Asynchronous Neighbor Discovery on Duty-Cycled Mobile Devices: Models and Schedules

Sixia Chen, *Member, IEEE*, Reynaldo Morillo, *Student Member, IEEE*, Yanyuan Qin, *Student Member, IEEE*, Alexander Russell, Ruofan Jin, *Member, IEEE*, Bing Wang, *Senior Member, IEEE*, and Sudarshan Vasudevan, *Member, IEEE*

***Abstract*— Neighbor discovery is a fundamental problem in wireless networks. In this paper, we study asynchronous neighbor discovery on duty-cycled mobile devices. Most existing studies develop integer schedules where time proceeds in discrete slots and a node is awake or asleep for an entire slot duration. We show that integer schedules can lead to significant waste of resources, and develop a generalized non-integer model, where time is continuous and a node may become awake or asleep at any point of time (subject to a few constraints) so that the resultant schedules can be significantly more efficient than integer schedules. In addition, we provide a reduction that transforms any schedule in the integer model to a corresponding schedule in the generalized non-integer model while reducing the discovery latency by up to a factor of two. Applying this reduction, an optimal schedule in the integer model becomes an optimal schedule in the non-integer model. We further demonstrate the practicality of non-integer schedules in a testbed, and compare the worst-case discovery latency of several existing schemes under both integer and non-integer models. Last, we establish a family of lower bounds for the best achievable latency guarantee. These lower bounds are applicable to both integer and non-integer models, covering both symmetric and asymmetric settings, and encompassing the existing lower bounds that are only for a subset of settings as special cases.**

***Index Terms*— Wireless networks, network management, neighbor discovery.**

I. Introduction

NEIGHBOR discovery is a fundamental problem in wireless networks. Specifically, the proliferation of mobile devices, e.g., smartphones, PDAs, and sensors, has fueled many novel applications that utilize opportunistic localized device-to-device communication, including mobile social networking [25], [31], [32], [36], proximity-based services [1], [45], media sharing [28], traffic uploading [13], asset management [33], emergency rescue [15], tracking encountering patterns of migrating animals [24], device-to-device communication in 5G cellular networks [12], [38], [39], and many more. These applications require mobile devices to discover each other as they move into each other's transmission range. Limited battery resources often mandate that these devices are duty-cycled. As a result, two nodes are only able to discover each other when both are awake. In addition, in practice, nodes typically have no means of coordination (e.g., through a shared server or dedicated control channel), their clocks may not be synchronized, and they cannot even rely on individual "identities" for the purposes of symmetry breaking.

We broadly refer to the above problem as *asynchronous* neighbor discovery on *duty-cycled* mobile devices. In this paper, we focus on *deterministic* neighbor discovery algorithms, where the schedules of the nodes are determined beforehand. In such algorithms, the worst-case discovery latency is bounded. In addition, since the nodes' schedules are deterministic, one node can precisely determine the future awake times of another node once they discover each other, which facilitates their later communications. We consider both *symmetric* setting, where the duty cycles of the nodes are the same, and *asymmetric* setting where the nodes have different duty cycles.

Most existing studies [11], [14], [17], [22], [41] develop *integer* schedules where time proceeds in discrete slots of length Δ, and a node is either awake/active for an entire slot or asleep/inactive for an entire slot (we use *awake* and *active* interchangeably, and use *asleep* and *inactive* interchangeably). Two studies [2], [29] break away from integer schedules by allowing "overflow" of active slots (i.e., an active slot is slightly lengthened to intersect the previous or subsequent slot), and demonstrate that the slight overflow can provide significant improvement.

In this paper, we develop a *generalized non-integer* model that is significantly more flexible than the integer model. We further provide a reduction that transforms *any* schedule in the integer model into a corresponding schedule in the non-integer model, while achieving significant performance gains. In addition, we present experimental and analytical results on the generalized non-integer model. In more detail, our main contributions are as follows:

Manuscript received April 17, 2019; revised November 14, 2019 and April 9, 2020; accepted April 13, 2020. Date of publication May 6, 2020; date of current version August 12, 2020. This work was supported by the National Science Foundation under Grant 1117427, Grant 1407205, Grant 1717432, and Grant 1925706. This article was presented at MobiHoc 2015 [7]. The associate editor coordinating the review of this article and approving it for publication was K. Zeng. *(Corresponding author: Bing Wang.)*

Sixia Chen is with the Computer Science Department, Central Connecticut State University, New Britain, CT 06050 USA.

Reynaldo Morillo, Yanyuan Qin, Alexander Russell, and Bing Wang are with the Computer Science and Engineering Department, University of Connecticut, Storrs, CT 06269 USA.

Ruofan Jin was with the Computer Science and Engineering Department, University of Connecticut, Storrs, CT 06269 USA. He is now with Google LLC.

Sudarshan Vasudevan is with LinkedIn, Inc., Mountain View, CA 94043 USA.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2020.2990764

- We develop a generalized non-integer model motivated by the observation that the integer model needlessly constrains the switching between active and inactive states on integer boundaries and, additionally, does not reflect an important constraint that arises in practice. In our proposed non-integer model, time is continuous: nodes may become active or inactive at any point in time, subject to a few constraints. In addition to time slot Δ that is commonly used in the integer model, we introduce a *required meeting time* δ: to discover one another, two nodes must be simultaneously awake for a continuous interval of duration at least δ. We show that when $\delta = \Delta/2$, the integer model does not introduce waste of resources and is appropriate. However, when $\delta \ll \Delta$ (e.g., for certain smartphones where Δ is one or two seconds, while δ is in milliseconds), the integer model can lead to significant waste of resources, while the non-integer model is much more efficient.
- We provide a general reduction that can transform *any* schedule in the integer model to a corresponding schedule in the generalized non-integer model. Using our reduction, all integer schedules in the literature can be directly converted into their corresponding non-integer schedules, while reducing the worst-case discovery latency by up to a factor of two. In addition, an optimal schedule in the integer model becomes an optimal schedule in the non-integer model under the reduction.
- We demonstrate the practicality of non-integer schedules and explore their advantages over the corresponding integer schedules using experiments in a tested. In addition, we explore several practical issues (e.g., how to select δ and Δ in practice) using the testbed.
- We compare the worst-case discovery latency of the integer and non-integer variants of several existing schemes in both symmetric and asymmetric settings. In addition, we relate non-integer schedules constructed using our proposed reduction with existing non-integer schedules [2], [29], and demonstrate that they are special cases of our reduction.
- We derive a family of lower bounds on the best achievable latency guarantees for both the integer and non-integer models. These are the first lower bounds that are characterized by both Δ and δ, and the first lower bounds that are applicable to both symmetric and asymmetric settings. The two existing lower bounds derived in [49] and [29] are only for the symmetric setting; they are special cases of our lower bounds by taking $\Delta = 1$, $\delta = \Delta/2$ ([49]) and $\delta \ll \Delta$ ([29]).

In this paper, we substantially extend the preliminary version [7] in two main aspects: (i) we use experiments in a testbed to demonstrate the practicality of non-integer schedules and provide insights of running such schedules in practical settings (Section IV), and (ii) we present detailed comparison of worst-case discovery latency of the integer and non-integer variants of several existing schemes (Section V). In addition, we relate the non-integer model with the recent development in slotless protocols (Section II).

The rest of the paper is organized as follows. Section II briefly describes related work. Section III introduces the non-integer model and proposes a general reduction to transform any schedule in the integer model to a schedule in the non-integer model. Section IV presents experimental results in a testbed to demonstrate the practicality of non-integer schedules. Section V compares the worst-case discovery latency of several neighbor discovery schemes under integer and non-integer models. Section VI presents a family of lower bounds for the non-integer model, which encompasses the lower bounds of integer models as special cases. Section VII-B discusses optimal integer and non-integer schedules for both symmetric and asymmetric settings. Last, Section VIII concludes the paper.

II. Related Work

Neighbor discovery has been studied extensively in the literature (e.g., [3], [4], [18], [27], [42]). In the following, we only briefly review the studies that are closest to ours, specifically, those that develop deterministic schedules for asynchronous neighbor discovery between two duty-cycled mobile devices.

A. Integer Schedules

Most existing studies develop integer schedules. The studies in [14], [41] propose neighbor discovery protocols based on cyclic quorum systems. The study in [22] expands the idea of cyclic quorum systems and proposes a scheme, CQS-pair, that is based on Singer difference sets (as pointed out in [7], the construction for CQS-pair, however, takes exponential time, and hence is only practical for very large duty cycles). Disco [11] and U-Connect [17] are two integer schedules, both based on prime numbers. In Disco, node i with duty cycle d_i chooses two primes, $p_{i1} < p_{i2}$, such that $1/p_{i1} + 1/p_{i2} \approx d_i$, and wakes up in the slots that are multiples of p_{i1} or p_{i2}, $i = 1, 2$. By virtue of the Chinese Remainder Theorem, the two nodes can discover each other with the worst-case discovery latency of $p_{11}p_{21}$, which is on the order of $c/(d_1\ d_2)$, c being a constant. In U-Connect, a node uses a single prime, p. It wakes up in the first $(p+1)/2$ slots as well as the slots that are multiples of p in every p^2 slots. For duty cycle d, the prime, p, is chosen so that $(3p+1)/(2p^2) \leq d$. Suppose node i has duty cycle d_i and chooses a prime p_i, $i = 1, 2$. Then the two nodes can discover each other with the worst-case latency of $1/(p_1\ p_2) \approx 2.25/(d_1\ d_2)$. We propose a general reduction that can convert any integer schedule (including all the above schedules) to their corresponding non-integer schedules, while reducing the worst-case discovery latency by up to a factor of two.

B. Non-Integer Schedules

The first study that breaks away from integer schedules is [2]. Specifically, Bakht *et al.* design a probing based approach, Searchlight, by leveraging the constant offset between periodic awake slots. The authors propose both integer and non-integer variants (by slightly lengthening an active

slot) of Searchlight, and show that the non-integer variant significantly outperforms the integer variant. In our study, we propose a generalized non-integer model that encompasses the assumptions in [2] and provide a reduction that transforms any integer schedule to a corresponding non-integer schedule. This reduction can transform the integer variant of Searchlight to a non-integer version, and achieves the same performance as the non-integer variant of Searchlight (see Section V).

Another study [29] adopts the same non-integer assumption in [2]. For the symmetric setting, the authors propose an optimal scheme, Diff-Codes, that is based on Singer difference sets (they also provide a greedy heuristic technique to deal with the cases when prime numbers are sparse, which leads to sub-optimal schedules for those cases). Our study is significantly more generalized than the above study. Specifically, applying our reduction to an optimal integer schedule based on Singer difference sets leads to a family of optimal non-integer schedules for the symmetric setting; Diff-Codes is a special instance in this family of optimal schedules (see Sections V and VII-B).

C. Lower Bounds

Zheng *et al.* [49] derive a lower bound for integer schedules in the symmetric setting; for asymmetric setting, their approach reduces to an NP-complete minimum vertex cover problem. Meng *et al.* [29] derive a new lower bound for the symmetric setting under a non-integer assumption (by allowing an active slot to overflow to an adjacent slot). The lower bounds in the above two studies are only applicable to the symmetric setting. The family of lower bounds that we establish is much more general, encompassing the lower bounds in the above two studies as special cases (see Section VI). Specifically, it is applicable to both integer and non-integer models, and both symmetric and asymmetric settings. To the best of our knowledge, our study is the first that derives lower bounds that are applicable to both symmetric and asymmetric settings. It is also the first that provides lower bounds that are characterized by both Δ and δ (two key parameters in the generalized model).

D. Slotless Protocols

Slotless or purely interval-based neighbor discovery protocols [16], [21] differ from slotted protocols (i.e., integer schedules) in that time is assumed to be continuous. A node scans (or listens to) a channel periodically every T_s time units, with each scanning interval of duration d_s, and transmits/advertises a beacon (with duration d_a) periodically every T_a time units. In such settings, node A_1 discovers another node A_2 when it receives a beacon from A_2, and vice versa. The scan intervals and advertisement intervals are decoupled, in contrast to the settings in slotted protocols, where beacons are transmitted at the beginning and end of a slot, while the remaining time of the slot is the listening interval. In our proposed non-integer model, time is continuous in that the awake interval does not need to last for an entire slot duration, and beacons are transmitted at the beginning and end of an awake interval (see Section III). In both the slotted/integer model and our proposed non-integer model, two nodes discover each other close in time (within one slot for the integer model and within one awake interval for the non-integer model), while in slotless protocols, node A_1 can discover node A_2 much earlier than when node A_2 discovers node A_1 (and vice versa), i.e., the two-way discovery latency can be significantly larger than one-way discovery latency. The studies in [16], [21] propose strategies to reduce two-way discovery latency, e.g., once one node A_1 discovers another node A_2 (by receiving a beacon from A_2), A_1 adjusts its advertisement schedule so that A_2 can discover A_1 quickly (i.e., to achieve faster two-way discovery). Recent efforts [19], [20] analyze the conditions under which slotless protocols can provide deterministic discovery latency. Specifically, the study in [20] presents, for the first time, a mathematical theory that computes the neighbor discovery latencies for all possible parametrizations of the devices, and shows that upper bounds on the latency can be guaranteed for all parametrizations, except for a finite number of singularities. In this paper, we establish a family of lower bounds on the discovery latency of slotted/integer schedules and non-integer schedules.

E. Other Studies

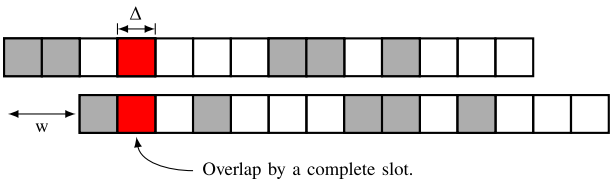
The studies in [46], [47] propose schemes that are based on existing asynchronous neighbor discovery schemes to further reduce energy consumption. The studies in [34], [43] allow beacons to be sent in non-active slots, which differ from the assumptions in this paper. The study in [37] proposes a generic framework that incorporates existing deterministic protocols, and allows flexibility in adjusting parameters. The study in [5] optimizes the duty cycle granularity of quorum and co-primality based protocols. Two studies [23], [48] propose neighbor discovery schemes for mobile duty-cycled devices that require clock synchronization, and hence are not applicable to asynchronous settings. A recent study [44] designs techniques for robust neighbor discovery in the presence of strong interference. All the above studies (including our work) focus on pairwise neighbor discovery between two duty-cycled nodes. Several studies [6], [9] consider neighbor discovery among a group of duty-cycled nodes, with the focus on leveraging the group setting to achieve faster discovery of new or hidden nodes in the network.

III. Integer and Non-Integer Models

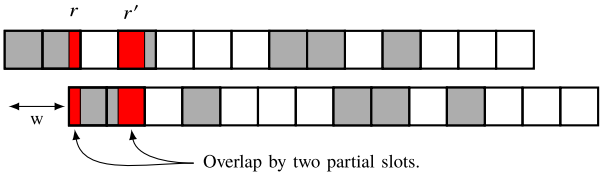
In this section, we briefly describe the integer model that is adopted by most existing studies. We then present an intersection property that holds for any valid schedule in the integer model, and use it to motivate the definition of the generalized non-integer model. After that, we establish a general reduction that transforms any schedule in the integer model to a corresponding schedule in the non-integer model.

A. Integer Model

In the integer model, time is discretized into slots of width Δ. A node is awake or asleep in a slot. When it is asleep, its radio is off and it cannot listen or transmit



(a) The intersection pattern resulting from an integral shift, $w = 2$.



(b) The intersection pattern resulting from a fractional shift, $w = 1.7$.

Fig. 1. Illustration of the integer model and the intersection property of the integer model. The shaded slots represent the awake slots. For simplicity, the two nodes in this figure have the same schedule except for the shift.

any packet. When it is awake, it can listen or transmit packets. Consider two nodes, A_1 and A_2. Each node has a fixed *duty cycle*, denoted as $d_i < 1$, $i = 1, 2$. The *schedule* of a node is represented by the slots when the node is awake. Let S_i denote the schedule of node A_i. For simplicity, we index the slots with the natural numbers $0, 1, 2, \ldots$. Then $S_i \subset \{0, 1, 2, \ldots\}$. We do not assume that the clocks of the two nodes are synchronized. As a result, the slot boundaries of the two nodes may not be aligned, as illustrated in Fig. 1, where the slot boundaries are aligned in Fig. 1(a), while not aligned in Fig. 1(b).

As in existing studies [2], [11], [17], [49], we assume that two nodes *discover* each other when they have an overlapping awake slot. This can be achieved, e.g., by assuming that a node sends a beacon message at the beginning and end of an awake slot. Since their slots may not be aligned, the overlapping duration of these two slots may be Δ or less than Δ, as shown in Figures 1(a) and (b), respectively, where the red regions represent the overlapping durations. The *discovery latency* is the time from when two nodes are in the transmission range of each other until their first discovery of each other.

B. Integer Model: Intersection Property

We next show that any valid schedule in the integer model has the following property:

Property 1: Consider two nodes using a schedule in the integer model for neighbor discovery. Suppose that when they come into each other's transmission range, their clocks read t_1 and t_2, respectively. Without loss of generality, suppose $t_2 > t_1$. Let $w = t_2 - t_1$ denote the relative time shift of the two nodes. Let S_1 and S_2 denote the schedules of these two nodes, respectively. Suppose the schedules guarantee discovery in time n. Then, for any w, we find one of two possible cases:

- The shift w is a multiple of Δ, and there exists an entire time slot during which both nodes are awake.
- The shift w is non-integral with respect to Δ (i.e., it is not a multiple of Δ), and there exist two time periods (among the first n) during which both nodes are awake; furthermore, the durations of these time periods, r and r', sum to Δ. Hence $r \geq \Delta/2$ or $r' \geq \Delta/2$.

This property is illustrated in Fig. 1. The first case in the above property represents the case when the slot boundaries of the two nodes are aligned, and is straightforward. We therefore only describe why the second property holds. To simplify the exposition, we assume $\Delta = 1$ (when $\Delta \neq 1$, one can scale time by $1/\Delta$ and apply the explanation below). Since w is non-integral, consider w as a fractional number between two integers z and $z + 1$. We shall show that the schedules of the two nodes, S_1 and S_2, have (at least) two overlaps, one of which has length $(w-z)$ and the other of which has length $z+1-w = 1-(w-z)$. We know that if S_2 started at z, then there would be a complete time slot (say s_1 in the schedule of S_2) so that S_1 and S_2 overlap. Since now S_2 starts at w, they will just overlap for a portion of s_1, which has length $1-(w-z) = z + 1 - w$. Similarly, if S_2 started at $z + 1$, there would be a complete time slot (say s_2 in the schedule of S_2 which is different from s_1) in which S_1 and S_2 overlapped. Since, in fact, S_2 is shifted to the left by length $z+1-w$, the overlap is actually a portion of length $1 - (z + 1 - w) = w - z$. It is clear that the sum of the two overlapping portions is one, and one of them has length at least half, as desired.

The above property indicates that there exists an overlapping duration of at least $\Delta/2$ when the two nodes are both awake under any circumstances. If the minimum amount of time required by the two nodes to discover each other when they are both awake is far below $\Delta/2$, then the overlap of at least $\Delta/2$ can lead to significant waste of resources. The above observation prompts us to define an additional parameter, the *required meeting time* $\delta > 0$, which is the minimum amount of time required by two nodes to discover each other when they are both awake.

In practice, the required meeting time, δ, depends on the device platform (including hardware, operating system and network stack). As we shall see in Section IV, our measurements on one mobile platform indicates that δ is around 20 ms for that platform; for other platforms, the value of δ varies in milliseconds [11], [17] or significantly less than one millisecond [33]. For a given δ, it is clear that the slot length, Δ, needs to be at least δ to allow successful discovery. In fact, based on the intersection property described earlier, Δ needs to be at least 2δ. In addition, the choice of Δ may also be subject to certain system constraints. Specifically, let *switching interval* represent the amount of time required for a node to change its state from asleep to awake or vice versa. In the integer model, since a node needs to be awake or asleep for an entire slot length Δ (and then may transit to another state), the value of Δ needs to be at least as large as the switching interval so that a node can finish switching the state at the end of a slot. The switch interval may be on the order of milliseconds [11], [17] or submillisecond [33]. However, for certain platforms, the switch interval can be significantly larger (e.g., one or two seconds on some smartphones [2]), leading to large Δ. For such platforms, since Δ can be much larger than δ, as described earlier, the waste caused by using an integer schedule is substantial. For instance, when

$\Delta = 1$ second, and $\delta = 10$ ms, then two nodes may have an overlapping awake duration larger than 500 ms, while only 10 ms is needed for neighbor discovery.

Motivated by the above observations, we propose a generalized model that considers both δ and Δ. This model provides a unified setting that accommodates the characteristics of a diverse range of wireless devices. It meaningfully considers continuous time, which is less restrictive than the integer model. In the rest of the paper, we assume that the choices of δ and Δ satisfy that $\delta \leq \Delta/2$, which is needed to ensure discovery. When $\delta = \Delta/2$, the integer model is appropriate since it does not lead to waste of resources. Specifically, by Property 1, the two time periods when both nodes are awake, r and r', may satisfy $r = r' = \Delta/2$, which is exactly δ, the needed amount of duration for discovery. When $\delta \ll \Delta/2$, as mentioned earlier, the integer model can lead to significant waste of resources.

C. Generalized Non-Integer Model

We now define the generalized non-integer model, which does not require a node to be awake or asleep for an entire slot duration. As we shall see, it encompasses the integer schedule as a special case, while allowing much more flexibility in defining schedules. Let Δ and δ be as defined above. We assume a node alternates in asleep and awake intervals. Let S denote the schedule for a node to be awake. Since a node does not need to wake up at the beginning of a slot or go to sleep at the end of a slot, we may consider arbitrary schedules of the form $S \subset \mathbb{R}$, where S consists of a set of disjoint intervals. That is, $S = \bigcup_\ell [a_\ell, b_\ell)$, each $b_\ell - a_\ell \geq \Delta$ so that the schedule satisfies the Δ switching constraint, and $b_\ell - a_\ell$ does not need to be a multiple of Δ; hence we refer to such schedules as *non-integer* schedules. Two nodes discover each other successfully only when their awake intervals overlap for a duration of at least δ. Specifically, consider a pair of nodes with schedules S_1, S_2, and their clocks read t_1 and t_2 when they come into the transmission range of each other. Let $L(S_1, S_2; t_1, t_2)$ denote the corresponding discovery latency. Then $L(S_1, S_2; t_1, t_2)$ is the smallest time at which the two schedules overlap at an interval of width at least δ. That is,

$$L(S_1, S_2; t_1, t_2) = \min \left\{ t \;\middle|\; \begin{array}{l} [t + t_1 - \delta, t + t_1) \subset S_1 \text{ and} \\ [t + t_2 - \delta, t + t_2) \subset S_2 \end{array} \right\}.$$

We define the *discovery time* for such a pair of schedules to be the worst case over all t_1 and t_2. That is,

$$L(S_1, S_2) = \min_{t_1, t_2 \geq 0} L(S_1, S_2; t_1, t_2).$$

The above model includes the integer model as a special case, i.e., when $b_\ell - a_\ell = \Delta$ for all ℓ. As mentioned earlier, when $\delta = \Delta/2$, using the integer model is appropriate, while when $\delta < \Delta/2$, the integer model can lead to waste of resources, particularly when $\delta \ll \Delta$. In the rest of the paper, we assume the integer model is used when $\delta = \Delta/2$ and the non-integer model is used when $\delta < \Delta/2$ (recall we only consider the cases that $\delta \leq \Delta/2$, which is the condition required for discovery).

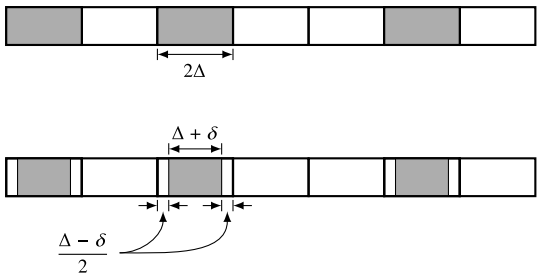


Fig. 2. Illustration of the reduction that converts an integer schedule to a corresponding non-integer schedule.

D. Converting Integer Schedules to Non-Integer Schedules

We now propose a general reduction that can be applied to any integer schedule to obtain a corresponding non-integer schedule with significantly improved performance when $\delta \ll \Delta/2$. The reduction is motivated by the following observation. Consider an integer schedule with slot length Δ. We can construct a new schedule by "trimming" the amount of time that a node is awake in an active slot from Δ to $\Delta/2 + \delta$. The new schedule still guarantees that two nodes can discover each other. This is because, by Property 1, if the two nodes discover each other by time n according to the original schedule, then there exists a time interval of at least $\Delta/2$, during which both of them are awake. After "trimming," since the amount of time a node is awake in an active slot is $\Delta/2+\delta$, the new schedule can still guarantee at least δ overlap during which both nodes are awake, which is sufficient for neighbor discovery. The duty cycle of the new schedule is $(\Delta/2+\delta)/\Delta$ of that of the original schedule, which is approximately a half when $\delta \ll \Delta$. The new schedule, however, may not be viable since the amount of awake time, $\Delta/2+\delta$, in a slot may become lower than the switching interval of a node, i.e., a node may not be able to switch the state, from asleep to awake and vice versa, within $\Delta/2 + \delta$. This violation can be easily fixed by specifying that the original integer schedule uses slot length of 2Δ instead of Δ; the "trimming" reduces the amount of time that a node is awake in an active slot from 2Δ to $\Delta + \delta$, which is larger than the switching interval Δ.

Summarizing the above, we propose the following reduction to convert an integer schedule to a non-integer schedule with duty cycle d. We first construct an integer schedule with duty cycle $d' = d(2\Delta)/(\Delta + \delta)$ and slot length 2Δ. We then apply a "trimming" procedure so that a node is only awake for a duration of $\Delta + \delta$ in each active slot so that the duty cycle is reduced to $d'(\Delta + \delta)/(2\Delta) = d$, as desired. Fig. 2 illustrates one "trimming" procedure, where the "trimming" is by removing $(\Delta - \delta)/2$ on each end of an active slot. In general, the "trimming" only needs to ensure that there remains a continuous length of $\Delta/2 + \delta$ awake time in an active slot. Many ways of "trimming" can be used. For instance, another way of "trimming" is removing $(\Delta - \delta)$ at the beginning or end of an active slot. In Section VII-A, we present a reduction that maps an optimal integer schedule to an optimal non-integer schedule.

For a given duty cycle, the non-integer schedule constructed as above can lead to significantly lower discovery latency

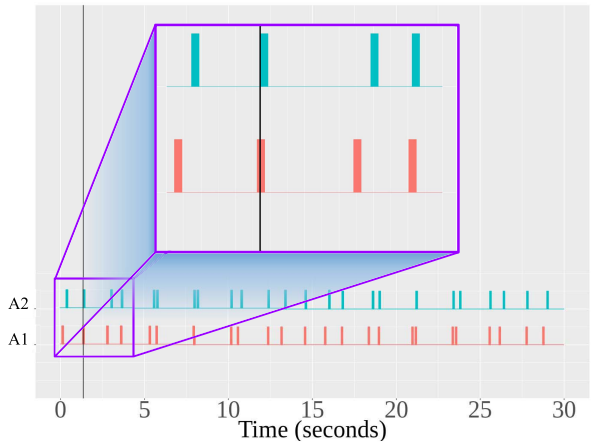


Fig. 3. Illustration of neighbor discovery between two nodes that run non-integer schedules.

compared to the original integer schedule when $\delta \ll \Delta/2$. The reason is as follows. Consider two nodes with duty cycles d_1 and d_2. As mentioned in Section II (and to be described in more detail in Section V), for many existing integer schedules, the worst-case discovery latency is $c/(d_1\ d_2) \cdot \Delta$, where c is a constant. Optimal schedules also have this property (see Sections VI and VII-B). When $\delta \ll \Delta$, where it is desirable to use non-integer schedules, we have $d_i' = d_i(2\Delta)/(\Delta + \delta) \approx d_i/2$, $i = 1, 2$. Then the discovery latency of the non-integer schedule constructed from the integer schedule is $c/(d_1' d_2') \cdot 2\Delta = c/(4d_1\ d_2) \cdot 2\Delta = c/(2d_1\ d_2) \cdot \Delta$, only half of the discovery latency of the original integer schedule. We shall illustrate the worst-case discovery latency of several non-integer schedules constructed from existing (integer model based) neighbor discovery schemes in Section V, and derive a precise lower bound on the worst-case discovery latency of non-integer schedules in Section VI.

Fig. 3 shows an example of neighbor discovery between two nodes, A_1 and A_2, that run non-integer schedules. The original integer schedules of these two nodes are constructed using Disco [11]. Applying the reduction as described above, we transform the integer schedules to the corresponding non-integer schedules (the "trimming" is as shown in Fig. 2). In this example, $\Delta = 100$ ms, $\delta = 20$ ms, and both nodes have duty cycle of 10%. The shaded bars represent the intervals when a node is awake. The long vertical line represents the first time when these two nodes discover each other, which is also shown in the zoomed-up subplot in Fig. 3 (marked by the purple box) along with the adjacent time slots.

IV. Experimental Results

We have implemented several neighbor discovery schemes (both the integer schedule and the corresponding non-integer schedule following the reduction in Section III-D) in a testbed. Our results below significantly extend the preliminary version [7] by demonstrating the practicality of non-integer schedules and the benefits of non-integer schedules over integer schedules in real experimental settings. We further provide insights on choosing the parameters (δ and Δ) using the testbed.

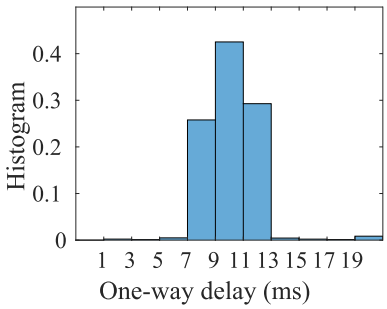


Fig. 4. One-way delay from a sender to a receiver in the testbed.

A. Experiment Setup

Our testbed contains three Raspberry Pi 3 Model B devices that are placed in the transmission range of each other. Two nodes run neighbor discovery schedules to discover each other; the third one serves as the *aggregator*, which sends the control commands to the other two nodes, and collects the experimental results. Each device has two network interface cards: an integrated BCM43438 WiFi interface that features single-band 2.4 GHz IEEE 802.11b/g/n, and a 100 Mbps Ethernet interface. The Ethernet interfaces of the three devices are used to control the experiments. Specifically, we connect the Ethernet interfaces to a switch; the control commands related to the experiments are sent from the aggregator to the other two devices via the Ethernet interfaces through the switch so that the latency is negligible. The wireless interfaces of the two nodes that perform neighbor discovery are configured in the ad-hoc mode for neighbor discovery. They are set to communicate over channel 1 using the IEEE 802.11 MAC protocol, CSMA/CA, for medium access. The neighbor discovery is achieved through broadcast to emulate the real-world scenario, where two nodes do not know each other's IP address before neighbor discovery is complete.

The operating system on each node is Raspbian 8. We program all the nodes using Python, leveraging the RPC/RMI (Remote Process Call/Remote Method Invocation) module Pyro (Version 4.60) [10]. This approach allows us to define the aggregator and nodes as objects, and makes them easily portable to other systems.

B. Determining δ

Recall that δ represents the required meeting time, i.e., the minimum amount of time required by two devices to discover each other when they are both awake. In our testbed, node A_1 is discovered by A_2 when a beacon transmitted by A_1 is received by A_2, and vice versa. Suppose that at time t, both nodes are awake, and A_1 sends a beacon to A_2. It takes one-way delay for the beacon to reach A_2, and hence the minimum overlap in their awake intervals, δ, corresponds to one-way delay. Note that δ is larger than the transmission latency of the wireless card since the media access follows a random access protocol, which introduces additional delays for a beacon to reach the receiver.

To obtain one-way delay, we perform the following experiments in the testbed. We set two nodes to serve as sender and

receiver, respectively. The sender transmits UDP packets periodically over the ad-hoc wireless network to the receiver. Each packet contains three fields, node ID, sending time and a sequence number. In addition, we set up a third node as a sniffer, dedicated to listening and logging all the packets it hears. All the nodes are synchronized using Network Time Protocol (NTP) so that we can obtain one-way latency from sending to receiving a packet (we verified that the clock differences of the nodes are within 2 ms). The sniffer records the time when it receives a packet, and then obtains the one-way delay of the packet as the difference of the sending time (carried by the packet) and the receiving time.

In each experiment, the sender transmits 1200 UDP packets periodically at the interval of 100 ms. We repeated the experiments 10 times at randomly chosen times of day and observe similar one-way delay distribution across the runs. Fig. 4 shows the histogram of the one-way delay over the 10 runs. We observe that most of the one-way delays are between 7 to 13 ms, and only a small percentage (0.8%) of the one-way delays is above 20 ms. Based on the above measurement results, we set $\delta = 20$ ms for all the experiments involving non-integer schedules in the testbed.

We remark that one-way delay is platform dependent. Existing studies that used other platforms such as Mote sensors [6], [11], software defined radios [29], [30], low-power RFID platforms [8], [26], and other embedded platforms [17], [33] reported much lower one-way delay. Correspondingly, for those platforms, δ can be set as a much smaller value.

C. Neighbor Discovery Results

We now report the results of neighbor discovery using the testbed. Specifically, we have implemented several representative schemes: Disco [11], U-Connect [17], Searchlight [2], and Singer [49], in the testbed. For each scheme, we report the results under both integer and non-integer schedules. For a given setting, the two nodes, A_1 and A_2, follow pre-determined schedules for neighbor discovery. Specifically, a node alternates in asleep and awake modes: when it is asleep, it does not transmit or listen to packets; when it is awake, it broadcasts a beacon (a UDP packet with the node ID) at the beginning and end of the awake interval, and listens for the rest of the awake interval. As mentioned earlier, δ is set to 20 ms, based on measurements results reported in Section IV-B; Δ is set to 100 ms, significantly larger than δ to demonstrate the benefits of non-integer schedules over integer schedules (we have also explored the setting when Δ is 200 ms and observed similar trends). To emulate real-world scenarios, the slot boundaries of the two nodes are not necessarily aligned (rather, the offset between the slot boundaries of the two nodes is selected randomly). When A_1 discovers A_2, it sends the discovery latency (i.e., the time when the discovery happens minus the time when the neighbor discovery process starts) to the aggregator. Similarly, when A_2 discovers A_1, it sends the discovery latency to the aggregator. The aggregator records the discovery latency as the maximum of the above two values.

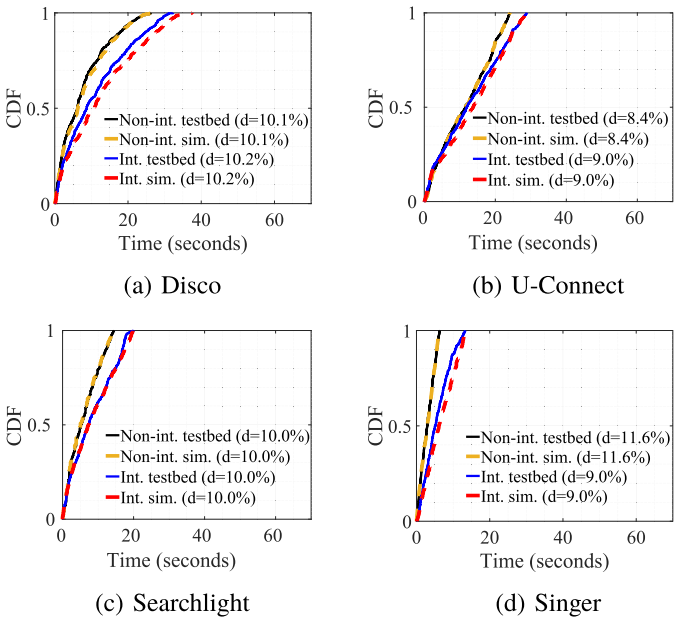


Fig. 5. Experimental results for neighbor discovery in the symmetric setting.

To validate the results from the testbed, we further implement the various schemes in a simulator that we developed in MATLAB. Neighbor discovery in the simulator is modeled as follows. For integer schedules, the discovery is achieved when the two nodes wake up in the same slot, which is the standard notion of discovery in the literature. For non-integer schedules, the discovery is achieved when the two nodes' awake durations overlap for at least δ.

We first report the results in the symmetric setting when the two nodes have the same duty cycle. Fig. 5 plots the cumulative distribution function (CDF) of the discovery latency for the various schemes. For each scheme, the figure shows the results for four cases, two cases under the non-integer model (one from simulation and the other from the testbed), and the other two cases under the integer model (again one from simulation and the other from the testbed). For each case, the results are obtained from 500 runs (using schedules with random starting offset). The duty cycle in a particular setting is determined by choosing the parameter(s) so that the resultant duty cycle is the closest to 10%. Since the two nodes have the same duty cycle, we represent the duty cycle as d; the value of d is marked for each case in the figure. U-Connect and Singer both take a single prime number as the parameter. Due to the sparsity of prime numbers, the duty cycles under these two schemes are further away from 10% than those under Disco (which takes two primes) and Searchlight (which takes an integer parameter that does not need to be a prime). We observe that, for each setting, the discovery latency obtained from the testbed is close to that from the simulation, particularly for non-integer schedules. For integer schedules, the discovery latency from the testbed tends to be lower than that from the simulation (the study in [2] made the same observation), since the model of neighbor discovery used in the simulation is stricter than what is needed in practice—in practice, two nodes discover each other as long

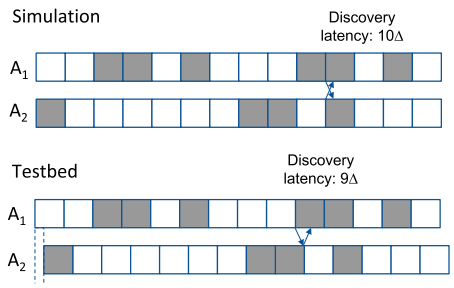
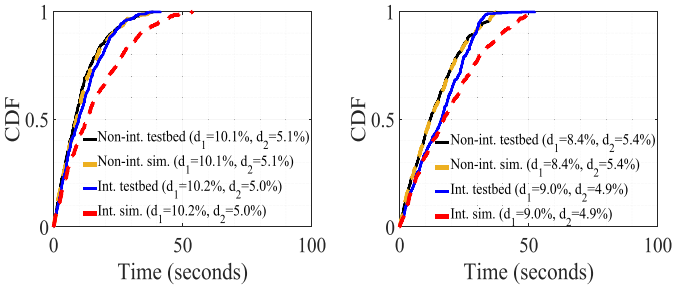


Fig. 6. An example that illustrates the difference of discovery latency in the simulation and the testbed in the integer model.



(a) Disco (b) U-Connect

Fig. 7. Experimental results for neighbor discovery in the asymmetric setting.

as they hear beacons from each other; their awake duration do not need to overlap for one time slot as modeled in the simulation. Fig. 6 illustrates the above difference in the integer model using an example. In the example, the discovery latency in the simulation is 10Δ when both nodes are awake in the 10th slot, while the latency in the testbed is 9Δ, when the two nodes have partial overlap in their awake slots. For the non-integer model, the overlap between two nodes when they are both awake tends to be shorter (the design is to ensure only δ overlap), and hence the results under the simulation and the testbed are closer compared to those in the integer model. For all the schemes, we observe that, indeed, non-integer schedules lead to much lower discovery latency than integer schedules.[1]

Fig. 7 shows the results in the asymmetric setting. The results of two schemes, Disco and U-Connect, are shown in the figure. For each scheme, the parameters of the scheme are chosen so that the duty cycle of one node is close to 10% and the duty cycle of the other node is close to 5% (the actual duty cycles for each setting are marked in the figure). We observe similar results as those in the symmetric setting: for the non-integer model, the results from the testbed are similar to those from the simulation; for the integer model, the discovery latency under the testbed tends to be lower than that from the simulation, again because of the more conservative assumption in the simulation setting. The simulation results show that the discovery latency under the non-integer model is significantly lower than that under the integer model; the results under the testbed are less clear due to the stochastic one-way latency and other non-deterministic factors in the testbed.

[1]For U-Connect, the discovery latency under the non-integer model is not strictly lower than that under the integer model (see Fig. 5(b)). This is because the non-integer schedule uses a lower duty cycle than the integer schedule.

V. Worst-Case Discovery Latency

Having demonstrated the practicality of non-integer schedules and their benefits over integer schedules, we now focus on worst-case discovery latency of non-integer and integer schedules (we only consider deterministic schedules in this paper, which provides a guarantee on the worst-case discovery latency). Specifically, we consider several existing schemes: Disco [11], U-Connect [17], Searchlight [2], and schemes using Singer difference sets [22], [49], and show their worst-case discovery latency in the non-integer model, compared to the worst-case discovery latency of the integer counterpart.

A. Symmetric Setting

We first consider the symmetric setting, i.e., when the two nodes have the same duty cycle, d. Table I lists the worst-case discovery latency of four schemes in both the integer and non-integer models. In the following, we show that for all the schemes, the worst-case discovery latency of the non-integer schedule is approximately half of the corresponding latency of the integer schedule when $\delta \ll \Delta$. Specifically,

- For Disco [11], in the integer model, a node chooses two different primes, p_1 and p_2 so that $1/p_1 + 1/p_2 = d$; in the non-integer model, a node chooses two primes, p_1' and p_2' so that $(1/p_1' + 1/p_2') \cdot (\Delta + \delta)/(2\Delta) = d$. Following the recommendation in [11], we choose p_1 and p_2 to be balanced (i.e., they are approximately equal), leading to $p_1 \approx p_2 \approx 2/d$ for the integer schedule, and $p_1' \approx p_2' \approx 1/d$ for the non-integer schedule when $\delta \ll \Delta$. Therefore, from Table I, we see that the worst-case discovery latency of the integer schedule is approximately $4/(d^2) \times \Delta$, while the worst-case discovery latency of the non-integer schedule is approximately $1/(d^2) \times 2\Delta$, which is half of that of the integer schedule when $\delta \ll \Delta$.
- For U-Connect [17], a node chooses a single prime, p, so that $\frac{3p+1}{2p^2} = d$ in the integer model, while choosing a prime, p', satisfying $\frac{3p'+1}{2p'^2} \cdot \frac{\Delta+\delta}{2\Delta} = d$ in the non-integer model. When $\delta \ll \Delta$, we have $p' \approx p/2$ and hence, from Table I, the worst-case discovery latency of the non-integer schedule, $p'^2 \times 2\Delta$, is approximately half of the corresponding value, $p^2 \times \Delta$ in the integer schedule.
- Searchlight [2] uses a single integer, t, satisfying $\frac{2}{t} = d$ in the integer model, while using integer t' satisfying $\frac{2}{t'} \cdot \frac{\Delta+\delta}{2\Delta} = d$ in the non-integer model. When $\delta \ll \Delta$, since we have $t' \approx t/2$, the worst-case discovery latency of the non-integer schedule, $(t'^2/2) \times 2\Delta$, is approximately half of the corresponding value, $(t^2/2) \times \Delta$, in the integer schedule.
- Singer algorithm [22], [49] uses a single prime, p, satisfying $\frac{p+1}{p^2+p+1} = d$ in the integer model, while using prime p' satisfying $\frac{p'+1}{p'^2+p'+1} \cdot \frac{\Delta+\delta}{2\Delta} = d$ in the non-integer model. Again when $\delta \ll \Delta$, since $p' \approx p/2$, the worst-case discovery latency of the non-integer schedule, $(p'^2 + p' +$

TABLE I
WORST-CASE DISCOVERY LATENCY IN THE SYMMETRIC SETTING

	Integer schedule		Non-integer schedule	
Scheme	**Parameter(s)**	**Worst-case discovery latency**	**Parameter(s)**	**Worst-case discovery latency**
Disco	p_1, p_2 s.t. $\frac{p_1+p_2}{p_1 p_2} = d$	$p_1 p_2 \times \Delta$	p'_1, p'_2 s.t. $\frac{p'_1+p'_2}{p'_1 p'_2} \cdot \frac{\Delta+\delta}{2\Delta} = d$	$(p'_1 p'_2) \times 2\Delta$
U-Connect	p s.t. $\frac{3p+1}{2p^2} = d$	$p^2 \times \Delta$	p' s.t. $\frac{3p'+1}{2p'^2} \cdot \frac{\Delta+\delta}{2\Delta} = d$	$p'^2 \times 2\Delta$
Searchlight	t s.t. $\frac{2}{t} = d$	$(t^2/2) \times \Delta$	t' s.t. $\frac{2}{t'} \cdot \frac{\Delta+\delta}{2\Delta} = d$	$(t'^2/2) \times 2\Delta$
Singer	p s.t. $\frac{p+1}{p^2+p+1} = d$	$(p^2+p+1) \times \Delta$	p' s.t. $\frac{p'+1}{p'^2+p'+1} \cdot \frac{\Delta+\delta}{2\Delta} = d$	$(p'^2+p'+1) \times 2\Delta$

TABLE II
WORST-CASE DISCOVERY LATENCY IN THE ASYMMETRIC SETTING

	Integer schedule		Non-integer schedule	
Scheme	**Parameters**	**Worst-case discovery latency**	**Parameter(s)**	**Worst-case discovery latency**
Disco	$p_{i1}, p_{i2}, i = 1, 2$ s.t. $\frac{p_{i1}+p_{i2}}{p_{i1}p_{i2}} = d_i$	$\min(p_{11}p_{21}, p_{11}p_{22}, p_{12}p_{21}, p_{12}p_{22}) \times \Delta$	$p'_{i1}, p'_{i2}, i = 1, 2$ s.t. $\frac{p'_{i1}+p'_{i2}}{p'_{i1}p'_{i2}} \cdot \frac{\Delta+\delta}{2\Delta} = d_i$	$\min(p'_{11}p'_{21}, p'_{11}p'_{22}, p'_{12}p'_{21}, p'_{12}p'_{22}) \times 2\Delta$
U-Connect	$p_i, i = 1, 2$ s.t. $\frac{3p_i+1}{2p_i^2} = d_i$	$p_1 p_2 \times \Delta$	p'_1, p'_2 s.t. $\frac{3p'_i+1}{2p'^2_i} \cdot \frac{\Delta+\delta}{2\Delta} = d_i$	$p'_1 p'_2 \times 2\Delta$

$1) \times 2\Delta$, is approximately half of the corresponding value, $(p^2+p+1) \times \Delta$, in the integer schedule.

We further remark that the above "trimming" based non-integer Singer schedule has the same performance as that of Diff-Codes [29]. This is because Diff-Codes expands each slot in the schedule from Singer sets to two consecutive slots, an active slot followed by an inactive slot, and an inactive slot followed by another inactive slot. When overflowing an active slot to its subsequent slot, it coincides with a special form of "trimming" (i.e., by "trimming" an amount $\Delta - \delta$ at the end of a 2Δ slot).

We next present numerical results on the worst-case discovery latency for the above four schemes under the original integer schedules and the "trimmed" non-integer schedules. The duty cycle d is varied in a wide range, from 0.5% to 5%, and δ/Δ is set to 0.1. Fig. 8(a) plots the results for three schemes, Disco, U-Connect, and Singer. Since all these three schemes take primes as parameters, the density of samples under lower duty cycles is higher than that under higher duty cycles due to higher density of primes. We observe that for all three schemes, the worst-case discovery latency of the non-integer schedule is indeed approximately half of that of the integer schedule. Fig. 8(b) plots the worst-case discovery latency of three variants of Searchlight, where the duty cycle values are chosen evenly (this can be easily achieved since Searchlight uses integers as parameters). Two variants, Searchlight-NS (no striped probing) and Searchlight-S (with striped probing), were proposed in [2]. Specifically, Searchlight-NS provides an integer schedule, while Searchlight-S provides a non-integer schedule since it allows each active slot to "overflow" by δ. The third variant, referred to as "Searchlight trim," is constructed by us, by converting Searchlight-NS to a non-integer schedule following our proposed reduction in Section III-D. We observe that the performance of "Searchlight trim" overlaps with that of Searchlight-S, both reducing the worst-case

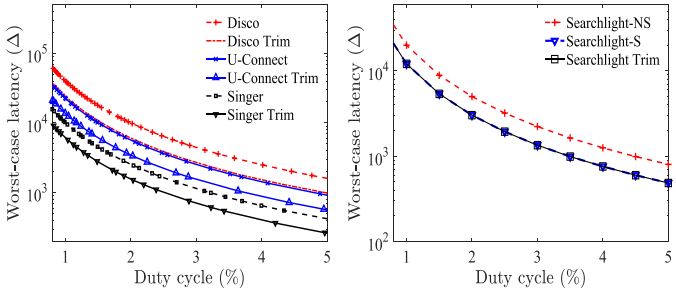


Fig. 8. Symmetric setting: worst-case discovery latency of integer schedules and their corresponding non-integer schedules.

discovery latency of the integer schedule, Searchlight-NS, by approximately a factor of two.

B. Asymmetric Setting

Table II shows the worst-case discovery latency for two schemes, Disco and U-Connect, in the asymmetric setting, where the two nodes have duty cycles, d_1 and d_2, and $d_1 \neq d_2$. For both schemes, the worst-case discovery latency under the non-integer model is approximately half of that under the integer model when $\delta \ll \Delta$. Specifically,

- For Disco, when $\delta \ll \Delta$, the two primes, p'_{i1} and p'_{i2}, chosen by node i, $i = 1, 2$, in the non-integer schedule satisfy that $p'_{i1} \approx p_{i1}/2$ and $p'_{i2} \approx p_{i2}/2$, where p_{i1} and p_{i2} are the two primes in the integer schedule (we assume that a node chooses either a balanced or unbalanced pair of primes; the preference between balanced versus unbalanced pair is the same in the integer and non-integer schedules). Hence, from Table II, the worst-case discovery latency of the non-integer schedule, $\min(p'_{11}p'_{21}, p'_{11}p'_{22}, p'_{12}p'_{21}, p'_{12}p'_{22}) \times 2\Delta$, is approximately half of the corresponding value,

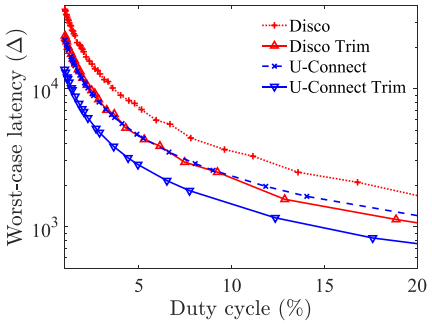


Fig. 9. Asymmetric setting: worst-case discovery latency of integer schedules and their corresponding non-integer schedules ($d_1 = 1\%$ and d_2 is varied from 1% to 20%).

$\min(p_{11}p_{21}, p_{11}p_{22}, p_{12}p_{21}, p_{12}p_{22}) \times \Delta$ for the integer schedule.

- For U-Connect, when $\delta \ll \Delta$, the prime, p_i', chosen by node i, $i = 1, 2$, in the non-integer schedule is approximately half of the prime, p_i, in the integer schedule, and hence the worst-case discovery latency of the non-integer schedule $p_1' p_2' \times 2\Delta$ is approximately half of the corresponding value, $p_1\ p_2 \times \Delta$, for the integer schedule.

Fig. 9 shows some numerical results when $\delta/\Delta = 0.1$, and the duty cycle of one node is fixed to be 1%, while the duty cycle of the other node varies from 1% to 20%. The results for Disco assumes that a node uses two balanced primes. We observe numerically that the worst-case discovery latency of the non-integer schedule is indeed approximately half of that of the integer schedule.

VI. Lower Bounds

We next develop a general lower bound in the non-integer model. This lower bound is characterized by both Δ (slot length) and δ (required meeting time), satisfying $\delta \le \Delta/2$. It applies to the general case where two nodes have duty cycles, d_1 and d_2, which may be the same or different, corresponding to symmetric or asymmetric setting. In the following, we first present and prove the lower bound, and then remark that it leads to a family of lower bounds. While derived in the non-integer model, the lower bound encompasses the integer model as a special case, since when $\delta = \Delta/2$, no "trimming" needs to be done, and the model becomes an integer model.

Theorem 2: Consider two nodes that have schedules S_1, S_2 in the non-integer model with parameters Δ, δ, where $\delta \le \Delta/2$. Assume that S_1, S_2 have duty cycles d_1, d_2. Let $L(S_1, S_2)$ denote the neighbor discovery latency under schedules S_1, S_2. Then to guarantee discovery within time n, the duty cycles d_1 and d_2 need to satisfy that $d_1 d_2 \ge \Delta^2/(2n(\Delta - \delta))$. Conversely, for given d_1, d_2, the discovery latency is at least $\Delta^2/(2d_1 d_2(\Delta - \delta))$.

Proof: Consider two schedules $S_1 \subset \mathbb{R}^+$ and $S_2 \subset \mathbb{R}^+$ with duty cycles d_1 and d_2, respectively. We suppose that these two schedules achieve discovery latency n, and proceed to prove that we must then have

$$n \ge \frac{\Delta^2}{2d_1 d_2(\Delta - \delta)}. \tag{1}$$

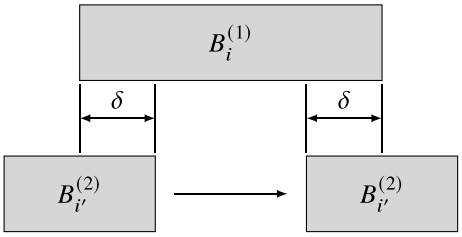


Fig. 10. Overlap occurrences from a pair of awake intervals between the two nodes.

Select two large times $T_1, T_2 \gg n$ for which $T_j + n \notin S_j$ (for each $j = 1, 2$). Since node j has duty cycle d_j, we have

$$\left|S_1 \cap [0, T_1 + n)\right| \le d_1(T_1 + n), \text{and}$$
$$\left|S_2 \cap [0, T_2 + n)\right| \le d_2(T_2 + n),$$

where the notation $|\cdot|$ is defined as follows: suppose $A = \bigcup_i [\alpha_i, \beta_i)$ is a disjoint union of half-open intervals, then $|A| = \sum_i (\beta_i - \alpha_i)$. We may express each of these "prefixes" of the schedules $S_j \cap [0, T_j + n)$ as a disjoint union of some (maximal) intervals:

$$S_1 \cap [0, T_1 + n) = \bigcup_{i=1}^{k_1} B_i^{(1)}, \qquad S_2 \cap [0, T_2 + n) = \bigcup_{i=1}^{k_2} B_i^{(2)},$$

where each $B_i^{(j)}$ is the ith awake interval appearing in S_j. By definition, $B_i^{(j)}$ has the form $[\alpha, \beta)$. Let $\ell_i^{(j)} = |B_i^{(j)}|$, i.e., $\ell_i^{(j)}$ represents the length of the ith awake interval in S_j. Then, summarizing the above, for each node j,

$$|S_j \cap [0, T_j + n)| = \sum_i \ell_i^{(j)} \le d_j(T_j + n).$$

In the following, we extend the notation $|A|$, defined above for subsets $A \subset \mathbb{R}$, to subsets of $\mathbb{R} \times \mathbb{R}$. Specifically, if $A = [\alpha^{(1)}, \beta^{(1)}) \times [\alpha^{(2)}, \beta^{(2)})$, we define $|A| = (\beta^{(1)} - \alpha^{(1)}) \cdot (\beta^{(2)} - \alpha^{(2)})$. If A is a disjoint union $\bigcup_i A_i$, where each A_i is such a product of intervals, we define $|A| = \sum_i |A_i|$. In addition, for a subset $S \subset \mathbb{R}$ and an element $r \in \mathbb{R}$, we let $S + r = \{s + r \mid s \in S\}$ (and likewise define $S - r$).

As S_1 and S_2 achieve discovery latency n, for any $(t_1, t_2) \in [0, T_1) \times [0, T_2)$ the intersection of $(S_1 - t_1) \cap [0, n)$ and $(S_2 - t_2) \cap [0, n)$ must contain an interval of width at least δ. Recall that $B_i^{(1)}$ and $B_{i'}^{(2)}$ represent awake intervals for nodes 1 and 2, respectively. Fixing the position of interval $B_i^{(1)}$, observe that the interval $B_{i'}^{(2)}$ overlaps with $B_i^{(1)}$ at an interval of width δ for a collection of times of length $\ell_i^{(1)} + \ell_{i'}^{(2)} - 2\delta$. The above is illustrated in Fig. 10: for a given $B_i^{(1)}$, to ensure $B_i^{(1)}$ and $B_{i'}^{(2)}$ overlap for at least δ, the leftmost and rightmost possible positions of $B_{i'}^{(2)}$ are marked in the figure; the interval between these two positions, which is of length $\ell_i^{(1)} + \ell_{i'}^{(2)} - 2\delta$, includes all possible positions of $B_{i'}^{(2)}$ to ensure that $B_i^{(1)}$ and $B_{i'}^{(2)}$ overlap for at least δ. It follows that

$$\left| \left\{ (t_1, t_2) \;\middle|\; \begin{array}{l} B_i^{(1)} - t_1 \text{ meets } B_{i'}^{(2)} - t_2 \\ \text{at an interval of width } \delta \\ \text{in } [0, n) \end{array} \right\} \right| \le n(\ell_i^{(i)} + \ell_{i'}^{(2)} - 2\delta).$$

Let k_1 denote the total number of awake intervals for node 1 in $[0, T_1]$. Similarly, let k_2 denote the total number of awake

intervals for node 2 in $[0, T_2]$. As every pair $(t_1, t_2) \in [0, T_1) \times [0, T_2)$ must be covered by some pair of intervals as above, we have

$$\begin{aligned} T_1 T_2 &\leq \sum_{i,i'} n \left(\ell_i^{(1)} + \ell_{i'}^{(2)} - 2\delta \right) \\ &= n \left(k_1 \sum_{i'} \ell_{i'}^{(2)} + k_2 \sum_{i} \ell_i^{(1)} - 2k_1 k_2 \delta \right) \\ &\leq n(k_1 d_2 (T_2 + n) + k_2 d_1 (T_1 + n) - 2k_1 k_2 \delta) \,. \end{aligned}$$

We conclude that

$$n \geq \frac{T_1 T_2}{k_1 d_2 (T_2 + n) + k_2 d_1 (T_1 + n) - 2k_1 k_2 \delta} \,. \tag{2}$$

Additionally, we recall that an awake interval has length at least Δ, and hence

$$k_1 \Delta \leq d_1 (T_1 + n) \quad \text{and} \quad k_2 \Delta \leq d_2 (T_2 + n) \,. \tag{3}$$

To complete the proof, we establish an upper bound on the denominator of (2), defined as

$$\phi(k_1, k_2) \triangleq k_1 d_2 (T_2 + n) + k_2 d_1 (T_1 + n) - 2k_1 k_2 \delta$$

subject to the constraints (3). Note that, assuming k_2 satisfies (3),

$$\begin{aligned} \frac{\partial}{\partial k_1} \phi(k_1, k_2) &= d_2 (T_2 + n) - 2\delta k_2 \\ &\geq d_2 (T_2 + n) - 2\delta \frac{d_2 (T_2 + n)}{\Delta} \\ &= d_2 (T_2 + n) \left(1 - \frac{2\delta}{\Delta} \right) \geq 0 \,, \end{aligned}$$

where the first inequality is due to the constraint of k_2 in (3), and the second inequality is due to our assumption that $2\delta/\Delta \leq 1$. It follows that for any choice of k_2 consistent with (3), the function ϕ is maximized by taking k_1 as large as possible—that is, so that the constraint (3) for k_1 is tight. An analogous computation of $\partial\phi/\partial k_2$ establishes the corresponding statement for k_2: In particular, $\phi(k_1, k_2)$ is maximized when the constraints (3) are tight. Thus

$$\phi(k_1, k_2) \leq 2 d_1 \; d_2 (T_1 + n)(T_2 + n) \frac{\Delta - \delta}{\Delta^2} \,.$$

Finally, in light of (2), we conclude that

$$n \geq \frac{\Delta^2}{2 \; d_1 \; d_2 (\Delta - \delta)} \cdot \frac{T_1 T_2}{(T_1 + n)(T_2 + n)} \,.$$

As we are free to select T_j as large as we wish, when T_j approaches infinity, the second factor in the right hand of the above inequality approaches 1, and hence inequality (1) follows, as desired. ∎

Remarks: The lower bound in Theorem 2 leads to a family of lower bounds, encompassing existing lower bounds in [29], [49] as special cases. It further provides insights into the relationship between the lower bounds of integer and non-integer models, and optimal schedules in both models.

- *Non-Integer Model, Symmetric Setting:* When $d_1 = d_2$, the required duty cycle to achieve a discovery latency of n becomes $\Delta/\sqrt{2n(\Delta - \delta)}$. When $\Delta = 1$ and $\delta \ll \Delta$ (the assumptions in [29]), it recovers the lower bound of $1/\sqrt{2n}$ in [29], which is a lower bound of the non-integer model in the symmetric setting. The derivation of [29] assumes the symmetric setting specifically. Our lower bound, on the other hand, applies to both symmetric and asymmetric settings, and permits a general model characterized by the two parameters Δ and δ that account for practical constraints.
- *Integer Model, Symmetric Setting:* As mentioned earlier, when $\delta = \Delta/2$, a non-integer schedule becomes an integer schedule since no "trimming" is needed in that case. When $d_1 = d_2 = d$ and $\delta = \Delta/2$, the lower bound on discovery latency is Δ/d^2 (or the lower bound on the needed duty cycle is $1/\sqrt{n}$ when $\Delta = 1$), which recovers the lower bound for integer schedules in the symmetric setting [49].
- *Integer and Non-Integer Models, Asymmetric Setting:* Our lower bound is also applicable to the asymmetric setting, i.e., when $d_1 \neq d_2$. As a special case, when $d_1 \neq d_2$ and $\delta = \Delta/2$, the lower bound on discovery latency is $\Delta/(d_1 d_2)$, which is, to be best of our knowledge, the *first* known lower bound for integer schedules in the asymmetric setting.
- *Integer and Non-Integer Models, Lower Bounds, Symmetric and Asymmetric Settings:* The lower bound on the discovery latency is $\Delta/(2d_1 d_2)$ when $\delta \ll \Delta$, while it is $\Delta/(d_1 d_2)$ when $\delta = \Delta/2$ (i.e., under the integer model). This is because, when $\delta \ll \Delta$, using an integer schedule essentially overestimates δ by assuming that $\delta = \Delta/2$, which can lead to discovery latency no better than $\Delta/(d_1 d_2)$, twice as the lower bound of the discovery latency of the non-integer model (i.e., $\Delta/(2d_1 d_2)$). The above observations on lower bounds apply to both symmetric and asymmetric settings. They further confirm that when $\delta \ll \Delta$, integer schedules can lead to significant waste of resources, while non-integer schedules are much more efficient.
- *Integer and Non-Integer Models, Optimal Schedules, Symmetric Setting:* In the symmetric setting (i.e., $d_1 = d_2 = d$), as indicated in Table I, the integer variant of the Singer algorithm chooses $p \approx 1/d$, leading to a worst-case discovery latency of approximately $((1/d)^2 + 1/d + 1)\Delta$, which achieves the lower bound Δ/d^2 for integer schedules when $d \ll 1$; the non-integer variant chooses $p \approx 1/(2d)$, leading to a worst-case discovery latency of approximately $((1/2d)^2 + 1/(2d) + 1) \times 2\Delta$, which achieves the lower bound $\Delta/(2d^2)$ for non-integer schedules when $d \ll 1$. The above results indicate that the Singer algorithm provides an optimal integer schedule in the symmetric setting, and its corresponding non-integer variant provides an optimal non-integer schedule in the symmetric setting; a point that we will return to in Section VII-A.
- *Integer and Non-Integer Models, Optimal Schedules, Asymmetric Setting:* In the asymmetric setting, we are not aware of an optimal schedule (with discovery latency meeting the lower bound) in either the integer or non-integer model. See further discussion in Section VII-B.

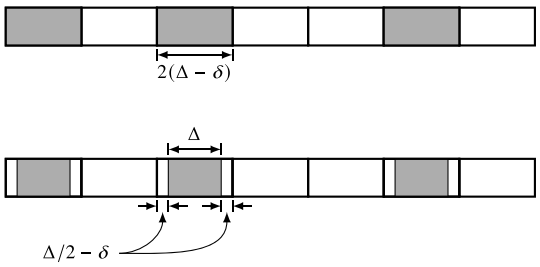


Fig. 11. Illustration of an optimal reduction that converts an integer schedule to a corresponding non-integer schedule.

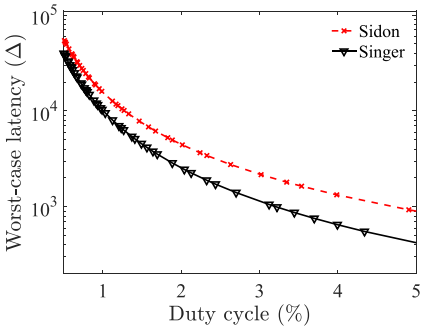


Fig. 12. Worst-case discovery latency of two optimal schemes (based on Singer and Sidon difference sets) for the symmetric setting.

VII. Optimal Schedules

In this section, we first demonstrate that the reduction in Section III-D converts an optimal integer schedule to an optimal non-integer schedule. We then discuss optimal schedules for symmetric and asymmetric settings, respectively. Our focus is on periodic schedules, as in most existing studies.

A. Optimal Integer and Non-Integer Schedules

Consider an optimal integer schedule for two nodes with duty cycles d_1 and d_2. When the slot length is 2Δ, based on the lower bound in Section VI, for the two nodes to discover each other in time n, we must have

$$d_1 d_2 \geq \frac{2\Delta}{n}. \tag{4}$$

Under the reduction in Section III-D, the duty cycle of a node in the non-integer schedule, denoted as d_i', satisfies

$$d_i' = d_i \frac{\Delta+\delta}{2\Delta}, i = 1, 2, \tag{5}$$

$$d_1' d_2' = d_1 d_2 \left(\frac{\Delta+\delta}{2\Delta}\right)^2. \tag{6}$$

Substituting (4) into (6) yields

$$d_1' d_2' \geq \frac{2\Delta}{n}\left(\frac{\Delta+\delta}{2\Delta}\right)^2 = \frac{(\Delta+\delta)^2}{2n\Delta} \approx \frac{\Delta^2}{2n(\Delta-\delta)},$$

where the last approximation holds since $\delta \ll \Delta$ (we only consider the non-integer model with $\delta \ll \Delta$, see remarks in Section VI). The new non-integer schedule therefore meets the lower bound for the non-integer model, and hence is an optimal non-integer schedule.

In fact, we can use a more exact reduction (see Fig. 11), which divides time into slots of length $2(\Delta-\delta)$ and "trims" a total amount of $\Delta - 2\delta$ in each active slot. This reduction still ensures discovery since the length of the awake time in an active slot is Δ, while the ensured overlap is at least $2(\Delta-\delta)/2 = \Delta-\delta$. Repeating the above derivation, we have

$$d_1\ d_2 \geq \frac{2(\Delta-\delta)}{n},$$

$$d_i' = d_i \frac{\Delta}{2(\Delta-\delta)}, \quad i = 1, 2,$$

$$d_1' d_2' \geq \frac{2(\Delta-\delta)}{n}\left(\frac{\Delta}{2(\Delta-\delta)}\right)^2 = \frac{\Delta^2}{2n(\Delta-\delta)}.$$

That is, the constructed non-integer schedule matches the lower bound in the non-integer model exactly, and hence is optimal.

B. Optimal Schedules

Our discussion below only considers optimal integer schedules, since using the reduction in Section III-D, they can be readily converted to optimal non-integer schedules.

1) Symmetric Setting: Two schemes have been shown to be optimal for the symmetric setting. The first scheme is based on Singer difference sets [22], [29], [49]; our earlier study [7] provides an efficient polynomial-time algorithm to construct such schedules based on the existence proof in [35]. The second scheme is based on *Sidon sets* (see, e.g., [40] for a detailed description of the theory); the construction of the scheme is detailed in [7]. Fig. 12 plots the worst-case discovery latency of these two schemes when the duty cycle of the two nodes is varied from 0.5% to 5%. We see that these two schemes have similar worst-case discovery latency for small duty cycles; for larger duty cycles, the latency under the Sidon set construction is larger than that under the Singer set construction due to the differences of the lower order terms in these two constructions [7]. Both the Singer set and Sidon set constructions are based on prime numbers, and hence only support a limited granularity of duty-cycles.

2) Asymmetric Setting: In the asymmetric case, for two nodes with duty cycles d_1 and d_2, $d_1 \neq d_2$, the lower bound of discovery latency in the integer model is $\Delta/(d_1\ d_2)$ (see Theorem 2). A simple optimal schedule is as follows. Each node picks the closest prime number, p_i, so that $p_i \leq 1/d_i$ and $p_1 \neq p_2$. Then a node wakes up in the slots that are the multiples of p_i. By the Chinese Remainder Theorem, since $p_1 \neq p_2$, for any time shift between these two nodes, they can meet in time $p_1\ p_2 \approx \Delta/(d_1\ d_2)$. This simple optimal schedule, however, only works when $p_1 \neq p_2$. Hence it does not work for the symmetric setting or when two nodes have similar duty cycles (which can lead to $p_1 = p_2$). In practice, we need a scheme that works for both asymmetric and symmetric settings since the nodes do not know whether they have the same or different duty cycles beforehand.

Existing schemes that are based on prime numbers overcome the limitation in the above simple scheme through adding mechanisms. Specifically, in Disco [11], each node uses two primes instead of one prime; in U-Connect [17], each

node uses one prime, but is awake for a continuous period of time in addition to the multiples of the prime. Neither of them is optimal. The asymmetric schedules in [2], [29] are not optimal either. Finding a schedule that is optimal in the asymmetric setting while also applicable (or even optimal) for the symmetric setting remains an open question that is left as future work.

VIII. Conclusion

In this paper, we first presented an intersection property of integer schedules and showed that integer schedules can lead to significant waste of resources. We then developed a generalized non-integer model, which relaxes the constraints in the integer model by assuming time to be continuous and nodes may become active or inactive at any point in time, subject to a few constraints. In addition, we provided a reduction that transforms any schedule in the integer model to a corresponding schedule in the non-integer model, while achieving significant performance gains. Using experiments in a testbed, we demonstrated the practicality of non-integer schedules and their advantages over the corresponding integer schedules. We further presented the worst-case discovery latency of the integer and non-integer variants for several existing schemes. After that, we established a new family of lower bounds for the best achievable latency guarantee that applies to both the integer and non-integer models, covering both symmetric and asymmetric settings. At the end, we discussed optimal neighbor discovery schedules for both the symmetric and asymmetric settings.

Acknowledgment

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] L. Aalto, N. Göthlin, J. Korhonen, and T. Ojala, "Bluetooth and WAP push based location-aware mobile advertising system," in *Proc. 2nd Int. Conf. Mobile Syst., Appl., Services (MobiSYS)*, 2004, pp. 49–58.

[2] M. Bakht, M. Trower, and R. H. Kravets, "Searchlight: Won't you be my neighbor?" in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, 2012, pp. 185–196.

[3] S. A. Borbash, A. Ephremides, and M. J. McGlynn, "An asynchronous neighbor discovery algorithm for wireless sensor networks," *Ad Hoc Netw.*, vol. 5, no. 7, pp. 998–1016, Sep. 2007.

[4] D. Burghal, A. S. Tehrani, and A. F. Molisch, "On expected neighbor discovery time with prior information: Modeling, bounds and optimization," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 339–351, Jan. 2018.

[5] L. Chen, R. Fan, K. Bian, M. Gerla, T. Wang, and X. Li, "On heterogeneous neighbor discovery in wireless sensor networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 693–701.

[6] L. Chen *et al.*, "Group-based neighbor discovery in Low-Duty-Cycle mobile sensor networks," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 1996–2009, Aug. 2016.

[7] S. Chen, A. Russell, R. Jin, Y. Qin, B. Wang, and S. Vasudevan, "Asynchronous neighbor discovery on duty-cycled mobile devices: Integer and non-integer schedules," in *Proc. 16th ACM Int. Symp. Mobile Hoc Netw. Comput.*, 2015, pp. 47–56.

[8] T. Chen, J. Ghaderi, D. Rubenstein, and G. Zussman, "Maximizing broadcast throughput under ultra-low-power constraints," in *Proc. CoNext*, 2016, pp. 1–15.

[9] R. Cohen and B. Kapchits, "Continuous neighbor discovery in asynchronous sensor networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 1, pp. 69–79, Feb. 2011.

[10] I. de Jong. *Pyro: Python Remote Objects*. Accessed: Apr. 2019. [Online]. Available: https://pythonhosted.org/Pyro

[11] P. Dutta and D. Culler, "Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, 2008, pp. 61–84.

[12] P. Gandotra and R. K. Jha, "Device-to-Device communication in cellular networks: A survey," *J. Netw. Comput. Appl.*, vol. 71, pp. 99–117, Aug. 2016.

[13] B. Han, P. Hui, V. S. A. Kumar, M. V. Marathe, J. Shao, and A. Srinivasan, "Mobile data offloading through opportunistic communications and social participation," *IEEE Trans. Mobile Comput.*, vol. 11, no. 5, pp. 821–834, May 2012.

[14] C. S. Hsu, J. R. Jiang, Y. C. Tseng, and T. H. Lai, "Quorum-based asynchronous power-saving protocols for IEEE 802.11 ad hoc networks," *Mobile Netw. Appl.*, vol. 10, nos. 1–2, pp. 169–181, 2005.

[15] J.-H. Huang, S. Amjad, and S. Mishra, "Cenwits: A sensor-based loosely coupled search and rescue system using witnesses," in *Proc. 3rd Int. Conf. Embedded Netw. Sensor Syst.*, 2005, pp. 180–191.

[16] C. Julien, C. Liu, A. L. Murphy, and G. P. Picco, "BLEnd: Practical continuous neighbor discovery for Bluetooth low energy," in *Proc. 16th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2017, pp. 105–116.

[17] A. Kandhalu, K. Lakshmanan, and R. R. Rajkumar, "U-connect: A low-latency energy-efficient asynchronous neighbor discovery protocol," in *Proc. 9th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, 2010, pp. 350–361.

[18] A. Keshavarzian, E. Uysal-Biyikoglu, F. Herrmann, and A. Manjeshwar, "Energy-efficient link assessment in wireless sensor networks," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 1751–1761.

[19] P. H. Kindt, M. Saur, and S. Chakraborty, "Slotless protocols for fast and energy-efficient neighbor discovery," 2016, *arXiv:1605.05614*. [Online]. Available: https://arxiv.org/abs/1605.05614

[20] P. H. Kindt, M. Saur, M. Balszun, and S. Chakraborty, "Neighbor discovery latency in BLE-like protocols," *IEEE Trans. Mobile Comput.*, vol. 17, no. 3, pp. 617–631, Mar. 2018.

[21] P. H. Kindt, D. Yunge, G. Reinerth, and S. Chakraborty, "Griassdi: Mutually assisted slotless neighbor discovery," in *Proc. 16th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2017, pp. 93–104.

[22] S. Lai, B. Zhang, B. Ravindran, and H. Cho, "CQS-pair: Cyclic quorum system pair for wakeup scheduling in wireless sensor networks," in *Principles of Distributed Systems*. Berlin, Germany: Springer, 2008, pp. 295–310.

[23] D. Li and P. Sinha, "RBTP: Low-power mobile discovery protocol through recursive binary time partitioning," *IEEE Trans. Mobile Comput.*, vol. 13, no. 2, pp. 263–273, Feb. 2014.

[24] T. Liu, C. M. Sadler, P. Zhang, and M. Martonosi, "Implementing software on resource-constrained mobile sensors: Experiences with Impala and ZebraNet," in *Proc. 2nd Int. Conf. Mobile Syst., Appl., Services*, 2004, pp. 256–269.

[25] *Locast*. Accessed: Apr. 2019. [Online]. Available: http://www.lokast.com

[26] R. Margolies, G. Grebla, T. Chen, D. Rubenstein, and G. Zussman, "Panda: Neighbor discovery on a power harvesting budget," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[27] M. J. McGlynn and S. A. Borbash, "Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks," in *Proc. 2nd ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, 2001, pp. 137–145.

[28] L. McNamara, C. Mascolo, and L. Capra, "Media sharing based on colocation prediction in urban transport," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2008, pp. 58–69.

[29] T. Meng, F. Wu, and G. Chen, "On designing neighbor discovery protocols: A code-based approach," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 1689–1697.

[30] T. Meng, F. Wu, and G. Chen, "Code-based neighbor discovery protocols in mobile wireless networks," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 806–819, Apr. 2016.

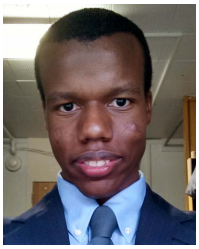
[31] M. Motani, V. Srinivasan, and P. S. Nuggehalli, "Peoplenet: Engineering a wireless virtual social network," in *Proc. 11th Annu. Int. Conf. Mobile Comput. Netw.*, 2005, pp. 243–257.

[32] A. K. Pietiläinen, E. Oliver, J. Lebrun, G. Varghese, and C. Diot, "MobiClique: Middleware for mobile social networking," in *Proc. 2nd ACM Workshop Online Social Netw.*, Aug. 2009, pp. 49–54.

[33] A. Purohit, N. Priyantha, and J. Liu, "WiFlock: Collaborative group discovery and maintenance in mobile sensor networks," in *Proc. IPSN*, 2011, pp. 37–48.

[34] Y. Qiu, S. Li, X. Xu, and Z. Li, "Talk more listen less: Energy-efficient neighbor discovery in wireless sensor networks," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.
[35] D. R. Stinson, *Combinatorial Designs: Constructions and Analysis*. New York, NY, USA: Springer-Verlag, 2003.
[36] *Nintendo 3ds's Streetpass*. Accessed: Apr. 2019. [Online]. Available: http://www.nintendo.com/3ds/features
[37] W. Sun, Z. Yang, K. Wang, and Y. Liu, "Hello: A generic flexible protocol for neighbor discovery," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 540–548.
[38] J. Tang, M. Dabaghchian, K. Zeng, and H. Wen, "Impact of mobility on physical layer security over wireless fading channels," *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 7849–7864, Dec. 2018.
[39] J. Tang, H. Wen, K. Zeng, R.-F. Liao, F. Pan, and L. Hu, "Light-weight physical layer enhanced security schemes for 5G wireless networks," *IEEE Netw.*, vol. 33, no. 5, pp. 126–133, Sep. 2019.
[40] T. Tao and V. Vu, "Additive combinatorics," in *Number 105 in Cambridge Studies in Advanced Mathematics*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[41] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks," in *Proc. IEEE INFOCOM*, Jun. 2002.
[42] S. Vasudevan, M. Adler, D. Goeckel, and D. Towsley, "Efficient algorithms for neighbor discovery in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 69–83, Feb. 2013.
[43] K. Wang, X. Mao, and Y. Liu, "BlindDate: A neighbor discovery protocol," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 949–959, Apr. 2015.
[44] F. Wu, T. Meng, A. Li, G. Chen, and N. H. Vaidya, "Have you recorded my voice: Toward robust neighbor discovery in mobile wireless networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1432–1445, Jun. 2018.
[45] Z. Yang, B. Zhang, J. Dai, A. C. Champion, D. Xuan, and D. Li, "E-smalltalker: A distributed mobile system for social networking in physical proximity," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst.*, Jun. 2010, pp. 468–477.
[46] D. Zhang *et al.*, "Acc: Generic on-demand accelerations for neighbor discovery in mobile applications," in *Proc. ACM SenSys*, 2012, pp. 169–182.
[47] D. Zhang, T. He, F. Ye, R. K. Ganti, and H. Lei, "EQS: Neighbor discovery and rendezvous maintenance with extended quorum system for mobile sensing applications," in *Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst.*, Jun. 2012, pp. 72–81.
[48] L. Zhang and D. Guo, "Neighbor discovery in wireless networks using compressed sensing with Reed–Müller codes," in *Proc. Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw.*, May 2011, pp. 154–160.
[49] R. Zheng, J. C. Hou, and L. Sha, "Asynchronous wakeup for ad hoc networks," in *Proc. 4th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, 2003, pp. 35–45.

Sixia Chen (Member, IEEE) received the master's and Ph.D. degrees from the Computer Science and Engineering Department, University of Connecticut. She is currently an Assistant Professor with the Department of Computer Science, Central Connecticut State University. Her research interests include design and analysis of algorithms, combinatorial optimization, and pseudorandomness.



Reynaldo Morillo (Student Member, IEEE) received the B.S. degree in computer science and engineering from the University of Connecticut, where he is currently pursuing the Ph.D. degree with the Computer Science and Engineering Department. His interests are in networks and systems, network security, and ubiquitous computing.

Yanyuan Qin (Student Member, IEEE) received the B.S. degree in automation from the Nanjing University of Aeronautics and Astronautics, China, in 2011, and the M.S. degree in control science and engineering from Shanghai Jiao Tong University, China, in 2014. He is currently pursuing the Ph.D. degree with the Computer Science and Engineering Department, University of Connecticut. His research interests are in software defined networking and wireless networks.

Alexander Russell was born in Philadelphia, PA, USA, in 1969. He received the B.A. degree in mathematics and computer science from Cornell University in 1991, the S.M. degree in computer science from the Massachusetts Institute of Technology in 1993, and the Ph.D. degree in mathematics from the Massachusetts Institute of Technology in 1996. He is currently a Professor of computer science and engineering with the University of Connecticut.

Ruofan Jin (Member, IEEE) received the B.S. and M.S. degrees in computer science and engineering from Beihang University in 2007 and 2010, respectively, and the Ph.D. degree from the University of Connecticut in 2015. His research interests are in the areas of wireless networks and performance optimization.

Bing Wang (Senior Member, IEEE) received the B.S. degree in computer science from the Nanjing University of Science and Technology, China, in 1994, the M.S. degree in computer engineering from the Institute of Computing Technology, Chinese Academy of Sciences, in 1997, and the M.S. degrees in computer science and applied mathematics and the Ph.D. degree in computer science from the University of Massachusetts, Amherst, in 2000, 2004, and 2005, respectively. She is currently a Professor with the Computer Science and Engineering Department, University of Connecticut. Her research interests are computer networking and distributed systems. She received an NSF CAREER Award in February 2008.

Sudarshan Vasudevan (Member, IEEE) received the B.E. degree in computer science and engineering from the National Institute of Technology, Trichy, India, in 2000, and the M.S. and Ph.D. degrees in computer science from the University of Massachusetts, Amherst, in 2003 and 2006, respectively. He was a Researcher with Bell Labs, Murray Hill, NJ, USA. He is currently a Staff Software Engineer with LinkedIn (part of Microsoft, Inc.) His current research interests include building large-scale distributed systems, randomized algorithms, and performance evaluation.