Blender: Toward Practical Simulation Framework for BLE Neighbor Discovery

Yukuan Ding
The Hong Kong University of Science and Technology, Huawei Technologies

Tong Li*
Renmin University of China

Jiaxin Liang
Huawei Technologies

Dan Wang
The Hong Kong Polytechnic University

ABSTRACT

In the widely used Bluetooth Low-Energy (BLE) neighbor discovery, the parameter configuration of neighbor discovery directly decides the results of the trade-off between discovery latency and power consumption. Therefore, it requires evaluating whether any given parameter configuration meets the demands. The existing solutions, however, are far from satisfactory due to unsolved issues. In this paper, we propose Blender, a simulation framework that produces a determined and full probabilistic distribution of discovery latency for a given parameter configuration. To capture the key features in practice, Blender provides adaption to the stochastic factors such as the channel collision and the random behavior of the advertiser. Evaluation results show that, compared with the state-of-art simulators, Blender converges closer to the traces from the Android-based realistic estimations. Blender can be used to guide parameter configuration for BLE neighbor discovery systems where the trade-off between discovery latency and power consumption is of critical importance.

CCS CONCEPTS

• **Networks** → **Sensor networks**; **Network simulations**.

KEYWORDS

Bluetooth Low Energy; Broadcast; Scan

ACM Reference Format:
Yukuan Ding, Tong Li, Jiaxin Liang, and Dan Wang. 2022. Blender: Toward Practical Simulation Framework for BLE Neighbor Discovery. In *Proceedings of the International Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM '22), October 24–28, 2022, Montreal, QC, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3551659.3559052

1 INTRODUCTION

Along with the increasing contacts among mobile devices, the idea of Internet of Everything (IoE) is harvesting increasing attentions and had been widely applied. Whether a connection for wireless audio, portable devices, tracking assets or automating buildings, Bluetooth Low-Energy (BLE)[1] is the innovative force that is ready to be the ultimate solution for IoE over the next few decades. It is reported that the number of BLE enabled devices has reached over 5 billion by 2021, and increase rapidly every year [2]. As the prerequisite stage of IoE, neighbor discovery is a process where a device seeks to first contact neighbors in the BLE signal range. The metrics of neighbor discovery, such as discovery latency and power consumption, may significantly impact the performance of applications. For example, low-latency neighbor discovery is the essential control plane component that decides the delay of the data transmission among wireless terminals [3]. Low-power neighbor discovery is also a critical factor that must be considered in some proximity tracing systems [4–6].

Generally, the neighbor discovery process involves the interactions between an advertiser and a scanner. It is well studied that the configuration of neighbor discovery parameters such as scan interval, scan window, and advertise interval directly decide the trade-off between power consumption and discovery performance. For example, a larger advertise interval and lower scan duty cycle (i.e., $\frac{scan\ window}{scan\ interval}$) might get lower power consumption, but the time to discovery (i.e., discovery latency) may also be unacceptably large (see §3.1).

Compared with the studies that explore how to define neighbor discovery protocols [7–9] or to set the discovery parameters[10–14], much less focus has been cast on evaluating whether a given parameter setting meets the demands. A straightforward way is to repeatedly conduct real-world experiments which has a low efficiency and a high cost, much less the bias induced by wireless interference. Another way is build a mathematical model that provides a function to obtain a deterministic discovery latency from a certain set of parameters. However, the state-of-the-art modeling work [15] only provides the lower bound of discovery latency, while the modeling of distribution of all discovery latency values is still an open issue. Alternatively, it is more controllable and reproducible to build a simulator that simulates the behavior of advertiser and scanner according to the parameter configurations.

In the previous decade, there exist various parameter evaluation methodologies for BLE neighbor discovery [16–21]. However, these solutions are far from satisfactory and face the following problems which are never solved in this field. First of all, prior works are not consistent with the reality without considering the stochastic factors in the neighbor discovery system (see §3.3). Second, some of these solutions fall into the category of random sampling methods,

*Tong Li is the corresponding author (tong.li@ruc.edu.cn).

MSWiM '22, October 24–28, 2022, Montreal, QC, Canada

ACM ISBN 978-1-4503-9479-6/22/10…$15.00
https://doi.org/10.1145/3551659.3559052

which have to trade off between precision and overhead (see §3.4). Third, only providing the mean value or lower bound of the latency cannot meet specific evaluation requirement (see §3.2).

In this paper, we propose Blender[1] as a further step towards practical simulation framework for BLE neighbor discovery. First, Blender includes the analysis of the impacts from channel collision and the random behavior of the advertiser, which capture the key features of nearby communication in BLE neighbor discovery (see §5.3 and §5.4). Second, Blender adopts the discrete timing methodology in order to achieve simplified but accurate analysis (see §5.1). Third, the discrete concept also allows Blender to traverse through all possibilities and result in a determined output that cannot be achieved through random sampling, where a method of *Case Projection* is designed to avoid a brute-force traverse (see §5.2). Finally, Blender provides higher confidence to be applied as the parameter selection tool when concerning the full distribution rather than average/maximum.

Our evaluation results show that the CDF produced by Blender converges closer to that of latency in practical scenarios. This can be attributed to the integrated analysis of the stochastic factors. To the best of our knowledge, Blender is the first practical framework that simultaneously has the ability to provide precise, computationally efficient, and complete simulation of the BLE neighbor discovery systems.

The rest of the paper is organized as follows. §2 summarizes the related work. §3 depicts the background of BLE neighbor discovery and the motivation of this paper. §4 provides the simulation framework of Blender and §5 gives the detailed design of Blender. We evaluate Blender in §6 and conclude this paper in §7.

2 RELATED WORK

In the previous decade since BLE neighbor discovery participated in production scenarios, various optimal parameter selection strategies have been proposed. For example, [10] suggests co-prime parameters to achieve a promised discovery. In [11–14], scan and advertise intervals vary according to given sequences. There are also schemes specifically designed for BLE neighbor discovery [16–25]. These above approaches adopt different parameter evaluation methods. A pragmatic idea is realistic measurement[22, 24]. However, significant amount of work can be required in analysis when distinguishing the individual parameter's impact on performance [26]. Simultaneously, each of the sampling processes consumes physical wall-clock time that is supposed to repeat for more than thousands of times to alleviate the impact of randomness. To overcome the disadvantages in real-world experiments, model-based methodologies are proposed. Nevertheless, the existing modeling approaches can only produce the average or maximum latency values and cannot satisfy the need when the CDF of latency is important (see 3.2). In addition, there lack models that can provide parameter selection basis under more practical scenarios with stochastic factors that can significantly increase modelling complexity. Simulations, which focus on representing the discovery process and often preserve simplified analysis, can be a suitable methodology to cover these deficiencies.

[1]The open-source implementation is maintained at https://github.com/litonglab/blender-neighbor-discovery

While few studies have explored the development of simulation tools for BLE neighbor discovery, a primitive way of simulation is to randomly sample from all possible range-entrance cases, which however has some deficiencies (see §3.4). An alternative strategy based on discrete timing is proposed in [19]. We named it TMC17. Although TMC17 can produce a fixed CDF of discovery latency values without random sampling, stochastic factors are excluded from the simulation. This paper recuperates the above vulnerabilities. With consideration of stochastic factors (i.e., the packet loss and random advertising delay), Blender can produce the possibilities of each possible latency value and therefore form a determined latency CDF.

3 BACKGROUND AND MOTIVATION

3.1 Parameter Configuration Matters

In BLE neighbor discovery, there are interactions between an advertiser and a scanner. The advertiser broadcasts three consecutive Protocol Data Unit (PDU) respectively in three channels (i.e., CH37, CH38, CH39), which are referred to as an advertising event. The discovery succeeds when a BLE scanner achieves to capture the complete packet in an advertising event after the scanner enters the radio range of the advertiser.

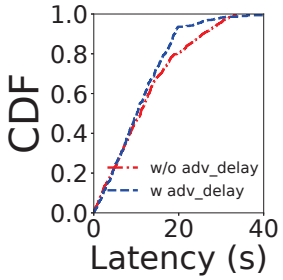
This ideally immediate success, however, can largely consume the battery capacity. In practice, BLE-like discovery protocols adopt duty-cycling in signal broadcasting and scanning. In other words, the devices would periodically switch off their radio activities, where the average ratio of the active time of a device is its duty-cycle. Apparently, with duty-cycling, discovery often cannot be completed as soon as the scanner enters the radio range of the advertiser, because the states (i.e., active or asleep) of the devices are random at this range-entrance event E_0 as the devices are equipped with unsynchronized clocks. Therefore, a discovery latency L is defined as the period from E_0 to the discovery.

Conceptually, more frequent advertising and scanning with higher energy consumption may lead to higher possibility for the advertiser and scanner to be simultaneously activated [27]. However, in BLE, high duty-cycle can also result in unwanted results, where an obvious example is the destined failure (i.e., infinite discovery latency) when the advertiser and scanner are running in parallel and are never activated at the same time. Since energy-saving is critical in mobile devices, parameter selection is of vital importance, which helps to filter out the inefficient parameters that consume high power but result in unacceptable discovery latency.

3.2 A Full Distribution of Latency Estimation is Required

In previous parameter selection tools for BLE neighbor discovery, the evaluation of the parameters focuses on the mean or the worst-case latency (i.e., the longest discovery latency under a parameter setting) [15]. However, a metric equipped with probabilistic features can be required. In some services such as proximity tracing [4, 5] and assets tracking [28, 29], the accomplishment of discovery becomes a expectation rather than a necessity due to the restriction of the length of time reserved for discovery.

Specifically, there is an expected rate $p\%$ of successful discovery within a given time L, where good parameters here are supposed



(a)

	Success Rate	
Latency	With *adv_delay*	Without *adv_delay*
<14 s	60%	60%
<20 s	95%	80%
<34 s	100%	98%

(b)

Figure 1: (a) Success Rate vs. Latency Restriction via Random Sampling ($T_a = 1975ms, T_d = 600ms, d_s = 60ms$). (b) Summary.

to reach this expectation. This quantified requirement is suitable to be represented through a CDF. In a latency CDF, each probability value p corresponds to a latency value L, where L is the p-percentile latency. Here, reaching a success rate of no lower than $p\%$ in L time is equivalent to that, the p-percentile latency being smaller than L. A full distribution of latency values is therefore necessary to provide any p-percentile latency.

3.3 Neighbor Discovery Simulation Suffers From Stochastic Factors

Packet loss rate. Packet loss is a common unideality in networking systems, especially in wireless networks [30, 31]. In BLE neighbor discovery process, channel collision is one of the major sources of packet loss. Despite the radio frequency interference, physical barriers can also increase the packet loss rate [32]. With duty-cycling, the opportunity of an advertise event occurring during an active scan window become rare. However, if this opportunity occurs with packet loss, the discovery that should be successful would fail. The system will have to wait until the next such opportunity before trying again to complete a discovery, which prolongs the discovery latency.

Random advertising delay. In practical BLE neighbor discovery, the interval between two consecutive advertise events is not a constant, which is longer than the settled advertise interval by a random period of time within $10ms$, called the *adv_delay*. With *adv_delay*, the advertise events are shifted from the original duty-cycled positions. The consequence of this shift can be complex. The scanner would be able to catch some advertise events that originally are not covered by a scan window, and simultaneously fails some other originally achievable discoveries.

While theoretical analysis is vulnerable to predict the impact of *adv_delay* on the CDF of discovery latency, simulation is a direct and efficient way to quantify the significance of this influence. A preliminary result from random sampling is shown in Figure 1(a), where the CDF figure of the same scan and advertise setting can be significantly changed by adopting *adv_delay*. As summarized in Figure 1(b), these changes result in different rates of successful discovery with same latency restriction, which can mislead the evaluation of the applied parameter set.

3.4 Random Sampling is Far From Satisfactory

Although random sampling enjoys convenience in implementation through focusing on reproducing the discovery process and omitting the analysis on the specific effects produced by the parameters or stochastic factors, there are still drawbacks to this approach.

Misjudgment in Parameter Selection. With random sampling, the results often vary in two simulations. This uncertainty of output is the nature of random sampling, and can incorrectly rule out some parameters. In production, parameters that cannot reach the threshold of the given success rate would be excluded by automated scripts.However, deviation in the CDF can be caused by random sampling and lead to fallacious drop of acceptable parameters, which is non-negligible especially when there are few candidate parameters. The only way to alleviate this deviation is to expand the sample size, which nevertheless results in linear growth of computation time.

Vulnerability in Distinguishing Individual Impacts. In the discovery process, the discovery latency is affected by a pile of variables, including the range-entrance situation, the scan and advertise duty-cycles, and the stochastic factors. While random sampling works through imitating the advertiser and scanner's consecutive behaviors, all the impacts of these variables are mixed together. A systematical analysis can help reveal individual variable's impact, which can further benefit the more precise modeling in practical scenarios.

Based on the observations above, we present the practical simulation framework, Blender, that simultaneously has the ability to provide precise, computationally efficient, and full simulation of the BLE neighbor discovery systems.

4 BLENDER OVERVIEW

4.1 Concept of Discrete Simulation

Due to the complexity of analysis based on wall-clock, which is a continuous dimension, Blender adopts discrete modeling of time. In other words, the timeline is split into timestamps of a minimum unit and the advertise and scan actions only start and stop at these discrete timestamps. When the time unit converges to zero, the time system in Blender can represent the wall-clock. As in BLE neighbor discovery the scan/advertise intervals are often parameterized in hundreds to thousands of milliseconds, 1ms is settled as the default time unit, which preserves computational friendliness with acceptable loss of simulation accuracy as shown in the later realistic evaluation. The minimum time unit can shrink until it reaches the value of d_a, which is a period from dozens to hundreds of microseconds[19].A further shrunk time unit would conceptually reform a simulated advertise event as a period longer than single time unit, which breaks the framework's principle.

4.2 The Architecture of Blender

The architecture of Blender is displayed as a flow diagram in Figure 2, which contains the following essential modules.

Input and Output. In Blender, the input contains five parameters, the scan interval length T_s, scan window length d_s, advertise interval length T_a, the packet loss rate F_p, and the maximum *adv_delay* value R_d settled for advertiser (10ms by default). The simulation's

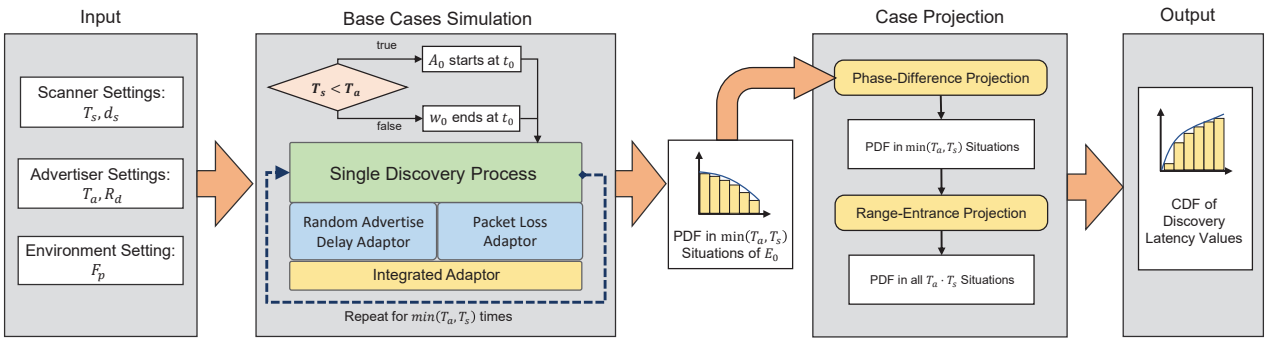


Figure 2: Overview of Blender Architecture

output is a cumulative probability distribution of all possible discovery latency values.

Base Case Simulation. The input parameters are first passed into the base cases simulation module, which considers $min(T_a, T_s)$ range-entrance cases (i.e., base cases) out of $T_a \cdot T_s$ ones (see §5.1). These base cases have either the first advertise event or the (end of) first scan window fixed at the range-entrance event, depending on the relationship between T_a and T_s. In each of the base cases, a single discovery process produces either the corresponding discovery latency, or a PDF of all possible latency values under current case if stochastic factors are considered.

Case Projection. While the base cases simulation module only considers a small partition of all possible range-entrance cases, the relationships between the base cases and all other cases can be analyzed. The case projection module can generate the latency PDF of all cases by time-efficient calculation (e.g., single addition) from the PDF produced by the base case simulation, from which the desired output of Blender (i.e., the latency CDF) can be derived.

Packet Loss Adaptor. When $F_p > 0$, each single discovery process needs to be equipped with a packet loss adaptor. With the packet loss adaptor, the result from each base case becomes a latency PDF rather than a latency value.

Random Advertising Delay Adaptor. Similar to packet loss adaptor, the result from each base case is also converted to a latency PDF when an *adv_delay* adaptor is active (i.e, $R_d > 0$). In addition, when packet loss is simultaneously considered, an integration of the two adaptors is specifically designed (see Algorithm 2).

5 DESIGN DETAILS

This section presents the design details of specific modules in Blender. We first introduce the base case simulation and the single discovery process in §5.1. Then in §5.2, the case projection module that can produce the complete CDF is illustrated, which is the key to optimizing the simulation's time complexity. Finally, the adaption methods of the packet loss and the random advertise delay are discussed in §5.3 and §5.4.

5.1 Base Case Simulation

Before diving into the base cases simulation, we introduce the single discovery process as shown in Figure 3. In such a process, two new attributes φ_a and φ_s that can describe a range-entrance

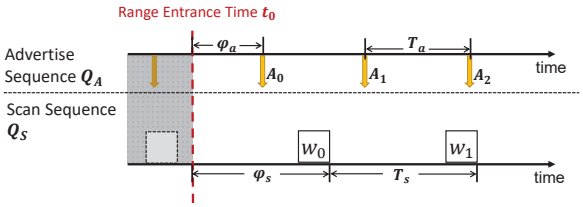


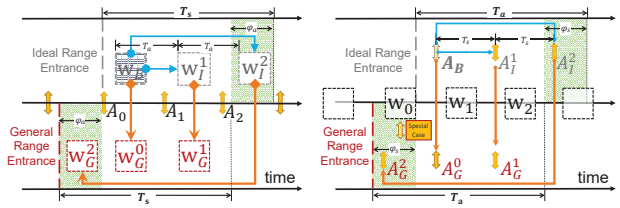
Figure 3: Attributes in a Single Discovery Process

case are introduced, which respectively annotate the time from the range-entrance event to the first following advertise event and the end of the first following scan window. With T_a, T_s, φ_a, and φ_s, an advertise sequence Q_A and a scan sequence Q_S can be formed, where Q_A is a list containing the timestamps (relative to the range-entrance event) of advertise events and Q_S is the one containing the scan windows' active time. A time sweeper similar to [19] will then traverse through Q_A and Q_S from range-entrance, and the time from range-entrance to an event covered in both Q_A and Q_S (when no packet loss considered) becomes the discovery latency $L(\varphi_a, \varphi_s)$.

The base cases simulation runs the single discovery process for different (φ_a, φ_s) pairs. Totally, there are $\varphi_a \cdot \varphi_s$ range-entrance cases with $\varphi_a \in [0, T_a)$ and $\varphi_s \in [0, T_s)$. If φ_a overflows from this range, the actual first advertise event will occur T_a earlier than A_0 and this A_0 will be A_1 (i.e., the second advertise event after range-entrance). The deduction for φ_s is in the same way. In the base cases simulation, either φ_a or φ_s will be fixed to be zero depending on whether $T_a > T_s$ (for case production in §5.2, where the unfixed one keeps ranging and forms $min(\varphi_a, \varphi_s)$ pairs of (φ_a, φ_s), which are the base cases.

5.2 Case Projection

Although a complete CDF of discovery latency can be generated through considering all $T_a \cdot T_s$ cases as base cases, it can often require the time-consuming single discovery process to execute millions of times. Considering the further time consumption when float number calculations are introduced along with random factors, a time reduction method is presented in this section with the principle to derive the latency values in the remaining cases through simple calculations.



(a) $T_a < T_s$ (b) $T_a > T_s$

Figure 4: Examples of the equivalence-relation-based Case Projection.

We first discuss the condition when $T_a < T_s$. As an initial step, φ_a is fixed to 0, which is annotated as Ideal Range-Entrance and infers that the first advertise event A_0 in Q_A occurs immediately at the range-entrance event. The first scan window w_0 has T_s possible positions depending on the value of φ_s. Each of these positions is alternatively described by $t_{w_0} \in (0, T_s]$, where t_{w_0} is the closing timestamp of w_0 relative to range-entrance. By passing a t_{w_0} as range-entrance situation into simulation, a corresponding timestamp D_{w_0} (relative to t_0, the timestamp of the range-entrance event) where the discovery occurs can be produced. As $T_a < T_s$, there must be a pair of $t_{w_0^i}$ and $t_{w_0^{i+1}}$ that are T_a distance apart, whose difference to A_i and A_{i+1} are the same. Thus, the Q_S represented by $t_{w_0^i}$ has the same phase difference to Q_A as that represented by $t_{w_0^{i+1}}$. In other words, the wake-up and sleep schedule of scanner and advertiser are identical after $t_{w_0^i}$ and $t_{w_0^{i+1}}$, indicating that $D_{w_0^i} - t_{w_0^i} = D_{w_0^{i+1}} - t_{w_0^{i+1}}$. We can then have the following statement.

Theorem 5.1. *When A_0 is fixed at the range-entrance event, for every $t_{w_0^i} \in (T_a, T_s]$, the discovery latency can be represented as*

$$\begin{aligned} L(0, t_{w_0^i}) &= D_{w_0^i} = D_{w_0^{i-1}} - t_{w_0^{i-1}} + t_{w_0^i} \\ &= D_{w_0^{i-1}} + T_a = L(0, t_{w_0^{i-1}}) + T_a, \end{aligned} \tag{1}$$

where $L(x, y)$ is the discovery latency with $\varphi_a = x$ and $\varphi_s = y$.

Based on Equation (1), the latency of all Ideal Range-Entrance cases $L(0, t_{w_0^i})(t_{w_0^i} \in (T_a, T_s])$ can be derived from those of the base cases $L(0, t_{w_0^0})(t_{w_0^0} \in (0, T_a])$ as shown in Figure 4(a) (the Phase-Difference Projection):

$$L(0, t_{w_0^i}) = L(0, t_{w_0^0}) + i \cdot T_a (i = \lfloor \frac{t_{w_0^i}}{T_a} \rfloor) \tag{2}$$

After considering all phase differences between Q_A and Q_S with fixing $\varphi_a = 0$, the next procedure is to involve the phase difference between Q_A and range-entrance by left-shifting the range-entrance event. Cases with this shifted range-entrance event are referred to as General Range-Entrance cases. After shifting A_0 away from range-entrance as shown in Figure **??**, $t_{w_0} \in (0, T_s]$ is still valid. Compared to the original interval of t_{w_0} when $\varphi_a = 0$, this new interval includes a new range $(0, \varphi_a]$ and excludes $(T_s, T_s + \varphi_a]$. For the other $t_{w_0} \in (\varphi_a, T_s]$, the discovery latency can be directly derived from those produced by (2). Also, each $t_{w_0} \in (0, \varphi_a]$ can be projected to the excluded range $(T_s, T_s + \varphi_a]$, where the discovery latency values have already been produced as well. Specifically, we have the following statement as the Range-Entrance Projection:

Theorem 5.2. *$\forall t_{w_0^i} \in (0, T_s]$, $\varphi_a \in (0, T_a)$, the discovery latency can be represented as*

$$L(\varphi_a, t_{w_0}^{\varphi_a}) = L(0, t_{w_0}^0) + \varphi_a, \tag{3}$$

where $t_{w_0}^0$ and $t_{w_0}^{\varphi_a}$ are in the same Q_S on wall-clock time (In other words, $t_{w_0}^0 + \varphi_a \equiv t_{w_0}^{\varphi_a} \ mod \ T_s$).

Proof. *For every $t_{w_0}^{\varphi_a} \in (\varphi_a, T_s]$, the corresponding positions of the first scan window relative to Q_A is the same as the positions represented by $t_{w_0}^0 \in (0, T_s - \varphi_a]$, which results in the same discovery time from the window. While the discovery latency $L(0, t_{w_0}^0)$ is relative to the original range-entrance (i.e., A_0) when $\varphi_a = 0$, the new latency $L(\varphi_a, t_{w_0}^{\varphi_a})$ will be φ_a larger than the former one because of the φ_a left-shift of the range-entrance event. For $t_{w_0}^{\varphi_a} \in (0, \varphi_a]$, a subsequent $t_{w_1}^{\varphi_a} = t_{w_0}^{\varphi_a} + T_s$ can be found in $(T_s, T_s + \varphi_a]$, which represents a w_1 in the same Q_S as w_0. Since no advertise event occurs during $[0, \varphi_a)$, it is certain that a successful discovery cannot occur in W_0. Hence $L(\varphi_a, t_{w_0}^{\varphi_a})$ is determined by subsequent scan windows in the same Q_S. Thus, we can derive that*

$$L(\varphi_a, t_{w_0}^{\varphi_a}) = L(\varphi_a, t_{w_1}^{\varphi_a}) = L(0, t_{w_0}^0) + \varphi_a \tag{4}$$

when $t_{w_0}^0 \in (T_s - \varphi_a, T_s]$.

By combining Equations (2) and (3), a formula is derived to get all $T_a \cdot T_s$ discovery latency values from $\{L(0, t_{w_0^i}) | t_{w_0^i} \in (0, T_a]\}$. Theoretically, the trivial case $T_a = T_s$ can also cope with this formula (and the precedent ones), which is however commonly not suggested in parameter tuning because it can often lead to parallel advertising and scanning and a high probability of infinite discovery latency.

Theorem 5.3. *When $T_a < T_s$, given T_s, T_a, d_s, any case (φ_a, φ_s) can be projected to T_a base cases and retrieve its discovery latency by:*

$$\begin{aligned} &L_{T_s,T_a,d_s}(\varphi_a, \varphi_s) = \\ &\quad L_{T_s,T_a,d_s}(0, (\varphi_s - \varphi_a + T_s) \ mod \ T_s \ mod \ T_a) + \\ &\quad \varphi_a + \lfloor \frac{(\varphi_s - \varphi_a + T_s) \ mod \ T_s}{T_a} \rfloor \cdot T_a \end{aligned} \tag{5}$$

When $T_a > T_s$, the simulation and latency value projection procedures can be re-applied with switching roles of Q_A and Q_S. For example, now t_{w_0} is initially fixed at the range-entrance, which produces a function to project the cases $A_0^0 \in [0, T_s)$ to $A_0^i \in [T_s, T_a)$ similar to Equation (2) where the attributes for scan and advertise are exchanged.

When the range-entrance starts to left-shift from w_0, a modification is required to resolve a special case as shown in Figure 4(b). When $T_a < T_s$ for any φ_a, no discovery can occur in $[t_0, t_0 + \varphi_a)$ since no advertise event exists in this interval. When $T_a > T_s$, however, a scan window can cover a partition of $[t_0, t_0 + \varphi_s)$. If A_0 occurs in that time period, $L(A_0, \varphi_s)$ will not be equivalent to $L(A_1, \varphi_s)$ with A_1 as the subsequent advertise event in the same Q_A as A_0, but become φ_a instead. This affects Equation (3), which together with the above reformed Equation (2) constructs the projection function below.

Theorem 5.4. *When $T_a > T_s$, given $T_s, T_a, d_s, \varphi_a, \varphi_s$, the discovery latency can be projected to T_s pre-calculated latency values from simulation by: $L_{T_s,T_a,d_s}(\varphi_a, \varphi_s) =$*

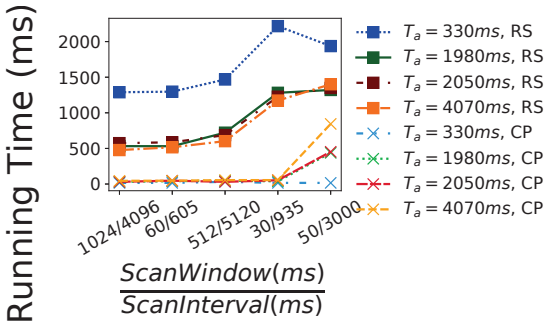


Figure 5: Running Time Comparison of Case Projection (CP) with Random Sampling (RS) (n=50000)

$$\begin{cases} \varphi_a, & \varphi_s > 0 \wedge 0 < \varphi_s - \varphi_a < d_s \\ L^* + \varphi_s + \lfloor \frac{(\varphi_a - \varphi_s + T_a) \bmod T_a}{T_s} \rfloor \cdot T_s, & otherwise \end{cases} \tag{6}$$

where $L^* = L_{T_s,T_a,d_s}((\varphi_a - \varphi_a + T_a) \bmod T_a \bmod T_s, 0)$.

To better understand the benefit of case projection, we conduct experiments to compare the running time between random sampling and case projection, in which 20 sets of scan and advertise parameters are run using Python implementation. Figure 5 shows the results. While the major drawbacks of random sampling include the long running time due to repetitive sampling before the result resembles the theoretical CDF, case projection can reduce the running time by 2-10 times.

5.3 Packet Loss Adaption

While the latency distribution under ideal conditions can be produced through the base cases simulation and case projection, the impact of packet loss on the discovery latency distribution is not considered. In this section, we make amendments to the ideal simulation in order to provide a loss-involved probabilistic distribution with a given packet loss rate F_p.

When an advertise event is covered by a scan window but fails to accomplish a discovery event due to packet loss, the discovery latency will grow until the time when a scan window covers another advertise event. This re-coverage is destined because Q_A and Q_S are periodic sequences, which indicates a common recurrence period of $lcm(T_a, T_s)$. After that period, an advertise event relocates in the same relative position δ_d in a scan window, which however may not be the only situation for a re-coverage to occur. As the scan window lasts for a period, the re-coverage can also occur when an advertise event is located in another relative position in the scan window. Therefore, the extent of growth of discovery latency at every loss-induced discovery failure depends on δ_d. Through fixing A_0 at t_0 and running a 'modified' (i.e., ignoring the discovery at t_0) simulation with all d_s possible w_0 that can cover A_0, an array indexed by the relative position δ_d can be formed to contain the corresponding latency growth values.

The next procedure is to add latency growth to the latency values in the original distribution of lossless situation. Through Equation (5) or (6), each of the $T_a \cdot T_s$ latency values are derived from one of the initially produced $min(T_a, T_s)$ values and, preserves the same δ_d. Therefore, it is appropriate to only add latency growth to the $min(T_a, T_s)$ values while the remaining values can be derived through the original projection. Under $T_a < T_s$, the process of latency addition is operated as follows.

First, retrieve the set of lossless latency values $S_{ideal} = \{L(0, t_{w_0}) \mid t_{w_0} \in (0, T_a]\}$. This time the corresponding δ_d on discovery is produced along with the latency values.

Second, through using the d_s possible w_0 candidates that can cover an A_0 at t_0, produce a set of discovery latency values ignoring the coverage at t_0 together with the corresponding $\delta_d^{'}$ on discovery. In other words, all possible intervals between two adjacent discovery (without loss) and δ_d situations at the two sides are calculated and stored in $I = \{(L(0, \delta_d), \delta_d^{'}) | \delta_d \in [0, d_s)\}$.

Third, for each $L(0, t_{w_0})$ in S, the increased latency value $L_1(0, t_{w_0})$ caused by the failure of the first discovery can be calculated as $L(0, t_{w_0}) + I_L$ where I_L is the latency value in a $(L(0, \delta_d), \delta_d^{'})$ pair, whose δ_d is identical to the corresponding δ_d of $L(0, t_{w_0})$. As the success of discovery follows binary distribution and thus the probability of failing the i^{th} discovery is $(1 - F_p)^{i-1} F_p$, the weight of value $L_1(0, t_{w_0}, F_p)$ in the result set is increased by F_p. The other values that can be derived from $L_1(0, t_{w_0})$ through (5) are also increased in weight by F_p. If the second discovery attempt fails again, a new I_L determined by current δ_d at the first discovery attempt will be selected to grow the latency. This will be iterated until the discovery latency reached a pre-settled limitation. Notice that under $T_a > T_s$, the only amendment is to repeat the third step for the special cases in Figure 4(b).

5.4 Random Advertising Delay Adaption

The *adv_delay* introduced in BLE brings randomness to the periodic intervals of an advertiser. This randomness first affects A_0, where the range of φ_a expands to $[0, T_a + R_d)$ as the largest possible advertise interval is now $T_a + R_d$. However, the probabilities of φ_a being each value in its range are no longer identical. The advertise interval T_{A_0} right before A_0 now has R_d candidate values with equal possibilities. For each possible T_{A_0}, φ_a follows a uniform distribution in $[0, T_{A_0})$, which can always cover $[0, T_a)$. Therefore, the probability distribution of φ_a can be divided into two parts, $\varphi_a \in [0, T_a)$ and $\varphi_a \in [T_a, T_a + R_d)$. The total possibility of the second part, produced by simple analysis, can be at a magnitude of 10^{-3}, indicating that ignoring these possibilities could reduce computation time with negligible impact on the integrity of the simulation result. Therefore, Blender mainly focuses on modeling the effect of *adv_delay* on the advertiser's time control after t_0.

As there are various advertise intervals, Q_A formed with a given φ_a no longer consists of determined timestamps but probabilistic distributions for each advertise event in it. This uncertainty of the advertise events' position in time may result in various discovery latency values following a specific PDF as an output of the latency producer. Therefore, the latency producer is attached with a so-called *adv_delay* Accumulation Module. We first focus at the new form of Q_A. While A_0 is at a determined timestamp, A_1 would occur inclusively between $A_0 + T_a$ and $A_0 + T_a + R_d$ with each discrete value having an occurrence possibility of $\frac{1}{R_d}$. Each possible timestamp of A_1 can derive a range of possible timestamps of A_2 in similar way, where each values is equipped with a probability of $\frac{1}{R_d^2}$. By integrating all those ranges of A_2, the PDF of A_2's possible

Algorithm 1 The Summation Accumulator.

1: $accum\_layers \leftarrow [[1, 1, 1, ..(totally\ R_d + 1\ ones).., 1]]$
2: $layer\_sums \leftarrow []$
3: **function** GetProbability( )
4: **Require**: $i\ of\ A_i,\ ts\_pos$
5: **Compute**:
6: Run NextLayer() $until\ i > length(accum\_layers)$
7: $cur\_layer \leftarrow accum\_layers[i-1]$
8: **Return**:
9: $Probability \leftarrow \frac{cur\_layer[ts\_pos]}{layer\_sums[i-1]}$
10: **end function**
11: **function** NextLayer( )
12: **Compute**:
13: $last\_layer \leftarrow accum\_layers[-1]$
14: $new\_layer \leftarrow$ empty array of length $(length(last\_layer) + R_d)$
15: $new\_layer[j] \leftarrow Sum(last\_layer[j : j - R_d])$
16: $layer\_sums\ add\ layer\_sums[-1] * R_d$
17: **end function**

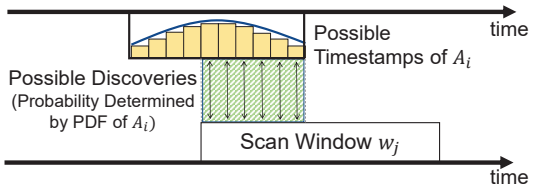


Figure 6: Probabilistic Discovery with *adv_delay*

timestamps (ranging in $[A_0 + 2 \cdot T_a, A_0 + 2 \cdot (T_a + R_d)]$) can be generated.

A difficulty in latency producing is that, as the probabilities of A_{i+1}'s timestamps are determined by those of A_i, it is hard to produce a list of PDF for $A_0, A_1, A_2...$ and retrieve the required probability from the list in each call of latency producer. The process to generate the probabilities of A_{i+1}'s candidate timestamps can be time consuming due to the exponential growth. We adopt a so-called Summation Accumulator (see Algorithm 1) as a solution. The algorithm stores the probability of possible positions for each A_i as the i^{th} layer, and dynamically derives the $i + 1^{th}$ layer.

While the timestamps of advertise events become nondeterministic with a single φ_a, the discovery events also become probabilistic. Figure 6 shows an example case of discovery judgement. An advertise event A_i is able to be discovered only when the timestamp range it belongs to has an intersection with a scan window. A discovery latency value can be produced by each of the timestamps in this intersection, and is attached with the same probability of A_i being at that timestamp. This intersection is then considered to be "discovered", probably leaving a range of A_i outside of it. This "undiscovered" range is then used to generate the probabilities of A_{i+1}'s candidate timestamps and the assignment of probability to latency continues in the same way as for A_i. A general procedure of discovery judgement is shown in Algorithm 2, which can be applied when loss is simultaneously considered. Specifically, line 6 and 7 process the situations without/with packet lost.

In addition, the case projection requires simple amendments when *adv_delay* is involved. For $T_a < T_s$, the Phase-Difference Projection (Equation 2 will no longer based on adding multiples of T_a, since the actual interval between the advertise events is probabilistically ranging from T_a to $T_a + R_d$. Therefore, the value to be added is a probabilistic one, whose probability can be calculated

Algorithm 2 Discovery with Loss and Delay.

1: Fetch cur_accum_layer and cur_layer_sum from current layer's status in Summation Accumulator
2: **Require**: $ts\_pos\ of\ A_i$
3: **Compute**:
4: **if** ts_pos covered by scan window **then**
5: $ts\_prob \leftarrow GetProbability(i, ts\_pos)$
6: $P(latency = A_i) += ts\_prob * (1 - F_p)$
7: $cur\_accum\_layer[ts\_pos] *= F_p$
8: **end if**
9: NextLayer()

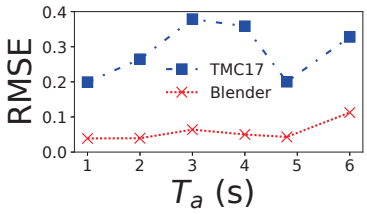
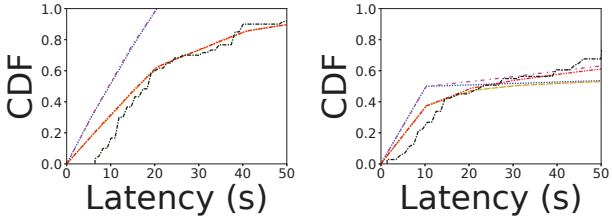


Figure 7: Overall Performance



(a) $T_a = 1860ms$ (Valley) (b) $T_a = 2050ms$ (Non-Peak)

Figure 8: The CDFs of the case study in which the configurations $T_s = 5120ms, d_s = 512ms$, and two T_a.

by the summation accumulator. Due to the same reason, the Range-Entrance Projection for $T_a > T_s$ also changes the addition of T_a to adding each value in $T_a + R_d$ with the same probability.

6 EVALUATION

To validate Blender's functionality, the simulation results are compared with the real experiment traces. Meanwhile, the simulation method TMC17 in [19] is also selected as the state-of-art approach for comparison. Note that the simulation in TMC17 only considers the ideal scenarios.

Setup. The practical testbed for real-world measurement consists of two Android smartphones, which are installed with a controller application that manipulates BLE signal broadcasting/scanning. The application is developed based on an official example [33] with an additional module. The module lets one phone advertise the BLE packets N times and another phone continuously scan the advertising channel. In this way, the scanner can estimate the channel's packet loss rate by checking the percentage of successfully received advertising packets over N.

Overall Performance. The root-mean-square error (RMSE) is introduced as the metric to quantify the CDFs of the simulation results and the measured results of the real testbed. Figure 7 shows the results with growing advertise interval under the *LOW_ENERGY* scan mode defined in [34]. When considering both stochastic factors, Blender results in much smaller RMSE values than the simulation

that only considers the ideal scenario (i.e., TMC17) in an environment with 5% – 45% packet loss rate, regardless of the selected advertising interval.

Case Study. Based on the analysis in [16], the average/maximum discovery latency values in BLE NDP have peaks and valleys with different parameter configurations. Figure 8(a) depicts the CDFs of several methods when an advertising interval that leads to a 'valley' is selected. While Figure 8(b) shows the CDFs when a 'non-peak' interval value is chosen. Although the gap between the two advertising interval values is less than 200ms , their corresponding latency performances are significantly different, where the valley results are better in p-percentile latency ($p > 50$). For both examples where the packet loss and the random advertising delay are considered, the simulation result of Blender shows better convergence to the Android measurement results than that of TMC17. Note that we have also examine the cases with other scan settings, and the results remain similar. In conclusion, Blender achieves improvement on the ability to approach the representation of practical BLE NDP situations in general cases compared to ideal simulation methodologies.

7 CONCLUSION

This paper proposes Blender to produce a full distribution of latency estimation, while avoiding the insufficiency of the legacy way of random sampling. To better fit the realistic estimation, Blender further takes into account packet loss rate and random advertising delay, which captures the key features of BLE-based neighbor discovery in practice. One of our ongoing work is applying Blender to the Offline Finding Network (OFN) [29]. OFN is a recent application introduced by industrial pioneers (e.g., Apple, Samsung, Tile, Nutspace and Nut Technology, Gigaset, etc.) that utilizes the nearby crowd-sourcing smartphones as scanners to conduct BLE neighbor discovery to find offline and lost BLE-enabled devices (e.g., earphones, tags, and watches). Blender acts as the simulation and evaluation tool for broadcasting/scanning parameters configuration, which is of critical importance in OFN services with the trade-off between power consumption and discovery latency.

ACKNOWLEDGMENT

Ding's work is partially done during the internship in Huawei. We would like to thank Kai Zheng, Wenjun Li, Zelin Chen, Baolong Li, Jiangshui Hu for their extensive and valuable feedback. Li's work is supported by the National Natural Science Foundation of China (No.62202473). Wang's work is supported by GRF 15210119, 15209220, 15200321, 15201322, ITF-ITSPITS/070/19FP, and C5018-20G.

REFERENCES

[1] Bluetooth specifications. https://www.bluetooth.com/, 2022.
[2] Bluetooth low energy devices market volume worldwide 2013-2020. https://www.statista.com/statistics/750569/worldwide-bluetooth-low-energy-device-market-volume/, 2022.
[3] Tong Li, Kai Zheng, Ke Xu, Rahul Arvind Jadhav, Tao Xiong, Keith Winstein, and Kun Tan. Revisiting acknowledgment mechanism for transport control: Modeling, analysis, and implementation. *IEEE/ACM Transactions on Networking*, 29(6):2678–2692, 2021.
[4] Huawei contact shield. https://developer.huawei.com/consumer/en/doc/development/system-Guides/contactshield-introduction-0000001057494465, 2020.
[5] Exposure notification. https://www.google.com/covid19/exposurenotifications, 2020.
[6] Meng Shen, Yaqian Wei, and Tong Li. Bluetooth-based covid-19 proximity tracing proposals: An overview. arXiv2008.12469, 2020.
[7] S Gallo, Laura Galluccio, Giacomo Morabito, and Sergio Palazzo. Rapid and energy efficient neighbor discovery for spontaneous networks. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 8–11, 2004.
[8] Zainab R Zaidi, Marius Portmann, and Wee Lum Tan. Analysis of link break detection using hello messages. In *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 143–150, 2011.
[9] Dimitrios Amaxilatis, Georgios Oikonomou, and Ioannis Chatzigiannakis. Adaptive neighbor discovery for mobile and low power wireless sensor networks. In *Proceedings of the 15th ACM International Conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 385–394, 2012.
[10] Prabal Dutta and David Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *ACM SenSys*, pages 71–84, 2008.
[11] Arvind Kandhalu, Karthik Lakshmanan, and Ragunathan Rajkumar. U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol. In *ACM/IEEE IPSN*, pages 350–361, 2010.
[12] Mehedi Bakht, Matt Trower, and Robin Hilary Kravets. Searchlight: Won't you be my neighbor? In *ACM Mobicom*, pages 185–196, 2012.
[13] Wei Sun, Zheng Yang, Keyu Wang, and Yunhao Liu. Hello: A generic flexible protocol for neighbor discovery. In *IEEE INFOCOM*, pages 540–548, 2014.
[14] Lin Chen, Ruolin Fan, Kaigui Bian, Mario Gerla, Tao Wang, and Xiaoming Li. On heterogeneous neighbor discovery in wireless sensor networks. In *IEEE INFOCOM*, pages 693–701, 2015.
[15] Philipp H Kindt and Samarjit Chakraborty. On optimal neighbor discovery. In *ACM SIGCOMM*, pages 441–457. 2019.
[16] Philipp Kindt, Daniel Yunge, Robert Diemer, and Samarjit Chakraborty. Precise energy modeling for the bluetooth low energy protocol. *arXiv preprint arXiv:1403.2919*, 2014.
[17] Andreina Liendo, Dominique Morche, Roberto Guizzetti, and Franck Rousseau. Ble parameter optimization for iot applications. In *IEEE ICC*, pages 1–7, 2018.
[18] Antonio Del Campo, Lorenzo Cintioni, Susanna Spinsante, and Ennio Gambi. Analysis and tools for improved management of connectionless and connection-oriented ble devices coexistence. *MDPI Sensors*, 17(4):792, 2017.
[19] Philipp H Kindt, Marco Saur, Michael Balszun, and Samarjit Chakraborty. Neighbor discovery latency in ble-like protocols. *IEEE TMC*, 17(3):617–631, 2017.
[20] Wha Sook Jeon, Made Harta Dwijaksara, and Dong Geun Jeong. Performance analysis of neighbor discovery process in bluetooth low-energy networks. *IEEE ToVT*, 66(2):1865–1871, 2016.
[21] Bingqing Luo, Feng Xiang, Zhixin Sun, and Yudong Yao. Ble neighbor discovery parameter configuration for iot applications. *IEEE Access*, 7:54097–54105, 2019.
[22] Texas Instruments. Measuring bluetooth low energy power consumption. In *Application Note AN092*, 2010.
[23] Jia Liu, Canfeng Chen, and Yan Ma. Modeling neighbor discovery in bluetooth low energy networks. *IEEE communications letters*, 16(9):1439–1441, 2012.
[24] Hojin Lee, Dongseok Ok, Joonsub Han, Iksoon Hwang, and Kangtae Kim. Performance anomaly of neighbor discovery in bluetooth low energy. In *IEEE ICCE*, pages 341–342, 2016.
[25] Keuchul Cho, Gisu Park, Wooseong Cho, Jihun Seo, and Kijun Han. Performance analysis of device discovery of bluetooth low energy (ble) networks. *Computer Communications*, 81:72–85, 2016.
[26] Bingqing Luo, Yudong Yao, and Zhixin Sun. Performance analysis models of ble neighbor discovery: A survey. *IEEE Internet of Things Journal*, 8(11):8734–8746, 2020.
[27] Johanna Nieminen, Carles Gomez, Markus Isomaki, Teemu Savolainen, Basavaraj Patil, Zach Shelby, Minjun Xi, and Joaquim Oller. Networking solutions for connecting bluetooth low energy enabled machines to the internet of things. *IEEE network*, 28(6):83–90, 2014.
[28] Mira Weller, Jiska Classen, Fabian Ullrich, Denis Waßmann, and Erik Tews. Lost and found: Stopping bluetooth finders from leaking private information. In *ACM WiSec*, page 184–194, 2020.
[29] Apple airtag. https://www.apple.com/sg/airtag/, 2021.
[30] Hui Xie and Tong Li. Revisiting loss recovery for high-speed transmission. In *IEEE WCNC*, pages 1987–1992, 2022.
[31] Tong Li, Ke Xu, Hanlin Huang, Xinle Du, and Kai Zheng. Wip: When rdma meets wireless. In *IEEE WoWMoM*, pages 177–180, 2022.
[32] Tong Li, Kezhi Wang, Ke Xu, Kun Yang, Chathura Sarathchandra Magurawalage, and Haiyang Wang. Communication and computation cooperation in cloud radio access network with mobile edge computing. *CCF Transactions on Networking*, 2(1):43–56, 2019.
[33] Android samples. https://github.com/android/connectivity-samples/tree/main/BluetoothAdvertisements, 2021.
[34] Android ble documentation. https://developer.android.com/reference/android/bluetooth/le/ScanSettings, 2022.