slot*. The primary advantage is in added flexibility for BLEnd to meet application requirements with the lowest possible energy consumption. Second, we *directly account for constraints from the BLE stack and the nature of communication in dense mobile environments*. We create an *optimizer* (Section 4) that determines the parameter settings that provide optimal continuous neighbor discovery w.r.t. a node's battery lifetime, given an application-desired service level agreement. This optimizer is built around a novel analytical model (Section 5) that accounts for *i)* idiosyncrasies of BLE, including accurate power consumption of BLE operations derived from laboratory experiments, and *ii)* packet collisions, which profoundly affect the behavior of neighbor discovery. Alongside this mathematical model, we provide an accurate simulator that we exploit to show the protocol functionality in a variety of scenarios (Section 6) as well as its performance in comparison to two reference protocols (Section 7). Finally, we describe our implementation of BLEnd on a *standard, unmodified BLE stack* (Section 8), and analyze its performance (Section 9). Our experimental results confirm not only that BLEnd is more versatile and efficient than its competitors, but also that our model, simulator, and implementation are in good agreement, enabling the immediate use of BLEnd in applications. Section 10 ends the paper with brief concluding remarks.

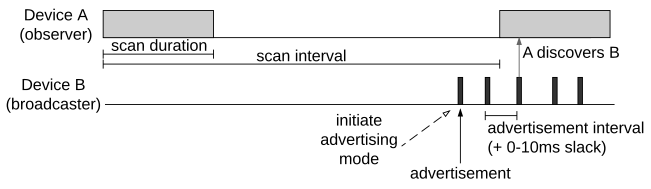


Figure 1: BLE Discovery.

2 BACKGROUND AND MOTIVATION

One of our motivations for BLEnd is that existing *continuous* neighbor discovery approaches are incompatible with BLE in various ways. This section discusses elements of BLE that are critical when using it for neighbor discovery. We then describe how neighbor discovery is commonly done in BLE, notably *not* in a continuous manner. We discuss the possibilities of using BLE as the underlying technology for existing approaches, laying the foundation for our novel continuous neighbor discovery protocol, BLEnd.

BLE in the Abstract. BLE is an obvious candidate to support continuous neighbor discovery for commodity applications, due to its low power and wide availability [8, 15]. We take as a premise the need to work with and around the BLE standard [2]. Figure 1 overviews key elements of BLE discovery, in which one side acts as an *observer* and the other as a *broadcaster*. The broadcaster emits an *advertisement event*, typically 1-3ms long depending on hardware, during which it sends a beacon on one of the three advertisement channels then waits briefly for a *scan response* on the same channel. Each advertisement event may perform this process on one, two, or all three of BLE's advertisement channels. Advertisement events are triggered periodically based on the *advertisement interval*. BLE also automatically adds a 0-10ms *random slack* to the advertisement interval. This slack is engineered into BLE to reduce the potential for simultaneous advertisers to collide many times in a row.

On the receiving side, a BLE observer listens continuously for a *scan duration*, repeating this *scan event* periodically based on a *scan interval*. The scan duration must be shorter than the scan interval; the advertisement and scan interval must be at least 20ms long; all three values are parameters from the application layer. Each scan event listens on exactly one advertisement channel; subsequent scan events are required to cycle through the three advertising channels. A scan event detects an advertising event only if the advertisement is sent on the matching channel; therefore, in mapping continuous neighbor discovery onto BLE, we send every beacon on all three advertisement channels to guarantee that it is captured by any listening device, regardless of the listener's scan channel.

BLE in the Concrete. Our approach to continuous neighbor discovery is generic to devices supporting the BLE specification [2], including many modern Android devices. Our evaluation uses TI SensorTag (www.ti.com/sensortag), an inexpensive sensing device with a complete BLE radio and networking stack representative of many realizations of the BLE specification, especially on IoT-style devices. Here, we highlight elements of the SensorTag BLE implementation relevant to continuous neighbor discovery.

Basic "off-the-shelf" BLE discovery expects one or more nodes to act as broadcasters and another to act as observer. However, the specification allows a single device to assume both roles, if supported by the hardware. The SensorTag does support applications that take on the multiple roles, switching between them without restarting the BLE stack. This capability is common in many other BLE implementations, both on lightweight devices and on Android.

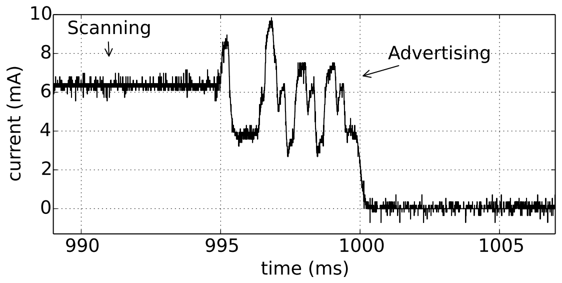


Figure 2: Oscilloscope trace showing scanning then advertising on the SensorTag.

Figure 2 shows an oscilloscope trace of a segment of BLE operation on a CC2650STK SensorTag. Oscilloscope measurements in this paper were captured with a Picoscope 2204A acquisition device. Current traces result from measuring the voltage drop over a 10Ω low side shunt resistor placed in series with the SensorTag. In this trace, our device begins as an observer, scanning one advertisement channel. At time 994.84ms, the application instructs the device to cease observing and start advertising. Here and throughout the paper we use non-connectable advertisements, as the intent is simply to announce presence, not to initiate a connection for sending data. The advertisement event starts at 996.26ms and ends at 999.46ms, sending a single advertisement on each of BLE's advertising channels. This advertisement event lasts for $b = 3.2$ms. Using this measurement setup, we recover the instantaneous currents of scanning and advertising, respectively, as $I_{scan} = 6.329$mA and $I_{adv} = 5.725$mA. We also measured a stand-by current $I_{idle} = 80.64\mu$A, although the TI SensorTag datasheet indicates an expected value of 1μA.

3 BLEnd

We start from the premise that many applications require only one node of a pair to detect the other's presence and design support for uni-directional discovery (U-BLEnd). We then show how this is easily adapted to bi-directional discovery. In both cases, the goal is to achieve continuous neighbor discovery without the device's radio having to be continuously active.

Uni-directional Discovery: U-BLEnd. Our initial goal is to guarantee that one of every pair of devices will discover the other within a given time, the *discovery latency*, while simultaneously minimizing the energy consumed in discovery activities. The protocol behavior is a simple, repeating sequence of listening intervals (scans, in BLE) and beacon transmissions (advertisement events). We term the duration of this repeating sequence the *epoch*, *E*.

On a node, U-BLEnd exploits the first half of every epoch to attempt to discover or be discovered, then remains inactive for the second half, with the radio in stand-by. The radio is also switched to stand-by whenever the node is not listening or transmitting during the first half epoch, conserving as much energy as possible.

As shown in Figure 3, an epoch always begins with a listening interval whose length depends on application requirements, most critically the discovery latency. Immediately following the listen

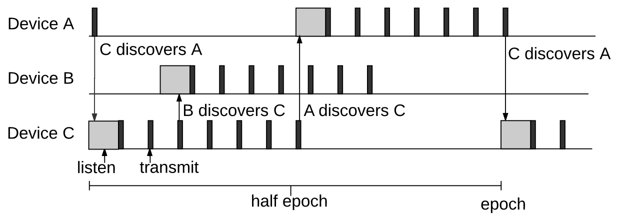


Figure 3: U-BLEnd: Uni-directional discovery.

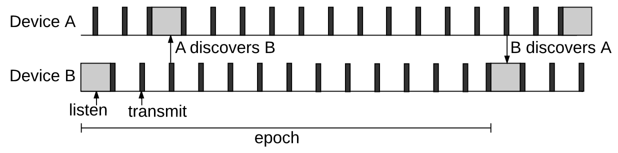


Figure 4: F-BLEnd: Full-epoch bi-directional discovery.

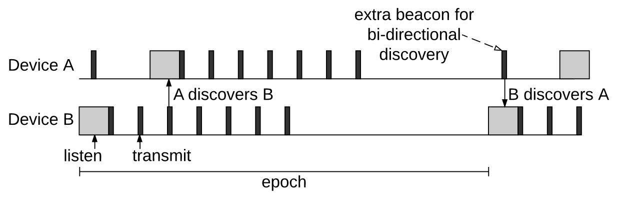


Figure 5: B-BLEnd: Efficient bi-directional discovery.

interval, BLEnd begins a sequence of beacon transmissions that lasts for the remainder of the first half of the epoch, with the last scheduled beacon falling just inside the second half of the epoch. The interval between the beginning of adjacent beacons is at most the length of the listen interval, thus ensuring that if the listening interval of another node overlaps with the active portion of this node, the listener receives at least one beacon and discovery occurs, as shown with arrows. Further, the active portion of the epoch, defined as the time from the beginning of the listening interval to the end of the last beacon, is greater than half of the epoch. This configuration *guarantees* that the active portions of two independent nodes overlap, resulting in detection. Interestingly, it *may* also result in bi-directional discovery, as shown in Figure 3 for A and C.

A key element of BLEnd relative to existing continuous neighbor discovery protocols is its direct consideration of the practical constraints of BLE. In the context of uni-directional discovery, there are two important elements. First, to provide guarantees on discovery latency, a node's beacon must be entirely contained within a listener's listening interval, and BLE beacons have non-negligible duration. Setting $A = L$, i.e., the beaconing interval equal to the listening interval does not consider the case in which the listener might receive a partial beacon at either the beginning or the end of its listening interval, a situation that may prevent discovery.

Second, BLE adds random slack to the application-specified advertising interval. Therefore, when U-BLEnd specifies an advertising interval of A, beacons may be spread as much as $A + s$ apart, where s is the maximum random slack added. Because this may make A greater than L, a listening interval may fall entirely between two beacons, preventing discovery.

We cater to these observations by defining $A = L - b - s$.

Full-epoch Bi-directional Discovery: F-BLEnd. To naïvely implement bi-directional discovery, one can simply continue beaconing throughout the entire epoch; we call this variant full-epoch BLEnd, F-BLEnd. In Figure 3, B would not discover A in U-BLEnd as B's listening interval falls inside the inactive portion of A's epoch. Instead, as shown in Figure 4, discovery happens in F-BLEnd, as B's listening interval overlaps one of A's additional beacons.

Efficient Bi-directional Discovery: B-BLEnd. Adding *all* of the beacons in the inactive half of the epoch is unnecessary; a more efficient strategy is employed by our last protocol variant, B-BLEnd. Consider discovery between two nodes A and B. If the U-BLEnd schedule allows A to detect B in one epoch, the goal of B-BLEnd is for B to detect A in the next epoch. To this end, any beacon added by B is unnecessary, as B has already been detected by A. Instead, it is sufficient that A adds exactly *one* beacon—the one falling in B's listening interval, as shown in Figure 5. This is possible if beacons include a small bit of information, i.e., the time between the start of the epoch and the beacon transmission. Combining this information with the knowledge about the duration of the epoch and of the advertisement interval enables a receiving node (A in our case) to compute the start time of the discovered node's next listening interval and selectively schedule only the one beacon, out of all those that F-BLEnd would add, that lands inside the other node's listening interval, thus allowing bi-directional discovery.

In scenarios with more than two nodes, the worst case requires a node to activate all of the beacons in the inactive half of the epoch, as in F-BLEnd. However, we observe that a single added beacon may allow discovery by more than one neighboring node if the listening intervals of multiple neighbors overlap with the beacon. Put another way, a new beacon is not necessarily needed for every neighbor. On the other hand, when a node is among few neighbors, correspondingly few beacons will be activated by B-BLEnd—none if the node is alone. It is this observation that makes BLEnd particularly suitable for continuous operation: *to reduce the energy cost of continuous neighbor discovery, BLEnd effectively adapts to the natural dynamics in the density of neighboring nodes.*

4 OPTIMIZING LATENCY VS. LIFETIME

Using BLEnd in real applications requires that trade-offs between latency and energy consumption are made explicit. Application developers often expect zero latency and infinite lifetime, which is clearly impossible. The tool we describe next, the *BLEnd optimizer*, enables developers to quickly explore these tradeoffs and select a protocol configuration most suited to the application requirements.

The latter may vary widely. An application that monitors proximity of visitors to museum exhibits [9] targets a discovery latency of seconds, which is more expensive in terms of energy but acceptable since devices can be easily recharged after each museum visit. On the other hand, wildlife monitoring [12] must accept a discovery latency of up to a minute, as devices are animal-borne and expected to last months, if not years. In our experience, neither latency nor lifetime requirements are cast in stone; they are selected as a compromise between application- and system-level concerns.

In this respect, our work has two assets relative to the state-of-the-art. First, the configuration space of existing protocols is severely limited by the coarse-grained discretization induced by slots and other constraints (e.g., choosing prime numbers [4, 7]). In contrast, BLEnd's slotless operation provides remarkably more

Table 1: Parameters used in model and optimizer.

Parameter	Description
Input parameters: Application requirements	
mode	uni-directional vs. bi-directional discovery
Λ	maximum discovery latency
P	minimum discovery probability
N	maximum number of nodes in a collision domain
Input parameters: BLE stack	
b	duration of an advertisement event (beacon)
s	maximum random slack for an advertisement event
Output parameters: BLEnd configuration	
E	duration of the epoch
A	duration of the advertising interval
Derived output parameters	
L	duration of the scan interval (listening)
n_b	number of advertisements per epoch

configuration options. Second, the impact of collisions, which directly result in missed discoveries, is routinely neglected by models underlying existing protocols; their latency and lifetime estimates are therefore deceptive, as they do not match what is possible in actual implementations. In contrast, the BLEnd model in Section 5 explicitly accounts for collisions, enabling the optimizer to more realistically determine the best BLEnd configuration.

Application requirements constitute a service-level agreement of sorts that must be honored by the BLEnd configuration generated by the optimizer. The requirements input to the optimizer are shown in Table 1, along with the expected outputs. A developer must specify the maximum discovery latency Λ and minimum discovery probability P allowed in the application, along with the maximum number N of nodes in a collision domain. P is defined for B-BLEnd and F-BLEnd as the probability that each node discovers all of its neighbors within Λ; for U-BLEnd, instead, P is the probability that at least one node in a pair discovers the other.

The inputs also include system-level parameters. The duration b of a BLE advertisement event may change depending on the hardware. Hereafter we consider $b = 3.2$ms as measured on our platform when using all three channels, which yields lower discovery latency and better resilience to collisions. The random slack introduced by the BLE stack is set to $s = 10$ms as per the specification [2].

The optimizer returns a configuration $\langle E, A \rangle$, i.e., the advertisement interval and epoch length, that satisfies the application requirements and minimizes energy consumption. For instance, assume the developer requires bi-directional discovery with $\Lambda = 2000$ms, $P = 0.95$, $N = 15$. The output configuration $E = 667$ms, $A = 71$ms guarantees that *i)* in the worst case where 15 nodes are all within range of one another, each of them has a 95% probability of discovering the others within 2s, and *ii)* this is achieved with the minimal energy consumption. The optimizer also outputs derived parameters: the scan interval $L = A + b + s$ and the number of advertisements, which is $n_b = \lfloor \frac{E}{2A} \rfloor - 1$ in the uni-directional case and $n_b = \lfloor \frac{E}{A} \rfloor - 1$ in the bi-directional (worst) case.

The optimizer has two fundamental components. The first is a model of the drain of electrical charge during one epoch:

$$Q(E, A) = I_{scan}L + I_{adv}n_b b + I_{idle}(E - L - n_b b) \tag{1}$$

where I_{adv} is the (average) instantaneous current consumed by the radio when advertising, I_{scan} when scanning, and I_{idle} when not engaged in neighbor discovery. These values must be measured for a given hardware, as we did for the SensorTag in Section 2. I_{idle} is determined by application specifics, e.g., whether the device senses, computes, or engages in other communication. In Section 6, we report lifetime values derived with $I_{idle} = 0\mu$A, as we are focused on the application-independent neighbor discovery functionality, and $I_{idle} = 80.64\mu$A, as measured on the SensorTag.

The second component of the optimizer is a model that, for a given BLEnd configuration $\langle E, A \rangle$, estimates the discovery probability P_d as a function of all the optimizer inputs, *by considering collisions*. This model is one of the contributions of this paper, and is presented in detail in Section 5.

Based on these components, the optimizer performs an exhaustive search across all possible $\langle E, A \rangle$ configurations. We test each possible epoch value smaller than the target latency, $E \leq \Lambda$; for each value E, we test all advertisement intervals allowed by the BLE specification [2], i.e., $A \geq 20$ms. Further constraints trivially rule out degenerate combinations, e.g., $E \leq A$ or when an epoch cannot accommodate a complete BLEnd schedule. For each pair $\langle E, A \rangle$ the optimizer computes the expected discovery probability P_d according to the model in Section 5. If $P_d \leq P$, i.e., the expected discovery probability does not satisfy the one targeted by application requirements, the $\langle E, A \rangle$ configuration is discarded as invalid. Otherwise, the current drain $Q(E, A)$ is computed; the optimizer outputs a valid $\langle E, A \rangle$ that minimizes this value.

The search space determined by the values of E and A is explored in (configurable) increments of 1ms, as this is close to the resolution of timers commonly found in BLE devices. Along with the other constraints mentioned above, this limits the computational overhead; our R implementation of the optimizer computes the optimal configurations for the scenarios considered in this paper in at most a few minutes on a common laptop.

5 MODELING DISCOVERY PROBABILITY

We next derive a model of the discovery probability P_d using the parameters in Table 1. Discovery probability is intimately intertwined with the probability of colliding beacons. Other factors, most notably interference in the crowded 2.4GHz band that BLE uses, can also cause a beacon to be lost. However, the three channels (37–39) used for advertising are defined by the BLE specification [2] *i)* to be different from those used for actual communication, and *ii)* to avoid interference with the most commonly used WiFi channels. Further, to the best of our knowledge, no existing work analyzes the impact of collisions of the beacons within continuous neighbor discovery; these collisions are the primary source of beacon loss, and we analyze these impacts in Section 5.1. The discovery probability is also affected by specifics of the BLE communication stack, most notably the random slack, as we discuss in Section 5.2.

5.1 Analyzing the BLEnd Schedule

We first derive discovery probability by considering the chance that beacons collide due uniquely to the BLEnd schedule.

In the following, the same model handles uni-directional and bi-directional discovery; the only difference is the number C of nodes in the collision domain. In the uni-directional case (U-BLEnd), $C = \frac{N}{2}$ on average because a node's schedule does not have beacons in the second half of the epoch. In the bi-directional case we assume $C = N$. F-BLEnd always schedules beacons throughout the second epoch, while B-BLEnd schedules beacons on-demand only if and when needed. However, B-BLEnd intends that all $N - 1$ nodes that are not the listener attempt to "hit" the listen interval with a beacon. Therefore, our model focuses on F-BLEnd because *i)* it provides the worst case for both collisions and energy consumption, and *ii)* it yields a more tractable model.

Within an epoch, collisions among beacons are harmful only when they occur within another node's listen interval; beacon collisions not overlapping with the listen interval of some node are irrelevant. Therefore, we focus on the listen interval and observe that, due to the structure of a BLEnd schedule, there are at most $C - 1$ other nodes that have a beacon scheduled within another given node's listen interval.

Consider one of these $C - 1$ senders, with a beacon start time t falling in the listen interval. Because successful discovery requires the listen interval to overlap entirely with the beacon, this sender is not discovered by the listener if another beacon (from another sender) overlaps even partially with the first one. This collision happens if the other sender chooses a beacon start time in the interval $(t - b, t + b)$. Therefore, $2b$ of the possible options for beacon start times cause collisions. Because beacon start times must be chosen in the interval $[0, L - b]$ to ensure that the entire beacon is received before the listening interval ends, the probability that some other sender collides with our selected sender is $\frac{2b}{L-b}$.

The discovery probability can therefore be computed as

$$P_{nc} = \left(1 - \frac{2b}{L - b}\right)^{\gamma} \tag{2}$$

where $\gamma = C - 2$ is the number of beacons potentially colliding with the chosen sender within the given listen interval. Since each sender is expected to send exactly one beacon during the listen interval, γ is simply the number C of nodes in the collision domain, minus the listener and the sender for which we are computing the discovery probability.

5.2 Accounting for BLE Specifics

The model above assumes that two consecutive beacons are spaced exactly by the advertisement interval A. However, if this were the case in BLE, two colliding advertisements would collide forever. To avoid this, BLE adds a "random slack" $r \in [0, s]$ to the start time of all advertisements in a sequence, except the first one [2]. This slack has a subtle yet significant impact on discovery probability.

Discovery Is Possible Across Epochs. Arguably, the most important effect of the slack is that it unlocks the possibility that a beacon experiencing a collision in the first epoch is discovered in a later epoch, as the random slack moves the colliding beacons slightly relative to each other. Therefore, we model the discovery probability across epochs and its interplay with the maximum latency Λ.

In principle, based on (2), the probability of discovering a given node across k epochs is simply:

$$P_{d,k} = 1 - (1 - P_{nc})^k \tag{3}$$

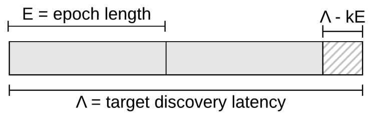


Figure 6: Relationship between Λ, E, and the "spillover".

where $(1 - P_{nc})^k$ is the probability that a node is *not* discovered across k epochs.

On the other hand, the only constraint relating the maximum latency and the epoch is that $E \leq \Lambda$, to guarantee that a complete BLEnd schedule can unfold. As a consequence, Λ is not necessarily a multiple of E. This implies that a node not discovered in the first k epochs may be discovered in the "spillover" $\Lambda - kE$; Figure 6 shows the relationship between Λ, E, and this spillover.

We observe that, within a given epoch, the rate at which nodes discover each other is constant, as the phases between nodes' epochs are independent. Therefore, the discoveries that occur in the spillover can be quantified simply by multiplying the discovery probability P_{nc} by the fraction of epoch $\frac{\Lambda - kE}{E}$ corresponding to the spillover. Moreover, the spillover increases the likelihood of discovery only if the node was *not* discovered in the first k epochs. Therefore, the probability that a node is discovered *for the first time* in the spillover is

$$P_{d,sp} = (1 - P_{nc})^k \left(\frac{\Lambda - kE}{E}\right) P_{nc} \tag{4}$$

and the overall probability of discovery within Λ is

$$P_d = P_{d,k} + P_{d,sp} \tag{5}$$

We next extend this simple formulation to account for other subtleties induced by the random slack.

Extra Beacons. In the presence of random slack, a listen interval of $L = A+b$ could miss discoveries even in the absence of collisions. Figure 7 shows an example, where a sender's beacon scheduled at the very end of a listen interval is "pushed out" of it by a slack $r > 0$. This is why BLEnd sets the listen interval to $L = A + b + s$, to guarantee that a beacon always overlaps a listen interval. On the other hand, this choice may lead to situations where *two* beacons from the same sender fall in the listen interval of the same listener, as also shown in Figure 7. As a result, the number of beacons potentially colliding in (2) becomes $\gamma > C - 2$.

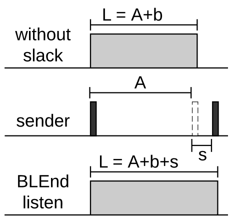


Figure 7: Compensating for slack.

To quantify the impact of these extra beacons, we observe that they *may* occur only when a beacon is within s of the beginning (or end) of the listen interval. The sum $b + A$ of the beacon duration and the advertisement interval leaves enough room for the next (or previous) beacon to fall within L. Each node has a $\frac{s}{L-b}$ chance of choosing such a beacon starting time. However, the extra beacon is generated only in half of the cases, on average. To see why, consider Figure 7. The first beacon occurs at a start time $t = 0$ w.r.t. the beginning of the listen interval, and only the choice of the maximum slack $r = s$ *prevents* an extra beacon from occurring; in the dual case where the first beacon is sent at $t = s$ into the listen interval, only a slack of $r = 0$ *causes* an extra beacon.

This fraction of extra beacons can therefore be modeled as

$$\sigma = \frac{1}{2}\left(\frac{s}{L-b}\right) \tag{6}$$

and must be added to those potentially colliding "naturally", i.e., because of the base BLEnd schedule, as discussed in Section 5.1. This is accounted for by modifying the exponent of (2) into

$$\gamma = (C-2)(1+\sigma) \tag{7}$$

Beacon Start Time Dependence. In Section 5.1, two beacons colliding in one epoch would collide in all subsequent ones. In essence, it is as if we modeled the discovery probability in the first epoch, which then remains the same because each node behaves according to its periodic schedule. The random slack introduced by BLE mitigates this situation by randomly "nudging" each beacon; beacons that collide in an epoch are no longer *always* colliding in the subsequent ones, and can be discovered across multiple epochs. However, the beacons that collided in the first epoch are still *more likely* to collide also in subsequent epochs, simply because they are kept close to each other by the random slack. The model in (3)–(5) does not account for this dependency across epochs, as it assumes that colliding beacons may always appear anywhere across the entire interval $L - b$.

The probability P_{nc} of *not* colliding in the first epoch is the same as in (2). However, for epochs $k > 1$, this probability becomes $P_{nc,W} \leq P_{nc}$, due to the fact that the starting time of beacons is constrained by the schedule in the first epoch. The discovery probability across k epochs in (3) becomes

$$P_{d,k} = 1 - (1-P_{nc})(1-P_{nc,W})^{k-1} \tag{8}$$

To derive the expression of $P_{nc,W}$, we observe that a beacon occurring at time t_i in the first epoch occurs within an interval W centered on t_i in subsequent epochs, due to the presence of the random slack. W represents the new *window of contention* for colliding beacons. If $W \geq (L-b)$, then the window is the same size or larger than the window used when assuming the beacons were randomly tossed in the interval $[0, L-b]$, and therefore P_{nc} in (2) still holds. However, when $W < L$, the probability of collisions *increases*, i.e., $P_{nc,W} \leq P_{nc}$, because beacons that collided once are more likely to collide again as they are somewhat loosely synchronized.

Estimating W is complicated by the fact that the slack introduced by BLE is added *relative to the start time of the previous beacon*, and therefore the offset among the same beacons in different epochs compounds across all beacons within an epoch. Consider the situation in Figure 8 and recall that, as per the BLE specification, the first beacon has no slack. Assume the extreme case where the value of the slack for all four remaining beacons is $r = 0$ for epoch 1 and $r = 10$ for epoch 2. Looking at the start time of a given beacon in

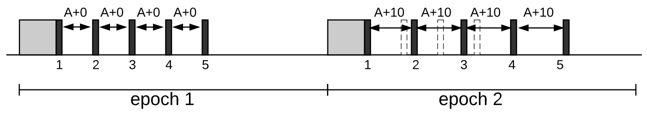


Figure 8: Compound slack. In epoch 2, positions of beacons from epoch 1 are shown with dashed lines.

each epoch, it is clear that the offset across epochs increases with the position of the beacon; beacon 2 occurs 10ms later in epoch 2 w.r.t. epoch 1, while beacon 5 occurs 40ms later. This of course could go either direction; swapping the choice of r for the two epochs would result in beacon 5 occurring 40ms *earlier* in epoch 2 w.r.t. epoch 1.

More generally, assume that a sender's i^{th} beacon is sent at time t_i in the listener's first epoch. Then the sending time for the same sender's i^{th} beacon in the listener's second epoch (or any epoch $k > 1$) falls in the interval:

$$[t_i - (i-1) \times s, t_i + (i-1) \times s]$$

This expression captures the *maximum* window of contention for the same beacon across two consecutive epochs. The *average* interval is only *half* of the above, since it is 0 for the first beacon, s for the second one, $2s$ for the third, and $s(i-1)$ for the i^{th} beacon.

Using these insights, and the fact that n_b is the total number of beacons sent in an epoch, we compute the *average* size W of the window of contention across all beacons in the epoch:

$$W = \frac{s\sum_{i=1}^{n_b-1} i}{n_b - 1} = s\frac{n_b}{2} \tag{9}$$

We can then compute the probability of *not* having a collision in an epoch $k > 1$ as

$$P_{nc,W} = \begin{cases} \left(1 - \frac{2b}{W}\right)^{\omega} & \text{if } W < (L-b) \\ P_{nc} & \text{otherwise} \end{cases} \tag{10}$$

The first expression is similar to (2), with the denominator $L - b$ replaced by W. The number of beacons potentially colliding within the window W is

$$\omega = 1 + \frac{W}{L-b}(\gamma - 1) \tag{11}$$

Recall that $P_{nc,W}$ accounts for the discovery probability after a beacon collision in the first epoch; the first term represents such a collider, which is bound to fall within W. The second term represents the fraction of the γ beacons from (7) that may fall in the window W surrounding the sender's beacon, minus the collider already considered in the first term. This formulation may underestimate the case where multiple beacons collide in the first epoch; in practice, this model already returns good estimates, as shown in Sections 6 and 9.

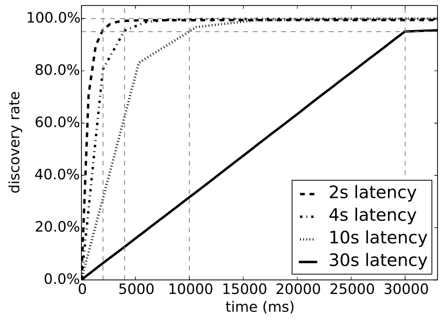
The complete expression of the discovery probability P_d remains the one in (5). However, its component due to the spillover $P_{d,sp}$ in (4) must also be modified along the same reasoning that led to the modified expression of $P_{d,k}$ in (8):

$$P_{d,sp} = (1-P_{nc})(1-P_{nc,W})^{k-1}\left(\frac{\Lambda - kE}{E}\right)P_{nc,W} \tag{12}$$

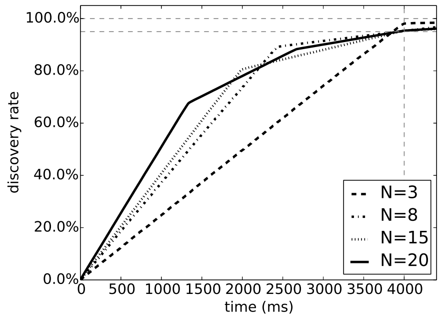
The first two factors account for the probability that the beacon collides in all the first k epochs, and the last captures the probability that the beacon does not collide in the spillover, accounting for the loose synchronization due to the slack.

The final expression of discovery probability across k epochs from (5) can therefore be rewritten as:

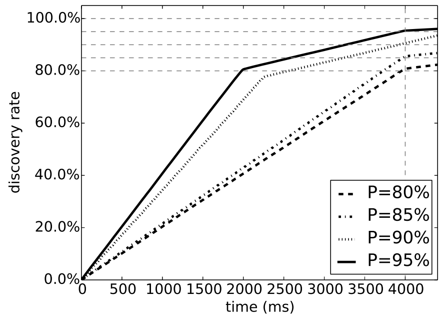
$$P_d = 1 - (1-P_{nc})(1-P_{nc,W})^{k-1}\left(1 - \left(\frac{\Lambda - kE}{E}\right)P_{nc,W}\right) \tag{13}$$



(a) $N = 15$, $P = 95\%$



(b) $\Lambda = 4s$, $P = 95\%$



(c) $N = 15$, $\Lambda = 4s$

Figure 9: Model-simulator validation for F-BLEnd.

Table 2: Model validation via simulation. Λ is in seconds; E and A are in milliseconds; probabilities P, P_{sim} and duty cycle DC are in percentage; lifetime is in days, assuming a battery capacity of 320mAh. We show the lifetime for both $I_{idle} = 0\mu A$ (LT_0) and $I_{idle} = 80.64\mu A$ (LT_{80}).

Configuration (optimizer)					**Simulation results**									
Input			**Output**		**F-BLEnd**					**B-BLEnd**				
N	Λ	P	E	A	P_{sim}	P_{sim}–P	DC	LT_0	LT_{80}	P_{sim}	P_{sim}–P	DC	LT_0	LT_{80}
3	4	95	3995	111	98.06	+3.06	5.93	37.45	30.61	97.16	+2.16	4.58	47.69	37.12
8	4	95	2430	106	95.25	+0.25	7.89	27.82	23.85	93.56	−1.44	6.74	32.21	27.01
15	4	95	1995	121	95.37	+0.37	9.38	23.19	20.37	93.91	−1.09	8.68	24.91	21.68
20	4	95	1335	91	95.36	+0.36	11.27	19.34	17.34	94.36	−0.64	10.54	20.59	18.34
15	2	95	667	71	95.74	+0.74	17.09	12.70	11.80	95.51	+0.51	16.32	13.25	12.28
15	4	95	1995	121	95.37	+0.37	9.38	23.19	20.37	93.91	−1.09	8.68	24.91	21.68
15	10	95	5363	149	95.11	+0.11	5.14	42.94	34.17	93.02	−1.98	4.36	50.12	38.57
15	30	95	29969	555	95.04	+0.04	2.49	86.93	57.22	93.37	−1.63	2.25	95.35	60.75
15	4	95	1995	121	95.37	+0.37	9.38	23.19	20.37	93.91	−1.09	8.68	24.91	21.68
15	4	90	2309	110	90.58	+0.58	8.22	26.70	23.03	88.28	−1.72	7.30	29.81	25.31
15	4	85	4000	174	85.51	+0.51	6.50	33.43	27.87	82.69	−2.31	5.95	36.30	29.83
15	4	80	3939	128	80.74	+0.74	5.96	37.00	30.30	76.97	−3.03	5.08	42.96	34.19

6 SIMULATING BLEND

To confirm the correctness of our model, we created a discrete event simulator in Java that steps through the BLEnd schedules of multiple nodes, simulating discoveries. The simulator accounts for collisions (or optionally ignores them), considers the three BLE advertisement channels, and implements all three versions of BLEnd. The simulator allows us to cross-validate the model underlying the optimizer with the protocol behavior observed in simulation.

Simulation Setup. We ran our simulations using the same assumptions made in the model. Further, we guarantee that any two nodes' epochs do not start within b time of each other, as this aspect is currently not captured by our model. Our evaluation tests the three dimensions of application requirements (i.e., discovery latency, node density, and discovery probability) by fixing two of them and varying the other. For each combination, we plot the cumulative distribution function (CDF) of discovery latencies and show target probabilities (as horizontal, dashed lines) and target latencies (as vertical, dashed lines); each curve represents 10,000 independent simulation runs. For lifetime, we assume a battery capacity of 320mAh—the same of the SensorTag we use in Section 9.

Model-simulator Validation: Bi-directional Discovery. Figure 9 shows the simulation results for F-BLEnd, which, among the BLEnd variants, is most accurately represented by the model in Section 5. We note a number of inflection points in each curve, caused by the fact that the target latency may be composed of multiple epochs; inside each epoch, BLEnd shows a constant discovery rate that decreases in each subsequent epoch as fewer nodes remain to be discovered. In all tested scenarios, our simulator produces discovery probabilities and latencies in line with the established targets. This is seen in Figure 9 by noting the difference between each CDF and the target discovery probability (horizontal line) at the target latency (vertical line). Table 2 shows the same data numerically, showing that the simulated F-BLEnd discovery probability is always higher than the target for the associated latency; the difference is always below 1%, except for the first combination.

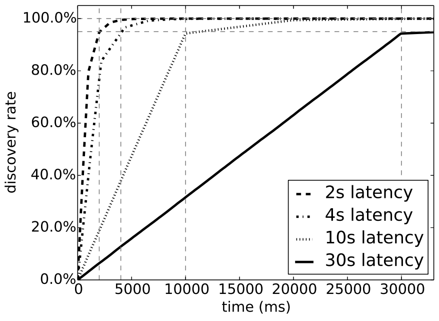
Table 2 shows also the results for B-BLEnd, whose CDFs are here omitted due to space limitations, confirming that the model and the associated optimizer are effective at generating realistic BLEnd configurations enabling bi-directional discovery while accounting for BLE constraints and beacon collisions. Further, Table 2 also allows us to quantify the benefits brought by the on-demand beacon scheduling of B-BLEnd w.r.t. the naïve solution provided by F-BLEnd. B-BLEnd offers 4 to 27% improvement in lifetime w.r.t F-BLEnd in the configurations studied, therefore confirming quantitatively that it provides a more efficient solution.

In the case of B-BLEnd, however, Table 2 shows that the simulator yields a discovery probability slightly lower than the target, except for two of the combinations. This is due to the fact that our model and the associated optimizer are based on F-BLEnd, for the reasons mentioned in Section 5, and therefore do not completely capture all intricacies of B-BLEnd.

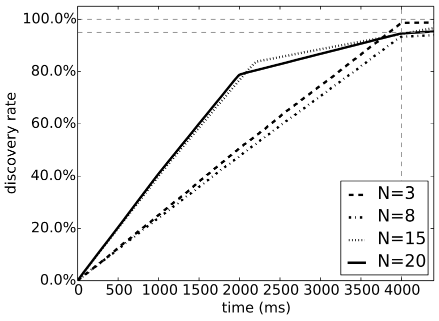
In particular, a subtle dependence among beacons occurs in B-BLEnd that does not exist in F-BLEnd. Remember that F-BLEnd schedules all beacons in all cases and that, in general, when $P <$ 100% the optimization does not require discovery of *all* nodes. Therefore, a valid configuration output by the optimizer may allow A to not detect B due to a collision. While this results in a discovery loss that is properly accounted in F-BLEnd, in B-BLEnd it may lead to a secondary loss that the model does not consider. Specifically, bi-directional discovery is achieved in B-BLEnd by the explicit addition of a beacon, triggered by detection. Therefore, if A does not detect B due to a collision, the additional beacon is not scheduled and B does not discover A, thus reducing the discovery rate. This motivates future work to improve the accuracy of our model and optimizer by explicitly considering the subtleties of B-BLEnd.

Model-simulator Validation: Uni-directional Discovery. We now investigate the relative performance of U-BLEnd. Although the use cases of uni-directional and bi-directional discovery differ, the expectation is that uni-directional discovery will save a significant amount of energy. These measurements also validate the application of the model to uni-directional U-BLEnd.

Figure 10 and Table 2 show that our simulation results come very close to the target discovery probability P in most cases. Recall that P is computed in U-BLEnd as the probability that at least one of each pair of nodes discovers the other within Λ time. U-BLEnd does miss the target discovery latency by a small amount in all but two parameter combinations. The reason is similar to the one discussed for B-BLEnd; essentially, the discovery rate in U-BLEnd is penalized twice for each missed detection. Nevertheless, these results do indicate that, when bi-directional discovery can be ascertained offline, U-BLEnd provides significant benefits over B-BLEnd. Meeting the same application requirements in terms of N, Λ, P with U-BLEnd and B-BLEnd yields comparable discovery rates, but U-BLEnd lifetime that is up to 1.8x longer than the one of B-BLEnd.



(a) $N = 15, P = 95\%$



(b) $\Lambda = 4s, P = 95\%$

Figure 10: Model-simulator validation for U-BLEnd.

Table 3: Model validation via simulation, for U-BLEnd. Units are the same as in Table 2.

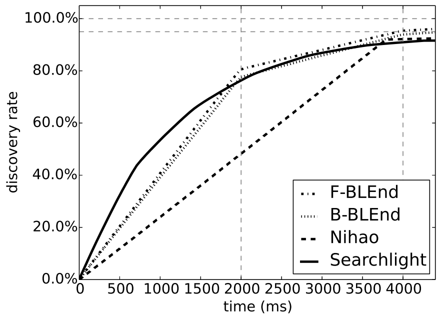
Configuration (optimizer)					**Simulation results**				
Input			**Output**		**U-BLEnd**				
N	Λ	P	E	A	P_{sim}	P_{sim}–P	DC	LT_0	LT_{80}
3	4	95	3999	80	98.51	+3.51	4.24	52.27	39.84
8	4	95	3983	83	93.26	−1.74	4.25	51.98	39.66
15	4	95	2197	69	94.36	−0.64	5.98	36.68	30.09
20	4	95	1998	72	94.58	−0.42	6.42	34.07	28.31
15	2	95	1001	51	94.74	−0.26	9.39	23.24	20.40
15	4	95	2197	69	94.36	−0.64	5.98	36.68	30.09
15	10	95	10000	232	94.24	−0.76	3.15	68.71	48.71
15	30	95	29951	234	94.26	−0.74	1.51	146.59	78.15
15	4	95	2197	69	94.36	−0.64	5.98	36.68	30.09
15	4	90	3995	111	88.97	−1.03	4.52	48.30	37.48
15	4	85	3999	80	84.89	−0.11	4.24	52.27	39.84
15	4	80	3999	80	84.89	+4.89	4.24	52.27	39.84

7 COMPARING TO THE STATE OF THE ART

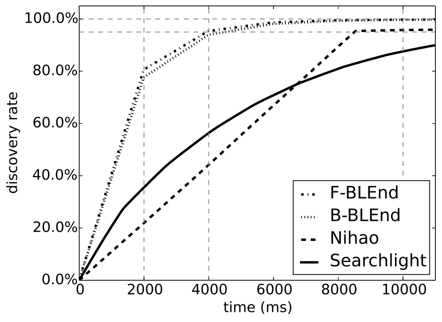
In our simulator, we also implemented both Searchlight [1] and Nihao [13] exactly as described in the literature. This allows us to compare against these recent protocols that are considered the best performing state of the art protocols but do not directly support BLE and whose implementations are not publicly available.

In both protocols, a key parameter is the slot duration, which is also the "unit of measure" of latency and duty cycle. Recall that duty cycle is a commonly used proxy for energy consumption; for clarity we report both duty cycle and battery lifetime for all protocols. Unfortunately, neither paper provides guidance on selecting an appropriate slot size. However, in Nihao α captures the ratio between the advertisement event duration and the slot duration; the value $\alpha = 0.054$ is used in [13]. In our hardware, a BLE advertisement event lasts 3.2ms, yielding a slot duration of 59.26ms. We use this value for both protocols.

Searchlight builds a schedule around a fixed period of t slots, which contains one "anchor" active slot at the beginning and a second "probe" slot somewhere in the first half of the period. The value of t relates directly to the schedule's duty cycle ($\frac{2}{t}$) and target discovery latency ($\frac{t^2}{2}$). A (Balanced) Nihao schedule is instead built around a parameter n; one period of the schedule consists of n^2 active slots. The node beacons at the beginning of every n^{th} active slot and then listens for the first n slots. The value of n is



(a) Fixed latency: $N = 15$, $P = 95\%$, $\Lambda = 4s$



(b) Fixed duty cycle: $N = 15$, $P = 95\%$, $DC = 8.77\%$

Figure 11: Comparing BLEnd against the state of the art.

similarly tied to duty cycle ($\frac{1+\alpha}{n}$) and latency (n^2). In the following, we leverage these relationships to set t and n based on a target discovery latency (or duty cycle) equivalent to that achieved by BLEnd in a given configuration.

Protocol comparison. Since Searchlight and Nihao both provide bi-directional discovery, we compare only our F-BLEnd and B-BLEnd variants against them. For all protocols, we use a target latency $\Lambda = 4s$. For BLEnd, we set a target discovery probability of $P = 95\%$ with $N = 15$ nodes; the other protocols attempt to reach 100% because, unlike BLEnd, they do not offer any alternative.

Figure 11a shows the results; notably, neither Searchlight nor Nihao reach 100% discovery, despite that this is their goal. The reason is two-fold: if two nodes choose nearly aligned slot start times, the nodes are always sending simultaneously and never discover each other; and the probability of collisions among beacons is not accounted for in these protocols. Instead, the BLEnd variants do take these aspects into account, via the model in Section 5; BLEnd succeeds in meeting a discovery rate around around the 95% targe. Figure 11a also shows that Searchlight most quickly discovers nodes at the beginning, though all protocols reach better than 92% discovery probability at or before the target latency. Nihao reaches its inflection point before the target discovery because its period is computed as n^2 slots; to hit a target latency, one must divide the latency into slots then take the square root. We conservatively choose n to be less than the square root as doing otherwise would cause Nihao to miss the target. In Figure 11a, Nihao uses $n = 8$; with a value of $n = 9$, Nihao misses the target latency by 850ms. This highlights the significant benefit that BLEnd is not artificially constrained by a rigid slotted structure.

Although all protocols, despite their different goals and configurations, have roughly the same discovery probability at the target latency, it is worth analyzing the expended energy. Table 4 shows the percentage of neighbors discovered, duty cycle, and resulting lifetime (based on different values of I_{idle} as in Table 2) when all protocols are configured with a target latency $\Lambda = 4s$, as considered thus far. Both BLEnd variants clearly outperform the competitors, with an expected lifetime for B-BLEnd that is 1.5x and 2x higher than Nihao and Searchlight, respectively.

To offer a dual perspective, we configured Searchlight and Nihao with a target duty cycle instead of a target latency, to observe their performance when given roughly the same energy budget of B-BLEnd. Therefore, we configured both to achieve a target duty cycle as close as possible to $DC = 8.68\%$. The results are shown in the last two rows of Table 4 and in Figure 11b, and show that BLEnd clearly outperforms the competition; both Searchlight and Nihao discover almost half of the nodes when given roughly the same energy budget as BLEnd. The duty cycles of Searchlight and Nihao are actually slightly higher than B-BLEnd; this is because their slotted operation significantly reduces the configuration options for these protocols. We chose the closest matching configuration, which while having a slightly higher energy usage still perform significantly worse in terms of discovery probability. This confirms that *BLEnd, thanks to its flexible unslotted operation and the ability to take into account collisions, efficiently uses the available energy budget to meet the target discovery and latency application requirements.*

A Stress-Test for BLEnd. To push the limits of B-BLEnd, we considered a configuration inspired by a small conference scenario ($N = 100$, $\Lambda = 10s$ and $P = 90\%$). First, when the configuration from the optimizer is fed to the simulator, B-BLEnd reaches 91.94% discovery by the target latency, exceeding the target discovery probability. Next, we observe that optimizer output yields an epoch $E = 3334$ms and an advertising interval $A = 444$ms. This implies that each epoch contains only 6 advertisements, a surprising result given the high node density. This schedule goes against the *talk more, listen less* principle driving Nihao, showing that "talking more" is not required at high density, as the same beacon enables discovery by multiple nodes. Interestingly, BLEnd is flexible enough to accommodate the advertising strategy best suited to the node density at hand, as configured via the optimizer.

8 A PRACTICAL IMPLEMENTATION

In this section, we relate the abstract description of the BLEnd protocols in Section 3 to our specific implementation on the TI

Table 4: Discovery probability vs. lifetime for $\Lambda = 4s$. BLEnd variants are configured with $N = 15$, $P = 95\%$. Notation and units are the same as in Table 2.

Protocol	P_{sim}	DC	LT_0	LT_{80}
F-BLEnd	95.37	9.38	23.19	20.37
B-BLEnd	93.91	8.68	24.91	21.68
Searchlight (target: latency)	90.96	18.52	11.53	10.79
Nihao (target: latency)	92.16	13.20	16.12	14.70
Searchlight (target: duty cycle)	56.36	9.66	22.11	19.53
Nihao (target: duty cycle)	44.19	8.81	24.16	21.11

Table 5: Implementation validation. Λ is in seconds; E and A in milliseconds; P and P_{eval} are percentages.

Configuration (optimizer)					**Results**				**Configuration** (optimizer)					**Results**	
Input			**Output**		**F-BLEnd**		**B-BLEnd**		**Input**			**Output**		**U-BLEnd**	
N	Λ	P	E	A	P_{eval}	$P_{eval} - P$	P_{eval}	$P_{eval} - P$	N	Λ	P	E	A	P_{sim}	$P_{eval} - P$
3	2	95	2000	77	97.70	+2.70	95.98	+0.98	3	2	95	2000	53	95.90	+0.90
8	2	95	1055	66	95.18	+0.18	95.72	+0.72	8	2	95	1991	83	92.86	−2.14
3	4	95	3995	108	96.58	+1.58	94.16	−0.83	3	4	95	4000	77	96.43	+1.43
8	4	95	2430	106	95.48	+0.48	94.39	−0.61	8	4	95	3983	83	95.28	+0.28
3	10	95	9975	172	97.22	+2.22	93.16	−1.84	3	10	95	9999	122	98.81	+3.81
8	10	95	9986	253	96.26	+1.26	93.00	−2.00	8	10	95	9999	122	94.89	−0.11

CC2650STK SensorTag. The latter is based on the CC2650 wireless MCU, which contains a 32-bit ARM Cortex-M3 processor, a 128kB programmable memory, 20kB of SRAM, and a complete system-on-chip BLE solution. The radio module uses a 2.4GHz RF transceiver fully compatible with BLE 4.2, with a receiver sensitivity of -97dBm and a range up to 50m/160ft.

Uni-directional Discovery. Our implementation of U-BLEnd delegates the protocol timing largely to the timers available in the BLE stack. An epoch always begins with a listening interval, implemented by initiating a BLE scan for the `scanDuration` established by the optimizer, $L = A + b + s$. As soon as the listening interval ends (indicated by the `GAP_DEVICE_DISCOVERY_EVENT` generated by the BLE stack), the protocol initiates advertising, with the `advertisingInterval` from the optimizer, A. A single BLE advertisement event duplicates the advertisement on BLE's three advertising channels (37, 38, and 39).

Bi-directional Discovery. To implement F-BLEnd atop U-BLEnd, we simply let the BLE advertising continue until the end of the epoch. However, the more efficient B-BLEnd requires each advertisement beacon to include the time until the next listen. The receiver uses this to activate the correct additional beacon for bidirectional discovery. Unfortunately, updating advertisement data inside a BLE beacon is non-trivial. Advertising must be stopped, the data updated, then advertisement can be restarted. As a result, in B-BLEnd, all timing between beacons (including BLE's random slack) is handled in application space.

Storing Data into Advertisements. A major constraint of implementing continuous neighbor discovery on BLE without pairing is that all exchanged information must fit inside an advertisement. In BLE, an entire beacon is always sent, regardless of whether it contains usable application data. Practically, SensorTag BLE advertisements have 31B of application-writeable data [16]. In BLEnd, we use 5B to identify packets as belonging to the BLEnd protocol (though less could conceivably be used) and 2B to carry a unique node identifier, enabling other nodes to determine which node has been discovered. B-BLEnd also needs to include the time to the next listen interval (2B) to enable receivers of the beacon to opportunistically schedule their beacons in the second half epoch.

9 REAL-WORLD EVALUATION

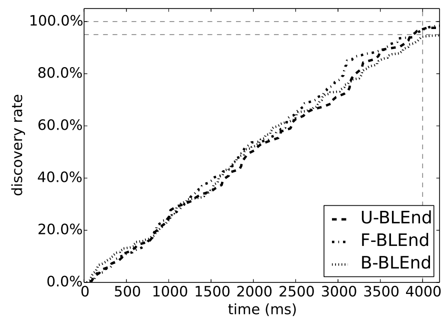
In this section, we report results from experiments with the SensorTag using the implementation described above. Our goal is to verify that our implementation matches the results from the simulator and the predictions of the model in terms of meeting the application requirements concerning discovery probability P within a given latency Λ and for a given node density N. We setup an indoor testing environment with 8 SensorTags in a 4×2 grid with inter-node spacings of 30cm/11.8in. We used an additional node that continually scanned the three BLE advertisement channels. This node was connected to a desktop computer for data collection.

In a run, each node first performs an initialization (e.g., setting BLEnd parameters, initializing the BLE stack, etc.) then emulates a *random arrival* in which it remains inactive for a random time $t_r \in (0, E)$ before starting to run the BLEnd protocol. This process simulates a real scenario in which participants move into range of one another at different times. To achieve randomness, we use the True Random Number Generator (TRNG) from the CC2650 MCU.

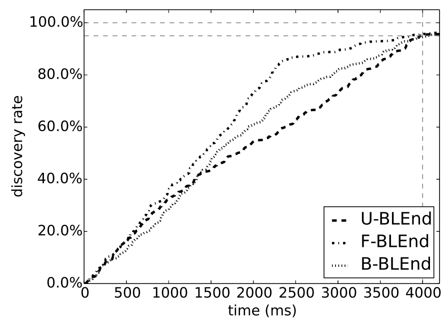
In our evaluation, we use the remaining space available in beacons to carry information useful to our experiments. To monitor energy consumption, we include the SensorTag battery level (2B) in each beacon. In addition, when a node discovers a new neighbor for the first time, it adds the corresponding timestamp (2B) to the beacon payload. Given the available space in the beacon, a node can convey information for up to 10 discovered neighbors. However, note that this 10-node limitation is only an artifact of data collection for our experimental setup and is not a restriction on BLEnd itself, which can support an arbitrarily large number of nodes. In our experiments, the sink node scans for these beacons and transfers the results to the desktop using the SimpleLink Debugger DevPack.

Table 5 reports tests for 3- and 8-node experiments with a target discovery probability $P = 95\%$ and different values for the target latency Λ. We used the optimizer to generate the $\langle E, A\rangle$ configuration for each BLEnd variant. The results are the average of 30 experiments for each combination of parameters. By considering the difference $P_{eval} - P$ between the measured and target discovery probability we see that our experiments track the target discovery rate within a few percentage points and, for all three protocols are in line with the expectations from the simulation. F-BLEnd always exceeds the target discovery probability as in simulation, while B-BLEnd and U-BLEnd narrowly miss the target in some cases, again similar to the results from simulation. As already discussed, this stems from the fact that the model in Section 5 does not capture all the subtleties of these two BLEnd variants.

Figure 12 provides a finer-grained perspective by showing the empirical CDFs of all three BLEnd variants for 3- and 8-node experiments with a fixed target latency Λ=4s and discovery probability of $P = 95\%$. We again show the target latency as a vertical dashed line and target discovery probability as a horizontal dashed line. Figure 12a shows that, with $N = 3$, the discovery rate of all protocols



(a) $N = 3$, $P = 95\%$, $\Lambda = 4s$



(b) $N = 8$, $P = 95\%$, $\Lambda = 4s$

Figure 12: Experimental evaluation of BLEnd.

steadily increases up to the target. This is a consequence of the fact that the epoch identified by the optimizer is very close ($E = 3995$ms for F-BLEnd and B-BLEnd) or equal (U-BLEnd) to the target latency Λ, as collisions are rare. Increasing the density to $N = 8$ bears little effect on U-BLEnd, whose epoch $E = 3983$ms remains close to Λ, and whose discovery rate is only marginally affected, as shown in Figure 12b. On the other hand, the optimal epoch becomes significantly smaller ($E = 2430$ms) for the bi-directional cases, to account for the increase in number of beacons and therefore collisions. F-BLEnd shows an inflection point precisely at that this point; discoveries occurring in the second epoch are due to collisions in the first one. In contrast, the discovery rate in B-BLEnd is smoother as this protocol sends significantly fewer beacons than F-BLEnd in the second half of the epoch. However, discoveries occur at a slower rate, precisely due to the fewer beacons in the first epoch and the need to explicitly schedule beacons in the second epoch to enable discovery. This is visualized effectively by Figure 12b, showing that the rate at which nodes are discovered in B-BLEnd sits between F-BLEnd and U-BLEnd—a consequence of its design that trades off the speed at which nodes are discovered for better energy consumption. In any case, for all BLEnd variants, the shape of the discovery rate curves matches very closely the results from simulation, as shown, e.g., in Figure 9b and 10b.

10 CONCLUSION

We presented BLEnd, a protocol designed with practical concerns in mind. On one hand, we choose BLE as a communication platform and design a protocol that, at its core, takes BLE's peculiar constraints into account. On the other hand, we put application users at the center and devise a protocol that is easily configurable to meet application requirements. These requirements include a target discovery latency, and consider, for the first time in the literature, the practical impact of collisions on continuous neighbor discovery.

The cornerstone of BLEnd is the realization that unidirectional discovery can be accomplished very efficiently without precluding its extension to bidirectional discovery, if and when needed. Further, the resulting protocol is adaptive, in that it automatically adjusts the amount of beaconing as density increases. Our evaluation in simulation shows quantitatively that BLEnd is significantly more performant than state of the art protocols once the latter are placed in the context of real practical constraints. Our design is reified in an implementation on TI SensorTags, but the key element enabling practical use is the companion optimizer, which enables users to identify the best configuration for a given set of application requirements. The optimizer is based on an analytical model of BLEnd, and we showed that model, simulator, and implementation are in good agreement, thereby enabling immediate use in applications.

ACKNOWLEDGMENTS

The authors would like to thank Davide Giovanelli (FBK) for his help with energy measurements and Ştefan Gună (now at Amazon) for his work on neighbor discovery [6] that inspired the approach described in this paper. The authors also thank the anonymous referees for their valuable comments and helpful suggestions. This work has been partially supported by FBK Smart Community, the CLIMB project, the municipality of Trento, and the National Science Foundation, CNS-1239498.

REFERENCES

[1] M. Bakht and R. Kravets. 2012. SearchLight: Won't You Be My Neighbor?. In *Proc. of Mobicom*. 185–196.
[2] Bluetooth SIG. 2014. Bluetooth Core Specification 4.2. https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439. (December 2014).
[3] T.K. Choudhury. 2004. *Sensing and modeling human networks*. Ph.D. Dissertation. Massachusetts Institute of Technology.
[4] P. Dutta and D. Culler. 2008. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proc. of SenSys*. 71–84.
[5] D. Giovanelli, B. Milosevic, C. Kiraly, A.L. Murphy, and E. Farella. 2016. Dynamic group management with Bluetooth Low Energy. In *Proc. of ISC2*. 1–6.
[6] Ş. Gună. 2011. *On Neighbors, Groups and Application Invariants in Mobile Wireless Sensor Networks*. Ph.D. Dissertation. University of Trento.
[7] A. Kandhalu, K. Lakshmanan, and R. Rajkumar. 2010. U-Connect: A Low-Latency Energy-Efficient Asynchronous Neighbor Discovery Protocol. In *Proc. of IPSN*.
[8] J. Liu, C. Chen, Y. Ma, and Y. Xu. 2013. Energy Analysis of Device Discovery for Bluetooth Low Energy. In *Proc. of VTC*.
[9] C. Martella, A. Miraglia, M. Cattani, and M. van Steen. 2016. Leveraging proximity sensing to mine the behavior of museum visitors. In *Proc. of PerCom*.
[10] M.J. McGlynn and S.A. Borbash. 2001. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *Proc. of MobiHoc*. 137–145.
[11] Y. Michalevsky, S. Nath, and J. Liu. 2016. MASHaBLE: Mobile applications of secret handshakes over Bluetooth LE. In *Proc. of Mobicom*. 387–400.
[12] G.P. Picco, D. Molteni, A.L. Murphy, F. Ossi, F. Cagnacci, M. Corra, and S. Nicoloso. 2015. Geo-Referenced Proximity Detection of Wildlife with WildScope: Design and Characterization. In *Proc. of IPSN*. 238–249.
[13] Y. Qiu, S. Li, X. Xu, and Z. Li. 2016. Talk More Listen Less: Energy Efficient Neighbor Discovery in Wireless Sensor Networks. In *Proc. of Infocom*.
[14] B. Shaw, M. Bicket, B. Elliott, B. Fagan-Watson, E. Mocca, and M. Hillman. 2015. Children's Independent Mobility: An International Comparison and Recommendation for Action. Project Report. Policy Studies Institute, London. (2015).
[15] M. Siekkinen, M. Hiienkari, J.K. Nurminen, and J. Nieminen. 2012. How Low Energy is Bluetooth Low Energy? Comparative Measurements with ZigBee/802.15.4. In *Proc. of WCNC Workshops*. 232–237.
[16] Texas Instruments. 2016. Bluetooth Low Energy Beacons (Application Report). http://www.ti.com/lit/an/swra475a/swra475a.pdf. (October 2016).
[17] K. Wang, X. Mao, and Y. Liu. 2015. BlindDate: A Neighbor Discovery Protocol. *IEEE Transactions on Parallel and Distributed Systems* 26, 4 (2015), 949–959.