Asynchronous Neighbor Discovery on Duty-cycled Mobile Devices: Integer and Non-Integer Schedules

Sixia Chen
Central Connecticut State University
schen@ccsu.edu

Alexander Russell
University of Connecticut
acr@cse.uconn.edu

Ruofan Jin
University of Connecticut
ruofan@uconn.edu

Yanyuan Qin
University of Connecticut
yanyuan.qin@uconn.edu

Bing Wang
University of Connecticut
bing@engr.uconn.edu

Sudarshan Vasudevan
Palo Alto Networks
svasudevan@paloaltonetworks.com

ABSTRACT

Neighbor discovery is a fundamental problem in wireless networks. In this paper, we study asynchronous neighbor discovery between duty-cycled mobile devices. Each node is duty-cycled, i.e., its radio may only be active for a small fraction of the time. The duty cycles of the nodes can be the same or different, leading to symmetric or asymmetric cases of the neighbor discovery problem. In addition, the setting is asynchronous, i.e., clocks of different nodes may not be synchronized. Most existing studies assume an integer model (where time proceeds in discrete steps); two recent studies break away from this assumption, which allows them to develop significantly more efficient schemes. Our study improves the state-of-the-art in three main fronts. Firstly, we develop a generalized non-integer model (where time is continuous) that permits unified treatment of the assumptions in existing studies. We also provide a reduction that transforms any schedule in the basic integer model to a corresponding schedule in the generalized non-integer model while improving the performance by a factor of two. Applying this reduction, an optimal schedule in the integer model becomes an optimal schedule in the non-integer model. Thirdly, we establish a new family of lower bounds for the best achievable latency guarantee in the non-integer model. They are applicable to both symmetric and asymmetric settings, and encompass the lower bounds for the integer model as special cases. Finally, we develop a novel optimal construction based on Sidon sets for the symmetric setting. Our approach differs from the approaches taken by all existing studies, and provides a new direction for constructing neighbor discovery schedules.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communication*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Performance

MobiHoc'15, June 22–25, 2015, Hangzhou, China.

http://dx.doi.org/10.1145/2746285.2746297.

General Terms

Algorithm, Design, Performance, Theory

Keywords

Neighbor discovery, deterministic algorithms, Sidon set, difference set

1. 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 [14, 22, 18, 19], proximity-based services [1, 26], media sharing [16], traffic uploading [6], asset management [20], emergency rescue [8], tracking encountering pattern of migrating animals [13], and many more. These applications require mobile devices to rendezvous as they move into each other's transmission range. On the other hand, limited battery sources often mandate these devices to be duty-cycled, making them unable to discover other nodes while being asleep. 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 act depending on individual "identities" (since the identities are not known before rendezvous).

We broadly refer to the above problem as *asynchronous* neighbor discovery on *duty-cycled* mobile devices. The duty cycles of the nodes can be the same or different, resulting in *symmetric* or *asymmetric* neighbor discovery, respectively. In such settings, it is possible to construct simple and powerful randomized algorithms (e.g., [15]); however, these schemes do not provide a bound on worst-case discovery latency and—rather remarkably—deterministic schemes can achieve better performance, even when compared with the "average case" performance of randomized algorithms. In this paper, we focus on *deterministic* algorithms, where the schedules of the nodes are determined beforehand and the worst-case discovery latency is bounded.

Most existing studies [5, 9, 11, 24, 7] develop *integer* schedules where time proceeds in discrete slots of length Δ, and a node is either awake for an entire slot or asleep for an entire slot. Two recent studies [3, 17] 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 improve the state-of-the-art on three main fronts: (i) we develop a generalized *non-integer* model that permits unified treatment of existing studies, and demonstrate that *any* integer schedule can be converted into a non-integer schedule while improving the performance by a factor of two, (ii) we derive a family of lower bounds in the generalized non-integer model, and (iii) we develop a novel optimal construction for the symmetric setting.

Our main contributions are as follows:

- We develop a generalized non-integer model motivated by the observation that the integer model needlessly constrains switching to occur 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 switching latency Δ, we also 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. This generalized non-integer model permits unified treatment of the assumptions in existing studies: the basic integer model used in most existing studies arises as when $\delta = \Delta/2$; the model of [3, 17] arises when $\delta \ll \Delta$.

- We provide a general reduction that can transform *any* schedule in the basic integer model to a corresponding schedule in the generalized non-integer model. The reduction reduces the duty cycles of the nodes by a factor of two while preserving the discovery latency. Applying this reduction, an optimal schedule in the integer model becomes an optimal schedule in the non-integer model.

- We derive a family of lower bounds on the best achievable latency guarantees in the non-integer model (they reduce to lower bounds for the integer model when $\delta = \Delta/2$). 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 [30] and [17] are only for the symmetric setting; they are special cases of our lower bounds by taking $\Delta = 1$, $\delta = \Delta/2$ ([30]) and $\delta \ll \Delta$ ([17]).

- We propose a novel optimal construction based on the theory of *Sidon sets* for the symmetric setting. All existing optimal schemes [30, 11, 17] are based on Singer difference sets. Our construction uses a different approach, and provides a new direction for constructing neighbor discovery schedules.

Our optimal scheme based on Sidon sets is directly applicable in practice (in fact, we have implemented it on smartphones). The non-integer schedules are particularly important for mobile devices with $\delta \ll \Delta$ (e.g., current smartphones where Δ is one or two seconds while δ is in milliseconds) since they require significantly lower duty cycle for the same discovery latency (or lead to much faster discovery under the same duty cycle) compared to integer schedules. Using our reduction, all integer schedules in the literature can be directly converted into their corresponding non-integer schedules.

The rest of the paper is organized as follows. Section 2 briefly describes related work. Section 3 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 4 presents a family of lower bounds for the non-integer model. Section 5 presents the Sidon sets construction. Finally, Section 6 concludes the paper and describes future work.

2. RELATED WORK

Neighbor discovery has been studied extensively in the literature (e.g., [15, 4, 10, 25]). In the following, we only briefly review the studies that are closest to ours, specifically, those that develop deterministic schedules for asynchronous neighbor discovery on duty-cycled mobile devices.

Most existing studies develop integer schedules. Zheng et al. [30] derive a lower bound for the symmetric setting, and show the existence of optimal schedules through *Singer* difference sets [21]. Lai et al. [11] expand the idea of cyclic quorum systems [24, 7], and propose a scheme, CQS-pair, that is also based on Singer difference sets. The construction for CQS-pair, however, takes exponential time, and hence is only practical for very large duty cycles. The general lower bound derived in our study encompasses the lower bound in [30] as a special case. We also provide a polynomial-time construction for Singer difference set based on the existence proof in [21]. In addition, we propose a novel optimal scheme based on Sidon sets for the symmetric setting; it is the first optimal scheme that does not rely on Singer difference sets.

Two other existing integer schedules are Disco [5] and U-connect [9]. In Disco, for duty cycle d, a node selects two different primes p_1, p_2 so that $1/p_1 + 1/p_2 \approx d$ and wakes up in the slots that are multiples of p_1 or p_2. Suppose node i has duty cycle d_i and chooses two primes, $p_{i1} < p_{i2}$, such that $1/p_{i1} + 1/p_{i2} \approx d_i$, $i = 1, 2$. Then 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 $1/(p_1 p_2) \approx 2.25/(d_1 d_2)$. Both Disco and U-Connect provide integer schedules that are applicable to both symmetric and asymmetric settings. We propose a general reduction that can convert these integer schedules to their corresponding non-integer schedules while reducing the worst-case discovery latency by a factor of two.

The first study that breaks away from integer schedules is [3]. Specifically, Bakht et al. design a simple probing based approach, Searchlight, by leveraging the constant offset between periodic awake slots. The authors propose both integer and non-integer variants 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 [3] 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 3.4).

A recent study [17] adopts the same non-integer assumption in [3]. The authors derive a new lower bound for the symmetric setting as well as 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). The general lower bound that we derive is applicable to both symmetric and asymmetric settings,

encompassing the lower in bound [17] as a special case. In addition, 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 (the optimal schedules that are based on Singer difference sets) is a special instance. In addition, we also develop a novel optimal scheme based on Sidon sets; the first optimal scheme that does not use Singer difference sets.

As other related work, the studies in [28, 27] propose schemes that are based upon an existing asynchronous neighbor discovery scheme to further reduce energy consumption. Two recent studies [29, 12] propose neighbor discovery schemes on mobile duty-cycled devices. Both of them however require clock synchronization, and hence are not applicable to asynchronous settings.

3. 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*. We then establish a general reduction that transforms any schedule in the integer model to a corresponding schedule in the non-integer model; the reduction *reduces duty cycle by a factor of two while preserving the discovery latency.*

3.1 Integer Model

In this model, time is discretized into slots of width Δ, a hardware-dependent parameter that we refer to as the *switching interval*. Specifically, Δ is the time required by nodes to change their states (from asleep to awake and vice versa). On certain mobile devices, Δ may be on the order of milliseconds [5, 9] or even a fraction of a millisecond [20], while it can be over a second for other mobile devices (e.g., Δ is one or two seconds on current smartphones due to limitation of the hardware [3]).

A schedule may then be described by calling for a node to be *awake* during some of these slots and *asleep* during the others. Indexing the slots with the natural numbers $0, 1, 2, \ldots$, a schedule is simply determined by a subset S of $\mathbb{N}$; such a schedule we call an *integer schedule*. We shall typically constrain nodes to adopt schedules with bounded *duty cycles*: they must be awake for no more than a bounded fraction of the time. Formally, we say that an integer schedule $S \subset \{0, 1, \ldots\}$ has duty cycle d if there is an infinite sequence of times $\tau_1 < \tau_2 < \cdots$ so that

$$\frac{|S \cap \{0, 1, \ldots, \tau_i - 1\}|}{\tau_i} \leq d\,.$$

Consider two nodes, A_1 and A_2, each with a fixed *duty cycle*, denoted $d_i < 1$. We may determine the behavior of these nodes by associating with each of them an integer schedule $S_i \subset \{0, 1, 2, \ldots\}$ meeting the duty cycle constraint; then the node A_i is awake only during those slots indexed by the schedule S_i. In general, we do not assume that the clocks of the two nodes are synchronized and so must entertain the possibility of arbitrary (even non-integer) shifts of the nodes' schedules. For this reason, it is convenient to treat the schedules S_i as subsets of the real numbers $\mathbb{R}$ in the obvious way: associate the integer s with the interval $[s\Delta, (s+1)\Delta)$. (We use the notation $[a, b)$ to denote the "half open" interval $\{x \mid a \leq x < b\}$.) Then, for the integer schedule S we define

$$S^{\Delta} = \bigcup_{s \in S} \big[s\Delta, (s+1)\Delta\big) \subset \mathbb{R}.$$

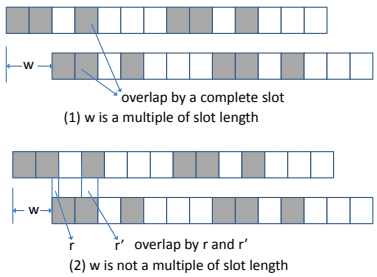

Figure 1: Illustration of the intersection property in the basic integer model.

Consider the circumstances where the nodes A_1 and A_2 come into each other's range at the moment when A_1's clock reads t_1 and A_2's clock reads t_2. The time to discovery for these particular shifts is then given by the smallest t for which both nodes are awake: $t_1 + t \in S_1$ and $t_2 + t \in S_2$. We write this quantity as

$$L(S_1, S_2; t_1, t_2) = \min\{t \mid t_1 + t \in S_1 \text{ and } t_2 + t \in S_2\}\,.$$

For two schedules S_1 and S_2, then, this motivates the following definition of *discovery latency*, the worst case time to discovery taken over all shifts:

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

We remark that—in general—this quantity may not even be well-defined (that is, there may be shifts for which the schedules never overlap). The basic problem is to construct schedules S_i, satisfying the duty-cycle constraints, that minimize the discovery latency $L(S_1, S_2)$.

Periodic schedules. One natural way to define an integer schedule is to fix a finite-length schedule and periodically repeat this sequence. More specifically, fixing a subset $\check{S} \subset \mathbb{Z}_n$ of the integers modulo n, one can define an integer schedule $S = \{z \mid z \bmod n \in \check{S}\} \subset \mathbb{Z}$. One advantage of schedules S defined in this way is that their duty cycle can be determined as the simple ratio $|\check{S}|/n$. As a matter of notation, we blur the distinction between $\check{S}$ and S, referring to the finite set $\check{S}$ as a schedule.

3.2 Motivation for Non-Integer Model

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

PROPERTY 1. *Consider two nodes using a schedule in the basic integer model for neighbor discovery. Suppose the schedule guarantees discovery in time n. Let w denote the relative time shift of the two nodes. Then, for any w, we find one of two possible cases:*

- *The shift w is an integer, and there exists an entire time slot during which both nodes are awake.*
- *The shift w is non-integral, 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 either r or r' is longer than $\Delta/2$.*

This property is illustrated in Fig. 1. The first case in the above property is straightforward. We therefore only describe why the second property holds. For ease of exposition, we

assume $\Delta = 1$ (when $\Delta \neq 1$, we just need to divide time by Δ and the explanation below applies similarly). 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, denoted as S_1 and S_2, have (at least) two overlaps, one of which has length $w-z$ and the other has length $z+1-w$. 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 part 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) that S_1 and S_2 overlapped. Since the real case is to shift S_2 to the left by length $z+1-w$, the overlap is actually part of s_2 that is with 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 prompts us to define a further hardware-dependent parameter, the *required meeting time* $\delta > 0$: This is the minimum amount of time required by two devices to discover each other when they are both active. The basic integer model implicitly assumes that $\delta = \Delta/2$. While this assumption is reasonable for some wireless devices that have compatible δ and Δ (e.g., both in a few milliseconds or less than one millisecond [5, 9, 20]), it can be quite pessimistic for other wireless devices where $\delta \ll \Delta/2$ (for instance, Δ is one or two seconds while δ is in milliseconds on current smartphones [3]). We therefore 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. When $\delta = \Delta/2$, it reduces to the basic integer model. In the rest of the paper, we assume $\delta \leq \Delta/2$.

3.3 Non-Integer Model

Fix Δ and δ as above. As we have relaxed the assumption that schedules adhere to integer boundaries, we may consider arbitrary schedules of the form $S \subset \mathbb{R}$; we assume, however, that each schedule S consists of a set of disjoint intervals, $S = \bigcup_\ell [a_\ell, b_\ell)$, and that each $b_\ell - a_\ell \geq \Delta$ so that the schedule satisfies the Δ switching constraint. We refer to such schedules as *non-integer* schedules. The quantity δ then determines the discovery time as follows. Consider a pair of schedules S_i and a pair of shifts $t_1, t_2 \geq 0$ $(i = 1, 2)$. For these two shifts, we define the time to discovery to be the smallest time at which the two schedules overlap at an interval of width at least δ:

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

As above, we define the *discovery time* for such a pair of schedules to be the worst case over all shifts:

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

As above we may consider periodic schedules. Specifically, if the schedule S repeats with period $n \in \mathbb{R}$ it can be represented as a subset $\check{S} \subset [0, n)$. We say that such a periodic schedule has duty cycle d if $|\check{S}|/n \leq d$, where $|\check{S}| = \sum_\ell (b_\ell - a_\ell)$. As above, we notationally blur the distiction between S and the "finite length" schedule $\check{S}$, which describes one "period" of S. Again the goal is, given duty cycles d_i, to design periodic schedules $S_i \subset [0, n_i)$ (so that S_i has period n_i), that minimize the resulting discovery latency.

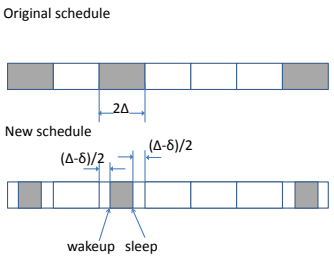

Figure 2: Illustration of the reduction that converts an integer schedule to a corresponding non-integer schedule.

3.4 General Reduction from Integer Schedules to Non-integer Schedules

We propose a general reduction that can be applied to any integer schedule to obtain a corresponding (lower duty-cycle) non-integer schedule. 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 amount of overlap while both nodes are awake, which is sufficient for neighbor discovery. The above construction significantly reduces the duty cycle: in the new schedule, the duty cycle is $d_i(\Delta/2+\delta)/\Delta$, approximately half of d_i when $\delta \ll \Delta$, where d_i is the original duty cycle of node i, $i = 1, 2$. The new schedule, however, violates the switching interval constraint (since the amount of awake time, $\Delta/2+\delta$, in a slot is less than Δ). This violation can be easily fixed by assuming 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 Δ to $\Delta/2+\delta$. Fig. 2 illustrates the "trimming" procedure. In the figure, 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. Besides the example shown in Fig. 2, there are many other ways of "trimming", e.g., removing $(\Delta-\delta)$ at the beginning or end of an active slot.

The above reduction converts an integer schedule to a non-integer schedule while preserving the discovery latency and reducing the duty cycle by (approximately) a factor of two. We next describe how to obtain a non-integer schedule from an integer schedule so that the duty cycle remains to be the same while the discovery latency is significantly reduced. Suppose the duty cycle is d. The new non-integer schedule is constructed as follows. We first construct an integer schedule with duty cycle of $d(2\Delta)/(\Delta+\delta)$ and slot length of 2Δ. We then apply the above "trimming" procedure (see Fig. 2) so that the duty cycle is reduced to d, as desired. For two nodes with duty cycles d_1 and d_2, if the worst-case discovery time of the original integer schedule is $c/(d_1 d_2) \cdot \Delta$, where c is a constant (many existing schedules have this property; and as we shall see, optimal schedules also have this property), then the discovery latency of the new non-integer schedule

is $c/(4d_1d_2) \cdot 2\Delta = c/(2d_1d_2) \cdot \Delta$, only half of the original value.

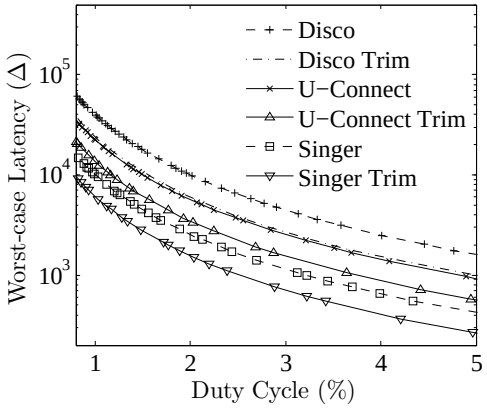

Figure 3: Performance of integer schedules and their corresponding non-integer schedules (symmetric case).

The above reduction can be applied to *any* integer schedule. We next illustrate it using three existing schemes, Disco [5] U-Connect [9], and schemes using Singer difference sets [30, 11], that provide integer schedules. Using the above reduction, we convert the integer schedules to their corresponding non-integer schedules. Fig. 3 plots the worst-case discovery latency versus duty cycle for these three schemes in the symmetric case. The results of both the original integer schedules and the "trimmed" non-integer schedules are plotted in the figure. We observe that for all three schemes, the worst-case discovery latency of the non-integer schedule is only half of that of the integer schedule. When "trimming" Singer sets based integer schedules using our reduction, the performance of the resultant non-integer schedules coincides with that of Diff-Codes [17] (for the schedules of duty cycles that are constructed using Singer sets). This is because Diff-Codes expands each slot in the schedule from Singer sets to two consecutive slots, an active slot followed by a sleeping slot, and a sleeping slot followed by another sleeping 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).

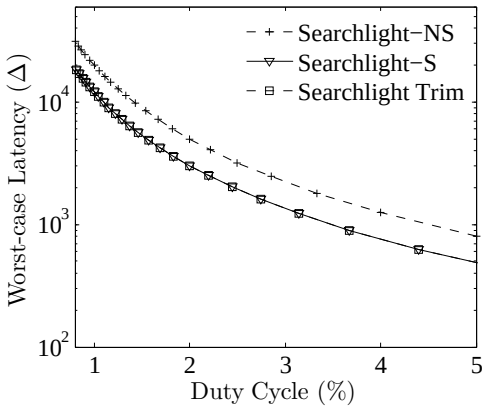

Figure 4: Performance of the integer and non-integer schedules for Searchlight (symmetric case).

The study in [3] proposes both integer and non-integer variants of Searchlight, referred to as Searchlight-NS (no striped probing) and Searchlight-S (with striped probing), respectively. We can convert Searchlight-NS to a non-integer schedule, referred to as "Searchlight trim," following our proposed reduction. Fig. 4 plots the worst-case discovery latency versus duty cycle for these three schemes for symmetric duty cycles. We observe the performance of "Searchlight trim" overlaps with that of Searchlight-S, both reducing the worst-case discovery latency of Searchlight-NS by a factor of two.

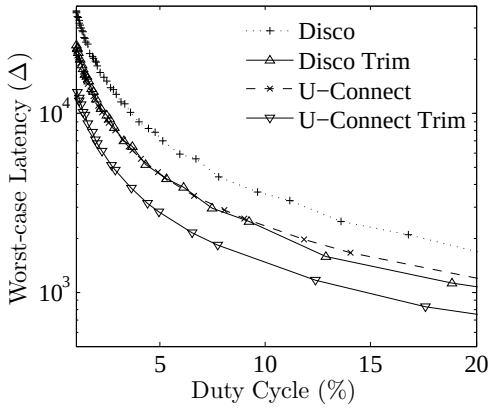

Figure 5: Performance of integer schedules and their corresponding non-integer schedules (asymmetric case).

The above examples illustrate applying the reduction in the symmetric case. The reduction also works for the asymmetric case. One example is shown in Fig. 5, where we plot the performance of Disco [5] and U-Connect [9], and their corresponding non-integer schedules under asymmetric duty cycles. The duty cycle of one node is fixed to be 1% while the duty cycle of the other node varies from 1% to 20%. We again observe the non-integer schedule reduces the worst-case discovery latency of the integer schedule by a factor of two.

4. LOWER BOUNDS

We next develop a general lower bound in the non-integer model. This lower bound is characterized by both Δ (switching time) and δ (required meeting time). It is for the general case where two nodes have duty cycles, d_1 and d_2, which can be the same or different. As we remark at the end, this lower bound leads to a family of lower bounds for both symmetric and asymmetric settings.

THEOREM 2. *Consider two schedules S_1, S_2 in the non-integer model with parameters Δ, δ. Assume that S_1, S_2 have duty cycles d_1, d_2 and, furthermore, $L(S_1, S_2) \leq n$, i.e., they guarantee discovery within time n. Then $d_1 d_2 \geq \Delta^2/(2n(\Delta - \delta))$. Conversely, the discovery latency is at least $\Delta^2/(2d_1d_2(\Delta - \delta))$.*

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 schedules achieve discovery latency n, and proceed to prove that we must then have

$$n \geq \frac{\Delta^2}{2d_1d_2(\Delta - \delta)}\,. \tag{1}$$

Select two large times $T_1, T_2 \gg n$ for which $T_j + n \notin S_j$ (for each $j = 1, 2$) and

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

(Recall that if $A = \bigcup_i [\alpha_i, \beta_i)$ is a disjoint union of half-open intervals, we define $|A| = \sum_i (\beta_i - \alpha_i)$.) 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 $B_i^{(j)}$ has the form $[\alpha, \beta)$. Let $\ell_i^{(j)} = |B_i^{(j)}|$, the length of the ith interval appearing in S_j; then, for each node j,

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

We pause briefly to introduce some further notation. 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 $S - r$). We also 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 $|A| = \sum_i |A_i|$.

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. Fixing the position of a particular 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 $\ell_i^{(1)} + \ell_{i'}^{(2)} - 2\delta$ (see Fig. 6). It follows that

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

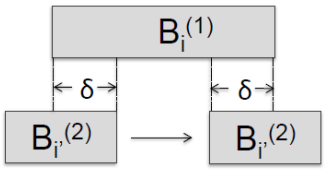

Figure 6: Overlap occurrences from a pair of intervals.

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

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

We conclude that

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

Additionally, we recall that any interval of activity must have width Δ, and hence

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

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

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

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

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

(Recall that $2\delta/\Delta \le 1$). 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. We conclude that

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

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

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

As we are free to select T_j as large as we wish, the inequality (1) follows, as desired. □

A few remarks area in order.

- When $\delta = \Delta/2$, the lower bound becomes the lower bound for the basic integer model. We see when $\delta \ll \Delta$ (the assumptions in [3, 17]), the lower bound is a factor 2 less than the lower bound for the integer model, showing the significant benefits of non-integer schedules. In the rest of the paper, non-integer model assumes $\delta \ll \Delta$.

- When $d_1 = d_2$, the lower bound on discovery latency becomes $\Delta/\sqrt{2n(\Delta - \delta)}$. When $\Delta = 1$ and $\delta \ll \Delta$, it reduces to the lower bound of $1/\sqrt{2n}$ in a recent study [17]. The derivation of [17] only considers symmetric settings. Our lower bound applies to both symmetric and asymmetric settings, and permits a quite general model, characterized by the two parameters Δ and δ that account for practical constraints.

- When $d_1 = d_2$ and $\delta = \Delta/2$, the lower bound on discovery latency is Δ/d^2, which coincides with the lower bound for integer schedules in the symmetric case [30]. When $d_1 \ne d_2$ 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 lower bound for integer schedules in the asymmetric case.

5. OPTIMAL SCHEDULES

In this section, we first demonstrate that the reduction in Section 3.4 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.

5.1 Optimal Integer and Non-integer Schedules

Consider an optimal integer schedule. Suppose two nodes have cycles d_1 and d_2. When the slot length is 2Δ, based on the lower bound in Section 4, 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 3.4, 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 4). 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. 7), which divides time into slots of length $2(\Delta-\delta)$ and "trim" 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

$$\begin{aligned} d_1 d_2 &\geq \frac{2(\Delta-\delta)}{n}\,, \\ d'_i &= d_i \frac{\Delta}{2(\Delta-\delta)}, i = 1, 2\,, \\ d'_1 d'_2 &\geq \frac{2(\Delta-\delta)}{n}\left(\frac{\Delta}{2(\Delta-\delta)}\right)^2 = \frac{\Delta^2}{2n(\Delta-\delta)}\,. \end{aligned}$$

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

5.2 Symmetric Case

Our discussion below only considers optimal integer schedules; using the reduction in Section 3.4, they can be readily converted to optimal non-integer schedules.

While existing studies [30, 11, 17] show optimal schedules can be achieved through Singer difference sets, the only explicit construction [11] in the networking community that we are aware of requires exponential running time. In the appendix, we describe an efficient polynomial-time construction based on the existence proof in [21]. In the following, we present a novel optimal construction that is based on the theory of *Sidon sets* (see, e.g., [23] for a detailed description of the theory). It is the first polynomial-time optimal algorithm that does not rely on Singer difference sets.

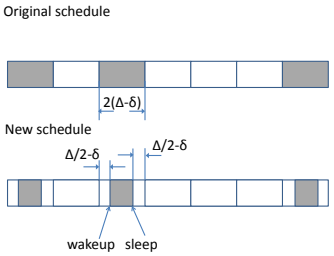

Figure 7: Illustration of an optimal reduction that converts an integer schedule to a corresponding non-integer schedule.

In the following, for ease of exposition, we regard the step length, Δ, as one unit of time and approach the dual problem: fixing a period n, how can one schedule a node's wake-up slots so that the schedule intersects with every rotation of the schedule and, furthermore, minimizes the duty cycle, d (determined by the number of active slots)? A solution to the dual problem can be immediately converted to a schedule for the original problem, i.e., for a given duty cycle, scheduling the wake-up times so that the discovery latency is minimized. Let S represent the schedule. Let $L(S) = L(S, S)$ represent the worst-case discovery latency.

Let $\mathbb{Z}_n$ denote the residue classes of the integers modulo n. We wish to construct a set $S \subseteq \mathbb{Z}_n$ of size $|S| \approx \sqrt{n}$ such that for any $\alpha, \beta \in \mathbb{Z}_n$,

$$|(\alpha + S) \cap (\beta + S)| \neq 0\,,$$

where $\alpha + S$ denotes the set $\{\alpha + s \mid s \in S\}$. Note that using this set S to define a periodic schedule immediately results in $L(S) \leq n$ (and $d = |S|/n$). To simplify our requirements on the set S, we observe the following:

LEMMA 3. *$|(\alpha+S)\cap(\beta+S)| \neq 0$ for any $\alpha, \beta \in \mathbb{Z}_n$ if and only if $\mathbb{Z}_n \subseteq S - S$, where $S - S$ denotes the set of differences $\{s_1 - s_2 \mid s_1, s_2 \in S\}$.*

PROOF. It is easy to see the statement that $|(\alpha+S)\cap(\beta+S)| \neq 0$ for any $\alpha, \beta \in \mathbb{Z}_n$ is equivalent with $|(\alpha+S)\cap S| \neq 0$ for any $\alpha \in \mathbb{Z}_n$. If $|(\alpha+S)\cap S| \neq 0$ for any $\alpha \in \mathbb{Z}_n$, then there exist two elements $s_1, s_2 \in S$ such that $\alpha + s_1 = s_2$; thus $\alpha = s_2 - s_1 \in S$, which proves one direction of the lemma. For the other direction, as $\mathbb{Z}_n \subseteq S - S$ for any $\alpha \in \mathbb{Z}_n$, we can find two elements $s_1, s_2 \in S$ such that $\alpha = s_2 - s_1$. Therefore $\alpha + s_1 = s_2 \in (\alpha + S) \cap S$, as desired. □

This allows us to focus our efforts on construction of *saturated difference sets* in $\mathbb{Z}_n$. Specifically, we say that a subset $S \subset \mathbb{Z}_n$ is a saturated difference set if $S - S \triangleq \{s_1 - s_2 \mid s_1, s_2 \in S\}$ contains every element of $\mathbb{Z}_n$.

The Sidon set construction. Consider a prime number p and let ϑ be a generator of the *multiplicative* group $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$ of units modulo p. Then the map $t \mapsto \vartheta^t$ is an isomorphism between $\mathbb{Z}_{p-1}$, the integers modulo $p-1$ under addition, and $\mathbb{Z}_p^*$, the multiplicative group of units modulo p. Consider the set

$$S_0 = \{(t, \vartheta^t) \mid t \in \mathbb{Z}_{p-1}\} \subset \mathbb{Z}_{p-1} \oplus \mathbb{Z}_p\,.$$

While the set S_0 is defined using the multiplicative structure of the ring of integers modulo p, our goal is to establish additive properties of the set (as a subset of the additive group $\mathbb{Z}_{p-1} \oplus \mathbb{Z}_p$).

LEMMA 4. $\{(a,b) \mid a,b \neq 0\} \cup \{(0,0)\} \subset S_0 - S_0$.

PROOF. It is easy to see that $(0,0) \in S_0 - S_0$. For a pair $(a,b) \in \mathbb{Z}_{p-1} \times \mathbb{Z}_p$ for which $a,b \neq 0$, consider the differences

$$(a+\ell, \vartheta^{a+\ell}) - (\ell, \vartheta^{\ell}) = (a, \vartheta^{\ell}(\vartheta^a - 1))$$

for $\ell \in \mathbb{Z}_{p-1}$. As $a \neq 0$, the sum $\vartheta^a - 1 \neq 0$ and it follows that $b \in \{\vartheta^{\ell}(\vartheta^a - 1) \mid \ell \in \mathbb{Z}_{p-1}\} = \mathbb{Z}_p^*$ as ϑ is a multiplicative generator. In particular, by choosing ℓ to be $\log_{\vartheta}(b/(\vartheta^a - 1))$ we find that $(a,b) = (a+\ell, \vartheta^{a+\ell}) - (a, \vartheta^a)$, as desired. (Here the notation $\log_{\vartheta} x$ denotes the unique exponent in the set $\{1, \ldots, p-1\}$ for which $\vartheta^{\alpha} = x$.) $\square$

Observe that since p and $p-1$ are relatively prime, $\mathbb{Z}_{p-1} \oplus \mathbb{Z}_p \cong \mathbb{Z}_{p(p-1)}$ by the Chinese remainder theorem, and a saturated difference set in $\mathbb{Z}_{p-1} \oplus \mathbb{Z}_p$ immediately yields a saturated difference set in the cyclic group $\mathbb{Z}_{p(p-1)}$. However, the set S_0 above is *not* saturated: differences of elements of the set miss a few evasive "slices" of $\mathbb{Z}_{p-1} \oplus \mathbb{Z}_p \cong \mathbb{Z}_{p(p-1)}$: the elements (a,b) where $a = 0$ or $b = 0$.

To complete the construction, we can apply an existing scheme in these small "slices." We recall here the basic details of U-Connect [9], which we adapt for this purpose. For a given $k > 0$, consider the set $A_k = \{k, 2k, \ldots, k^2\} \cup \{0, 1, 2, \ldots, k-1\} \subset \mathbb{Z}$. Observe that $\{-k^2, \ldots, k^2\} \subset A_k - A_k$ and hence that $A_k - A_k$ contains an element from every equivalence class of the integers modulo $2k^2 + 1$. To be precise, defining $A_{k,n} = \{a \bmod n \mid a \in A_k\}$ it follows immediately that $\mathbb{Z}_n \subset A_{k,n} - A_{k,n}$ so long as $n \leq 2k^2 + 1$.

To summarize, for the integer n, define $k_n = \lceil \sqrt{(n-1)/2} \rceil$ and $D_n = A_{k_n,n}$. By the discussion above, D_n is a saturated difference set in $\mathbb{Z}_n$ of size $2k_n$. As a function of n, this yields a saturated difference set of size $2\lceil \sqrt{(n-1)/2} \rceil \leq \sqrt{2n} + 2 = \sqrt{2n} + O(1)$.

Returning to our construction, define $S_a = \{(0,b) \mid b \in D_p\}$ and $S_b = \{(a,0) \mid a \in D_{p-1}\}$ (where D_n is as defined above), and observe now that the set $S \triangleq S_0 \cup S_a \cup S_b$ clearly has the property that $Z_{p-1} \oplus Z_p \subset S - S$. To summarize, this construction yields a saturated difference set in $Z_{p(p-1)} \cong Z_{p-1} \oplus Z_p$ of size no more than

$$\begin{aligned} & p - 1 + \sqrt{2p} + \sqrt{2(p-1)} \\ \leq\ & \sqrt{p(p-1)} + \sqrt{2p} + \sqrt{2(p-1)} \\ \leq\ & \sqrt{p(p-1)} + 3\sqrt[4]{p(p-1)}\,, \end{aligned}$$

as $p > 2$. (The fact that $\sqrt{2p} + \sqrt{2(p-1)} \leq 3\sqrt[4]{p(p-1)}$ for $p \geq 2$ follows immediately by expanding the fourth power of both sides.)

We record the results of this construction in the following theorem.

THEOREM 5. *For any n of the form $p(p-1)$ with $p > 2$, there is an explicit saturated difference set in $\mathbb{Z}_n$ of size no more than $\sqrt{n} + 3\sqrt[4]{n}$. For such a set, defining $d = |S|/n$, we have $L(S) \leq |S| = (1 + o(1))/d^2$.*

We use the word *explicit* in the above theorem to indicate that one can efficiently compute, given p, the elements of the difference set. The construction requires generating the elements of S_0, S_a, and S_b, which is straightforward given a generator ϑ of the cyclic group $\mathbb{Z}_p^*$. Such a generator can be found very quickly with randomization; deterministically, one can simply check the first $\sqrt{p}$ elements of $\mathbb{Z}_p$ to find one with the property that $x^{p-1} \equiv 1$ but $x^{(p-1)/r} \not\equiv 1$ for each prime r dividing $p-1$. See [2] for details.

It remains, of course, to represent the set S inside $\mathbb{Z}_{p(p-1)}$ via the isomorphism promised by the Chinese remainder theorem. For this purpose, note that

$$\begin{aligned} p &\equiv 1 \mod p-1 & (p-1)^2 &\equiv 0 \mod p-1 \\ p &\equiv 0 \mod p & (p-1)^2 &\equiv 1 \mod p\,. \end{aligned}$$

It follows immediately that the element $(a,b) \in \mathbb{Z}_{p-1} \oplus \mathbb{Z}_p$ is carried to the element $a \cdot p + b(p-1)^2 \mod p(p-1)$ in $\mathbb{Z}_{p(p-1)}$. As arithmetic in $\mathbb{Z}_p$, $\mathbb{Z}_{p-1}$ and $\mathbb{Z}_{p(p-1)}$ can be carried out in time $\text{poly}(\log p)$, we conclude that the entire set S can be computed in time $p \cdot \text{poly}(\log p)$.

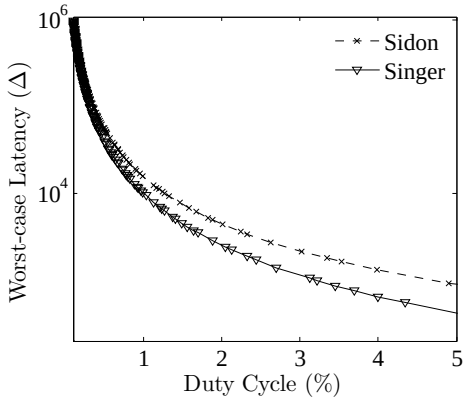

Figure 8: The worst case discovery latency of two optimal schemes (based on Singer and Sidon difference sets) for symmetric duty cycles.

Fig. 8 compares the performance of the two optimal constructions based on Singer and Sidon difference sets, respectively. The performance of Sidon set construction is very close to that of the Singer set construction for small duty cycles. The deviation under larger duty cycles is because the lower order terms in Sidon set construction, which can be improved using more efficient construction when filling the small "slices." We remark that like the other known constructions that achieve asymptotic optimality, this Sidon set construction requires n (and hence d) to have particular number-theoretic properties.

5.3 Asymmetric Case

In the asymmetric case, we again only consider optimal integer schedules for the same reason as described earlier. For two nodes with duty cycles d_1 and d_2, $d_1 \neq d_2$, the lower bound of discovery latency is $1/(d_1 d_2)$ (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 $p_1 p_2 \approx 1/(d_1 d_2)$. 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 leads 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 [5], each node uses two prime numbers instead of one prime number; in

U-Connect [9], each node uses one prime, but is awake for a continuous period of time in addition to the multiples of the prime numbers. Neither of them is optimal. The asymmetric schedules in [3, 17] 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.

6. CONCLUSION AND FUTURE WORK

In this paper, we first developed a generalized non-integer model for asynchronous neighbor discovery on duty-cycled mobile devices. This generalized model permits unified treatment of the assumptions in existing studies, and allows us to place various existing results into a single setting. We then provided a reduction that transforms any schedule in the basic integer model to a corresponding schedule in the generalized non-integer model while improving the performance by a factor of two. After that, we established a new family of lower bounds for the best achievable latency guarantee in the non-integer model. Last, we developed a novel optimal construction based on Sidon sets for the symmetric setting.

Our work motivates two directions of further work. The first is to develop optimal schedules for general duty cycles, d, in the symmetric setting. (As mentioned above, known constructions—including the new construction presented here—only provide control for a particular set of d.) One way to achieve this is through padding, in which case the analysis of the worst-case performance requires detailed analysis of the density of the relevant d. The second is to develop practical (close-to) optimal schedules for the asymmetric setting.

Acknowledgements

We gratefully acknowledge support from the National Science Foundation under grant 0835735, CAREER award 0746841, and grants 1117427 and 1407205. 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. We would like to thank the anonymous reviewers for their insightful comments.

7. REFERENCES

[1] L. Aalto, N. Göthlin, J. Korhonen, and T. Ojala. Bluetooth and WAP push based location-aware mobile advertising system. In *Proc. of MobiSys*, June 2004.

[2] E. Bach and J. Shallit. *Algorithmic Number Theory, Volume 1: Efficient Algorithms*. Foundations of Computing. MIT Press, 1996.

[3] M. Bakht, M. Trower, and R. H. Kravets. Searchlight: Won't you be my neighbor? In *Proc. of ACM MobiCom*, 2012.

[4] S. A. Borbash, A. Ephremides, and M. J. McGlynn. An asynchronous neighbor discovery algorithm for wireless sensor networks. *Ad Hoc Networks*, 5(7):998–1016, 2007.

[5] P. Dutta and D. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proc. of ACM SenSys*, 2008.

[6] B. Han, P. Hui, V. Kumar, M. Marathe, J. Shao, and A. Srinivasan. Mobile data offloading through opportunistic communications and social participation. *IEEE Transactions on Mobile Computing*, 11(5):821–834, 2012.

[7] C. S. Hsu, J. R. Jiang, Y. C. Tseng, and T. H. Lai. Quorum-based asynchronous power-saving protocols for ieee 802.11 and hoc networks. *ACM Journal on Mobile Networks and Applications (MONET)*, 10(1-2):169–181, 2005.

[8] J.-H. Huang, S. Amjad, and S. Mishra. CenWits: a sensor-based loosely coupled search and rescue system using witnesses. In *Proc. of ACM SenSys*, 2005.

[9] A. Kandhalu, K. Lakshmanan, and R. R. Rajkumar. U-Connect: a low-latency energy-efficient asynchronous neighbor discovery protocol. In *Proc. of IPSN*, 2010.

[10] A. Keshavarzian, E. Uysal-Biyikoglu, F. Herrmann, and A. Manjeshwar. Energy-efficient link assessment in wireless sensor networks. In *Proc. of IEEE INFOCOM*, March 2004.

[11] S. Lai, B. Zhang, B. Ravindran, and H. Cho. CQS-Pair: Cyclic quorum system pair for wakeup scheduling in wireless sensor networks. In *Proc. of Principles of Distributed Systems*, 2008.

[12] D. Li and P. Sinha. RBTP: low-power mobile discovery protocol through recursive binary time partitioning. *IEEE Transactions on Mobile Computing*, 13(2):263–273, 2014.

[13] T. Liu, C. M. Sadler, P. Zhang, and M. Martonosi. Implementing software on resource-constrained mobile sensors: experiences with Impala and ZebraNet. In *Proc. of ACM MobiSys*, 2004.

[14] Locast. `http://www.lokast.com`.

[15] M. J. McGlynn and S. A. Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *Proc. of ACM MobiHoc*, October 2001.

[16] L. McNamara, C. Mascolo, and L. Capra. Media sharing based on colocation prediction in urban transport. In *Proc. of ACM MobiCom*, September 2008.

[17] T. Meng, F. Wu, and G. Chen. On designing neighbor discovery protocols: A code-based approach. In *Proc. of IEEE INFOCOM*, 2014.

[18] M. Motani, V. Srinivasan, and P. S. Nuggehalli. PeopleNet: engineering a wireless virtual social network. In *Proc. of ACM MobiCom*, 2005.

[19] A. K. Pietiläinen, E. Oliver, J. Lebrun, G. Varghese, and C. Diot. MobiClique: middleware for mobile social networking. In *Proc. of ACM Workshop on Online Social Networks*, August 2009.

[20] A. Purohit, N. Priyantha, and J. Liu. WiFlock: collaborative group discovery and maintenance in mobile sensor networks. In *Proc. of IPSN*, 2011.

[21] D. R. Stinson. Combinatorial designs: Constructions and analysis. *Springer Verlag*, 2003.

[22] Nintendo 3ds's streetpass. `http://www.nintendo.com/3ds/features`.

[23] T. Tao and V. Vu. *Additive Combinatorics*. Number 105 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2006.

[24] 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. of IEEE INFOCOM*, 2002.

[25] S. Vasudevan, M. Adler, D. Goeckel, and D. Towsley. Efficient algorithms for neighbor discovery in wireless networks. *IEEE/ACM Transactions on Networking*. To Appear.

[26] Z. Yang, B. Zhang, J. Dai, A. Champion, D. Xuan, and D. Li. E-SmallTalker: A distributed mobile system for

social networking in physical proximity. In *Proc. of IEEE ICDCS*, June 2010.
[27] D. Zhang, T. He, Y. Liu, Y. Gu, F. Ye, R. K. Ganti, and H. Lei. Acc: generic on-demand accelerations for neighbor discovery in mobile applications. In *Proc. ACM SenSys*, 2012.
[28] 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. of IEEE ICDCS*, 2011.
[29] L. Zhang and D. Guo. Neighbor discovery in wireless networks using compressed sensing with Reed-Muller codes. In *Proc. of WiOpt*, 2011.
[30] R. Zheng, J. C. Hou, and L. Sha. Asynchronous wakeup for ad hoc networks. In *Proc. of ACM MobiHoc*, 2003.

APPENDIX

The Singer Difference Set Construction

DEFINITION 6. *A subset $D \subseteq \mathbb{Z}_v$ is called a $(v, k, 1)$-difference set if $|D| = k$ and, for any $g \in \mathbb{Z}_v \backslash \{0\}$, there exists exactly one pair $(x, y) \in D \times D$ such that $g \equiv x - y \mod v$.*

For a given v, the Singer difference set [21] has $k^2 + k + 1 = v$ and $k - 1$ is a prime power. It is the minimal difference set whose size, k, approximates the lower bound $\sqrt{v}$, i.e., $k \approx \sqrt{v}$. It can be directly used to construct an optimal neighbor discovery schedule for the symmetric setting. In the following, for completeness, we first prove the existence of Singer difference sets (the proof is adapted from that in [21]). We then describe an explicit polynomial time construction.

THEOREM 7. *Let q be a prime power. Then there exists an explicit $(q^2 + q + 1, q + 1, 1)$-difference set in $\mathbb{Z}_{q^2+q+1}$.*

PROOF. The finite field $\mathbb{F}_{q^3}$ is a three-dimensional vector space over $\mathbb{F}_q$ for any prime power q. Let $\mathcal{V}_1$ denote the collection of all one-dimensional subspaces of $\mathbb{F}_{q^3}$ and $\mathcal{V}_2$ denote the collection of all two-dimensional subspaces of $\mathbb{F}_{q^3}$. For each $B \in \mathcal{V}_2$, let $A_B \triangleq \{C \in \mathcal{V}_1 \mid C \subset B\}$. It is easy to check that $|\mathcal{V}_1| = q^2 + q + 1$, a quantity we call ℓ throughout the construction; furthermore $|\mathcal{V}_2| = |\mathcal{V}_1|$, as every two-dimensional space is dual to a unique one-dimension space under the bilinear map $\langle x, y\rangle = \mathrm{Tr}_{\mathbb{F}_{q^3}/\mathbb{F}_q}(xy) = (xy)^{q^2} + (xy)^q + xy$. It is also easy to check that $|A_B| = q + 1$ for each $B \in \mathcal{V}_2$ and that any distinct pair $(C_1, C_2) \in \mathcal{V}_1 \times \mathcal{V}_1$ is contained exactly in one two-dimensional space $B \in \mathcal{V}_2$.

Let ω denote a primitive element of $\mathbb{F}_{q^3}$ (that is, a generator for the multiplicative group $\mathbb{F}^*_{q^3}$). Define a mapping $\alpha : \mathbb{F}_{q^3} \to \mathbb{F}_{q^3}$ by $\alpha(x) = \omega x$. Note that α is $\mathbb{F}_q$-linear (and clearly bijective): it carries subspaces to subspaces. It is not difficult to show that α permutes the elements in $\mathcal{V}_1$ in a single cycle of length ℓ. Fixing a particular one-dimensional space C_0, we write $\mathcal{V}_1 = \{C_0, C_1, \cdots C_{\ell-1}\}$ with the convention that $\alpha(C_i) = C_{i+1 \mod \ell}$ and, hence, $C_i = \omega^i C_0$. (Note that this is well-defined because α^ℓ, corresponding to left-multiplication by ω^ℓ, has the identity action on the C_i.) Rather remarkably, it can be proved that α also permutes the elements in $\mathcal{V}_2$ in a single cycle of length ℓ.

Let B_0 denote a fixed two-dimensional subspace and define $D \triangleq \{i \mid C_i \subset B_0\} \subset \mathbb{Z}_\ell$. To see that D is a difference set, consider an element $g \in \mathbb{Z}_\ell$, $g \neq 0$. Now, the pair (C_0, C_g) is contained in exactly one two-dimensional subspace B; let i have the property that $\alpha^i B_0 = B$. Now it follows that $\alpha^{-i} B = B_0$ and both C_{-i} and C_{g-i} are contained in B_0; hence $-i$ and $g-i$ are both in D, with $g-i-(-i) \equiv g \mod \ell$. Uniqueness follows similarly. Thus D is a $(q^2+q+1, q+1, 1)$-difference set in $\mathbb{Z}_{q^2+q+1}$. □

Construction: For a prime power p, $\mathbb{F}_{p^3}$ can be realized as $\mathbb{F}_p[x]/(g(x))$ where $g(x) \in \mathbb{F}_p[x]$ is an irreducible polynomial of degree 3. Then the elements of $\mathbb{F}_{p^3}$ can be represented as polynomials in $\mathbb{F}_p[x]$ having degree at most 2. We may choose $C_0 = \mathbb{F}_p$, the constant polynomials; we then choose $B_0 \in \mathcal{V}_2$ to be $\mathrm{span}(1, x) = \{i + jx \mid i, j \in \mathbb{F}_p\}$. This yields the difference set

$$D = \{y \in \mathbb{Z}_{p^2+p+1} \mid \omega^y = ax + b \text{ for some } a, b \in \mathbb{F}_q\},$$

where ω is a primitive element of $\mathbb{F}_{p^3}$.

The remaining work is to find a irreducible polynomial $g(x)$ and a primitive element ω. It turns out that we can do both at once. Initially, factor $p^3 - 1$ such that $p^3 - 1 = p_1^{n_1} \cdots p_k^{n_k}$, where each p_i is a prime number. It suffices to identify a polynomial $g(x) = x^3 + bx^2 + cx + d$ (where $b, c, d \in \mathbb{F}_p$) so that: (i.) $x^{p^3-1} \equiv 1 \mod g(x)$, and (ii.) $x^{(p^3-1)/p_i} \not\equiv 1 \mod g(x)$ for each i. Such $g(x)$ is irreducible in $\mathbb{F}_p[x]$ and guarantees that x is a primitive element of $\mathbb{F}_{p^3}$. As such g are known to be sufficiently dense (in fact, they have density $\phi(p^3 - 1)/(3p^3)$, where ϕ is the Euler totient function), this can be carried out very quickly with a randomized algorithm; see [2]. As construction of the set S will take time polynomial in p anyway, it is also possible to carry out this search over all triples a, b and c. In either case, so long as fast modular exponentiation is used to compute the powers x^v, the construction can be carried out in time $p^3 \,\mathrm{poly}(\log p)$, as desired.

This construction can be applied as well for prime powers of the form $q = p^k$. The only new complication is that one must first construct the field $\mathbb{F}_q$, which can be accomplished using the same technique as described above: specifically, one chooses an irreducible polynomial $h(x) \in \mathbb{F}_p[x]$ of degree k and uses the set $\mathbb{F}_p[x]/(h(x))$ as a representation of $\mathbb{F}_q$, where addition is simply addition of polynomimals, and multiplication is given by polynomial multiplcation modulo the polynomial $h(x)$. The remainder of the construction proceeds as above.