On Design and Performance of Offline Finding Network

Tong Li[†‡], Jiaxin Liang[‡], Yukuan Ding[§‡], Kai Zheng[‡], Xu Zhang[¶], and Ke Xu[‖]
Renmin University of China[†], Huawei[‡], HKUST[§], Nanjing University[¶], Tsinghua University[‖]

Abstract—**Recently, such industrial pioneers as Apple and Samsung have offered a new generation of offline finding network (OFN) that enables crowd search for missing devices without leaking private data. Specifically, OFN leverages nearby online finder devices to conduct neighbor discovery via Bluetooth Low Energy (BLE), so as to detect the presence of offline missing devices and report an encrypted location back to the owner via the Internet. The user experience in OFN is closely related to the success ratio (possibility) of finding the lost device, where the latency of the prerequisite stage, i.e., neighbor discovery, matters. However, the crowd-sourced finder devices show diversity in scan modes due to different power modes or different manufacturers, resulting in local optima of neighbor discovery performance. In this paper, we present a brand-new broadcast mode called ElastiCast to deal with the scan mode diversity issues. ElastiCast captures the key features of BLE neighbor discovery and globally optimizes 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, as part of 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, can only work online with Internet access. Moreover, with the complicated built-in functions of communication and positioning, these devices are usually expensive and are with shorter battery life. To tackle this issue, industrial pioneers (e.g., Tile, Nutspace and Nut Technology, Apple, and Samsung) have offered an offline finding network (OFN) that leverages nearby online devices (a.k.a., finder devices) to help users to locate their lost devices even when the devices are offline. Take Apple as an example. Currently, Apple has over 1 billion iPhones that already have the Find My application [1] on them, 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 highlight its design challenges from the perspective of performance. Particularly, in OFN applications, the user experience is closely related to the success ratio (possibility) of finding the lost device, where the neighbor discovery latency matters (see §IV-A). Neighbor discovery is the prerequisite stage of OFN, in which a finder device seeks to first contact 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).

The finder devices are composed of a group of users that are nearby and under the Bluetooth signal coverage of the lost device. For 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). These finder devices therefore usually show diversity in scan modes. Then the question becomes seeking the controllable broadcast mode (pattern) that adapts to the diversity of the uncontrollable scan modes (see §IV-C).

The decision-making of broadcast mode matters in OFN on both discovery latency and power consumption. On one hand, a large broadcast interval may result in a large discovery latency, which significantly reduces the possibility of finding lost devices (see §IV-A). On the other hand, power consumption by the broadcaster is inversely proportional to the broadcast interval. For example, with the default broadcast interval of 2000 ms [10], the AirTag's CR2032 lithium coin battery life lasts for one year [3], however, halving the broadcast interval to 1000 ms may proportionally halve the battery life. Hence, a smaller broadcast interval means higher power consumption, reducing the applicability of OFN systems.

The legacy way of neighbor discovery that adopts the fixed

This work is supported by the NSFC Projects (No. 62202473 and No. 61932016), the China National Funds for Distinguished Young Scientists (No. 61825204), and the Beijing Outstanding Young Scientist Program (No. BJJWZYJH01201910003011). Li and Ding's work is partially done at Huawei. Jiaxin Liang is the corresponding author (jiaxin.liang@huawei.com).

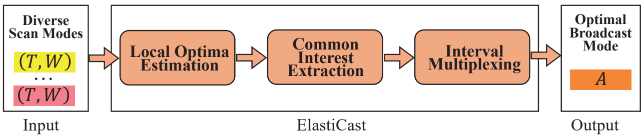


Fig. 1: Key modules in ElastiCast.

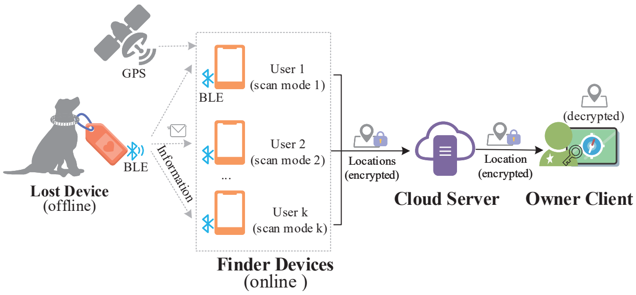


Fig. 2: Ecosystem overview of OFN.

broadcast mode is far from satisfactory, as it may result in local optima of neighbor discovery performance in the case of scan mode diversity, that is, the discovery latency is not bounded within power budget (see §IV-D). In this paper, we present 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 provides Blender [11], a neighbor discovery simulator that simulates the behavior of broadcasters and scanners. Its input contains multiple settings of scan modes among all possible finder devices, and its output is the set of latency distributions as the functions of the broadcast interval. Differ from the legacy way of *random sampling*, Blender adopts the equivalence relationship between cases to avoid uncertainty in outputs, significantly reducing estimation overhead (see §VI-A).

Common Interest Extraction makes full use of the non-linear relationship between discovery latency and power consumption (see §IV-B). It picks the common broadcast intervals that achieve minimized discovery latency among all scan modes. However, simply setting a single optimal broadcast interval might result in 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 adopts the intermixed use of multiple feasible broadcast intervals instead of the single one. It steps further 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 legacy way 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 shown to obtain an improvement of over 11% on the success ratio 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 §VII).

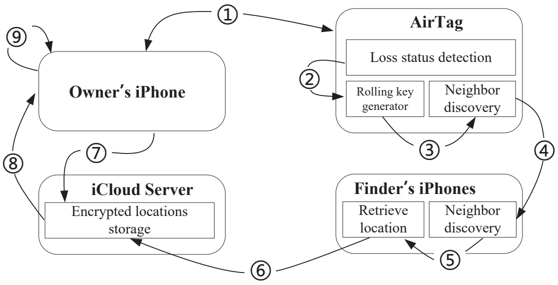
II. Related Work

Prior studies on OFN 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 key [3]. While Mira Weller et al. [5] proposed a symmetric key-based way called PrivateFind to provide end-to-end encryption of location data. The 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 examines 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 avoid detection by item safety alerts. Tonetto et al. [8] further present how OFN can be used to estimate large groups of people, which is beyond its originally designed purpose of tracking lost devices. In this paper, we do not focus on the privacy and security analysis 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.

Efficient neighbor discovery is characterized by achieving the shortest possible discovery latency for a given power budget. Towards this, a large number of broadcast and scan setting approaches have been proposed for general wireless neighbor discovery to date, see [12]–[22]. There are also studies specifically designed for BLE neighbor discovery [23]–[31]. 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. [31] 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. 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 overcomes the challenges induced by scan mode diversity.

III. Offline Finding Network: An Overview

Realizing your items have been missing is a sickening feeling and often one that elicits panic. For example, it is reported that nearly 2,000 phones are lost or stolen every single hour [32]. Moreover, millions of children or elderly people worldwide are going missing every year [33], [34], and one-third of all pets are reported missing in their lifetimes, and more than 80% are never found [35]. To locate missing devices even when offline, such industry leaders as Apple and Samsung are entering the market of OFN and are introducing their Bluetooth Tags, i.e., AirTag [3] and SmartTag [36]. These



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.

tags, as small battery-powered BLE devices without direct Internet access, can be attached to important items such as bags, keychains, or bikes.

In general, the ecosystem of OFN has 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. While it can also be a rich device (e.g., phone, tablet) without online access. The *Finder Devices* are composed of a group of users that are nearby and under 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 marked as lost (in **Step 2**), it generates rolling public keys periodically (e.g., 10 minutes) using a shared secret [37] (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**).

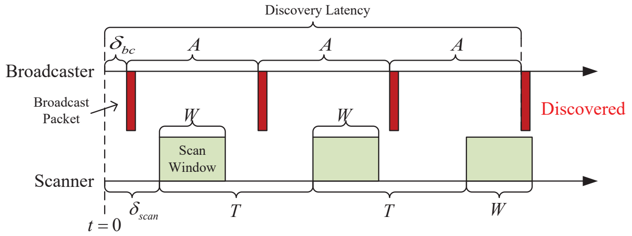


Fig. 4: The duty-cycling of BLE neighbor discovery. T, W, and A are the scan interval, scan window, and broadcast interval, respectively. δ_{bc} and δ_{scan} are the entrance time.

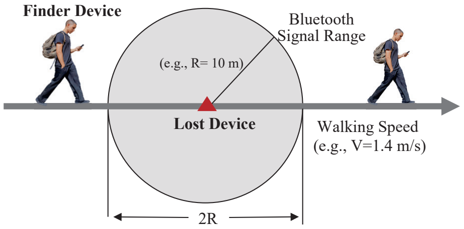


Fig. 5: Neighbor discovery between finder device and lost device in the walking scenario.

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 very natural extension to the conventional finding networks with crowd-sourcing.

IV. Neighbor Discovery Latency of OFN: Issues and Challenges

A. Neighbor Discovery Latency Matters in OFN

As the prerequisite stage of OFN, neighbor discovery is a 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 works in a duty-cycling paradigm (see Figure 4). It 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 point when both devices come into the range of reception (i.e., range-entrance time t = 0 in Figure 4).

As illustrated in Figure 4, the discovery latency is mainly bounded by 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 results in an interleaved activity pattern between the broadcaster and the scanner, which

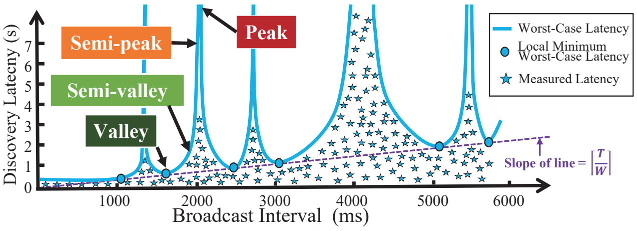


Fig. 6: The distribution of the worst-case discovery latency (upper bound). W = 1024 ms and T = 4096 ms (i.e., a representative scan mode labeled as BALANCED in modern Android systems [41]).

may induce unacceptably large discovery latency [31], [38]. Since both the lost devices and the finder devices in OFN are power-sensitive [3], [39], the large discovery latency may significantly reduce the possibility of finding lost devices.

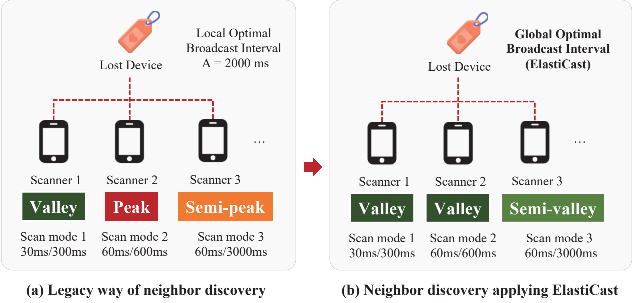
To explain this more clearly, we give an example of how a lost device is found by a finder device. As shown in Figure 5, a person with a finder device passes the Bluetooth signal range (e.g., a cycle area with a radius of $R = 10$ meters [40]) of a lost device at a walking speed (e.g., $V = 1.4$ meters per second)[1]. Taking into account 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. 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 [24], [42]. Recently, Kindt et al. [29] 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 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).

[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.



(a) Legacy way of neighbor discovery

(b) Neighbor discovery applying ElastiCast

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

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 [31] 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 do not violate the power constraint.

C. 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.

Dynamics. By default, the Android Open Source Project (AOSP) supports three scan modes labeled as LOW_LATENCY, BALANCED, and LOW_POWER [41]. 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 [43].

D. *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

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

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

$$Z = \min \hat{l}, \quad s.t. \quad A \geq A_{min} \tag{2}$$

where A is the selected broadcast interval, and A_{min} is the minimum broadcast interval, which is equivalently regarded as the power budget of a broadcaster in this paper.

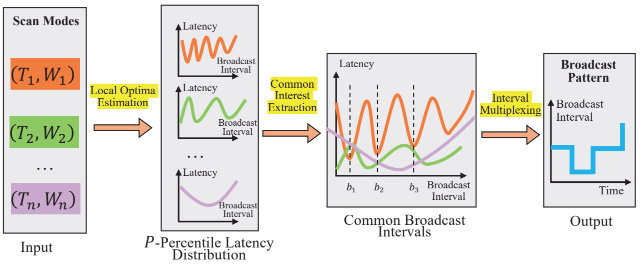


Fig. 8: The basic framework of data flow in ElastiCast.

B. *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 distribution[2]. A straightforward way is to repeatedly conduct real-world experiments [23], [26] which have low efficiency and a high cost, much less the bias induced by wireless interference [44]. 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 [31] 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, in this paper, we build a lightweight neighbor discovery simulation tool, Blender [11], 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 (1)) 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.

[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.

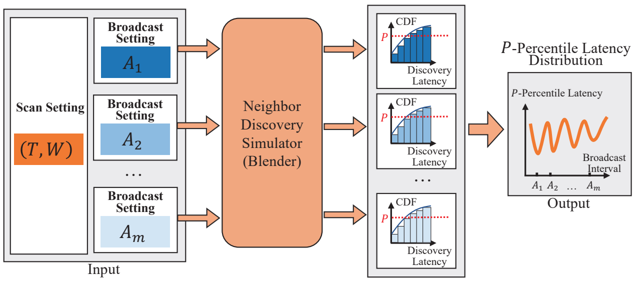


Fig. 9: The work progress of local optima estimation.

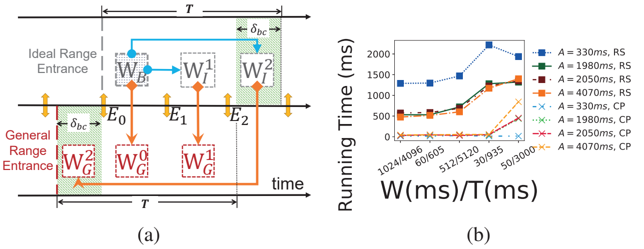


Fig. 10: (a) An example of the equivalence-relation-based Case Projection. (b) Blender (CP) outperforms random sampling (RS) (n=50000) in running time, therefore benefits the searching process of ElastiCast, especially when the searching space is large

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 [45], 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 12), 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

This section gives the design details of the specific modules in ElastiCast.

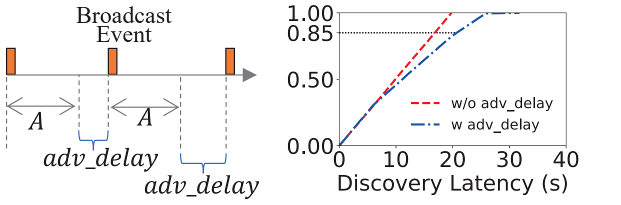
A. Local Optima Estimation

Since the local optima are estimated via comparing the P-percentile discovery latencies under a scan setting (T, W) with various broadcast intervals, the latency distributions are required to be calculated with given scan and broadcast settings. We built *Blender* (see [11]), a simulator that can simulate the behavior of the BLE broadcast and scan interaction in the BLE neighbor discovery process. As shown in Figure 9, Blender takes in an (A, T, W) combination and outputs the corresponding Cumulative Distribution Function (CDF) of discovery latency. The final output is the set that consists of the P-percentile latencies of all broadcast intervals.

A case refers to a pair of time offsets δ_{scan}, δ_{bc} from the time (i.e., range-entrance time $t = 0$ as shown in Figure 4) when the scanner enters the broadcaster's radio range. Given a case, the simulation continuously shifts the broadcast event time and scan window to see whether the broadcast packet arrives in between the scan window's start time and end time. As there can be a massive number of cases, the brute-force traversal through all cases can be unacceptable. Thus, the legacy way of simulation is random sampling, in which the simulator randomly runs a partition of them. However, tens of thousands of samples might be required to reach a stable output, which produces ad-hoc results and can be time-consuming.

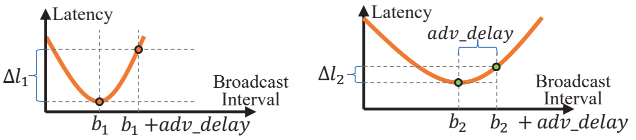
To overcome the limitation of random sampling, we propose a two-step approach, i.e., the Base Case Simulation and the Case Projection. Figure 10(a) illustrates an example of how it works. Although range-entrance time is decided by both δ_{scan} and δ_{bc}, without loss of generality, we make the broadcast event sequence fixed, for example, E_0 is the first broadcast event since the range-entrance time, then the problem becomes testing all the cases of different scan window positions. We first define the *ideal range-entrance situation* where the range-entrance time meets $\delta_{bc} = 0$ (i.e., grey dashed line), and define the *general range-entrance situation* where the range-entrance time meets $\delta_{bc} > 0$ (i.e., red dot line). Then we define the *base case* as the case whose scan window locates between E_0 and E_1 under the ideal range-entrance situation. In the example shown in Figure 10(a), W_B is the base case, and W_I^1, W_I^2, W_G^0, W_G^1, and W_G^2 are representative cases. For example, W_I^1 is the representative case where the scan window locates after E_1 under the ideal range-entrance situation, and W_G^2 is the representative case where the scan window locates in the green shadow area under the general range-entrance situation. We regard two cases are equivalent when (1) the scan windows have an equal gap to the corresponding broadcast event (e.g., W_B and W_I^1), or (2) the scan windows are in the same position of the scan interval (e.g., W_I^2 and W_G^2). In Blender, we first conduct the Base Case Simulation by producing the latency distribution of the base case. We then conduct the Case Projection by deriving the distribution of other cases based on their equivalent cases. For example, W_I^1 and W_G^0 are directly derived from W_B, and W_G^1 and W_G^2 are indirectly derived from W_I^1 and W_I^2, respectively.

To better understand how Blender outperforms random sampling, we conduct experiments where 20 sets of scan and broadcast parameters are run via simulation. Figure 10(b) shows the results. While the major drawbacks of random



(a) (b)

Fig. 11: (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. 12: Analysis on sensitivity to adv_delay.

sampling include the long running time due to repetitive sampling before the result resembles the theoretical CDF, Blender can reduce the running time by 2-10 times.

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 [31], the local minimum P-percentile latency is computed as

$$l_{min}^{P} = P \cdot \lceil \frac{T}{W} \rceil \cdot A \tag{3}$$

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

Definition 2: Semi-valley Area. We regard an A is within the semi-valley area if its P-percentile latency meets $l_A^P \leq \alpha \cdot l_{min}^P$, 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_A^P, the P-percentile latency of any given A. By comparing l_A^P and l_{min}^P 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 (1) and (2), 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:

$$\min \hat{l} = \sum_{i=1}^{n} \omega_i \cdot l_{i,j}, \quad s.t. \ \ b_j \geq A_{min}, \ \forall b_j \in \mathbf{B}^* \tag{4}$$

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 11(a), before each broadcast event [45]. 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 11(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 12, 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 achieves lower discovery latency than b_1. This greatly motivates the Interval Multiplexing as we will elaborate next.

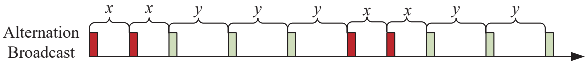


Fig. 13: An example of the Alternation Broadcast Pattern.

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 13 illustrates an example of the broadcast 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 13, 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 broadcast 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 (2) 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 further 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 [31].

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.

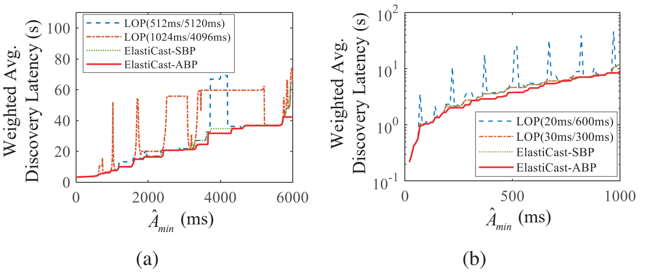


Fig. 14: Overall performance of ElastiCast. (a) Scan modes: $512ms/5120ms$ and $1024ms/4096ms$. (b) Scan modes: $20ms/600ms$ and $30ms/300ms$.

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.

VII. Evaluation

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.

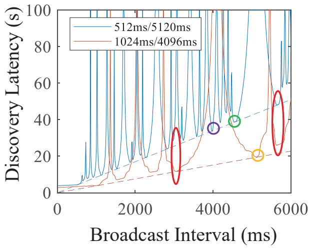
Parameters. We run tests for broadcast intervals that meet $A \in [A_{left}, A_{right}]$, where we recommend $A_{left} \geq 20$ ms and $A_{right} < \min\{10240ms,\ latency\_tolerance\}$ according to the whole range of broadcast intervals allowed in BLE. In this paper we set $A_{left} = 20$ ms and $A_{right} = 6000$ ms. The initial relaxation coefficient is set $\alpha = 1.2$.

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 (3)) 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 [31], 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 [41], and $20ms/600ms$ and $30ms/300ms$ are two types of scan modes adopted by the applications of HarmonyOS (e.g., HiLink [43])

[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.



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

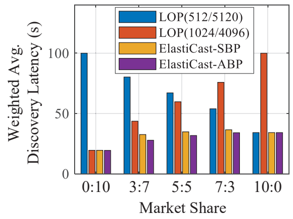
and iOS (e.g., Apple's Find My [3]), respectively. The market shares of both scan modes are 50%. "LOP ($20ms/600ms$)" refers to the scheme that locally optimizes latency for the scan mode of $20ms/600ms$, and so forth.

Figure 14 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 demonstrated that ElastiCast can always bound the neighbor discovery latency in the case of scan mode diversity. Specifically, Figure 14(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 14(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 15(a), we mark the selected/optimal broadcast interval(s) for each scheme with colored cycles. Figure 15(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 §VII-B, ElastiCast shows the stability in achieving global optima regardless of the types of scan modes.



(a)

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

(b)

Fig. 16: (a) Performance under different market shares. (b) Performance comparison between AirTag and ElastiCast.

How does the power budget impact stability? As shown in Figure 14, the power budget impacts the decision-making significantly, however, ElastiCast still approximates a bounded and globally minimized discovery 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 16(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 [39]. 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[4]. 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 16(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

[4] The measurement of the AirTag indicates the results under our specific types of scan modes (iOS systems may have different scan modes) without considering further optimization techniques applied in its commercial product.

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.

VIII. 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.

Acknowledgment

We are grateful for the conversations with and feedback from Lu Yan, Dan Wang, Jie Li, Neng Yang, Zelin Chen, Baolong Li, and Jiangshui Hu. We also thank the reviewers for their valuable comments.

References

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

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

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

[4] "Airtags are apple's next billion dollar business," 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] "Airtag analysis," https://adamcatley.com/AirTag.html.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

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

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

[20] 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.

[21] 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.

[22] 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.

[23] Texas Instruments, "Measuring bluetooth low energy power consumption," in *Application Note AN092*, 2010.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

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

[32] "How to easily find your lost android phone," https://www.asurion.com/connect/tech-tips/how-to-easily-find-your-lost-android-phone/.

[33] "The international centre for missing and exploited children," https://www.icmec.org/global-missing-childrens-center/imcd/.

[34] "Number of missing persons in the united states in 2020," https://www.statista.com/statistics/240387/number-of-missing-persons-files-in-the-us-by-age/.

[35] "U.s. missing pet epidemic," https://peeva.co/missing-pet-epidemic-facts-and-figures.

[36] "Smarttag," https://www.samsung.com/us/mobile/mobile-accessories/phones/samsung-galaxy-smart-tag-1-pack-black-ei-t5300bbegus/.

[37] 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.

[38] 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.

[39] "Huawei tag," https://consumer.huawei.com/cn/accessories/tag/.

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

[41] "Android ble scan settings apis," https://developer.android.com/reference/android/bluetooth/le/ScanSettings.

[42] 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.

[43] "Huawei hilink," https://iot.hilink.huawei.com/.

[44] 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.

[45] "Bluetooth specifications," https://www.bluetooth.com/.