Hello: A Generic Flexible Protocol for Neighbor Discovery

Wei Sun*, Zheng Yang†, Keyu Wang* and Yunhao Liu†
*CSE, Hong Kong University of Science and Technology
†School of Software and TNList, Tsinghua University
Email: {wsunaa, kwangad}@cse.ust.hk, {yang, yunhao}@greenorbs.com

***Abstract*—Neighbor discovery is crucial for both wireless sensor networks and mobile computing applications. The crux of the problem is to achieve energy efficiency, which has been demonstrated to be difficult by prior work. In this paper we propose Hello, a generic flexible protocol for neighbor discovery. With an unrestricted parameter, it serves as a generic framework that incorporates existing deterministic protocols. Under the framework, we expose optimal parameters for either symmetric or asymmetric duty cycles, which is the first to our knowledge. As a result, it exhibits great flexibility by adjusting itself to any configuration that best meets application demands. Besides, several techniques are applied to removing redundant discoveries without sacrificing the worst-case latency. We evaluate Hello through extensive simulation and real-world sensor experiments. The results show that Hello is highly energy-efficient under symmetric and asymmetric duty cycles. In particular, it is two times better than the state of the art in terms of the worst-case latency under asymmetric duty cycles.**

I. Introduction

Ever-developing fabrication technology has given birth to a wide variety of miniaturized low-cost multi-functional devices. This trend of evolvement nurtures two classes of far-ranging applications: Wireless Sensor Networks (WSNs) and Personal Mobile Computing (PMC). WSNs are composed of low-power wireless sensors with data collection, processing, and transmission capacities. They are typically applied to environmental monitoring [1], smart housing [2], object tracking [3], and the like. More computationally powerful devices, such as smartphones and tablets, satisfy the increasing needs of PMC. Notably popular PMC communities include mobile gaming (*e.g.*, PlayStation Vita [4] and MOGA [5]) and mobile social networking (*e.g.*, Nextdoor [6] and Highlight [7]).

Both classes of applications regard *neighbor discovery* as a fundamental function. Sensors in WSNs need to discover their neighbors for network connectivity maintenance [8], while many proximity-based PMC applications are built upon direct interconnection between devices. Inability to discover neighbors promptly may result in unawareness of neighbors or loss of transient encounters, which may severely degrade network or application performance. In practice, nodes (*e.g.*, sensors or smartphones) cannot afford to turn radios on all the time. Hence they have to make a compromise by turning radios on/off from time to time, with the portion of time in the ON state characterized by *duty cycle*. Typically, duty cycle and discovery latency are two key metrics by which energy efficiency is evaluated. However, there is a natural trade-off between them: a lower duty cycle usually leads to a higher discovery latency, and vice versa.

978-1-4799-3360-0/14/$31.00 © 2014 IEEE

As a result, energy-efficient neighbor discovery is nontrivial. Though clock synchronization via GPS [9] or NTP (Network Time Protocol) [10] greatly simplifies the problem, it requires either power-hungry operations or Internet access, not scalable for a wide range of applications. The problem gets more complicated as nodes can work under the same (symmetric) or different (asymmetric) duty cycles. A favorable mechanism should provide good performance under both scenarios.

There has been several proposals for asynchronous neighbor discovery that can work under both symmetric and asymmetric duty cycles. For example, a family of Birthday protocols [11] are representative of probabilistic designs. Disco [12] and U-Connect [13], which are both deterministic protocols, provide a latency bound by virtue of the Chinese Remainder Theorem. Another state-of-the-art deterministic protocol called Searchlight [14] achieves higher efficiency by leveraging constant offset between periodic active slots. Since the Birthday protocols fail to guarantee a worst-case latency, we limit our discussion to deterministic protocols in the following.

Despite their success in achieving efficiency to some extent, existing protocols may suffer from the following drawbacks. First, *their parameters are highly restricted*. Both Disco and U-Connect are built upon primes, limiting duty cycles to specific forms. Worse still, Searchlight requires duty cycles to be power-multiples of the smallest one, resulting in only a handful of options available (*e.g.*, at most four for a targeted span of duty cycles ranging from 1% to 10%). Second, *there are redundant discovery opportunities within a latency bound*. Such redundancy might help accelerate discovery in general, but rarely decreases the worst-case latency. Third, *they cannot adjust well to application demands*. Various applications may manifest different preferences towards low discovery latency under symmetric or asymmetric duty cycles. Even for a particular application, its preference may vary over time (*e.g.*, hiking logging [15] with APs or companions). Those protocols except Disco, however, cannot respond to such dynamics.

To mitigate these drawbacks, we propose Hello[1], a generic

[1]It is so named since "Hello" is a globally recognized (generic) word for greeting neighbors.

flexible protocol for neighbor discovery. Compared with the aforementioned protocols, it manifests the highest degree of freedom in parameter selection as one of its two parameters remains unrestricted. Given such freedom, we show that it can serve as a generic framework incorporating all the deterministic protocols mentioned above. By exploring its parameter space under the framework, we expose optimal parameters for symmetric or asymmetric duty cycles. By switching between these optimums, Hello exhibits great flexibility to respond to dynamic application demands. Moreover, discovery redundancy is removed by getting rid of superfluous active slots without compromising on the worst-case latency.

We evaluate the performance of Hello through extensive simulation and real-world sensor experiments, and the results are promising. Under symmetric duty cycles, Hello is highly comparable to the state-of-the-art Searchlight. Under asymmetric duty cycles, on the other hand, Hello significantly outperforms existing protocols. In particular, it is two times better than Searchlight in terms of the worst-case latency.

In summary, we outline the contributions of this work as follows:

- **A neighbor discovery protocol that exhibits the highest degree of freedom**. The least restriction is imposed on Hello, giving it a great many choices for the parameters.
- **An optimality analysis under a generic framework**. To our best knowledge, we are the first to design a generic framework incorporating existing deterministic protocols, and to expose optimal parameters for symmetric and asymmetric duty cycles.
- **A flexible solution to applications dynamics**. Hello can be fine-tuned to any configuration that best meets application demands.
- **Extensive evaluation through simulation and real-world experiments**. The results show that Hello is highly competitive under both symmetric and asymmetric duty cycles.

The rest of this paper is organized as follows. Section II briefly describes existing protocols for neighbor discovery. The design of Hello is detailed in Section III. We present the simulation studies and experimental results from its implementation in Section IV and V, respectively. Finally, we conclude the work in Section VI.

II. Related Work

Neighbor discovery without time synchronization is of major concern in research communities. Several communication protocols, such as BMAC [16] and SMAC [17], can serve this purpose. They employ lower-power listening with a fixed listening period, and assume a global agreement on duty cycle throughout the network. In practice, however, duty cycle is often locally determined by workloads or energy budgets, leading to various duty cycles among nodes. Therefore, neighbor discovery mechanisms that support both asynchronous and asymmetric duty cycles are in high demand.

In response, several neighbor discovery protocols have been proposed. They typically fall into two categories: probabilistic and deterministic. The best representative among probabilistic approaches is a family of Birthday protocols [11], where a node decides to transmit, listen, or sleep with probabilities. By virtue of their probabilistic nature, they support asymmetric duty cycles, but suffer from aperiodic and unpredictable discovery, leading to unbounded worst-case latency.

Deterministic protocols, on the other hand, guarantee a definite bound on discovery latency. One class of deterministic protocols is termed as Quorum-based protocols [18], [19]. They divide time into groups of m^2 continuous intervals. Within each group, the m^2 intervals are organized as a 2-dimensional $m \times m$ array in a row-major manner. Then a node can arbitrarily pick one column and one row of entries as active intervals. Clearly for any two nodes, this paradigm ensures two overlaps of active intervals within m^2 intervals. The initial quorum design [18] requires that m be a global parameter, and a later improvement [19] slightly relaxes this constraint to support two different schedules, of which both underperform in dealing with duty-cycle asymmetry.

Theoretically, Zheng et al. [20] formulated the symmetric neighbor discovery as the symmetric block design problem in combinatorics, and suggested the use of the Multiplier Theorem [21] to derive an optimal block design. With respect to duty-cycle asymmetry, they showed that this approach reduces to the minimum vertex-cover problem, which is NP-complete. Likewise, it is fundamentally limited to symmetric duty cycles.

Prime-based protocols [12], [13] overcome this limitation by leveraging the Chinese Remainder Theorem [22]. In Disco [12], each node selects a pair of different primes p and q such that the sum of their reciprocals is close to a desired duty cycle. It then wakes up at multiples of the individual prime p or q. Clearly for any two nodes, there exists at least one pair of different primes each from the individual node. Thus neighbor discovery is guaranteed within the least product of such prime pairs. For ease of design, U-Connect [13] uses a single prime p and requires nodes to wake up in the first $\frac{p+1}{2}$ slots every p^2 slots. This ensures discovery within the product of the two primes for any two nodes. One drawback of both protocols is that their parameters are fundamentally limited by prime selections.

The state-of-the-art Searchlight [14] explores another opportunity for neighbor discovery by leveraging constant offset between periodic active slots. There are two active slots in a cycle of t slots. The first (indexed from 0) is always an active slot called the anchor slot. The other active slot, called the probe slot, traverses from position 1 to $\lfloor \frac{t}{2} \rfloor$ across $\lfloor \frac{t}{2} \rfloor$ cycles (*i.e.*, a period). This design ensures a discovery within a period under symmetric scenarios. To maintain the constant offset under asymmetric scenarios, Searchlight restricts duty cycles to power-multiples of the smallest one. However, this restriction significantly limits its parameter t. For example, in a span of duty cycles ranging from 1% to 10%, at most four different duty cycles (say 1%, 2%, 4%, and 8%) are available for Searchlight.

III. Design

In this section we develop the design of Hello. We first present the schedule of active slots and derive its worst-case latency under symmetric duty cycles. A study on addressing duty-cycle asymmetry is then expounded. In what follows, we demonstrate that under symmetric scenarios, Hello serve as a generic framework incorporating existing deterministic protocols such as Quorum, Disco, U-Connect, and Searchlight. Under this framework, we explore for optimal parameters for symmetric or asymmetric scenarios. Finally, we show ways to reduce or utilize discovery redundancy. Note that we focus on initial discovery at this stage, and leave neighbor maintenance [23] as future work.

A. Schedule Design

Consider a question asking how to ensure the mutual discovery of two nodes within a period of c slots. Among many possible solutions, the most straightforward one may be to stay active in the first $(\lfloor c/2 \rfloor + 1)$ slots, slightly larger than half the period. By this means, at least one discovery is guaranteed within a period, regardless of the phase offset due to their asynchronous clocks. However, it suffers severely from a high duty cycle, which has to be over 50%.

An amendment can be made to this solution by compromising on the latency bound for a low duty cycle. We expand the period to cn slots, where n is a positive integer larger than 1. Starting from the c-th slot, the first slot every c slots is selected as a new active slot. In such a manner, the duty cycle can be reduced to a significant extent, depending on the value of n. Meanwhile, we can show that with a reduced duty cycle, a discovery is still guaranteed (proof given in the following section). In essence, this is where Hello originated.

Systematically, Hello is customized by two parameters: cycle length c and the number n of cycles in a period. During each period, a node wakes up at the first slot of each cycle, and at the consecutive slots indexed from 1 to $\lfloor c/2 \rfloor$ in the first cycle. For ease of reference, we refer to the former set of active slots as *guardians* and the latter as *patrols*. The worst-case discovery latency $\mathcal{L}_s$ for Hello under symmetric duty cycles is the period $T = cn$, as stated in Lemma 1 in the next section. We denote by $Hello(c, n)$ an instance of Hello with parameters c and n. Fig. 1 illustrates a schedule of $Hello(9, 3)$, where slots containing G or P indicate guardians or patrols, respectively.

We now take a quick look at Hello's duty cycle. There are n guardians and $\lfloor c/2 \rfloor$ patrols in each period of cn slots. Therefore, the duty cycle of Hello is given as

$$d = \frac{\lfloor c/2 \rfloor + n}{cn}. \tag{1}$$

Note that c and n are free to choose and independent from each other. *This provides Hello with the highest degree of freedom in parameter selection compared with existing deterministic protocols*, which require either parameter(s) to be prime (*e.g.*, Disco [12] and U-Connect [13]) or duty cycles to be power-multiples of the smallest one (*e.g.*, Searchlight [14]). Although we will impose a restriction on c in Section III-C to address duty-cycle asymmetry, Hello still remains highly autonomous since n is always kept unrestricted.

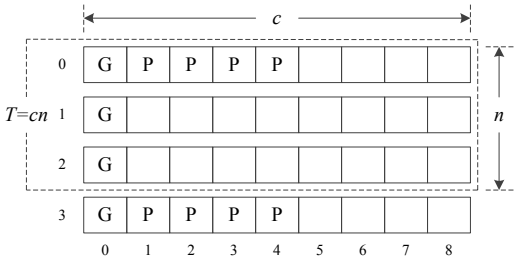


Fig. 1. Hello with $c = 9$ and $n = 3$.

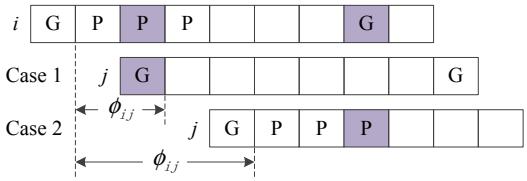


Fig. 2. Discovery for two cases under symmetric duty cycles.

B. Symmetric Discovery Latency

In this section, we give a proof of the worst-case discovery latency for Hello under symmetric duty cycles.

Lemma 1. *The worst-case discovery latency $\mathcal{L}_s$ for Hello under symmetric duty cycles with parameter pair (c, n) is cn slots.*

Proof: Consider any two nodes i and j adopting Hello for discovery with the same parameters c and n. Since both nodes wake up at multiples of c, the relative phase offset between their guardians, denoted by ϕ_{ij} ($\phi_{ij} \in [0, c)$), must be constant. When $\phi_{ij} = 0$, their guardians are in-sync with one another, and hence the worst-case latency is c slots. Next, we decompose other possibilities into two cases.

Case 1: $0 < \phi_{ij} \leq \lfloor c/2 \rfloor$. Within a period of cn slots, node i's patrols must meet a guardian of node j (highlighted in Fig. 2), as they traverse from position 1 to $\lfloor c/2 \rfloor$ in one cycle of the period. The worst-case latency of cn slots is encountered when they come within communication range right after a patrol of node i should have met a guardian of node j.

Case 2: $\lfloor c/2 \rfloor < \phi_{ij} < c$. The same reasoning as in Case 1 applies to this case, except that the roles of nodes i and j exchange (see Fig. 2).

With all scenarios considered, a discovery will take place within cn slots. ■

C. Duty-Cycle Asymmetry

In practice, duty cycles are often determined locally by workloads or energy budgets. As it is common that these determinants vary throughout the network and during the course of usage, neighbor discovery under asymmetric duty cycles becomes common and necessary. To address this practical

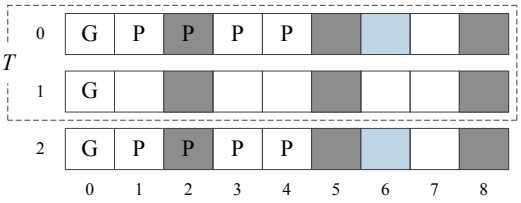


Fig. 3. Discovery guaranteed for $c_1 \geq 2c_2$.

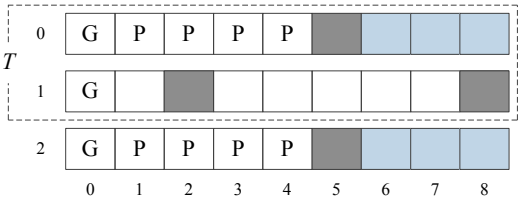


Fig. 4. Discovery not guaranteed for $c_2 < c_1 < 2c_2$.

concern, we discuss two possible scenarios that lead to duty-cycle asymmetry, and analyze several approaches to providing a reasonable latency bound.

1) ***Same*** **c,** ***Different*** **n***:* One possibility of the duty-cycle asymmetry is that nodes select the same cycle length but different numbers of cycles. Consider two nodes A and B with discovery schedule $Hello(c, n_1)$ and $Hello(c, n_2)$, respectively. Without loss of generality, we assume $n_1 > n_2$. Here one natural question arises: is a discovery between nodes A and B guaranteed within a worst-case bound? Fortunately the answer is yes. Through similar arguments as in the proof of Lemma 1, we can show that the worst-case discovery latency is cn_1 slots (proof omitted due to the page limit). Put another way, *two nodes operating asymmetrically with the same cycle length but different numbers of cycles can discover each other within the longer period of the two.* Nothing needs to be done for discovery guarantee under this scenario.

2) ***Different*** **c***:* Different selections of cycle lengths give the other possibility of the duty-cycle asymmetry. Consider two nodes A and B with discovery schedule $Hello(c_1, n_1)$ and $Hello(c_2, n_2)$, respectively, where we assume $c_1 > c_2$. Likewise, we are interested in whether a discovery is guaranteed within a worst-case bound. To study this question, we further decompose it into two cases.

Case 1: $c_1 \geq 2c_2$. Since nodes wake up at multiples of corresponding c, this condition implies that there is at least one of node B's guardians in the first half of each cycle of node A. Hence in each period of node A containing $c_1 n_1$ slots, at least a guardian or a patrol of node A meets a guardian of node B, leading to a mutual discovery. Fig. 3 exemplifies the case of $(c_1, n_1) = (9, 2)$ and $(c_2, n_2) = (3, 6)$, where the gray and blue slots are the guardians and patrols of node B, respectively. In this condition, a discovery is guaranteed within a fixed bound without further adjustments. In order to make this condition hold for any two asymmetric nodes, however, it requires that c be the power of 2 multiples of a base value, similar to Searchlight. This requirement is prohibitively restrictive in terms of freedom in parameter selection, resulting in sparse options available. Worse still, such sparsity disables the optimal parameter selections presented in Section III-E. Therefore, we have to deal with the case when this condition does not hold.

Case 2: $c_2 < c_1 < 2c_2$. Neighbor discovery may not be guaranteed in this case, as illustrated in Fig. 4 where $(c_1, n_1) = (9, 2)$ and $(c_2, n_2) = (6, 3)$. A possible solution is to require c_1 and c_2 to be prime. As they are also distinct, they are coprime by themselves. It follows from the Chinese Remainder Theorem [22] that the guardians of the nodes overlap once every $c_1 c_2$ slots. Though this requirement also restricts the selection of c to some extent, it provides much more room than Searchlight. For example, there are 21 primes for Hello ranging from 100 to 200, the targeted range of c at a single duty cycle of 1% (see Section III-E), but only at most 4 options available for Searchlight in a range of duty cycles from 1% to 10%.

In short, *Hello ensures asymmetric discovery latency within a fixed bound by requiring the cycle length c to be prime*. We summarize the whole analysis in Lemma 2.

Lemma 2. *The worst-case discovery latency $\mathcal{L}_{as}$ for Hello under asymmetric duty cycles with parameter pairs (c_1, n_1) and (c_2, n_2) is given as*

$$\mathcal{L}_{as} = \begin{cases} \max\{c_1 n_1, c_2 n_2\} & \text{if } c_1 = c_2 \\ c_1 c_2 & \text{if } c_1 \neq c_2 \end{cases}. \quad (2)$$

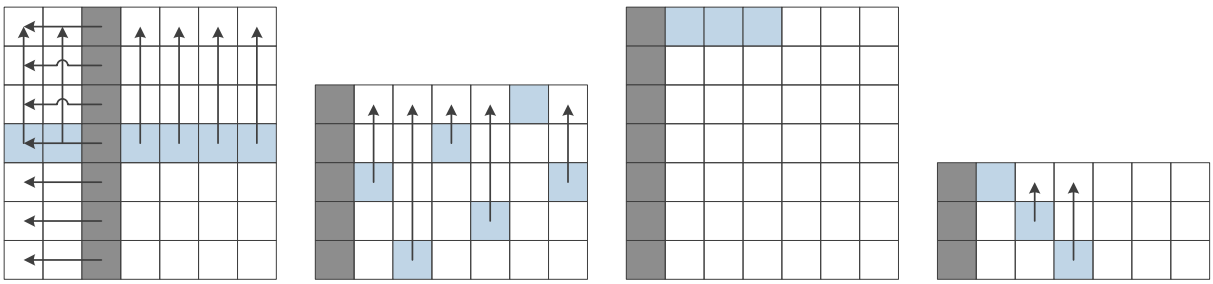
Proof: It can be easily proved by the preceding analysis. ■

D. A Generic Framework

From the design of Hello, we find some interesting similarities with existing deterministic protocols. For example, some of their active slots, say guardians in Hello and anchor slots in Searchlight, exhibit a repetitive pattern. In this section we study the interconnection between these protocols. To our surprise, it turns out that under symmetric duty cycles, *Hello can serve as a generic framework incorporating well-known protocols such as Quorum, Disco, U-Connect, and Searchlight*. In what follows, we show in turn how each is either a redundant variant or specific derivant of Hello.

1) ***Quorum****:* In Quorum, a node picks one column and one row of entries as active slots in an $m \times m$ array of consecutive slots. We observe that it makes no difference in terms of worst-case latency if each node picks the first column and the first row. Consequently, if the selected row/column are shifted to the first row/column (see Fig. 5(a)), Quorum becomes a redundant variant of Hello with additional patrols in the second half of the first cycle. That is, Quorum is derived from $Hello(m, m)$.

2) ***Disco****:* A node using Disco for neighbor discovery wakes up at multiples of individual prime p or q ($p \neq q$). Consider a period of pq slots that are organized into a $q \times p$



(a) Quorum ~ Hello(7,7) (b) Disco ~ Hello(7,5) (c) U-Connect ~ Hello(7,7) (d) Searchlight ~ Hello(7,3)

Fig. 5. Hello as a generic framework.

matrix. Clearly, the first column are all active slots. Since p and q are coprime, it is known that no two of the integers $q, 2q, \ldots, (p-1)q$ are congruent modulo p [24]. Therefore, there is exactly one active slot in each remaining column. Likewise, if those active slots are shifted to the first row, the worst-case latency remains unchanged and Disco amounts to a redundant variant of Hello (see Fig. 5(b)). Its archetype in Hello is $Hello(p, q)$, where p and q are primes.

*3) **U-Connect**:* It is a direct derivant from Hello in the form of $Hello(p, p)$ where p is a prime (see Fig. 5(c)).

*4) **Searchlight**:* In Searchlight, a nodes wakes up at anchor slots that come every t slots, and at probe slots that traverse from position 1 to $\lfloor t/2 \rfloor$ across $\lfloor t/2 \rfloor$ cycles. It is easy to find the correspondences between anchor slots and guardians, and between probe slots and patrols using a similar technique as before. Essentially, Searchlight is a particular derivant of Hello represented as $Hello(t, \lfloor t/2 \rfloor)$ (see Fig. 5(d)).

Overall, the above analysis shows that under symmetric duty cycles, Hello can be reduced to each of these protocols by tuning parameters. *Such generality enables Hello to explore the whole parameter space and search for optimal parameters to achieve the minimum duty cycle or minimum discovery latency*. The next section specifies how to tune the parameters for symmetric or asymmetric duty cycles.

E. Optimality Analysis

Up to now, one critical question has not been answered: since there are multiple combinations of the parameters for a duty cycle, how to select among them? Or, is there an optimal selection that leads to minimum discovery latency? Especially given Hello as a generic framework, an optimality analysis would shed light on how existing protocols and other derivants of Hello perform from a theoretic perspective. In this section, we derive optimal parameters separately for symmetric and asymmetric duty cycles, discuss a compromise between the two scenarios, and finally highlight Hello's flexibility in adjusting to application demands.

*1) **Optimum for symmetric duty cycles**:* Our objectives are to minimize the worst-case latency given a duty cycle, and to minimize the duty cycle given a latency requirement. These two objectives are both practical for real-world applications — nodes tend to work at a duty cycle according to workloads and energy budgets, while a latency requirement may be specified for the sake of application performance.

First consider two nodes that operate at the same duty cycle d and select the same parameters c and n for Hello. As proved in Section III-B, their worst-case discovery latency $\mathcal{L}_s$ is equal to cn slots. With Eqn. (1) and ignoring the round-down error, it follows that

$$\mathcal{L}_s = \frac{c^2}{2(cd-1)}. \tag{3}$$

Taking the first-order derivative of $\mathcal{L}_s$ and setting it to zero, we obtain a solution of

$$c = \frac{2}{d}. \tag{4}$$

It is easy to verify that *this solution gives the minimum of* $\mathcal{L}_s$. Considering the compatibility with duty-cycle asymmetry, c is supposed to select a prime close to $2/d$, and n be determined by c and Eqn. (1), which is close to $1/d$.

Through similar reasoning as above, the cycle length that achieves the minimum duty cycle given a latency requirement L is optimized at

$$c = \sqrt{2L}. \tag{5}$$

Likewise, c is supposed to select a prime close to $\sqrt{2L}$, and n be determined accordingly, which is close to $\sqrt{2L}/2$.

It is interesting to note that there is a relatively constant relationship between parameters c and n for both optimums: $c \approx 2n$. Moreover, we find that Searchlight *happens to* be the optimum for symmetric duty cycles as it is in the form of $Hello(c, \lfloor c/2 \rfloor)$.

*2) **Optimum for asymmetric duty cycles**:* Our objective is to minimize the worst-case latency given different duty cycles. Since latency requirements under asymmetric duty cycles cannot be met in a local fashion, we leave them short of discussion.

Consider two nodes that operate at different duty cycles d_1 and d_2. They select different sets of parameters (c_1, n_1) and (c_2, n_2), respectively, where c_1 and c_2 are primes. We assume $d_1 < d_2$. As shown in Section III-C, the worst-case discovery latency is given by Eqn. (2). Therefore, their optimal worst-case discovery latency $\mathcal{L}^*_{as}$ is equal to

$$\mathcal{L}^*_{as} = \min\left\{ \min_{c_1 \neq c_2} c_1 c_2,\ \min_{c_1 = c_2} \max\{c_1 n_1, c_2 n_2\} \right\}, \tag{6}$$

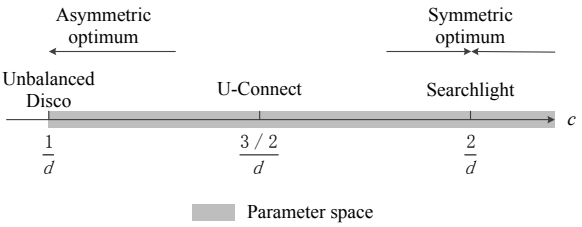


Fig. 6. Parameter space of cycle length c.

subject to

$$d_i = \frac{\lfloor c_i/2 \rfloor + n_i}{c_i n_i} \quad i = 1, 2. \tag{7}$$

To minimize $c_1 c_2$ when $c_1 \neq c_2$, we can minimize individual c_i. Note that Eqn. (7) implies that $c_i > 1/d_i$. By selecting a prime slightly larger than $1/d_i$, $c_1 c_2$ is minimized and it approaches $1/(d_1 d_2)$. On the other hand, when $c_1 = c_2$, it can be found from Eqn. (7) that $n_1 > n_2$ as $d_1 < d_2$. Thus the second component in Eqn. (6) is equal to $\min\{c_1 n_1\}$, or $2/d_1^2$ given by Eqn. (4). Since $1/(d_1 d_2)$ is at least two times smaller than $2/d_1^2$, the optimal worst-case discovery latency is close to $1/(d_1 d_2)$. Overall, *the nodes achieve the minimum worst-case latency if they select their cycle length c to be a prime slightly larger than the reciprocal of the duty cycle*. This result well explains why Disco performs best under asymmetric duty cycles when it selects unbalanced primes, of which one prime is very close to the reciprocal of the duty cycle.

*3) **A compromise**:* From the above analysis, we find that the two optimums for symmetric and asymmetric duty cycles cannot be achieved simultaneously as they require different values of c. Typically with a duty cycle d, the symmetric worst-case latency decreases as c goes from $1/d$ to $2/d$, but at the cost of increasing the asymmetric worst-case latency, and vice versa. We wonder if there is a balanced compromise between the two optimums.

Suppose that multiple nodes work at the duty cycle d and select c_1 or c_2 as the cycle length. We assume $c_1 \approx c_2 \approx c$ for a unified solution. A compromise can be made by balancing the symmetric and asymmetric worst-case latency:

$$\frac{c^2}{2(cd-1)} = c^2. \tag{8}$$

Clearly, $c = \frac{3}{2}/d$ is the solution. Coincidentally, this solution corresponds to U-Connect, where

$$d = \frac{3c+1}{2c^2} \approx \frac{3}{2}/c.$$

*4) **Optimal flexibility**:* Fig. 6 shows the parameter space of cycle length c, and indicates the locations to which existing deterministic protocols correspond. It can be seen that those protocols have already held a particular position within the space in question. However, *they all suffer from limited inflexibility in adjusting to application demands*. In particular, we observe that different applications may manifest different preferences towards latency under symmetric or asymmetric duty cycles. Even for a specific application, its preference may vary over time. One typical example is hiking logging [15], which benefits more from low symmetric latency when together with a group of companions, and more from low asymmetric latency when registering with anchor APs.

For U-Connect and Searchlight, their parameters are uniquely determined by duty cycles. Therefore, they cannot adjust themselves to dynamic application demands and are thus suitable for only a small class of applications. Though Disco exhibits flexibility to some extent via balanced or unbalanced primes, it is restricted since it requires two primes at the same time. Different from them, *Hello is highly flexible to select any configuration within the parameter space*. The cycle length c can be fine-tuned to approach the symmetric optimum or the asymmetric optimum according to application demands.

F. Slot Non-Alignment

In the previous discussion, we implicitly assume that the slots are aligned. In practice, this assumption rarely holds as nodes work independently and they lack any global time reference. To maximize the likelihood that overlaps of active slots will lead to discoveries, Hello employs the same beaconing strategy as Disco and Searchlight — *a node transmits a beacon at both the beginning and end of an active slot*. Yet this strategy also introduces a new source of discovery redundancy due to the two opportunities for active slots to overlap when slots are not aligned. We focus on the redundancy issue in the next section.

G. Redundancy

Though Hello gets rid of the redundant patrols as in Disco, there are still multiple opportunities for discovery within a worst-case bound. In this section we decompose them into two cases and study how to reduce or take advantage of them.

*1) **Redundancy from slot non-alignment**:* It was first noted by Searchlight [14] that beaconing at both the beginning and the end of an active slot leads to two discovery opportunities when slot boundaries are not aligned. Such redundancy was eliminated by striped probing — every even slot is probed and each active slot overflows by δ, a small amount sufficient to receive a leading beacon of another node. In the context of Hello, it translates into retaining patrols at only even slots. Besides reduced patrols, *the overflowing guardians contribute to more opportunities for discovery*. Specifically, under asymmetric duty cycles with $Hello(c_1, n_1)$ and $Hello(c_2, n_2)$, it yields two overlaps of guardians within $c_1 c_2$ slots by the Chinese Remainder Theorem. Since these two overlaps are usually spaced away from each other, the worst-case bound can be further contracted.

With striped probing, the duty cycle of Hello turns into

$$d_{sp} = \frac{\lceil \frac{\lfloor c/2 \rfloor}{2} \rceil + n}{cn} \tag{9}$$

Through similar reasoning as in the previous section, the optimal selections of the cycle length remain the same for both symmetric and asymmetric scenarios (proof omitted).

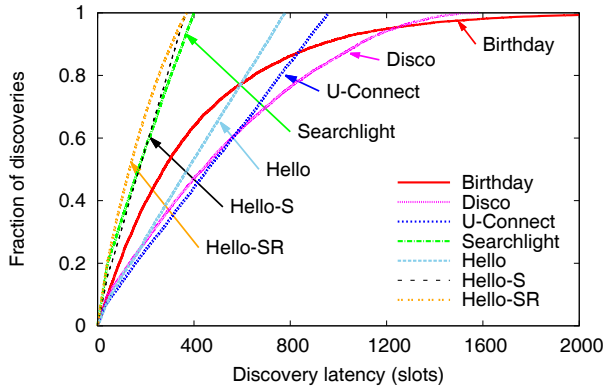


Fig. 7. CDF of discovery latency at a duty cycle of 5%.

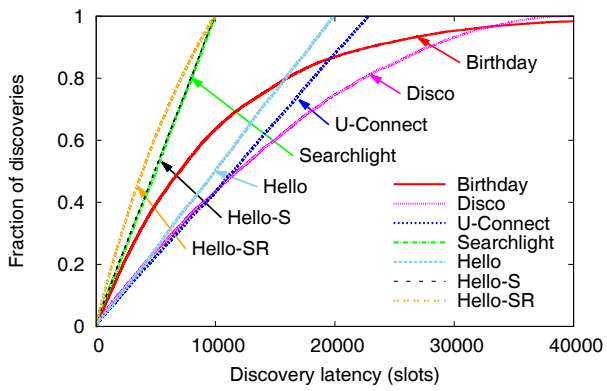


Fig. 8. CDF of discovery latency at a duty cycle of 1%.

*2) **Redundancy from repetitive patrols**:* Even with striped probing, discovery redundancy still remains under symmetric duty cycles. For example, when the phase offset between two nodes is two slots, one guardian-patrol overlap and multiple patrol-patrol overlaps lead to multiple discoveries. Since removing any guardian or patrol results in discovery miss for some phase offset, we consider how to utilize such redundancy for lowering the average-case latency. Note that the stair-like placement of patrols similar to Searchlight may not be optimal either — it only reduces the discovery latency for the phase offset in the form of $m(c+2)$, where $m = 1, \cdots, n-1$. At present we have not found a good formulation of this problem. Alternatively, we adopt randomization in placing patrols — *each patrol is put at a randomly selected cycle (row) but at the same relative position (column) within the cycle*. After randomization, the worst-case latency under symmetric duty cycles remains the same since the relative offset between guardians of the nodes remains unchanged. Under asymmetric duty cycles, on the other hand, such randomization does not increase the worst-case bound, different from Searchlight. This is because no adjustment is made to guardians that provide a guarantee for asymmetric discovery.

IV. Evaluation

In this section, we evaluate the performance of Hello relative to earlier work through simulation studies. Due to the trade-off between duty cycle and discovery latency, we consider and compare only discovery latency of different protocols at given duty cycles. In particular, we are interested in the CDF (Cumulative Distribution Function) of discovery latency under symmetric and asymmetric duty cycles. In order for the results to become independent from a specific hardware platform, we use the number of slots of equal size to measure latency.

To make a comprehensive comparison, we included almost all well-known protocols such as Birthday protocol, Disco, U-Connect, and Searchlight. Note that Searchlight with randomized probing increases the worst-case bound under asymmetric duty cycles, while the variant with restricted randomized probing is rarely practical since it only works with two different duty cycles. Therefore we considered Searchlight with striped probing. For a given duty cycle d, the Birthday protocol sets its wake-up probability to d. U-Connect determines its selection of the prime p close to $\frac{3}{2}/d$. Although a duty cycle of d may not be compatible with Searchlight, we ignore such limitation for comparison purposes, and select $2/d$ as its cycle length. As for Disco, we use its best configuration: balanced primes (close to $2/d$) under symmetric duty cycles and unbalanced primes (one close to $1/d$ and another much larger) under asymmetric duty cycles.

We simulated three versions of Hello: the original design as introduced in Section III-A, the Hello with striped probing, and the Hello with striped probing and randomization. These are denoted by Hello, Hello-S, and Hello-SR, respectively. Under symmetric duty cycles, the cycle length c is set to $2/d$ according to Eqn. (4), and the number of cycles n is calculated according to Eqn. (1) for Hello and Eqn. (9) for Hello-S and Hello-SR. Under asymmetric duty cycles, c is selected as a prime slightly larger than $1/d$, and n is calculated by the same rule as in symmetric scenarios.

We evaluated these protocols through a state-based simulation. Specifically, we used the worst-case alignment strategy for each protocol — slots were aligned for Birthday protocol, Disco and U-Connect, while they were not aligned for Searchlight and Hello. To reflect the overall distribution of discovery latency, we ran Birthday protocol, Disco, U-Connect, Searchlight, Hello and Hello-S 10000 times, and collected discovery latency with random initial clock readings. Due to the probabilistic placement of patrols in Hello-SR, we created 100 random placement strategies for all nodes, and ran it 1000 times for each strategy.

A. Symmetric Discovery

We compare first the performance of different protocols under symmetric duty cycles. We selected two typical duty cycles, namely 5% and 1%, and ran individual protocols at each of the duty cycles separately. According to the parameter selection rules described above, Disco selected $(37, 43)$ and $(191, 211)$ (balanced primes), U-Connect 31 and 151, Searchlight 40 and 200, Hello $(41, 19)$ and $(199, 100)$, Hello-S and

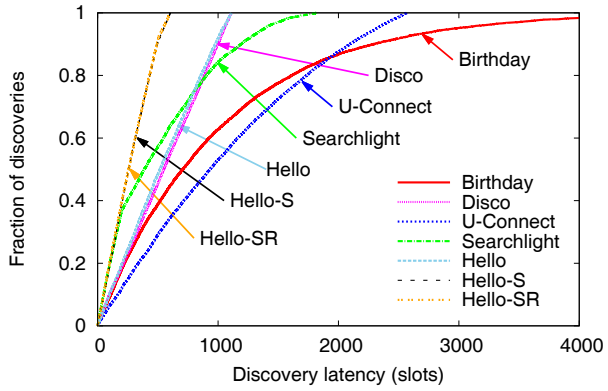


Fig. 9. CDF of discovery latency at the duty cycle of 10% and 1%.

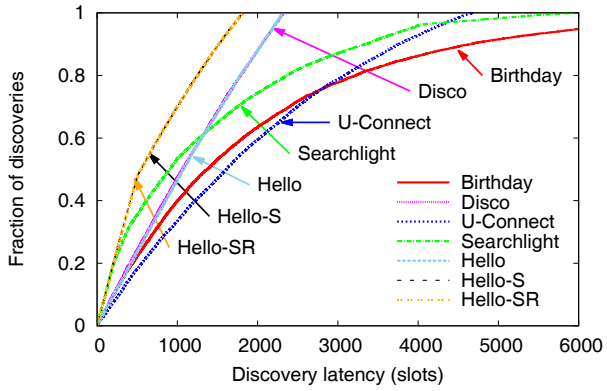


Fig. 10. CDF of discovery latency at the duty cycle of 5% and 1%.

Hello-SR $(41, 9)$ and $(199, 50)$, for the individual duty cycle respectively.

Fig. 7 and Fig. 8 show the CDF of the discovery latency at the two duty cycles, respectively. It can be observed that under both scenarios, *Hello-S and Hello-SR are comparable with the state-of-the-art Searchlight under symmetric duty cycles, while Hello-SR performs slightly better in terms of average-case latency*. Without striped probing, the original Hello still dominates U-Connect for selecting the optimal parameters. Disco exhibits large worst-case bound because of its large period length caused by redundancy in patrols. Similar to the results from earlier work, the Birthday protocol suffers from a long tail in discovery latency due to its probabilistic nature.

B. Asymmetric Discovery

Next we evaluate the performance of different protocols under asymmetric duty cycles. As reported by Searchlight, its performance varied under asymmetric duty cycles $(5\%, 1\%)$ and $(10\%, 1\%)$. We therefore include both scenarios for study. With a duty cycle of 10%, 5%, and 1%, Disco selected $(11, 101)$, $(23, 157)$ and $(101, 9973)$ (unbalanced primes), U-Connect 17, 31 and 151, Searchlight 20, 40 and 200, Hello $(11, 50)$, $(23, 73)$ and $(101, 5000)$, Hello-S and Hello-SR $(11, 30)$, $(23, 40)$ and $(101, 2500)$, respectively.

Fig. 9 depicts the CDF of discovery latency at the duty cycles of 10% and 1%. *Hello-S and Hello-SR perform the best all along, dominating Searchlight and all other protocols.* Their distribution curves coincide with each other because a discovery is usually achieved by an overlap of guardians. The original Hello is comparable with Disco using unbalanced primes. Searchlight excels the original Hello and Disco most of the time, but its worst-case latency is significantly larger[2]. All protocols, except the Birthday protocol, strictly dominate U-Connect at all times.

A similar conclusion can be drawn at the duty cycles of 5% and 1%, as illustrated in Fig. 10. The only exception is that U-Connect outperforms Searchlight in terms of worst-case discovery latency, the same observation as in [14].

[2]Searchlight reported the lowest latency in the absence of Disco with unbalanced primes.

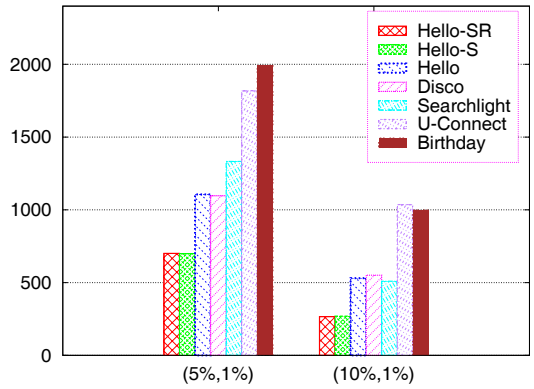


Fig. 11. Average discovery latency under asymmetric duty cycles of $(5\%, 1\%)$ and $(10\%, 1\%)$.

To characterize the extent to which Hello-S and Hello-SR improve over others, we plot the average latency for both scenarios in Fig. 11. On average, *Hello-S and Hello-R decrease the average latency by* 47% *compared with Searchlight in both scenarios, and by* 36% *and* 51% *compared with Disco at the two duty cycles, respectively*. They are also significantly (over 60%) better than U-Connect and Birthday protocol under both scenarios.

V. Implementation

To validate its feasibility and performance in the real world, we have implemented Hello on the TinyOS [25] platform[3] and Telos wireless sensor nodes [26]. In total eight sensor nodes are employed, with one particular node marked as the target node. The target node logged the discovery latency with the remaining nodes whenever the neighbor discovery process started. To accurately characterize the distribution of the latency regarding various phase offset, we deactivated the target node and restarted it with a random initial clock reading after all neighboring nodes were discovered.

According to the simulation results, we selected Disco and Searchlight for comparison with Hello-S under symmetric

[3]The implementation on the Android platform is still in progress.

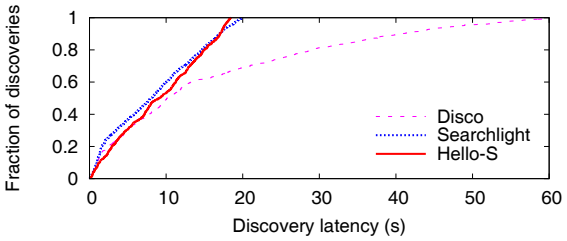


Fig. 12. Implementation: CDF of discovery latency at the duty cycle of 5%.

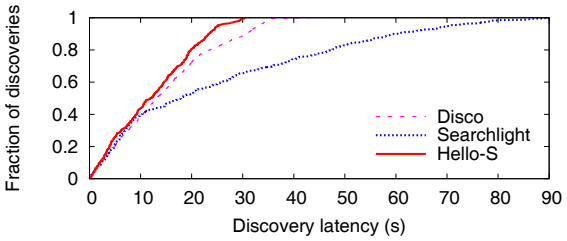


Fig. 13. Implementation: CDF of discovery latency at the duty cycle of 10% and 1%.

and asymmetric duty cycles. For symmetric duty cycles, we used a duty cycle of 5%, while for asymmetric duty cycles, we used that of 10% and 1% (the better combination for Searchlight). The parameters of each protocol at these duty cycles were determined as introduced in Section IV. From the sensor nodes, we observed that the transmission of a beacon containing node ID took around 1ms. To mitigate the impact of overflowing slots, we empirically set the slot length to 50ms (*i.e.*, $\delta = 2\%$). Each protocol was evaluated for an hour.

Fig. 12 and Fig. 13 depict the CDF of discovery latency under symmetric and asymmetric duty cycles, respectively. In general, *Hello-S demonstrates low latency under both scenarios, while Disco and Searchlight excel in one scenario only*. The worst-case latency under the duty cycle of 5% is bounded within 20s, and that under the asymmetric duty cycles of 10% and 1% is bounded within 30s. Moreover, Hello-S is two times better than Searchlight in terms of latency under the asymmetric duty cycle.

VI. Conclusion

We have presented Hello, a generic flexible protocol for neighbor discovery. It has been shown to be of high degree of freedom in parameter selection, high flexibility in adjusting itself to application demands, and high energy efficiency. As a generic framework, It incorporates existing deterministic protocols such as Quorum, Disco, U-Connect, and Searchlight. By exploring its parameter space, we have found optimal parameters for symmetric and asymmetric duty cycles. The results from the simulation and experiments show that Hello is highly competitive under both symmetric and asymmetric duty cycles. As future work, we will investigate its adoption in PMC applications and efficient neighbor maintenance after initial discovery.

Acknowledgment

This work is supported in part by the NSFC Major Program No. 61190110, National Basic Research Program of China (973) No. 2012CB316200, NSFC No. 61171067, 61133016, and 61272429, NSFC/RGC Joint Research Scheme No. 61361166009, and RFDP No. 20121018430.

References

[1] Y. Liu, X. Mao, Y. He, K. Liu, W. Gong, and J. Wang, "Citysee: not only a wireless sensor network," *IEEE Network*, vol. 27, no. 5, pp. 42–47, 2013.

[2] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse, "The smart thermostat: using occupancy sensors to save energy in homes," in *Proceedings of ACM SenSys*, 2010.

[3] H. Zhang, S. Tang, M. Li, and H. Ma, "Tracking and identifying burglar using collaborative sensor-camera networks," in *Proceedings of IEEE INFOCOM*, 2012.

[4] "PlayStation Vita," http://us.playstation.com/psvita/.

[5] "MOGA Mobile Gaming System," http://www.powera.com/Products/MOGA-Mobile-Gaming-System-Android.

[6] "Nextdoor," https://nextdoor.com/.

[7] "Hightlight," http://highlig.ht/.

[8] J. He, S. Ji, Y. Pan, and Y. Li, "Reliable and energy efficient target coverage for wireless sensor networks," *Tsinghua Science and Technology*, vol. 16, no. 5, pp. 464–474, 2011.

[9] T. Liu, C. M. Sadler, P. Zhang, and M. Martonosi, "Implementing software on resource-constrained mobile sensors: experiences with impala and zebranet," in *Proceedings of ACM MobiSys*, 2004.

[10] D. Li and P. Sinha, "RBTP: Low power mobile discovery protocol through recursive binary time partitioning," *IEEE Transactions on Mobile Computing*, published online, 2012.

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

[12] P. Dutta and D. Culler, "Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications," in *Proceedings of ACM SenSys*, 2008.

[13] A. Kandhalu, K. Lakshmanan, and R. R. Rajkumar, "U-Connect: a low-latency energy-efficient asynchronous neighbor discovery protocol," in *Proceedings of ACM/IEEE IPSN*, 2010.

[14] M. Bakht, M. Trower, and R. H. Kravets, "Searchlight: won't you be my neighbor?" in *Proceedings of ACM MobiCom*, 2012.

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

[16] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of ACM SenSys*, 2004.

[17] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proceedings of IEEE INFOCOM*, 2002.

[18] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks," in *Proceedings of IEEE INFOCOM*, 2002.

[19] S. Lai, B. Ravindran, and H. Cho, "Heterogenous quorum-based wake-up scheduling in wireless sensor networks," *IEEE Transactions on Computers*, vol. 59, no. 11, pp. 1562 –1575, 2010.

[20] R. Zheng, J. C. Hou, and L. Sha, "Asynchronous wakeup for ad hoc networks," in *Proceedings of ACM MobiHoc*, 2003.

[21] I. Anderson, *Combinatorial designs and tournaments*. Oxford University Press, 1998.

[22] I. Niven, H. Zuckerman, and H. Montgomery, *An introduction to the theory of numbers*. John Wiley & Sons, 2008.

[23] A. Purohit, B. Priyantha, and J. Liu, "WiFlock: Collaborative group discovery and maintenance in mobile sensor networks," in *Proceedings of ACM/IEEE IPSN*, 2011.

[24] K. H. Rosen, *Discrete mathematics and its applications*. McGraw-Hill, 2012.

[25] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *Proceedings of ACM ASPLOS*, 2000.

[26] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proceedings of ACM/IEEE IPSN*, 2005.