Toward Optimal Broadcast Mode in Offline Finding Network

Tong Li, *Member, IEEE,* Yukuan Ding, Jiaxin Liang, Kai Zheng, *Senior Member, IEEE,* Xu Zhang, Tian Pan, Dan Wang, *Senior Member, IEEE,* Ke Xu, *Fellow, IEEE*

***Abstract*—This paper proposes ElastiCast, a novel Bluetooth Low Energy (BLE) broadcast mode that reduces the neighbor discovery latency in offline finding networks (OFNs). ElastiCast adapts the broadcast mode of the lost devices to the scan modes of the finder devices, considering their diversity. We start with an overview of OFNs, followed by a detailed analysis of the issues and challenges of existing solutions, which motivates the design of ElastiCast. Then we provide Blender, a simulator that models the neighbor discovery behavior of different broadcasters and scanners. By adopting Blender, ElastiCast can be implemented with three components: Local Optima Estimation, Common Interest Extraction, and Interval Multiplexing, in which we capture the key features of BLE neighbor discovery and globally optimize the broadcast mode interacting with diverse scan modes. Experimental evaluation results and commercial product deployment experience demonstrate that ElastiCast is effective in achieving stable and bounded neighbor discovery latency within the power budget.**

***Index Terms*—Offline Finding Network, Neighbor Discovery, Bluetooth Low Energy, Scan Mode Diversity**

I. Introduction

Mobile devices, which are essential in our daily lives, might go missing at a time. Thus some wearable devices (e.g., My Buddy Tag, Moochies, Pocket Finder) are equipped with the feature of position tracking. These devices, however, require Internet access and are often costly and power-hungry, due to the complicated built-in communication components. To tackle this issue, some industrial pioneers (e.g., Tile, Nutspace and Nut Technology, Apple, and Samsung) have developed offline finding networks (OFN) that leverage nearby online devices (a.k.a., finder devices) to help users locate their lost devices even when the devices are offline. For instance, Apple has over 1 billion iPhones that already have the Find My application [1] installed. These iPhones have formed one of the largest crowd-sourced OFNs worldwide [2]. Apple's AirTag [3] is a tiny metal tracker that works with the Find My application, and can be attached to a keychain, dropped in a bag, or snapped onto luggage to keep track of these items. AirTag, together with its OFN service, is predicted as Apple's next billion-dollar business [4].

Nowadays, OFN remains in its infancy with plenty of unsettled issues. To the best of our knowledge, almost all prior works focus on privacy and security analysis of OFN systems [5]–[9]. Their goal falls into the category of enabling crowd search without leaking private data. This paper does not focus on the security and privacy issues of OFNs. Instead, we take a first step toward understanding the OFN framework (see §III) and identify its design challenges from the perspective of performance. Particularly, in OFN applications, the user experience is closely related to the success ratio (probability) of finding the lost device, which is significantly affected by the neighbor discovery latency (see §IV-A). Neighbor discovery is the prerequisite stage of OFN, in which a finder device tries to establish contact with the lost device in the BLE signal range. It involves interactions between broadcasters and scanners in a duty-cycling paradigm with three parameters: the broadcaster's broadcast interval (A), the scanner's scan window (W), and the scan interval (T).

Existing works typically focus on optimizing point-to-point BLE broadcasts and scans [10]–[18]. These prior works, however, fall into the category of setting broadcast mode in the case of homogeneous neighbor discovery, where all the scanners are with the same scan mode. Among them, Kindt et al. [18] have proposed the tight duty-cycle-dependent bounds on the worst-case discovery latency that no prior neighbor discovery parameter setting approaches can beat, and concluded there is no further potential to improve the relationship between latency and duty-cycle in homogeneous neighbor discovery. In this paper, we argue that it is still possible to optimize discovery latency with a power budget in OFN due to the *scan mode diversity* of finder devices.

The finder devices are composed of a group of users

This work is supported in part by the National Natural Science Foundation of China (NSFC) under No. 62202473, No. 62441230 and No. 62372053, the National Science Foundation for Distinguished Young Scholars of China under No.62425201, the Science Fund for Creative Research Groups of the NSFC under No. 62221003, the Key Program of the NSFC under No. 61932016 and No. 62132011. Dan Wang's work is supported by RGC GRF 15200321, 15201322, 15230624, ITC ITF-ITS/056/22MX, ITF-ITS/056/22MX, ITS/052/23MX, and PolyU 1-CDKK, G-SAC8. *(Corresponding author: Jiaxin Liang.)*

T. Li is with the DEKE Lab and Information School, Renmin University, Beijing, China, 100872. E-mail: tong.li@ruc.edu.cn

J. Liang and K. Zheng are with the 2012 Labs, Huawei, Shenzhen, China, 518129. E-mail: {jiaxin.liang, kai.zheng}@huawei.com

Y. Ding is with the Electrical Engineering, Mathematics and Computer Science (EEMCS), Delft University of Technology, Delft, The Netherlands, 2600 AA. E-mail: y.ding-1@tudelft.nl

X. Zhang is with the School of Electronic Science and Engineering, Nanjing University, Nanjing, China, 210023. E-mail: xzhang17@nju.edu.cn

T. Pan is with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China, 100876. E-mail: pan@bupt.edu.cn

D. Wang is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China, PQ708. E-mail: dan.wang@polyu.edu.hk

K. Xu is with the Department of Computer Science and Technology, Tsinghua University, Beijing, China, 100084. E-mail: xuke@tsinghua.edu.cn

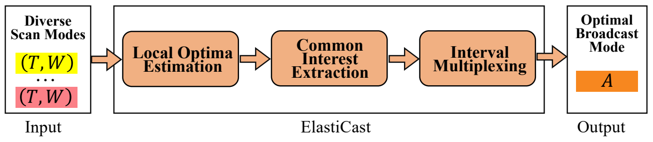


Fig. 1: Key modules in ElastiCast.

that are close to and within the Bluetooth signal coverage of the lost device. From the perspective of the operators in the ecosystem of OFN, the broadcast mode of the lost device is usually controllable (i.e., private brand), while the scan modes of finder devices are uncontrollable. The users of finder devices are crowd-sourced and are probably in different power modes (e.g., power-saving mode and high-performance mode) or from different manufacturers (e.g., Samsung, Huawei, Oppo, Vivo, and Xiaomi from the Android ecosystem). Thus, the scan modes of the finder devices are diverse. The challenge is to find the optimal broadcast mode (pattern) that adapts to the diversity of the uncontrollable scan modes (see §IV-D).

The broadcast mode affects both discovery latency and power consumption. On the one hand, a large broadcast interval may result in a large discovery latency, significantly reducing the possibility of finding lost devices (see §IV-A). On the other hand, the power consumption by the broadcaster is inversely proportional to the broadcast interval. For example, with the default broadcast interval 2000 ms [19], the AirTag's CR2032 lithium coin battery life lasts for one year [3]. Still, it may be halved if the broadcast interval is reduced to 500 ms (see Table I). Hence, a smaller broadcast interval means higher power consumption, reducing the applicability of OFN systems.

It seems intuitive that choosing the minimal A within the power budget would maximize discovery. However, multiple broadcast intervals yield local minimum latency for a given W and T on the scanner (see Figure 6). Therefore, choosing the minimal A may not always maximize discovery. On the other hand, the conventional method of neighbor discovery that adopts a fixed broadcast mode is far from satisfactory, as it may result in suboptimal performance of neighbor discovery in the case of scan mode diversity. In other words, its discovery latency is not bounded within the power budget (see §IV-E). This paper presents a brand-new broadcast mode called `ElastiCast`. As shown in Figure 1, ElastiCast deals with the scan mode diversity issues through the following modules: Local Optima Estimation, Common Interest Extraction, and Interval Multiplexing.

Local Optima Estimation uses Blender (originally proposed in [20]), a neighbor discovery simulator that mimics the behavior of broadcasters and scanners. Differing from the conventional method of *random sampling*, Blender uses specifically designed models to generate a deterministic full distribution of latency (see §IV-B and §VI-A). The input of estimation contains multiple settings of scan modes among all possible finder devices, and the output is the set of latency distributions as the functions of the broadcast interval.

Common Interest Extraction exploits the non-linear relationship between discovery latency and power consumption (see §IV-C). It selects the common broadcast intervals that achieve minimized discovery latency among all scan modes. However, simply choosing a single optimal broadcast interval might introduce bias due to the random advertising delay (i.e., adv_delay), a pseudo-random value with a range of 0 ms to 10 ms generated for each broadcast event (see §VI-B).

Interval Multiplexing overcomes the hurdles caused by adv_delay. It uses a mixture of multiple feasible broadcast intervals instead of a single one. It advances toward the global optima in the cases where adv_delay is non-negligible (see §VI-C).

Experimental evaluation results show that ElastiCast is effective in realizing stable and low-latency BLE neighbor discovery within the power budget in the case of scan mode diversity. Specifically, compared to the conventional method of local optima, ElastiCast reduces the discovery latency by 50% to 90% within the power budget in our case studies. Our commercial product applying ElastiCast is also improves the success rate by over 11% compared to the state-of-the-art AirTag in a real-world deployment. This can be attributed to the elasticity of ElastiCast, which adapts well to the scan mode diversity by searching for the best broadcast pattern that achieves the globally minimized discovery latency within the power budget (see §VIII).

The rest of this paper is organized as follows. §II reviews the related work. §III introduces the background and framework of OFN. §IV explains the design rationale of ElastiCast. §V summarizes the design of ElastiCast together with the Blender simulator, followed by the detailed procedure of ElastiCast in §VI. We evaluate the performance of Blender in §VII and the performance of ElastiCast in §VIII, and conclude the paper in §IX.

II. Related Work

Offline Finding Networks. Prior studies on OFNs mainly focus on how to enable crowd search without leaking private data [5]–[8]. For example, Apple proposed the asymmetric key-based way that encrypts location data with public keys [3], [8]. Mira Weller et al. [5] proposed a symmetric key-based way, called PrivateFind, to provide end-to-end encryption of location data. Other works have conducted an in-depth analysis of security and privacy issues based on these two ways. For example, Heinrich et al. [6] challenge OFN's security and privacy claims and examine the system design and implementation for vulnerabilities through reverse engineering. Mayberry et al. [7] show that OFN's threat model for antitracking is dangerously incomplete and presents three strategies that a malicious tracker can use to evade detection by item safety alerts. Tonetto et al. [8] further present how OFNs can be used to estimate large groups of people, which is beyond its original purpose of tracking lost devices. In this paper, we do not focus on the privacy and security aspects of OFN systems. Instead, we take a first step towards understanding the OFN framework and highlight its design challenges from the perspective of neighbor discovery performance.

Neighbor Discovery. Efficient neighbor discovery aims to achieve the shortest possible discovery latency for a given power budget. To this end, a large number of broadcast and scan setting approaches [21]–[31] have been proposed for general wireless neighbor discovery. For instance, following slotted neighbor discovery protocols where time is divided into fixed intervals that communication occurs within, Hello [27] identifies overlaps in active slots and offers a framework for abstracting parameter settings in active slot selections. Hedis [30] provides a better solution in asymmetric situations where the devices are operating in different duty-cycles. Diff-Codes [28] decreases worst-case and median latency through maximizing the likelihood of two devices being awake at the same time. However, despite the necessity of synchronization between devices in slotted protocols, latency distribution is hardly observed in the optimization problem statements. Besides traditional slotted protocols, there are studies specifically designed for BLE neighbor discovery [10]–[18], [32], [33]. Examples include [10], [13], where realistic measurements are adopted as the basis for searching the best parameter configurations. Meanwhile, [11] proposes the Effective-Scan-Window-based modeling for slotless BLE neighbor discovery, which, together with the subsequent works [12], [17], simulates and evaluates the discovery process. Kindt et al. [18] have proposed the tight duty-cycle-dependent bounds on the worst-case discovery latency and concluded there is no further potential to improve the relationship between latency and duty-cycle in homogeneous neighbor discovery. These prior works, however, fall into the category of setting broadcast mode in the case of homogeneous neighbor discovery, where all the scanners are with the same scan mode. To the best of our knowledge, this is the first work that captures the key features of OFN neighbor discovery where finder devices vary in different scan modes and overcome the challenges induced by scan mode diversity.

Discovery Latency Estimation. The correlation between parameter combinations of neighbor discovery, specifically BLE-like instances, and discovery latency is naturally complicated due to the flexibility of parameter selection, which arouses the requirement for calculating the discovery latency. The approaches adopt different methodologies including a pragmatic idea to conduct realistic measurements [10], [13]. To overcome the randomness and wall-clock time consumption in real-world experiments, more modeling-based methods are proposed, considering the limitation of theoretical analysis triggered by the complex interaction between the scanner and the broadcaster. The majority of recent modeling methods [34]–[36] utilize the *Chinese Remainder Theorem* (CRT) to calculate latency for every range-entrance situation. Besides, to either avoid redundant and inefficient calculations or reveal the impact of the parameter configuration on the discovery process, more analytical models have been proposed. For example, [37] builds the Partition model to group the offsets of the first scan window to the range-entrance event, and derives an average latency in terms of the number of broadcasting events before discovery. Unfortunately, the model is complicating the discovery analysis and is tailored for average and worst-case latency. Even if the model is reformed to produce the distribution of discovery latency, the fixed duty-cycle of the broadcaster that deeply entangles within the model and the derivation steps prevents the model to generalize to dynamic duty-cycle, including the broadcasting with random delay and the flexible broadcasting intervals. The Coverage Model from [18], though proposed earlier than many neighbor discovery models, manifests its conciseness in revealing the nature of the discovery process. It is utilized to formulate the minimum worst-cast latency at a given fixed duty-cycle or with flexible intervals but an expected duty-cycle. Nevertheless, the Coverage Model was delivered with excessive focus on the configurations that achieve the minimum possible worst-case latency, neglecting the other configurations as well as their performance assessment outside of the worst-case scenario. In other words, the potential of Coverage Model has not be fully leveraged by neither the original nor the subsequent studies.

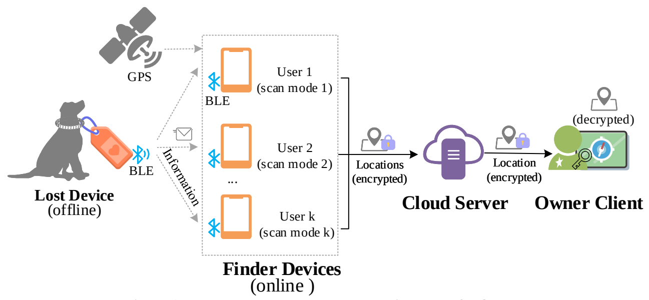


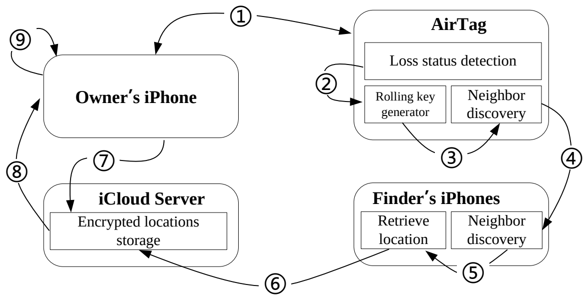
Fig. 2: Ecosystem overview of OFN.

III. Offline Finding Network: An Overview

The ecosystem of OFN consists of four types of roles. As shown in Figure 2, the *Lost Device* is usually a thin device (e.g., watch, headset, tag) without complicated built-in functions of communication and positioning. It can also be a rich device (e.g., phone, tablet) without online access. The *Finder Devices* are a group of users who are close to and within the Bluetooth signal coverage of the lost device. These devices are usually rich devices with built-in functions of communication and positioning, acting as volunteers to help the offline lost device report its location. The *Cloud Server* provides the storage service of the reported locations in the cloud. Since the location data is encrypted, the privacy of finder devices is protected. The *Owner Client* is usually a device with Internet access (e.g., phone, tablet) that decrypts the location data queried from the cloud server.

To understand how OFNs work in detail, we take Apple's Find My feature that works with Apple's iPhone and AirTag as a case study. The basic framework of Apple's Find My is illustrated in Figure 3. We can see that there are two additional components called *iCloud Server* and *Finder's iPhones* that help the *Owner's iPhone* find the lost *AirTag*.

For initialization, the owner binds AirTag to his/her iPhone, and the Find My application generates the Elliptic Curve key pair, whose public key is stored in the AirTag (**Step 1** in Figure 3). When the AirTag is detected and



Step 1: Generate public-private key pairs and bind AirTag to owner's iPhone
Step 2: Detect if the AirTag is lost. If yes, go to Step 3
Step 3: Generate rolling public keys periodically in minutes
Step 4: AirTag broadcasts public key as content via BLE and finder's iPhones recognize the key throughout neighbor discovery
Step 5: Retrieve current location and encrypt it with the broadcasted public key
Step 6: Upload the encrypted location record
Step 7: Generate the list of rolling public keys in the last days and query the iCloud server
Step 8: Return the encrypted locations records for the list of requested keys
Step 9: Decrypt location records with the private key and obtain an approximate location

Fig. 3: Basic framework of Apple's Find My.

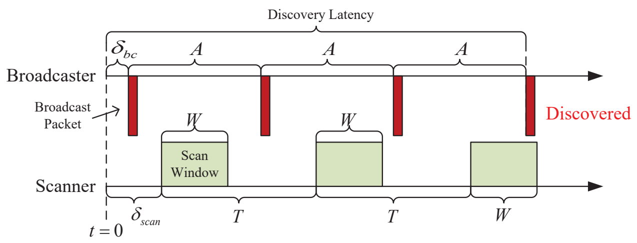


Fig. 4: The duty-cycling of BLE neighbor discovery. T, W, and A are the scan interval, scan window, and broadcast interval, respectively. δ_{bc} and δ_{scan} are the entrance time.

marked as lost (in **Step 2**), it generates rolling public keys periodically (e.g., 10 minutes) using a shared secret [38] (in **Step 3**). The AirTag broadcasts the public keys periodically (i.e, broadcast interval $A = 2000$ ms), and multiple nearby iPhones recognize and get the broadcasted public key via BLE neighbor discovery (in **Step 4**). These finder's iPhones retrieve their current location, encrypt the location with the public key (in **Step 5**) and upload the encrypted location record (in **Step 6**). To find the lost AirTag, the paired owner's iPhone generates the list of the rolling public keys that the AirTag would have used in the last days and queries the iCloud server (in **Step 7**). The server returns the encrypted location records (in **Step 8**) for the list of requested public keys, and finally, the owner's iPhone decrypts the location records with its private key and obtains an approximate location of the lost AirTag (in **Step 9**).

We have also investigated other OFNs. Although some of them (e.g., PrivateFind) adopt a different key management scheme, the basic workflow remains similar. Thus we have found that Apple's framework is representative in terms of the four major components of an OFN as illustrated in Figure 3, which is not surprising given it is a natural extension to the conventional finding networks with crowd-sourcing.

IV. Neighbor Discovery Latency of OFN: Issues and Challenges

This section discusses several observations on the issues and challenges in the neighbor discovery of OFN.

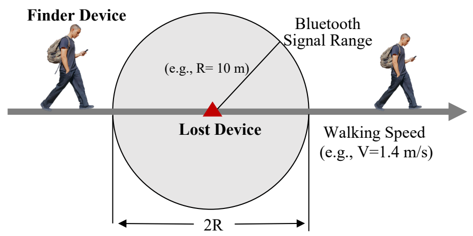


Fig. 5: Neighbor discovery between finder device and lost device in the walking scenario.

A. Neighbor Discovery Latency Matters in OFN

As the prerequisite stage of OFN, neighbor discovery is an essential process where a finder device seeks to first contact the lost device in the BLE radio range (i.e., **Step 4** in Figure 3). Generally, neighbor discovery involves the interactions between a broadcaster and a scanner, where the broadcaster broadcasts signals periodically and the scanner operates in a duty-cycling paradigm (see Figure 4). Neighbor discovery succeeds at the time when a scanner captures the complete packet in a broadcast event, which should occur immediately when the scanner device enters the radio range of the broadcaster device if the devices are scanning/broadcasting continuously. The discovery latency is measured from the time when both devices enter the range of reception. We refer to this outset as $t = t_0 = 0$ as in Figure 4, denoted by the entrance time or range-entrance event. From the range-entrance event, a broadcasting event for a packet to be broadcasted should occur after a period of δ_{bc}; a scanning window should occur after a period of δ_{scan}. Thereby, the discovery latency is defined as:

$$L = \delta_{bc} + \mathcal{N} \cdot A \tag{1}$$

where the $(\mathcal{N}-1)$-th broadcast packet is the first one, after the entrance time, overlapping within an active scan window. Specifically, a broadcast at time $t = \delta_{bc} + \mathcal{N} \cdot A$ will overlap with an active scan window if the following condition is satisfied:

$$(\delta_{bc} - \delta_{scan} + \mathcal{N} \cdot A) \bmod T < W$$

The modulo operation accounts for the periodic nature of the scanning schedule, effectively mapping the time difference into a single scanning interval. Therefore, $\mathcal{N}$ can formally defined as the minimal non-negative integer achieving the overlap:

$$\mathcal{N} = \min\left\{ n \in \mathbb{Z}_{\geq 0} \;\middle|\; (\delta_{bc} - \delta_{scan} + n \cdot A) \bmod T < W \right\}$$

As illustrated in Figure 4, the discovery latency is related to the broadcaster's broadcast interval (A), the scanner's scan window (W), and scan interval (T), where the scan duty cycle is computed by $D = \frac{W}{T}$. In general, power consumption is proportional to D and is inversely proportional to A, while a larger A or a lower D leads to an interleaved activity pattern between the broadcaster and the scanner, which may induce unacceptably large discovery latency [18], [39]. Since both the lost devices and the finder devices in OFN are power-sensitive [3], [40], the large discovery latency may significantly reduce the possibility of finding lost devices.

To explain this more clearly, we give an example of how a lost device is found by a finder device. As shown in

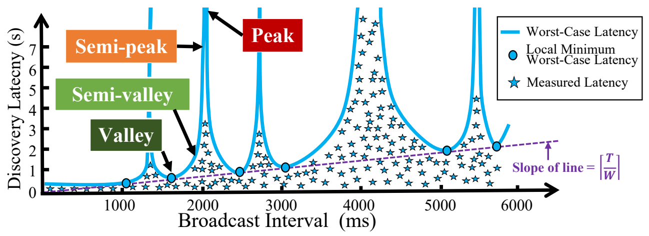


Fig. 6: The distribution of the worst-case discovery latency (upper bound). W = 1024 ms and T = 4096 ms (i.e., a representative scan mode labeled as BALANCED in modern Android systems [42]).

Figure 5, a person with a finder device passes the Bluetooth signal range (e.g., a cycle area with a radius of $R = 10$ meters [41]) of a lost device at a walking speed (e.g., $V = 1.4$ meters per second)[1]. Assuming the ideal case where a person walks along the diameter of the circle, the neighbor discovery latency should not exceed the time the person spends within the Bluetooth signal range (i.e., latency tolerance, $\frac{2R}{V} \approx 14$ seconds). Otherwise, the finder device fails to find the lost device, and the opportunity is wasted. Thus we conclude that neighbor discovery latency matters in OFN concerning the possibility of finding lost devices.

B. A Full Latency Distribution Requires Consideration

In previous parameter evaluation tools for BLE-like neighbor discovery, the rubric focuses on the mean or the worst-case latency (i.e., the longest discovery latency under a parameter setting) [18]. However, a metric equipped with probabilistic features can be required. In some services such as proximity tracing [43], [44] and assets tracking [5], [45], the accomplishment of discovery becomes an 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 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_P, which is the P-percentile latency. Here, reaching a success rate of no lower than $P\%$ in L time has an equivalent expression: the P-percentile latency being smaller than L. In order to provide any P-percentile latency, a full distribution of latency values is hence necessary. The importance and configuration of P is revisited in §V-B and §VI-B

While a series of previous works are mentioned to possess the functionality to generate full latency distribution, such distributions stem from an incomplete domain. As we claimed in §IV-A, the first scan window after the range-entrance event should also have a time offset δ_s from the range-entrance event rather than stick at time zero. While the previous distributions were generated upon a domain with a fixed first scan window, the refined range-entrance event will significantly enlarge the domain and can produce a more practical distribution. The refinement, however, cannot be directly applied to the existing methods. Since the simulation time for both CRT-based and effective-scan-window-based methods can be significantly prolonged due to their need to traverse the entire domain, this becomes impractical when a large number of parameter configurations need to be evaluated.

[1]A person's state of motion contains static scenario (e.g., still), low-speed scenario (e.g., walking), and high-speed scenario (e.g., running or riding vehicles). Since the static scenario can hardly suffer from performance issues, and the high-speed scenario is regarded as a rare case in OFN, in this paper we mainly focus on the most representative case of walking.

C. Discovery Latency is a Non-linear Function of Power Consumption

It has been well-studied that the trade-off between discovery latency and power consumption should be carefully handled to meet the application's required performance under the power constraint. Intuitively, discovery latency is inversely proportional to power consumption. However, this is only true when $A \leq W$, which is well studied in the prior studies [11], [46]. Recently, Kindt et al. [16] have exposed the non-linear relationship between discovery latency and power consumption when $A > W$. As shown in Figure 6, given a certain scan mode, i.e., scan window (W) and scan interval (T), there exists an upper bound of discovery latency in the worst case for each broadcast interval (A). As plotted in the blue line, the worst-case latency is a non-linear function of A.

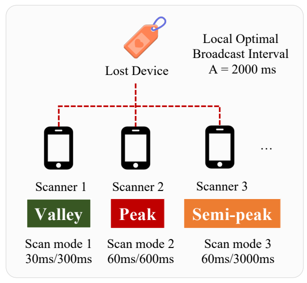
Since both broadcaster and scanner may randomly come into the range of reception, i.e., the range-entrance times δ_{bc} and δ_{scan} (see Figure 4) are the stochastic factors that decide the actual measured discovery latency. As a result, for a given A, the measured discovery latencies (plotted as blue stars) range from zero to the upper bound (i.e., worst-case latency).

In Figure 6, we further find that there exist multiple local minimum worst-case latencies (plotted as blue cycles). It is demonstrated in the prior work [18] that all the local minimum worst-case latencies follow a line with a slope of $\lceil \frac{T}{W} \rceil$. This reveals that the distribution of the worst-case latency varies with the scan mode. For the sake of description, we define that A is within "valley area" when A achieves the local minimum worst-case latency, and A is within the "semi-valley area", the "semi-peak area", and the "peak area", respectively, according to the increased latency compared to the corresponding local minimum worst-case latency.

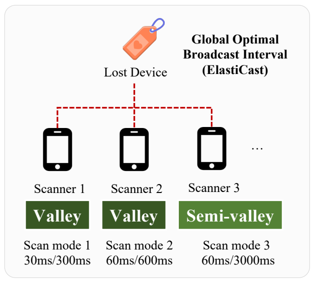
To summarize, the existence of multiple valley areas implies that we might fail to reduce discovery latency by simply reducing A. Instead, we need to "smartly" select A within valley areas while not violating the power constraint.

D. Finder Devices Vary in Scan Modes

As shown in Figure 2, the finder devices are composed of a group of crowd-sourced users that are nearby and under the Bluetooth signal coverage of the lost device. Hence the scan modes of finder devices are usually uncontrollable. As a result, these finder devices usually show diversity in scan modes. We further summarize the causes, i.e., dynamics and customization, of this diversity below.



(a) Legacy way of neighbor discovery



(b) Neighbor discovery applying ElastiCast

Fig. 7: ElastiCast aims to change from local optima to global optima.

Dynamics. By default, the Android Open Source Project (AOSP) supports three scan modes labeled as LOW_LATENCY, BALANCED, and LOW_POWER [42]. For a certain finder device, the scan mode might depend on whether the device is in power-saving mode or high-performance mode, whether the device screen is on or off, and whether the application is running in the foreground or background. For example, Android recommends applying LOW_LATECNY scan mode with a duty cycle of 100% only when the application is in the foreground, and LOW_POWER scan mode with a duty cycle of 10% when the application is in the background.

Customization. Some manufacturers might customize the scan mode in their products to achieve a better trade-off between scan frequency and power consumption. For example, iOS proposes a customized scan mode that scans 30 ms for every 300 ms in its OFN (i.e., Apple's Find My [3]), while HarmonyOS recommends a scan mode that scans 20 ms for every 600 ms in its HiLink protocols [47].

It is worth noting that forming a global finder network with different brands could help alleviate performance challenges brought about by scan mode diversity. However, both technical and non-technical issues arise. On the technical side, this approach cannot fully eliminate the diversity in scan modes. This is because even for phones of the same brand, the scanning mode may differ depending on whether the screen is on or off. On the non-technical side, each phone manufacturer aims to optimize the performance of its devices in specific scenarios, leading to the customization of their scan modes.

E. Scan Mode Diversity Results in Local Optima

As discussed above, for each scan mode we should carefully search an optimal broadcast interval to obtain the minimum upper bound of discovery latency. However, the distribution of the worst-case latency varies with the scan mode. As a result, an optimal broadcast interval for a certain scan mode might not be the optimal broadcast interval for another scan mode.

To explain this more clearly, in Figure 7 we give an example of neighbor discovery in the case of multiple scan modes. Specifically, three types of finder devices act as scanners, and the lost device acts as the broadcaster. Their scan modes are $30ms/300ms$, $60ms/600ms$, and $60ms/3000ms$, respectively. As shown in Figure 7 (a), when we set the broadcast interval A = 2000 ms according to the default settings in modern commercial products (e.g., AirTag), we find that A = 2000 ms is within the valley area of the scan mode 1, but A = 2000 ms is within the peak and semi-peak area of the scan modes 2 and 3, respectively. This reveals that the scan mode diversity results in local optima.

Based on these observations, the legacy way of neighbor discovery is far from satisfactory. In this paper, we aim to search for the global optimal broadcast pattern that makes the broadcast interval(s) within the valley or semi-valley area for all types of scan modes (see Figure 7 (b)). Thus we proposed ElastiCast as we will elaborate next.

V. ElastiCast Overview

The primary goal of ElastiCast is to achieve global optima by overcoming the challenges induced by scan mode diversity. In this section, we first formalize the problem of neighbor discovery with scan mode diversity, and then we introduce the framework of ElastiCast.

A. Problem Formalization

We model the overall performance of discovery latency by introducing the metric, *weighted average discovery latency*, denoted as $\hat{l}$. Given a set $\mathbf{S} = \{s_1, s_2, ..., s_n\}$ including n types of scan modes, where s_i is the i^{th} scan mode. Let ω_i and l_i be the market share and the discovery latency of the i^{th} scan mode, respectively. Here the market share represents the percentage of different types/states of finder devices in the market, which approximates the percentage of states of finder devices near a lost device. Then the weighted average discovery latency $\hat{l}$ is computed as follows

$$\hat{l} = \sum_{i=1}^{n} \omega_i \cdot l_i \tag{2}$$

As mentioned in the previous section, the duty cycle of a lost device cannot be vast due to power constraints. ElastiCast aims to minimize the weighted average discovery latency within the power budget. In order words, we aim to achieve

$$Z = \min \hat{l}, \quad s.t. \quad A \geq A_{min} \tag{3}$$

where A is the selected broadcast interval, and A_{min} is the minimum broadcast interval. As illustrated in Table I, the broadcaster's power consumption (battery life) increases proportionally with the broadcast interval. Hence, A_{min} can be equivalently regarded as the power budget of a broadcaster in this paper.

TABLE I: Battery life with different broadcast intervals.

Battery	Capacity	A	Battery Life
ER14250	2400 mAh	500 ms	60 months
CR2032	240 mAh	1000 ms	9 months
CR2032	240 mAh	2000 ms	12 months

It is worth noting that market share changes can affect the optimality of the undecided broadcast interval parameters in ElastiCast. This paper assumes that market share changes, calculated at a regional level, are relatively slow within most

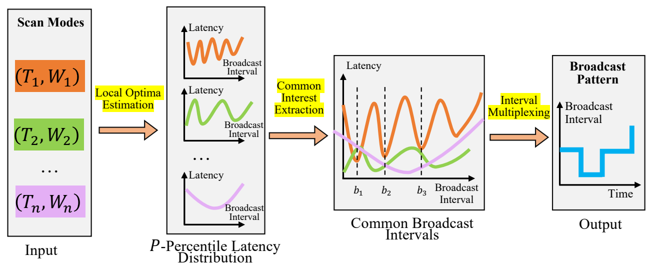


Fig. 8: The basic framework of data flow in ElastiCast.

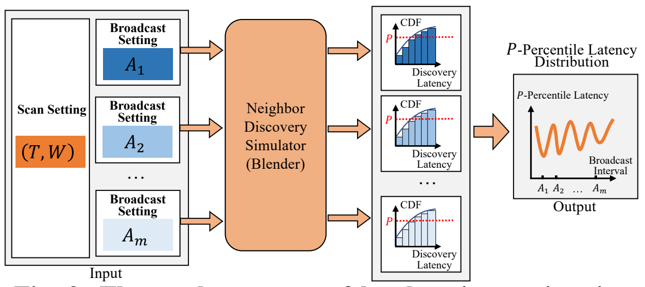


Fig. 9: The work progress of local optima estimation.

users' activity areas. However, as future work, we plan to offer a dedicated app, which will run on smartphones and connect to the TAG to configure its broadcast mode based on the dynamic market share of scanning modes.

B. The ElastiCast Framework

The framework of ElastiCast is displayed as a data flow diagram in Figure 8, which contains the input/output and has three intermediate modules primarily. In this section, we briefly introduce the design rationale of each module as below.

Input and Output. When applying ElastiCast in the neighbor discovery of OFN, the input contains multiple settings of scan modes among all possible finder devices. The output is the broadcast pattern of the lost device by multiplexing the feasible broadcast intervals.

Local Optima Estimation. For each scan mode, given a range of the broadcast intervals, the goal of this module is to generate the P-percentile ($P \in [0,\ 100]$) discovery latency from the latency distribution[2]. A straightforward way is to repeatedly conduct real-world experiments [10], [13] which have low efficiency and a high cost, much less the bias induced by wireless interference [48]. Another way is building 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 [18] only provides the bound of discovery latency, while the modeling of the distribution of all discovery latency values is still an open issue in this field. Controllable and reproducible, as shown in Figure 9, we build a lightweight neighbor discovery simulation tool, Blender [20], that simulates the behavior of the broadcaster and scanner according to the parameter configurations. Blender distinguishes itself from the legacy way of random sampling by applying equivalence-relation-based *Case Projection* according to the *Base-Case Simulation*, this not only avoids the costly brute-force traversal through all cases but also achieves more deterministic results than that of the random sampling.

Common Interest Extraction. For each scan mode, the broadcast intervals near the valley or semi-valley area are defined as the *interest* of this scan mode. As shown in Figure 8, the latency distributions vary with different types of scan modes. This module extracts the common interest by finding the common broadcast intervals (e.g., b_1, b_2, and b_3 in Figure 8) that achieve the locally minimized weight average discovery latency (i.e, Equation (2)) among all types of scan modes. This serves as a basis for global optimization of neighbor discovery performance in the case of scan mode diversity.

Interval Multiplexing. A naive way of ElastiCast is to apply a constant broadcast mode where a single broadcast interval is carefully chosen through the common interest extraction. However, as specified in the BLE standards [49], the interval between two consecutive broadcast events is not a constant but is mandatorily longer than the settled broadcast interval by a random period within 10 ms, called the random advertising delay (denoted by adv_delay, and $adv\_delay \in [0,\ 10]$ ms). As a result, a large adv_delay might induce a negative effect for the naive way of choosing a single broadcast interval. Specifically, a broadcast interval within the valley area might be shifted to a non-valley area due to the mandatory and random adv_delay (see §VI-B for more details). To tackle this issue, the interval multiplexing module steps further toward the global optima by dealing with adv_delay. The design rationale is that different broadcast intervals show different sensitivity to the adv_delay (see Figure 15), and multiplexing the complementary broadcast intervals might compensate for the negative effect induced by the adv_delay. Specifically, we apply the broadcast pattern selected from the *Single Broadcast Pattern* and *the Alternation Broadcast Pattern*. The decision-making of the broadcast pattern and the corresponding parameter settings are made according to the minimized weighted average discovery latency for a given power budget.

VI. Design Details

In this section, we proceed to the design details of the specific modules in ElastiCast. Notations are listed in Table II.

A. Estimate the Local Optima

The parameter selection scheme of ElastiCast relies on the awareness of P-percentile latency under the candidate broadcasting and scanning configurations, as discussed in §V-B. In this section, we first present a method along with its underlying principles illustrated, to produce the P-percentile latency under given broadcasting and scanning configuration. We then introduce the architecture of the Blender simulator built upon the aforementioned P-percentile latency generation scheme.

[2]By default, $P = 100$ refers to the worst-case discovery latency. However, in practical cases manufacturers may care more about a range of tail latencies (e.g., 95-percentile), thus we leave P as the customizable parameter.

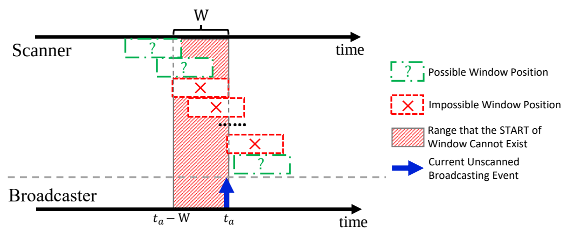

W
Scanner
time
Possible Window Position
Impossible Window Position
Range that the START of Window Cannot Exist
Current Unscanned Broadcasting Event
Broadcaster
$t_a - W$
t_a
time

Fig. 10: The exclusion of possible scan window positions with an uncaptured broadcast package.

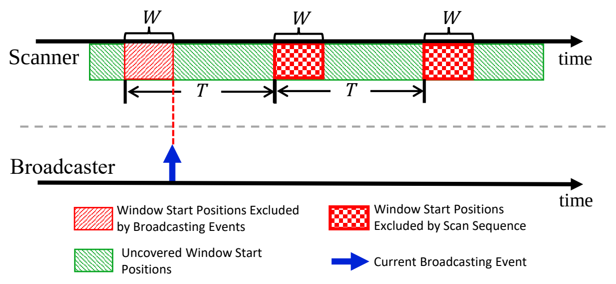

W
W
W
Scanner
time
T
T
Broadcaster
time
Window Start Positions Excluded by Broadcasting Events
Window Start Positions Excluded by Scan Sequence
Uncovered Window Start Positions
Current Broadcasting Event

Fig. 11: Coverage of possible scan window positions by the single broadcasting event.

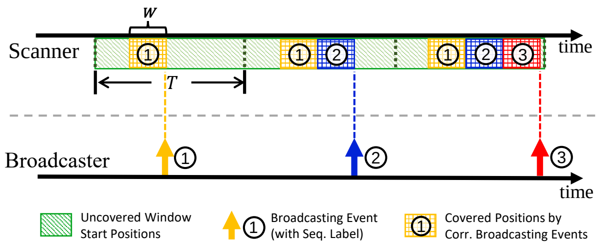

W
Scanner
time
T
Broadcaster
time
Uncovered Window Start Positions
Broadcasting Event (with Seq. Label)
Covered Positions by Corr. Broadcasting Events

Fig. 12: Coverage of possible scan window positions by multiple broadcasting events.

TABLE II: Notations

Name	Description
A	Broadcast interval
T	Scan interval
W	Duration of a scan window
δ_{bc}	Offset between broadcast and scan sequences
$\mathcal{A}_i$	The i-th broadcasting event in a given broadcasting sequence
$\mathcal{G}$	A time segment to be *covered*
ν	Overlap length between newly covered and previously covered periods in $\mathcal{G}$
Υ_i	Intermediate discovery latency produced by broadcasting event $\mathcal{A}_i$
Υ_f	Latency from the first broadcasting event to full coverage of segment $\mathcal{G}$
$P_l(\Upsilon_i)$	Collaborative probability of achieving intermediate latency Υ_i with $\delta_{bc} = 0$
$\mathcal{L}_\Upsilon$	List of latency-probability pairs $(\Upsilon_i, P_l(\Upsilon_i))$
ε	Deadline latency beyond which it is considered insignificant and all represented by ε
$\mathcal{Q}$	Root sequence of broadcasting intervals to be repeated with dynamic broadcast interval
$\mathcal{A}_j^{\mathcal{Q}}$	j-th broadcasting event in $\mathcal{Q}$ selected as the first broadcasting event after range entrance
τ_j	Interval between events $\mathcal{A}_j^{\mathcal{Q}}$ and $\mathcal{A}_{j-1}^{\mathcal{Q}}$ in $\mathcal{Q}$

1) Beyond Worst-Case Latency: The Latency Probability Distribution: To know the P-percentile latency for any P, it is necessary to retrieve the latency's probability distribution, over the stochastic phase difference between the sequences of broadcasting events/scanning windows. While neighbor discovery involves the interaction between two devices, an intuitive idea for producing the distribution is to reproduce the discovery process from a global perspective; inefficiently sampling from all possible phase difference values nevertheless can produce the latencies from start to timestamps that both devices are simultaneously awake. But furthermore with sampling, neither we achieved a deterministic distribution nor we understood the principles of how the broadcasting/scanning parameters are alternating the distribution.

***Exclusions* in the discovery process.** Rather than the random sampling method, we can bring benefit in determinism of the distribution and the estimation efficiency through shifting the perspective to the broadcaster . The gist of a broadcaster's point of view is to conjecture about the relative time position of the scan sequence to the broadcast sequence. In other words, we build a probabilistic distribution over the time offsets of all possible scan sequences to the given broadcasting sequence. Figure 10 illustrate the basic principle with assuming a broadcasting event occurring at time t_a. If it were captured by the scanner, a scan window must exist and envelop this broadcasting event. Conversely, it can be inferred from an uncaptured broadcasting event that there is a period of time where the scan window is certainly deactivated if no environmental interference occurs; if a broadcasting event occurs at t_a is not captured, the probability of any scan window to commence in the period $(t_a - W, t_a]$ is *excluded*. Noticeably, this exclusion of $(t_a - W, t_a]$ is not the end: the periodically repeating nature of the scan window also excludes $(t_a - W + i \cdot T, t_a + i \cdot T]$, $i \in \mathbb{Z}^+$, as shown in Figure 11. Therefore, as we traverse the broadcasting events, an increasing number of periods are excluded from being considered as starting timestamps for the scan windows, which we subsequently refer to as the *scan window's position.*

Coverage: Bridging Exclusion to Latency. How is the exclusion related to latency? We first consider the worst-case latency. To illustrate, we split the wall-clock time into identical segments $\mathcal{G}_i$, in other words, copies of a single segment $\mathcal{G}$, of length T. As shown in Figure 12, after range-entrance, a broadcasting event (yellow arrow with circled 1) occurs. As discussed in the last paragraph, if discovered is not accomplished with current broadcasting event, we can infer the exclusion of scan window's positions: a period of length W repeating in each segment. Notice that it is possible for this very first broadcasting event to achieve a discovery, it does not represent the worst case. As more broadcasting events occur, a new period will be excluded as long as discovery does not occur with the event; the segment will be gradually *covered* by the clusters of those excluded periods. Since the positions in $\mathcal{G}$ actually represent all different offsets between the scan windows sequence and the broadcasting events sequence, the longest time from the first broadcasting event to the discovery will be the time to exclude all possibilities for the offset, i.e., to cover $\mathcal{G}$. Conclusively, we formulate the Coverage Theory:

Theorem 6.1: Given the parameter set A, T, W, an infinite sequence of broadcasting events is used to generate clusters of impossible scan windows' positions. If the broadcasting events can fully cover the time-varying segment of length T within $\varrho = LCM(A,T)$ time (LCM stands for *Lowest Common Multiple*), there is a finite worst-case discovery latency, which is $A + \Upsilon_f$. A is the longest time from range-entrance to the first broadcasting event, and Υ_f is the time from the first broadcasting event to the broadcasting event that generates the last period to cover $\mathcal{G}$.

Expanded Coverage Theory for Full Latency Distribution. Theorem 6.1 was first presented in [18], unfortunately obscuring its underlying probabilistic nature in explanation: the gradual coverage of segment $\mathcal{G}$ is a process constructing the probability distribution of the scan sequence's offset to the broadcasting sequence. Therefore, Theorem 6.1 was not leveraged to generate the probability distribution of latency but only the worst-case counterpart. We propose the expansion of the Coverage Theory, illustrating how the latency distribution is produced respectively for the broadcaster with fixed and dynamic broadcast interval.

Latency Distribution for Fixed Broadcast Interval. With an overly focus on the worst-case latency, we considered the scenario that no discovery occurs during the coverage process of Segment $\mathcal{G}$. However, whenever a broadcasting event can contribute to a newly covered period in $\mathcal{G}$, it is possible to encounter an active scan window; the probability remains for some scan sequence offsets, in which a scan window starts within W before this broadcasting event. Consider a broadcasting event $\mathcal{A}_p$ in the left part of Figure 13, where it can be inferred that: the probability of a scan window to start in this period of length W covered by $\mathcal{A}_p$ is $\frac{W}{T}$. Therefore, an intermediate discovery latency, valued by Υ_m, is obtained as the time from the first broadcasting event $\mathcal{A}_0$ to $\mathcal{A}_m$, accompanied by a successful discovery with $\mathcal{A}_p$. Under the assumption $\delta_{bc} = 0$ (i.e., $P(\delta_{bc} = 0) = 1$), the probability for a scan window to overlap with $\mathcal{A}_p$ or the intermediate discovery latency to be Υ_p is:

$$P_l(\Upsilon_p) = P(\Upsilon_p \mid \delta_{bc} = 0) \cdot P(\delta_{bc} = 0) = \frac{W}{T} \tag{4}$$

Noticeably, any subsequent $\Upsilon_i (i \geq m)$ exists with the premise that no discovery occurs with $\mathcal{A}_j (j \leq m)$. Therefore, $\mathcal{A}_p$ still needs to produce a coverage to represent the case of an absent discovery. Each time a new period in $\mathcal{G}$ is to be covered, such as that produced by $\mathcal{A}_q$ in Figure 13, it may collide with a previously covered period. The probability for Υ_q then becomes $P_l(\Upsilon_q) = P(\text{Window Start} \in \mathcal{G}_{[n]} \setminus \mathcal{G}_{[\text{prev}]})$, with $\mathcal{G}_{[n]}$ as the new period to cover with $\mathcal{A}_q$ and $\mathcal{G}_{[prev]}$

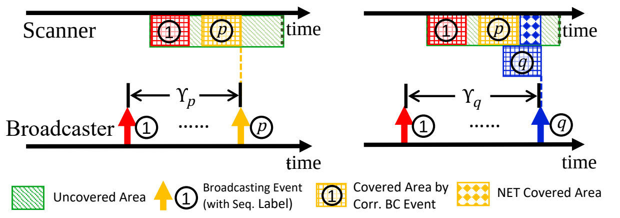


Fig. 13: Intermediate latency Υ_i w/ (left) and w/o (right) redundant coverage.

as the previously covered periods. Specifically, $P(\Upsilon_q)$ is calculated as $\frac{max(W-\nu,0)}{T}$ where ν is the length of $\mathcal{G}_q \cap \mathcal{G}_{\text{prev}}$ and $max(W - \nu, 0)$ is the length of *net* covered period $\mathcal{G}_q \setminus (\mathcal{G}_q \cap \mathcal{G}_{\text{prev}})$. The intersected period is not considered since if a scan window starts in that area, the discovery should haven been completed with a former broadcasting event.

With a list of broadcasting events $\{\mathcal{A}_i \mid i \in \mathbb{Z}\}$, the simulation will end up with a list $\mathcal{L}_\Upsilon$ of $(\Upsilon_i, P_l(\Upsilon_i))$ pairs. Besides, a deadline ε should be settled to represent all large and infinite latency, which are considered insignificant in the simulation result. Practically, an entry $(\varepsilon, 1 - \sum_i P_l(\Upsilon_i))$ is finally appended to $\mathcal{L}_\Upsilon$. This finite list will be expanded to a distribution over different range-entrance situations, i.e., $\delta_{bc} \in [0, A)$, while the generated list of pairs only describes the case $\delta_{bc} = 0$. Therefore, *case projection* becomes the process to involve all $\delta_{bc} \in [0, A)$ and produce the full latency distribution. Then the probability density function for a discovery latency L is formulated as:

$$P(L) = P(\delta_{bc} = \delta_{bc}^i) \cdot P_l(L - \delta_{bc}^i) = \frac{1}{A} \cdot P_l(L - \delta_{bc}^i), \tag{5}$$

where $\delta_{bc}^i = L - \Upsilon_i, \ \Upsilon_i \in \mathcal{L}_\Upsilon \wedge \Upsilon_i \leq L \wedge \Upsilon_i > L - A$

Latency Distribution for Dynamic Broadcast Interval. When fixed broadcast interval is applied, the broadcasting events are identical in terms of their relative positions within the sequence. Consequently, the coverage process of $\mathcal{G}$ is unaffected by which specific event in the sequence occurs first. However, with a flexibly duty-cycling broadcaster, this changes; as the broadcasting intervals vary, the interval between $\mathcal{A}_0$ and $\mathcal{A}_i$ also changes, leading to different values of Υ_i. A chance to recover this breakdown is that a determined broadcasting sequence in production scenarios should usually be recurrent, repeating a short sub-sequence stored in the limited ROM in Bluetooth modules. Therefore, a determined broadcasting sequence can be represented by a finite *root sequence* $\mathcal{Q}$, which records the intervals τ_j between a broadcasting event $\mathcal{A}_j^{\mathcal{Q}}$ and $\mathcal{A}_{j-1}^{\mathcal{Q}}$.

Stemming from the finite length of root sequence, the first broadcasting event after range-entrance can be enumerated from the root sequence, but with a probability $P(\mathcal{A}_j^{\mathcal{Q}}) = \frac{\tau_j}{\sum_k \tau_k}$ assigned. Similar to the situation of fixed broadcast interval, $\mathcal{L}_\Upsilon^j$ containing the $(\Upsilon_{j,i}, P(\Upsilon_{j,i} | \mathcal{A}_j^{\mathcal{Q}}))$ pairs is generated for each $\mathcal{A}_j^{\mathcal{Q}}$. Case projection is again conducted for each $\mathcal{L}_\Upsilon^j$ in the same way in Equation (5):

$$\begin{aligned} P(L|\mathcal{A}_j^{\mathcal{Q}}) &= \sum_i P(\delta_{bc,j} = \delta_{bc,j}^i) \cdot P_l(L - \delta_{bc,j}^i | \mathcal{A}_j^{\mathcal{Q}}) \\ &= \sum_i \frac{P(L - \delta_{bc,j}^i | \mathcal{A}_j^{\mathcal{Q}})}{\tau_j}, \end{aligned} \tag{6}$$

where we have:

$$\delta_{bc,j} \in [0, \tau_j),$$
$$\delta^i_{bc,j} = L - \Upsilon_{j,i},$$
$$\Upsilon_{j,i} \in \{\Upsilon_{j,u} \mid \Upsilon_{j,u} \in \mathcal{L}^j_{\Upsilon} \wedge \Upsilon_{j,u} \leq L \wedge \Upsilon_{j,u} > L - \tau_j\}.$$

Note that different from Equation (5), there can be multiple $\Upsilon_{j,i}$ possible to derive L after added by δ_{bc}. After marginalization, we derive the probability density function of a Latency L as:

$$P(L) = \sum_j P(L|\mathcal{A}^{\mathcal{Q}}_j) \cdot P(\mathcal{A}^{\mathcal{Q}}_j) = \sum_j \frac{\sum_i P_l(L - \delta^i_{bc,j}|\mathcal{A}^{\mathcal{Q}}_j)}{\sum_k \tau_k}, \tag{7}$$

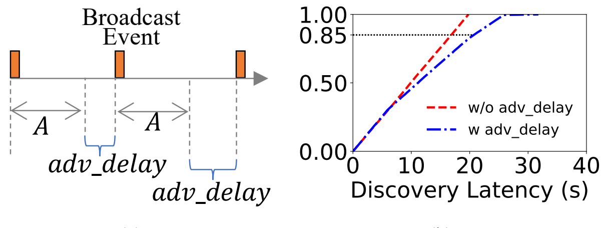
2) The Architecture of Blender: The reframed and expanded coverage theory renders feasible producing the deterministic latency distribution in a sampling-free manner, culminating in our practical discovery latency estimator Blender. This section briefly introduces the utility of Blender.

Input. The input of Blender mainly consists of configurations for broadcaster and scanner. For instance, the scanner configuration includes the scan interval T and scan window duration W; The broadcaster configuration includes two branches: the determined parameterization with a finite broadcasting sequence $\mathcal{Q}$ or broadcast interval A. In addition, several stochastic factors can be involved, including the interference configuration represented by an average signal interference rate and the maximum random broadcasting delay (i.e., adv_delay), which is further discussed in Appendices B and C.

Modules in Blender. Blender follows a two-step procedure in producing the latency distribution:

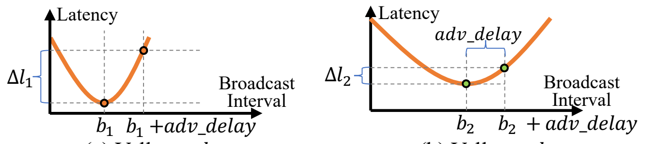
- **Coverage Simulation.** From the perspective of the broadcaster, a sequence of broadcasting events is simulated to estimate the wall-clock time offset between the scanning sequence and the start of this broadcasting sequence. With δ_{bc} fixed as 0, the simulation results in a list of several intermediate discovery latency and their conditional probability.
- **Case Projection.** The latency from coverage simulation accounts for a fraction of all possible latency, which is the reason to annotate their result as *intermediate*. The range-entrance case projection (referred to as case projection), for example with fixed broadcast interval, projects case $\delta_{bc} = 0$ to all $\delta_{bc} \in [0, A)$, meanwhile producing the overall latency as the summation of a range-entrance latency (i.e., time from range-entrance to the first broadcasting event) and each intermediate latency. When considering stochastic factors, the case projection requires adaptations, of which the explanation is complemented by the discussion of pragmatic adapters in the Appendix A.

The output is represented as a cumulative probability distribution of discovery latency in the complete domain, from which the P-percentile latency can be easily queried.



(a) (b)

Fig. 14: (a) The adv_delay added to broadcast interval. (b) The adv_delay induces negative effects. W = 60 ms, T = 600 ms, and A = 1980 ms is within the valley area.



(a) Valley at b_1 (b) Valley at b_2

Fig. 15: Analysis on sensitivity to adv_delay.

B. Common Interest Extraction

For each scan mode, the broadcast intervals near the valley or semi-valley area are defined as the *interest* of this scan mode. To search the global optimal broadcast intervals, we extract the common interest by finding the common broadcast intervals that achieve the locally minimized weight average discovery latency among n types of scan modes. First of all, we quantificationally give the definitions of the valley and semi-valley areas as below.

Definition 1: Valley Area. According to [18], the local minimum P-percentile latency is computed as

$$l^P_{min} = P \cdot \lceil \frac{T}{W} \rceil \cdot A \tag{8}$$

We then regard an A is within the valley area if its P-percentile discovery latency l^P_A equals to l^P_{min}.

Definition 2: Semi-valley Area. We regard an A is within the semi-valley area if its P-percentile latency meets $l^P_A \leq \alpha \cdot l^P_{min}$, where α is a relaxation coefficient ($\alpha > 1$).

For each scan mode $s_i \in \mathbf{S}, i = 1, 2, ..., n$, from the input of Common Interest Extraction, we can get the l^P_A, the P-percentile latency of any given A. By comparing l^P_A and l^P_{min} according to the definitions of valley area and semi-valley area, we can get the set of feasible broadcast intervals $\mathbf{B}_i$ that is within the valley or semi-valley areas. Then we compute the intersection $\mathbf{B}^*$ among n scan modes by $\mathbf{B}^* = \mathbf{B}_1 \cap \mathbf{B}_2 \cap ... \cap \mathbf{B}_i \cap ... \cap \mathbf{B}_n$.

Optimality Analysis. Through Common Interest Extraction, we obtain $\mathbf{B}^*$, a set of feasible broadcast intervals that are within the valley or semi-valley areas of all the n scan modes. According to Equations (2) and (3), from $\mathbf{B}^*$ we can always find an interval that achieves the minimal weighted average discovery latency across all scan modes, called the *Single Broadcast Pattern*. We then formulate this problem as follows:

$$\min \hat{l} = \sum_{i=1}^{n} \omega_i \cdot l_{i,j}, \quad s.t. \;\; b_j \geq A_{min}, \; \forall b_j \in \mathbf{B}^* \tag{9}$$

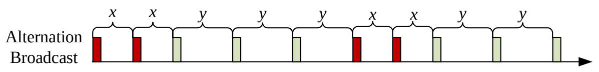


Fig. 16: An example of the Alternation Broadcast Pattern.

where $l_{i,j}$ is the worst-case discovery latency of scan mode s_i with the j^{th} feasible broadcast interval b_j ($b_j \in \mathbf{B}^*$). We further give an example of two types of scan modes s_1 and s_2 whose market shares are $\omega_1 = 30\%$ and $\omega_2 = 70\%$. Assume from Common Interest Extraction we obtain the intersection including 3 feasible broadcast intervals, i.e., $\mathbf{B}^* = \{b_1, b_2, b_3\} = \{1980\ ms, 2560\ ms, 4460\ ms\}$. Assume the power budget is $A_{min} = 2000$ ms, then b_1 is discarded due to power constraint. Assume from Local Optima Estimation, we have $l_{1,2} = 5$ s, $l_{2,2} = 20$ s, $l_{1,3} = 8$ s and $l_{2,3} = 10$ s, then we compute the weighted average discovery latency of b_2 and b_3 as $\hat{l}_2 = 15.5$ s and $\hat{l}_3 = 9.4$ s, respectively. As a result, $b_3 = 4460$ ms becomes the optimal broadcast interval that is recommended by the ElastiCast algorithm.

However, simply setting a single optimal broadcast interval using the way depicted above might result in bias. This is because the implementations of the most modern BLE chips set a mandatory random advertising delay, i.e., adv_delay as illustrated in Figure 14(a), before each broadcasting event [49]. adv_delay is not a constant and varies up to 10 ms. Thus, always adding a random time to the settled broadcast interval might result in a larger discovery latency than expected. Specifically, a broadcast interval within the valley area might be shifted to a non-valley area due to the mandatory and random adv_delay.

To explain this more clearly, we conduct experiments on how ElastiCast performs with and without adv_delay. Figure 14(b) shows the results. Through simulation, we have known A = 1980 ms is within the valley area for the scan mode with W = 60 ms and T = 600 ms, i.e., it obtains the local minimum worst-case discovery latency ideally. However, in practical cases with adv_delay the tail latency (e.g., 85-percentile latency = 21 s) is larger than that in the ideal cases without adv_delay (e.g., 85-percentile latency = 16 s). Hence, adv_delay induces a negative effect for ElastiCast.

For a more practical ElastiCast, our next goal is to seek a way how to avoid the negative effect of adv_delay. Based on our long-term and comprehensive observations, we find that different broadcast intervals show different sensitivity to adv_delay. As illustrated in Figure 15, assume the broadcast interval b_1 achieves the global minimal weighted average discovery latency among all scan modes. While compared with b_1, the broadcast interval b_2 is less sensitive to the adv_delay, which obtains a lower latency bias (i.e., $\Delta l_2 < \Delta l_1$). We thus infer that the combination between b_2 and b_1 improves the adaptability in the context of the random adv_delay. Specifically, when adv_delay is large, the broadcast interval b_2 might achieve lower discovery latency than b_1. This greatly motivates the Interval Multiplexing as we will elaborate next.

C. Interval Multiplexing

As discussed above, the Single Broadcast Pattern might fall short due to the existence of adv_delay. In this section, we explain how to step further toward the global optima by adopting the intermixed use of multiple broadcast intervals instead of the single one, thus we present the Alternation Broadcast Pattern as blow.

Figure 16 illustrates an example of the broadcasting event sequence when applying the Alternation Broadcast Pattern. In this pattern, two[3] phases with intervals x and y ($x, y \in \mathbf{B}^*$) appear interchangeably. We define ϕ_x and ϕ_y as the *repeated times* of intervals x and y, respectively. For example in Figure 16, we have $\phi_x = 2$ and $\phi_y = 3$. We define the *equivalent broadcast interval* (denoted by $\hat{A}$) as the average time between two consecutive broadcasting events. Then for the Alternation Broadcast Pattern, we have $\hat{A} = \frac{x\phi_x + y\phi_y}{\phi_x + \phi_y}$. When $\phi_x = 0$ or $\phi_y = 0$, the Alternation Broadcast Pattern falls back to the Single Broadcast Pattern.

Decision-Making. The decision-making is to select the best pattern and the corresponding parameter configuration from the Single Broadcast Pattern and the Alternation Broadcast Pattern. ElastiCast aims to achieve the minimized weighted average discovery latency within the power budget, therefore the constraint in Equation (3) can be updated as $\hat{A} \geq \hat{A}_{min}$, where $\hat{A}$ is the equivalent broadcast interval of broadcast pattern and $\hat{A}_{min}$ is the minimum equivalent broadcast interval.

D. Discussion

This section discusses the provisioning principles for such parameters as the latency percentile (P), the relaxation coefficient (α), and the repeated times (ϕ_x and ϕ_y).

Latency Percentile. In OFN applications, the user experience is closely related to the success ratio (possibility) of finding the lost device, where the P-percentile discovery latency matters. Although P are customizable, to better fit the problem of achieving the highest success ratio within the valley or semi-valley area, it is recommended to compute P by $P = \min\{\frac{latency\_tolerance}{l_{min}^{max}} \cdot 100, 100\}$, where $latency\_tolerance$ is the time a finder device spends within the BLE signal range in the walking scenario (see Figure 5), and $l_{min}^{max} = \lceil \frac{T}{W} \rceil \cdot A$ is the local minimum worst-case latency [18].

Relaxation Coefficient. In general, setting a large relaxation coefficient α (e.g., $\alpha = 5$) expands the solution space of feasible broadcast intervals but also induces significant search overhead. On the other hand, setting a small α (e.g., $\alpha = 1.001$) shortens search time but may fail to extract the common interest. Hence, $\mathbf{B}^* = \varnothing$ means the relaxation factor α is set too small. In this paper, we increase α by 10% until it meets $\mathbf{B}^* \neq \varnothing$. This assures ElastiCast can always obtain a globally feasible solution with a bounded overhead.

[3]Generally, ElastiCast can multiplex as many feasible broadcast intervals as possible (i.e., ≥ 2) to search for the best pattern and parameter configuration. However, considering the exploding solution space and deployment hurdles, this paper only focuses on patterns consisting of two intervals and leaves the patterns consisting of more than two intervals for future work.

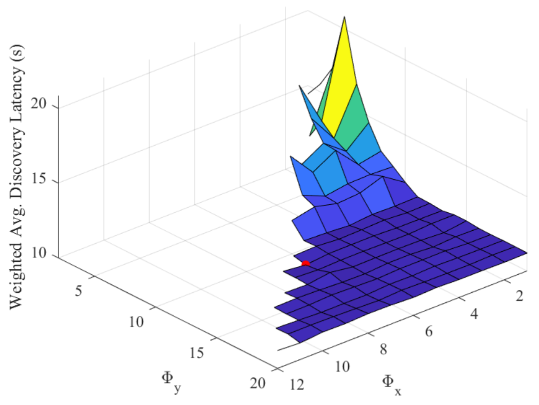


Fig. 17: An example of how to select the ϕ_x and ϕ_y combination. The settings of this experiment are the same as the one in Figure 20. The red dot achieves the lowest weighted average discovery latency with the ϕ_x and ϕ_y combination (6, 12).

Repeated Times. For the Alternation Broadcast Pattern, the repeated times ϕ_x and ϕ_y decide the phase duration proportion of the two broadcast intervals x and y. Note that ϕ_x and ϕ_y are not the input of ElastiCast but the output. The optimal ϕ_x and ϕ_y might vary with the inputs (e.g., scan modes) when the Alternation Broadcast Pattern is the preferred option. To select ϕ_x and ϕ_y, we straightforwardly search through combinations that result in a preferable percentile latency. As shown in Figure 17, we search within the ranges $\phi_x \in [1, \phi_x^{\max}]$ and $\phi_y \in [1, \phi_y^{\max}]$, where $\phi_x^{\max}$ and $\phi_y^{\max}$ are calculated based on the scanning modes. For example, assuming there are n scan modes, $\phi_x^{\max}$ is computed as $\phi_x^{\max} = [\max\{\text{LCM}(x, T_i) \mid i = 1, 2, \ldots, n\}]/x$. We then exhaustively search these combinations to select the optimal ϕ_x and ϕ_y that result in a preferable percentile latency.

Relationship among Alternation Broadcast Pattern, Coverage, and Latency Percentile. Interval multiplexing makes the CRT-based analysis unsuitable without a fixed broadcasting interval. Consequently, the latency estimation pivots to being governed by the extended Coverage Model. Importantly, the extended Coverage Model reveals the mechanism behind how dynamic broadcast interval overcomes the impact on P-percentile latency under adv_delay. While the model with dynamic broadcast interval shares the fundamental idea of *coverage* with fixed broadcast interval, we use the latter to simplify the following explanation. As implied in §VI-A1, the latency distribution directly correlates with the scan window position distribution. In other words, the P-percentile latency will be $\Upsilon_i + b$, where after broadcasting event $\mathcal{A}_i$, the coverage status of segment $\mathcal{G}$ should be $\frac{|\mathcal{G}_{\text{covered}}|}{|\mathcal{G}|} \geq P\%$. Therefore, lowering P-percentile latency is equivalent to faster achieving a coverage of $P\%$ for $\mathcal{G}$. The fastest coverages for any P are achieved by *valley* broadcast intervals, stemming from the observation that there are no redundantly covered periods in $\mathcal{G}$. However, adv_delay can deteriorate this optimality. While the position of a broadcasting event becomes probabilistic, the originally deterministic period to be covered in $\mathcal{G}$ becomes a distribution. Therefore, even if the broadcast interval belongs to *valley*, some of the samples from this distribution of periods could overlap with previously covered periods, slowing down the process towards $P\%$ coverage. When dynamic broadcast interval (e.g., interval multiplexing) is adopted, such a distribution of periods could be shifted along with a shifted broadcasting event, potentially reducing or even avoiding any redundant coverage caused by the periods in the distribution.

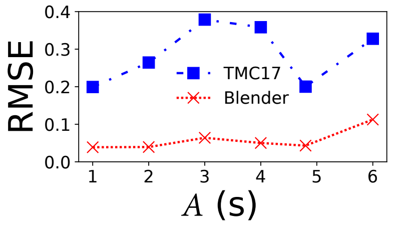


Fig. 18: RMSE of the distribution generated by Blender and ideal case simulator compared to the measurement traces.

VII. Evaluation of Blender

Comparison on Percentile Latency. To validate Blender's basic functionality, the simulation results are compared with the measurement traces. Since one of the major targets of Blender is to provide the percentile latency, the comparison focuses on 20, 50, and 80-percentile latency, which lay through the most part of the distribution and preliminarily reflects the ability of the model-based output to fit the traces. We select 6 configurations for broadcasters, and 2 for scanners, and measure in 2 physical locations with different average packet loss rates.

Measuring the Effectiveness of Stochastic Factor Adaptors. Blender has equipped several modules to adapt stochastic factors, such as the random advertising delay and environmental interference, which introduces extra analysis and computation burden on the framework. To prove the significance of these excessive efforts, we compare Blender with a simulator constructed upon the *effective-scan-window* [16] model for ideal cases, denoted as TMC17. The RMSE between the distributions generated to the measurement traces are compared to show the ability to reflect pragmatic scenarios.

Measurement Setup. The testbed for 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 [50] with an additional module to measure the average packet loss rate, which is a necessary parameter for the simulation. The loss measurement module keeps one phone advertising the BLE packets N times and another phone continuously scans 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.

TABLE III: Percentile latency between simulation (Sim) and measurement (Mea). (Parameter in milliseconds

Parameter Type	Broadcaster A	Scanner $\frac{W}{T}$	Avg. Loss Rate	P20 L(Sim)	P20 L (Mea)	P50 L(Sim)	P50 L(Mea)	P80 L(Sim)	P80 L(Mea)
Randomized Broadcasting	1140	$\frac{512}{5120}$	26%	3.10s	4.28s	7.72s	8.59s	16.4s	15.1s
	3250			8.83s	9.59s	22.6s	20.4s	43.1s	39.7s
	6530			17.7s	15.6s	44.3s	43.4s	>50s	>50s
	1140	$\frac{1024}{4096}$	43%	4.10s	4.38s	10.1s	11.9s	23.7s	27.5s
	3250			4.70s	2.78s	12.4s	13.7s	27.8s	31.1s
	6530			9.17s	10.1s	24.9s	21.3s	>50s	48.3s
Flexible Duty-Cycle[4]	1860×4 +2140×6	$\frac{512}{5120}$	26%	5.48s	6.5s	15.1s	14.1s	33.6s	32.0s
	1280×2 +3250×4 +4000×1		43%	11.0s	9.27s	28.8s	26.8s	>50s	>50s
	860×5 +6530×2	$\frac{1024}{4096}$		3.22s	3.96s	7.45s	9.38s	21.3s	19.1s

Evaluation results. Results are shown in Table III. For the parameters selected to produce the percentile latency, most of the results can act as a close estimation of the realistic situations. Although some of the 80-percentile latency shows a larger deviation, the trend of the real distribution is generally well-represented. To further represent the ability of the model to fit the realistic situation, the root-mean-square error (RMSE) is introduced as the metric to quantify the similarity between any two distributions. Figure 18 shows the results with growing advertise interval under the *LOW_POWER* scan mode defined in Android BLE module [51]. 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 on average 43% packet loss rate regardless of the selected advertising interval, which hence proves that the stochastic factors are worth handling. Conclusively, Blender is capable of being applied in real-world scenarios and provides satisfactory outputs.

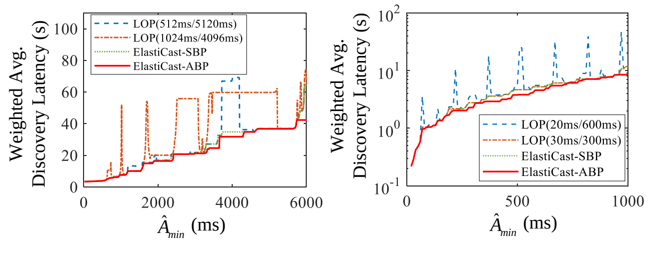
VIII. Evaluation of ElastiCast

A. Experiment Setup

Inputs. A type of scan mode is in the form of W/T. For example, "$1024ms/4096ms$" refers to the scan mode with a scan window of 1024 ms and a scan interval of 4096 ms. The evaluation inputs are $s_i \in \mathbf{S}$ $(i = 1, 2, ..., n)$, ω_i, and $\hat{A}_{min}$, where s_i is the i^{th} type of scan mode, ω_i is the market share of s_i, and $\hat{A}_{min}$ represents the broadcaster's power budget in the form of the minimum equivalent broadcast interval.

Parameters. We run tests for broadcast intervals that meet $A \in [A_{left}, A_{right}]$, where we recommend $A_{left} \geq 20$ ms and $A_{right} < \min\{10240ms,\ latency\_tolerance\}$ according to the whole range of broadcast intervals allowed in BLE. In this paper, we set $A_{left} = 20$ ms and $A_{right} = 10240$ ms. The initial relaxation coefficient is set $\alpha = 1.2$.

[4]Due to the limitation of Android BLE module, the flexible duty-cycle is currently measured from a random sampler imitating the detailed behavior of the BLE devices.



(a) (b)

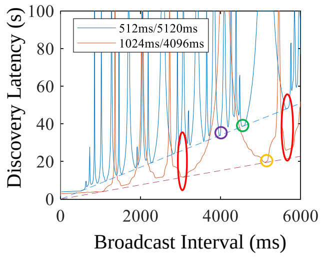
Fig. 19: Overall performance of ElastiCast. (a) Scan modes: $512ms/5120ms$ **and** $1024ms/4096ms$**. (b) Scan modes:** $20ms/600ms$ **and** $30ms/300ms$**.**

Schemes. Let **LOP** (Local OPtima) be the scheme that sets the broadcast interval that achieves the tight duty-cycle-dependent bounds on discovery latency (i.e., l_{min}^{P} in Equation (8)) for only one type of scan mode, i.e., LOP locally optimizes latency. Since no prior neighbor discovery parameter setting approaches can beat LOP in the case of homogeneous scan mode [18], we select LOP as the baseline for ElastiCast evaluation in the case of scan mode diversity. **ElastiCast-SBP** refers to the Single Broadcast Pattern and **ElastiCast-ABP** refers to the Alternation Broadcast Pattern.

B. Performance Improvement of ElastiCast

In this experiment, we investigate two representative scenarios, where $512ms/5120ms$ (i.e., LOW_POWER) and $1024ms/4096ms$ (i.e., BALANCED) are two types of default scan modes supported by most Android phones [42], and $20ms/600ms$ and $30ms/300ms$ are two types of scan modes adopted by the applications of HarmonyOS (e.g., HiLink [47]) and iOS (e.g., Apple's Find My [3]), respectively. The market shares of both scan modes are 50%. "LOP ($20ms/600ms$)" refers to the scheme that locally optimizes latency for the scan mode of $20ms/600ms$, and so forth.

Figure 19 shows the minimized weighted average discovery latency $\hat{l}$ (Y-axis) when applying each scheme within the power budget in the form of $\hat{A}_{min}$ (X-axis). It is demon-

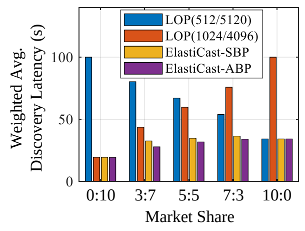


(a)

Flag	Scheme	Broadcast Interval(s)	
		x (ms)	*y* (ms)
🟣	LOP(512/5120)	4000	-
🟠	LOP(1024/4096)	5110	-
🟢	ElastiCast-SBP	4600	-
🔴	ElastiCast-ABP	2980	5620

(b)

Fig. 20: (a) A case study on latency distribution when $\hat{A}_{min} = 4000$ ms. (b) Examples of the selected broadcast interval(s) in different schemes.



(a)

	Success Ratio		
	Screen On	Screen Off	Overall
AirTag	68%	5.6%	71.48%
Elasti-Cast	72%	21%	82.52%

(b)

Fig. 21: (a) Performance under different market shares. (b) Performance comparison between AirTag and ElastiCast. (Screen On: $60ms/600ms$, Screen Off: $60ms/3000ms$)

strated that ElastiCast can always bound the neighbor discovery latency in the case of scan mode diversity. Specifically, Figure 19(a) shows the performance with two scan modes $512ms/5120ms$ and $1024ms/4096ms$. Both ElastiCast-SBP and ElastiCast-ABP outperform LOP in most cases. Particularly, compared to LOP($1024ms/4096ms$), both ElastiCast-SBP and ElastiCast-ABP reduce the discovery latency by up to 50% in some cases. Figure 19(b) shows the performance with two scan modes $20ms/600ms$ and $30ms/300ms$. We can see that, compared to LOP($20ms/600ms$), both ElastiCast-SBP and ElastiCast-ABP reduce the discovery latency by more than one order of magnitude (i.e., 90%) in some cases. Note that compared to ElastiCast-SBP, ElastiCast-ABP reduces the discovery latency by up to 40% in the scenario of $20ms/600ms$ and $30ms/300ms$, but only gains a margin benefit in the scenario of $512ms/5120ms$ and $1024ms/4096ms$. This can be attributed to the adv_delay that induces more negative effects in the scenario of $20ms/600ms$ and $30ms/300ms$.

To explain the benefit of ElastiCast more clearly, we give a case study on latency distribution when $\hat{A}_{min} = 4000$ ms. As shown in Figure 20(a), we mark the selected/optimal broadcast interval(s) for each scheme with colored cycles. Figure 20(b) further shows the exact values of broadcast intervals selected by corresponding schemes. For example, ElastiCast-SBP selects 4600 ms as the optimal broadcast interval, and ElastiCast-ABP selects both 2980 ms and 5620 ms ($\hat{A} = 4300$ ms) that appear interchangeably. Our decisions are then made by comparing the minimized discovery latency of all schemes.

C. Stability Analysis

How does the type of scan mode impact stability? First of all, we regard scan modes with different W or T as different types of scan modes even if they have the same duty cycle. This is because they have different valleys or semi-valley areas. For example, given $\hat{A}_{min} = 4000$ ms, although the duty cycles are both 10%, the optimal broadcast interval of LOP($30ms/300ms$) is 4030 ms, but the optimal broadcast interval of LOP($512ms/5120ms$) is 4000 ms. As shown in the experiment results in §VIII-B, ElastiCast shows stability in achieving global optima regardless of the types of scan modes.

How does the power budget impact stability? As shown in Figure 19, the power budget impacts the decision-making significantly, however, ElastiCast still approximates a bounded and globally minimized latency.

How does the market share impact stability? We further vary the market shares of scan modes. For example, “3:7” represents that the market share of $512ms/5120ms$ is 30%, and the market share of $1024ms/4096ms$ is 70%. Figure 21(a) shows that no matter how the market shares change, ElastiCast always approximates a bounded and globally minimized discovery latency.

D. Deployment Experience: HUAWEI Tag

ElastiCast has been deployed in Huawei's commercial-off-the-shelf (COTS) BLE devices, called HUAWEI Tag [40]. HUAWEI Tag acts as the role of Lost Device (see Figure 2) in the ecosystem of OFN. The Finder Devices are currently with two types of scan modes, i.e., $60ms/600ms$ and $60ms/3000ms$, which refer to the scan modes applied when the screens of the Finder Devices are on and off, respectively. Based on our long-term market statistics, the market shares of the scan modes approximate 33% and 67%, respectively.

We compare ElastiCast with Apple's AirTag which adopts a fixed broadcast interval of 2000 ms. We estimate the success ratio of finding the lost Tag with a latency tolerance of 14 seconds. Let N^+ be the number of tests that meet discovery latency of fewer than 14 seconds, and N^- otherwise. The success ratio is computed by $\frac{N^+}{N^+ + N^-}$. For a fair comparison, $\hat{A}_{min}$ varies in a small range of $[1950, 2050]$ ms. Figure 21(b) shows the results. It is demonstrated that ElastiCast outperforms AirTag no matter whether the screen is on or off. In a case with three Finder Devices nearby, ElastiCast obtains an improvement of over 11% on the overall success ratio.

The lesson we have learned during deployment is that the scanner rarely strictly follows the instructions of parameter settings. For example, when we set the scan interval as 600 ms, the actual scan interval might be 605 ms with a small random bias, which we believe is attributed to the hardware/-software task scheduling on the Finder Devices. Together with adv_delay, this may further impact the performance of the Single Broadcast Pattern. However, our evaluations demonstrate that the Alternation Broadcast Pattern can still compensate for the negative effect induced by the scanner side.

IX. Conclusion

This paper examined the framework design and performance optimization in OFN. Our study identifies the unique features as well as the fundamental design challenges in OFN neighbor discovery. Our proposed ElastiCast has proven to be effective in achieving stable and low-latency neighbor discovery within the power budget in the case of scan mode diversity. The authors have provided public access to the code of Blender at https://github.com/litonglab/blender-neighbor-discovery.

References

[1] "Apple's find my feature." [Online]. Available: https://www.apple.com/icloud/find-my/

[2] "A billion people now have iphones." [Online]. Available: https://www.aboveavalon.com/notes/2020/10/26/a-billion-iphone-users

[3] "Apple airtag." [Online]. Available: https://www.apple.com/sg/airtag/

[4] "Airtags are apple's next billion dollar business." [Online]. Available: https://www.forbes.com/sites/timbajarin/2021/04/20/airtags-are-apples-next-billion-dollar-business/?sh=75d236e65d18

[5] M. Weller, J. Classen, F. Ullrich, D. Waßmann, and E. Tews, "Lost and found: Stopping bluetooth finders from leaking private information," in *ACM WiSec*, 2020, p. 184–194.

[6] A. Heinrich, M. Stute, T. Kornhuber, and M. Hollick, "Who can find my devices? security and privacy of apple's crowd-sourced bluetooth location tracking system," *PoPETs*, vol. 2021, no. 3, pp. 227–245, 2021.

[7] T. Mayberry, E. Fenske, D. Brown, J. Martin, C. Fossaceca, E. C. Rye, S. Teplov, and L. Foppe, "Who tracks the trackers? circumventing apple's anti-tracking alerts in the find my network," 2021, p. 181–186.

[8] L. Tonetto, A. Carrara, A. Y. Ding, and J. Ott, "Where is my tag? unveiling alternative uses of the apple findmy service," in *IEEE WoWMoM*, 2022, pp. 1–10.

[9] Y. Zhao, K. Xu, H. Wang, B. Li, and R. Jia, "Stability-based analysis and defense against backdoor attacks on edge computing services," *IEEE Network*, vol. 35, no. 1, pp. 163–169, 2021.

[10] S. Kamath and J. Lindh, "Measuring bluetooth low energy power consumption," *Texas instruments application note AN092*, 2010.

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

[12] P. Kindt, D. Yunge, R. Diemer, and S. Chakraborty, "Precise energy modeling for the bluetooth low energy protocol," *arXiv preprint arXiv:1403.2919*, 2014.

[13] H. Lee, D. Ok, J. Han, I. Hwang, and K. Kim, "Performance anomaly of neighbor discovery in bluetooth low energy," in *IEEE ICCE*, 2016, pp. 341–342.

[14] K. Cho, G. Park, W. Cho, J. Seo, and K. Han, "Performance analysis of device discovery of bluetooth low energy (ble) networks," *Computer Communications*, vol. 81, pp. 72–85, 2016.

[15] W. S. Jeon, M. H. Dwijaksara, and D. G. Jeong, "Performance analysis of neighbor discovery process in bluetooth low-energy networks," *IEEE ToVT*, vol. 66, no. 2, pp. 1865–1871, 2016.

[16] P. H. Kindt, M. Saur, M. Balszun, and S. Chakraborty, "Neighbor Discovery Latency in BLE-Like Protocols," *IEEE TMC*, vol. 17, no. 3, pp. 617–631, 2018.

[17] A. Liendo, D. Morche, R. Guizzetti, and F. Rousseau, "Ble parameter optimization for iot applications," in *IEEE ICC*, 2018, pp. 1–7.

[18] P. H. Kindt and S. Chakraborty, "On optimal neighbor discovery," in *ACM SIGCOMM*, 2019, pp. 441–457.

[19] "Airtag analysis." [Online]. Available: https://adamcatley.com/AirTag.html

[20] Y. Ding, T. Li, J. Liang, and D. Wang, "Blender: Toward practical simulation framework for ble neighbor discovery," in *ACM MSWiM*, 2022, pp. 103–110.

[21] S. Vasudevan, J. Kurose, and D. Towsley, "On neighbor discovery in wireless networks with directional antennas," in *IEEE INFOCOM*, 2005, pp. 2502–2512.

[22] R. Zheng, J. C. Hou, and L. Sha, "Optimal block design for asynchronous wake-up schedules and its applications in multihop wireless networks," *IEEE TMC*, vol. 5, no. 9, pp. 1228–1241.

[23] Z. Zhang and B. Li, "Neighbor discovery in mobile ad hoc self-configuring networks with directional antennas: algorithms and comparisons," *IEEE TMC*, vol. 7, no. 5, pp. 1540–1549, 2008.

[24] S. Vasudevan, D. Towsley, D. Goeckel, and R. Khalili, "Neighbor discovery in wireless networks and the coupon collector's problem," in *ACM MobiCom*, 2009, pp. 181–192.

[25] N. Karowski, A. C. Viana, and A. Wolisz, "Optimized asynchronous multi-channel neighbor discovery," in *IEEE INFOCOM*, 2011, pp. 536–540.

[26] S. Vasudevan, M. Adler, D. Goeckel, and D. Towsley, "Efficient algorithms for neighbor discovery in wireless networks," *IEEE/ACM TON*, vol. 21, no. 1, pp. 69–83, 2012.

[27] W. Sun, Z. Yang, K. Wang, and Y. Liu, "Hello: A generic flexible protocol for neighbor discovery," in *IEEE INFOCOM*, 2014, pp. 540–548.

[28] T. Meng, F. Wu, and G. Chen, "On designing neighbor discovery protocols: A code-based approach," in *IEEE INFOCOM*, 2014, pp. 1689–1697.

[29] W. Sun, Z. Yang, X. Zhang, and Y. Liu, "Energy-efficient neighbor discovery in mobile ad hoc and wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1448–1459, 2014.

[30] L. Chen, R. Fan, K. Bian, M. Gerla, T. Wang, and X. Li, "On heterogeneous neighbor discovery in wireless sensor networks," in *IEEE INFOCOM*, 2015, pp. 693–701.

[31] Y. Qiu, S. Li, X. Xu, and Z. Li, "Talk more listen less: Energy-efficient neighbor discovery in wireless sensor networks," in *IEEE INFOCOM*, 2016, pp. 1–9.

[32] T. Li, B. Hu, G. Tu, J. Shuai, J. Liang, Y. Ding, Z. Li, and K. Xu, "Accelerating ble neighbor discovery via wi-fi fingerprints," in *IEEE INFOCOM WKSHPS*, 2023, pp. 1–2.

[33] Z. Li, Z. Yang, B. Hu, T. Li, B. Wu, Y. Ding, D. Xu, and K. Xu, "Rend: Toward reasoning-based ble neighbor discovery by integrating with wi-fi fingerprints," in *IEEE/ACM IWQoS*, 2024, pp. 1–6.

[34] B. Luo, J. Xu, and Z. Sun, "Neighbor discovery latency in bluetooth low energy networks," *Wireless Networks*, vol. 26, pp. 1773–1780, 2020.

[35] W. S. Jeon, M. H. Dwijaksara, and D. G. Jeong, "Performance analysis of neighbor discovery process in bluetooth low-energy networks," *IEEE ToVT*, vol. 66, no. 2, pp. 1865–1871, 2016.

[36] G. Shan and B.-h. Roh, "Performance model for advanced neighbor discovery process in bluetooth low energy 5.0-enabled internet of things networks," *IEEE ToIE*, vol. 67, no. 12, pp. 10 965–10 974, 2020.

[37] W. Bai, J. Chen, Y. Xu, F. Song, H. Wang, G. Li, and Y. Jiao, "Euclidean-division-based low-complexity precise analytical approach of ble-like neighbor discovery latency," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4184–4201, 2022.

[38] A. Antipa, D. Brown, A. Menezes, R. Struik, and S. Vanstone, "Validation of elliptic curve public keys," in *Public Key Cryptography*. Springer Berlin Heidelberg, 2002, pp. 211–223.

[39] K. Geissdoerfer and M. Zimmerling, "Bootstrapping battery-free wireless networks: Efficient neighbor discovery and synchronization in the face of intermittency," in *USENIX NSDI*, 2021, pp. 439–455.

[40] "Huawei tag." [Online]. Available: https://consumer.huawei.com/cn/accessories/tag/

[41] "What is the range of bluetooth and how can it be extended." [Online]. Available: https://www.scienceabc.com/innovation/what-is-the-range-of-bluetooth-and-how-can-it-be-extended.html

[42] "Android ble scan settings apis." [Online]. Available: https://developer.android.com/reference/android/bluetooth/le/ScanSettings

[43] "Exposure notification," 2020. [Online]. Available: https://www.google.com/covid19/exposurenotifications

[44] "Huawei contact shield," 2020. [Online]. Available: https://developer.huawei.com/consumer/en/doc/development/system-Guides/contactshield-introduction-0000001057494465

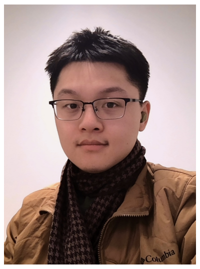
[45] "Apple airtag," 2021. [Online]. Available: https://www.apple.com/sg/airtag/

[46] J. Liu, C. Chen, and Y. Ma, "Modeling and performance analysis of device discovery in bluetooth low energy networks," in *IEEE GLOBECOM*, 2012, pp. 1538–1543.

[47] "Huawei hilink." [Online]. Available: https://iot.hilink.huawei.com/

[48] T. Li, K. Zheng, K. Xu, R. A. Jadhav, T. Xiong, K. Winstein, and K. Tan, “Tack: Improving wireless transport performance by taming acknowledgments,” in *ACM SIGCOMM*, 2020, pp. 15 – 30.
[49] “Bluetooth specifications.” [Online]. Available: https://www.bluetooth.com/
[50] “Android samples,” 2021. [Online]. Available: https://github.com/android/connectivity-samples/tree/main/BluetoothAdvertisements
[51] “Android ble documentation,” 2022. [Online]. Available: https://developer.android.com/reference/android/bluetooth/le/ScanSettings

Tong Li received his Ph.D. degree from Tsinghua University, Beijing, China, in 2017. He held a visiting scholar with the School of Computer Science and Electronic Engineering of University of Essex, UK, in 2014 and 2016. He is currently an associate professor at Renmin University of China. His research interests include network protocols, the Internet of Things, and distributed systems. He was a former chief engineer at Huawei, and was the 2021 Huawei’s Top Ten Inventions Award and the 2024 ACM SIGCOMM China Rising Star Award.



Yukuan Ding received his M.Sc degree in Information Technology from the Hong Kong University of Science and Technology, Hong Kong SAR, in 2023. He is currently pursuing his Ph.D. degree at Delft University of Technology. His research interests include long-range wireless networks, Artificial Intelligence of Things, and distributed systems.

Jiaxin Liang received his Ph.D. degree in Information Engineering from The Chinese University of Hong Kong, in 2021. He is currently a Principle R&D Engineer at Huawei in Computer Network and Protocol Research Laboratory. His research interests include network protocols, low-energy transmission, the Internet of Things (IoT), multiple access networks, and software-defined radios.

Kai Zheng received his Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2006. He is the Director of the Computer Network and Protocol Research Lab at Huawei Technologies. His research interests covered architectures and protocols for the next generation networks (e.g. 5G/IoT Networks, cloud-oriented data center networks, RDMA networks, and real-time multimedia networks, etc). He is a senior member of the IEEE.

Xu Zhang received the Ph.D. degree in computer science from the Department of Computer Science and Technology, Tsinghua University, China, in 2017. He is with the School of Electronic Science and Engineering, Nanjing University, China. His research interests include AI for computing and networking, multimedia networks, and Internet of Things. He was a recipient of the EU Marie Skłodowska-Curie Individual Fellowship and a co-recipient of the 2019 IEEE Broadcast Technology Society Best Paper Award.

Tian Pan received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, in 2015. He is currently an associate professor at the Beijing University of Posts and Telecommunications. His primary research interests include cloud data center networks, programmable data planes, and satellite networks.

Dan Wang received his B.Sc from Peking University, Beijing, M.Sc from Case Western Reserve University, Cleveland, OH, and Ph.D. from Simon Fraser University, Vancouver, Canada, all in Computer Science. He is a Professor of the Department of Computing, at The Hong Kong Polytechnic University. His research interests include Sensor Networks, Internet Architecture and Protocols. Recently, he has been working on Cloud Computing and Big data, to support interdisciplinary domains in Smart Cities. He is a senior member of the IEEE.

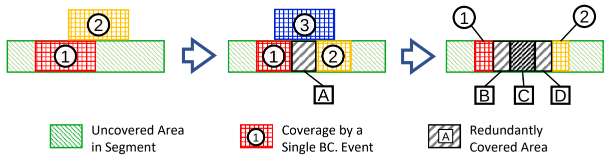
Ke Xu (Fellow, IEEE) received the Ph.D. degree from Tsinghua University, Beijing, China. He is currently a Full Professor at the Department of Computer Science and Technology, at Tsinghua University. He has published more than 200 technical articles in the research areas of next-generation internet, blockchain systems, and network security. He has won the IWQoS 2024 Best Paper Award and the Distinguished Paper Award at USENIX Security 2023 and 2024.

Algorithm 1: Probabilistic Coverage for Possible Scan Window Starting Positions

```
   Require: Index of $\alpha_j$-$j$, Range of $j$-$n$, $\tau_j$, Loss
            Rate-$F_p$, End Time-$\varepsilon$
   Result: List of $(\Upsilon_i, P(\Upsilon_i|\alpha_j))$ pairs
1  next_idx ← $j$ mod $n$;
2  timestamp ← 0;
3  itv_entries ← $EmptyList()$ of (interval, $\pi$);
4  while timestamp does not exceed $\varepsilon$ do
5      Conduct coverage on an interval
        $\sigma = (\texttt{timestamp} - W, \texttt{timestamp}]$;
6      if $\sigma$ does not collide with other intervals then
7          Append (timestamp, $\frac{W\cdot(1-F_p)}{T}$);
8          Record the interval $\sigma$ as an entry $(\sigma, F_p)$ in
            itv_entries;
9      else
10         for Every uncollided partition part_uncol
            of $\sigma$ do
11             $\bar{\nu_k}$← $len$(part_uncol)
12             Append (timestamp, $\frac{(\bar{\nu_k})\cdot(1-F_p)}{T}$);
13             Record the interval $\sigma$ as an entry
                (part_uncol, $F_p$) in itv_entries;
14         end
15         for Every collided partition part_col of $\sigma$
            do
16             prev_itv, $\pi_k$
                ← the entry of a former interval that
17             contains part_col;
18             Clip part_col from prev_itv inside
                itv_entries;
19             $\nu_k$← $len$(part_col)
20             Append (timestamp, $\frac{\nu_k\cdot\pi_k\cdot(1-F_p)}{T}$);
21             Record (part_col, $\pi_k \cdot F_p$) in
                itv_entries;
22         end
23     end
24 end
```

Appendix A
Producing Latency while Adapting Environmental Interference

The foundation of the Coverage Theorem and our elaboration is that an observer in the broadcaster's perspective has absolute confidence in its inference result of the scan window's positions. However, environmental interference introduces uncertainty, which indicates that an uncaptured broadcasting event may not exclude the existence of a scan window. While a systematic consideration of the interference factors can cast a high bias on specific environments and may overemphasize the minutiae of signaling stages, an average packet loss rate F_p can be a generalization about the impact of a complex of factors.



2
1
3
1
2
A
1
2
B
C
D
🟩 Uncovered Area in Segment
🟥 Coverage by a Single BC. Event
▨ Redundantly Covered Area

Fig. 1: The Probability Coverage Model Based on Clipping Redundantly Covered Areas

The key to integrating F_p into the existing coverage procedure is to maintain a probability model for the segment of possible scan window starting positions. Figure 1 illustrates the model's operation to process the covered area by a new broadcasting event. Originally, assume there was an area ① covered by an uncaptured broadcasting event A_1. With packet loss rate F_p given, A_1 has $100 \cdot (1 - F_p)\%$ confidence to exclude the scan window starting positions in area ①. In other words, the probability for A_1 to leave area ① uncovered is F_p. When A_2 induces a new area ②, a partition of area ② collides with area ①, annotated as square-A. While square-A is ignored and only the rest of area ② is considered in interference-free simulation, here it requires separate treatment since it has F_p chance to not be a redundant coverage. Therefore, the probability assigned to the intermediate latency Υ_2 with α_j as the first broadcasting event is the summation of the possibility of covering area ② (remaining) and square-A. From this instance, it can be deduced that if every covered area has a corresponding probability π, the intermediate latency has a probability:

$$P(\Upsilon_i|\alpha_j) = \sum_{k,i} \frac{\nu_k \pi_k (1-F_p)}{T} + \frac{(1-F_p)(W - \sum_{k,i} \nu_k)}{T} \quad (1)$$

where π_k is the π of the interval that ν_k lies in.

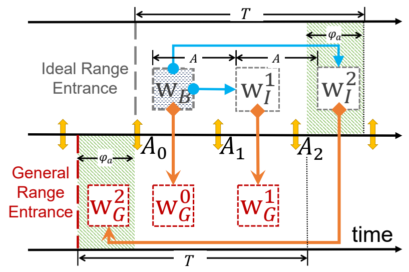
The subsequent case projection operation is exactly the same as the interference-free simulation since case projection is based on the range-entrance situation where packet loss is not involved. However, a left-over issue is the update strategy of π_k. Therefore, Algorithm 1 is presented to explain the whole coverage procedure.

Appendix B
2-Step Case Projection of Effective-Scan-Window-Based Models

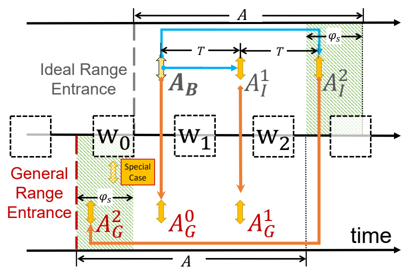
This section presents the design details of specific Case Projection steps of the Effective-Scan-Window-Based model in Blender. We first introduce the base case simulation and the single discovery process in §B-A. Then in §B-B and §B-C, the case projection module that can produce the complete CDF is illustrated, which is the key to improving the simulation efficiency.

A. Base Case Simulation

In the base case simulation, attributes δ_{bc} and δ_s, describing a range-entrance case, respectively annotate the time from the range-entrance event to the first following broadcasting event and the end of the first following scan window. With A, T, δ_{bc}, and δ_s, a time sweeper similar to [1] will then traverse through the timeline from range-entrance, and the time from



(a) $A < T$



(b) $A > T$

Fig. 2: Examples of the equivalence-relation-based Case Projection.

range-entrance to the first discovery (when no packet loss considered) becomes the discovery latency $L(\delta_{bc}, \delta_s)$.

B. Step Details in Case Projection

The purpose of **case projection** is to avoid redundant simulation of the discovery process. Instead, some of the range-entrance events share a similar discovery process (i.e., the relative position of the broadcasting events and scan windows on the timeline are the same), where the only difference is the offsets of the first broadcasting event and the first scan window from the range-entrance event. Intuitively, the latency in the non-base cases can be directly derived from the latency in the correlated base cases, which is called case projection. Case projection comprises two steps.

The Phase-Difference Projection. The cases when $A < T$ are used as example. As an initial step, δ_{bc} is fixed to 0, which is annotated as Ideal Range-Entrance and infers that the first broadcasting event A_0 occurs immediately at the range-entrance event. The first scan window w_0 has T possible positions depending on the value of δ_s. Each of these positions is alternatively described by $t_{w_0} \in (0, T]$, 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 the 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 $A < T$, there must be a pair of $t_{w_0^i}$ and $t_{w_0^{i+1}}$ that are 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 the scanner and broadcaster 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 B.1: When A_0 is fixed at the range-entrance event, for every $t_{w_0^i} \in (A, T]$, the discovery latency can be represented as

$$\begin{aligned} L(0, t_{w_0^i}) &= D_{w_0^i} = D_{w_0^{i-1}} - t_{w_0^{i-1}} + t_{w_0^i} \\ &= D_{w_0^{i-1}} + A = L(0, t_{w_0^{i-1}}) + A, \end{aligned} \quad (2)$$

where $L(x, y)$ is the discovery latency with $\delta_{bc} = x$ and $\delta_s = y$.

Based on Equation (2), the latency of all Ideal Range-Entrance cases $L(0, t_{w_0^i})(t_{w_0^i} \in (A, T])$ can be derived from those of the base cases $L(0, t_{w_0^0})(t_{w_0^0} \in (0, A])$ as shown in Figure 2(a) (the Phase-Difference Projection):

$$L(0, t_{w_0^i}) = L(0, t_{w_0^0}) + i \cdot A (i = \lfloor \frac{t_{w_0^i}}{A} \rfloor) \quad (3)$$

The Range-Entrance Projection. After considering all phase differences between Q_A and Q_S with fixing $\delta_{bc} = 0$, 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, $t_{w_0} \in (0, T]$ is still valid. Compared to the original interval of t_{w_0} when $\delta_{bc} = 0$, this new interval includes a new range $(0, \delta_{bc}]$ and excludes $(T, T + \delta_{bc}]$. For the other $t_{w_0} \in (\delta_{bc}, T]$, the discovery latency can be directly derived from those produced by (3). Also, each $t_{w_0} \in (0, \delta_{bc}]$ can be projected to the excluded range $(T, T + \delta_{bc}]$, where the discovery latency values have already been produced as well. Specifically, we have the following statement as the Range-Entrance Projection:

Theorem B.2: $\forall t_{w_0^i} \in (0, T]$, $\delta_{bc} \in (0, A)$, the discovery latency can be represented as

$$L(\delta_{bc}, t_{w_0}^{\delta_{bc}}) = L(0, t_{w_0}^0) + \delta_{bc}, \quad (4)$$

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

C. Integrated Case Projection Progress

By combining Equations (3) and (4), a formula is derived to get all $A{\cdot}T$ discovery latency values from $\{L(0, t_{w_0^i}) | t_{w_0^i} \in (0, A]\}$. Theoretically, the trivial case $A = T$ 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 broadcasting and scanning and a high probability of infinite discovery latency.

Theorem B.3: When $A < T$, given T, A, W, any case (δ_{bc}, δ_s) can be projected to A base cases and retrieve its discovery latency by:

$$\begin{aligned} L_{T,A,W}(\delta_{bc}, \delta_s) = & L_{T,A,W}\left(0, (\delta_s - \delta_{bc} + T) \bmod T\right) \bmod A \\ & + \delta_{bc} + \left\lfloor \frac{(\delta_s - \delta_{bc} + T) \bmod T}{A} \right\rfloor \cdot A \end{aligned}$$

When $A > T$, 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 $A_0^0 \in [0, T)$ to $A_0^i \in [T, A)$ similar to Equation (3) where the attributes for scan and broadcasting 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 2(b). When $A < T$ for any δ_{bc}, no discovery can occur in $[t_0, t_0 + \delta_{bc})$ since no broadcasting event exists in this interval. When $A > T$, however, a scan window can cover a partition of $[t_0, t_0 + \delta_s)$. If A_0 occurs in that time period, $L(A_0, \delta_s)$ will not be equivalent to $L(A_1, \delta_s)$ with A_1 as the subsequent broadcasting event in the same Q_A as A_0, but become δ_{bc} instead. This affects Equation (4), which together with the above reformed Equation (3) constructs the projection function below.

Theorem B.4: When $A > T$, given $T, A, W, \delta_{bc}, \delta_s$, the discovery latency can be projected to T pre-calculated latency values from simulation by:

$$L_{T,A,W}(\delta_{bc}, \delta_s) = \begin{cases} \delta_{bc} & \text{if } \delta_s > 0 \wedge 0 < \delta_s - \delta_{bc} < W \\ L^* + \delta_s + \left\lfloor \frac{(\delta_{bc} - \delta_s + A) \mod A}{T} \right\rfloor \cdot T & \text{otherwise} \end{cases} \quad (5)$$

where $L^* = L_{T,A,W}((\delta_{bc} - \delta_{bc} + A) \ mod \ A \ mod \ T, 0)$.

The benefit of the above 2-step case projection can be exposed by experiments to compare the running time between random sampling and case projection. 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, observed among 20 sets of scan and broadcasting parameter configurations using Python implementation.

Appendix C
Adapting Random broadcasting Delay and Environmental Interference with Effective-Scan-Window-Based Model

It is arduous to utilize the coverage theory when randomized broadcasting (i.e., the *adv_delay*) is involved, since the number of possible broadcasting sequences, differentiated by the intervals between consecutive broadcasting events, grows exponentially as more broadcasting events occur. The alternative is to simulate the whole discovery process between the broadcaster and the scanner from a global perspective for every range-entrance event.

The *adv_delay* introduced in BLE brings randomness to the periodic intervals of a broadcaster. This randomness first affects A_0, where the range of δ_{bc} expands to $[0, A + R_d)$ as the largest possible broadcasting interval is now $A + R_d$. However, the probabilities of δ_{bc} being each value in its range are no longer identical. The broadcasting interval T_{A_0} right before A_0 now has R_d candidate values with equal possibilities. For each possible T_{A_0}, δ_{bc} follows a uniform distribution in $[0, T_{A_0})$, which can always cover $[0, A)$. Therefore, the probability distribution of δ_{bc} can be divided into two parts, $\delta_{bc} \in [0, A)$ and $\delta_{bc} \in [A, A + R_d)$. 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 broadcaster's time control after t_0.

Algorithm 2: The Summation Accumulator.

```
1  accum_layers ←
       $[[1, 1, 1, ..(totally\ (R_d + 1)\ ones).., 1]]$ ;
2  layer_sums ← [];
3  Function GetProbability:
       Require: Index of $A_i$-$i$, Position of $A_i$ among its
                Possible Range-$ts\_pos$
       Result: Probability for $A_i$ to be at $ts\_pos$
4      while $i \leq length($accum_layers$)$ do
5          NextLayer();
6          $i + +$;
7      end
8      Run NextLayer() until
         $i > length(accum\_layers)$;
9      $cur\_layer \leftarrow accum\_layers[i - 1]$;
10     Return Probability← $\frac{\text{cur\_layer[ts\_pos]}}{\text{layer\_sums}[i-1]}$;
11 end
12 Function NextLayer:
13     last_layer←
         accum_layers[−1] new_layer ←
         $EmptyArray()\ of\ length($last_layer$)$+R$_d$;
14     new_layer[$j$]←Sum(last_layer[j:j-R$_d$]);
15     layer_sums += layer_sums[−1]$_d$;
16 end
```

As there are various broadcasting intervals, Q_A formed with a given δ_{bc} no longer consists of determined timestamps but probabilistic distributions for each broadcasting event in it. This uncertainty of the broadcasting 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 to a so-called *adv_delay* Accumulation Module. We first focus on the new form of Q_A. While A_0 is at a determined timestamp, A_1 would occur inclusively between $A_0 + A$ and $A_0 + A + R_d$ 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 a similar way, where each value 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 timestamps (ranging in $[A_0 + 2 \cdot A, A_0 + 2 \cdot (A + R_d)]$) 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 2) 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 broadcasting events become non-deterministic with a single δ_{bc}, the discovery events also

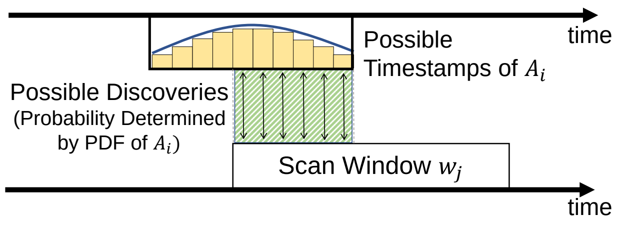


Fig. 3: Probabilistic Discovery with adv_delay

become probabilistic. Figure 3 shows an example case of discovery judgment. A broadcasting 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 judgment is shown in Algorithm 3, which can be applied when the loss is simultaneously considered. Specifically, lines 6 and 7 process the situations without/with packet loss.

It is worth noticing that, when adv_delay is involved, the case projection requires amendments. For example, when $A < T$, the Phase-Difference Projection (Equation 3 will no longer be based on adding multiples of A, since the actual interval between the broadcasting events is probabilistically ranging from A to $A + R_d$. Therefore, the value to be added is a probabilistic one, whose probability can be calculated by the summation accumulator. Due to the same reason, the Range-Entrance Projection for $A > T$ also changes the addition of A to adding each value in $A + R_d$ with the same probability. Meanwhile, these procedures may be equipped with versatile implementations, which can highly affect the complexity of the algorithm.

Algorithm 3: Updating the Summation Accumulator during the Loss-Involved Simulation.

```
Require: Position of $A_i$ among its Possible
         Range-ts_pos
1 Fetch cur_accum_layer and cur_layer_sum
    from
    current layer's status in Summation Accumulator;
2 if ts_pos is covered by current scan window then
3     ts_prob←GetProbability(i,ts_pos)
4     $P(latency = A_i) += $ ts_prob $\cdot (1 - F_p)$;
5     cur_accum_layer[ts_pos]$\times =$F$_p$;
6 end
```

References

[1] P. H. Kindt, M. Saur, M. Balszun, and S. Chakraborty, "Neighbor discovery latency in ble-like protocols," *IEEE TMC*, vol. 17, no. 3, pp. 617–631, 2017.