Ad Hoc Networks 77 (2018) 54–68

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

On heterogeneous duty cycles for neighbor discovery in wireless sensor networks

Lin Chen[a,b], Ruolin Fan[c], Yangbin Zhang[a], Shuyu Shi[d], Kaigui Bian[a,*], Lin Chen[e], Pan Zhou[f], Mario Gerla[c], Tao Wang[a], Xiaoming Li[a]

[a] Peking University, China
[b] Yale University, USA
[c] University of California, Los Angeles, USA
[d] National Institute of Informatics, Japan
[e] University of Paris-Sud, France
[f] Huazhong University of Science and Technology, China

ARTICLE INFO

Article history:
Received 10 November 2017
Revised 5 April 2018
Accepted 16 April 2018
Available online 27 April 2018

Keywords:
Neighbor discovery
Heterogeneous duty cycles
Wireless sensor networks

ABSTRACT

Neighbor discovery plays a crucial role in the formation of wireless sensor networks and mobile networks where the power of sensors (or mobile devices) is constrained. Due to the difficulty of clock synchronization, many asynchronous protocols based on wake-up scheduling have been developed over the years in order to enable timely neighbor discovery between neighboring sensors while saving energy. However, existing protocols are not fine-grained enough to support all *heterogeneous* battery duty cycles, which can lead to a more rapid deterioration of long-term battery health for those without support. Existing research can be broadly divided into two categories according to their neighbor-discovery techniques—the quorum-based protocols and the co-primality based protocols. In this paper, we propose two neighbor discovery protocols, called *Hedis* and *Todis*, that control the duty cycle granularity of quorum and co-primality based protocols respectively, by enabling the finest-grained control of heterogeneous duty cycles. We compare the two optimal protocols via analytical and simulation results, which show that the optimal co-primality based protocol (Todis) is not only simpler in its design, but also has a better performance.

1. Introduction

As human technology continues to advance at an unprecedented rate, there are more mobile wireless devices in operation than ever before. Many have taken advantage of the ubiquity of these devices to create mobile social network applications that use mobile sensing as an important feature [1,2]. These applications rely on their devices' capability to opportunistically form decentralized networks as needed. For this to happen, it is important for these devices to be able to discover one another to establish a communication link. In order to save energy, each of the devices alternates between active and sleeping states by keeping its radio "ON" for only some of the time [3]. This is challenging to achieve because two neighboring nodes have the opportunities of discovering each other only when both of their radios are "ON" at the same time; and with clock drifts, having set times for all the nodes to wake up at the same time is not trivial. Since clock synchronization is difficult in a distributed system, neighbor discovery must be done asynchronously. Over the years, the asynchronous neighbor discovery problem has been widely studied [4–13], and existing research mainly focused on satisfying the following three design requirements:

1. Guarantee neighbor discovery within a reasonable time frame;
2. Minimize the number of time slots for which the node is awake to save energy;
3. Match the nodes' wake-up schedules with their heterogeneous battery duty cycles[1] as closely as possible (i.e. finer duty cycle granularity).

Most existing solutions to this problem use patterned wake-up schedules to satisfy the first two requirements. We classify these solutions into two broad categories: (1) *quorum* based protocols that arrange the radio's time slots into a matrix and pick wake-

* Corresponding author.
E-mail address: bkg@pku.edu.cn (K. Bian).

[1] Duty cycle is the percentage of one period in which a sensor/radio is active.

https://doi.org/10.1016/j.adhoc.2018.04.007
1570-8705/

up times according to quorums in the matrix; and (2) *co-primality* based protocols that use number theory to choose numbered time slots as the radio's wake-up times.

In a quorum-based protocol, a node populates time slots into a matrix, where the elements in the matrix represent time slots the node takes to run a period of the wake-up schedule [14]. The specific arrangements of rows and columns depend upon the protocol scheme, which typically assign slots as "active" or "sleeping", such that it will ensure these chosen active time slots in the matrix of one node will overlap with those active ones of a neighboring node. Especially, when nodes have the same duty cycles, two nodes choosing active times from a row and a column respectively in the matrix will be ensured to achieve neighbor discovery regardless of clock drifts.

A co-primality based protocol directly takes advantage of properties of the Chinese remainder theorem (CRT) [15] to ensure that any two nodes would both be active in the same time slot [6]. Under these protocols, nodes wake up at time slots in multiples of chosen numbers (a.k.a. protocol parameters) that are co-prime to one another. Such a neighbor discovery protocol fails when nodes choose the same number that would compromise the co-primality. Thus, every node is allowed to choose several numbers and wake up at multiples of all of those chosen numbers, which guarantees that nodes discover one another within a bounded time/delay.

Up to now, all of the protocols incepted, be it quorum-based or co-primality based, fail to meet the third design requirement, as their requirements for duty cycles are too specific. As a quorum-based protocol, Searchlight [4] requires that the duty cycles be in the form $\frac{2}{n^i}$, where n is a fixed integer and $i = 1, 2, 3, \ldots$ (it only supports duty cycles of $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \ldots$ if $n = 2$). Therefore, it greatly restricts the choices of supported duty cycles due to the requirement for duty cycles to be in the form $\frac{2}{n^i}$. For a co-primality based protocol like Disco [6], it restricts duty cycles to be in the form $\frac{1}{p_1} + \frac{1}{p_2}$, where p_1 and p_2 are prime numbers. Such stringent requirements on duty cycles force devices to operate at duty cycles that they are not designed to operate at, thus shortening their battery longevity.

In this paper, we present two fine-grained neighbor discovery protocols, called *Hedis* (*heterogeneous discovery* as a quorum-based protocol) and *Todis* (*triple-odd based discovery* as a co-primality based protocol), that guarantee asynchronous neighbor discovery in a heterogeneous environment, meaning that each node could operate at a different duty cycle. We analytically compare these two protocols with existing state-of-the-art protocols to confirm their fine granularity in the support of duty cycles, and also compare them against each other as a comparison between the two general categories of neighbor discovery protocols (quorum vs. co-primality based protocols).

The rest of this paper is organized as follows. We formally define the problem as well as any necessary terms in Section 2, and give a taxonomy of current research efforts in this area in Section 3. In Sections 4 and 5, we present our optimizations for the quorum-based and co-primality based protocols respectively, and we evaluate them with simulations in Section 6. Finally, we conclude with Section 8.

2. Problem formulation

Here we define the terms and variables used to formally describe the neighbor discovery problem and its solution; and meanwhile we state the assumptions used in devising our protocols.

Wake-up schedule: We consider a time-slotted wireless sensor network where each node is energy-constrained. The nodes follow a *neighbor discovery wake-up schedule* that defines the time pattern of when they need to wake up (or sleep), so that they can discover their respective neighbors in an energy-efficient manner.

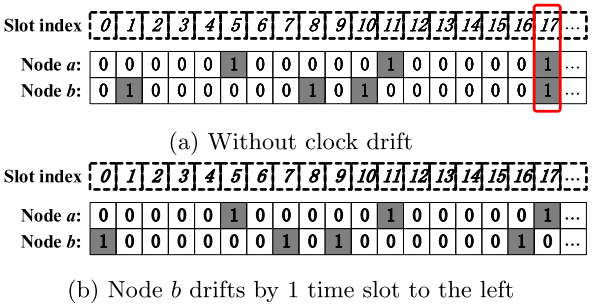


Fig. 1. An example of neighbor discovery: two neighbor discovery schedules are $\mathbf{s}_a = \{0, 0, 0, 0, 0, 1\}$ and $\mathbf{s}_b = \{0, 1, 0, 0, 0, 0, 0, 0, 1\}$. Without clock drift (a), the two nodes can discover each other every 18 time slots since $\text{lcm}(T_a, T_b) = 18$. With clock drift (b), neighbor discovery fails.

Definition 1. The neighbor discovery *schedule* (or simply schedule) of a node a is a sequence $\mathbf{s}_a \triangleq \{s_a^t\}_{0 \le t < T_a}$ of period T_a and

$$s_a^t = \begin{cases} 0 & a \text{ sleeps in slot } t \\ 1 & a \text{ wakes up in slot } t \end{cases}$$

We do not assume clock synchronization among nodes, therefore any two given nodes may have random clock drifts. We use the cyclic rotation of a neighbor discovery schedule to describe this phenomenon. For example, a clock drift by k slots of node a's schedule s_a is

$$rotate(\mathbf{s}_a, k) = \{r_a^t\}_{0 \le t < T_a},$$

where $r_a^t = s_a^{(t+k) \bmod T_a}$.

Definition 2. The *duty cycle* δ_a of node a is the percentage of time slots in one period of the wake-up schedule where node a is active (node a wakes up), defined as

$$\delta_a = \frac{|\{0 \le t < T_a : s_a^t = 1\}|}{T_a}.$$

For example, a node that wakes up on average in one slot for every 2 time slots has a duty cycle of 50%.

The importance of duty cycle matching: In a wireless mobile sensor network, each sensor node may have a different duty cycle due to various factors. By adjusting the duty cycles of a sensor, one is able to exploit the tradeoff between conserving battery power and packet forwarding capacity. A smaller duty cycle consumes less power because the radio is powered on for less of the time; however, because the radio is off for so long, the node cannot spend as much time transmitting packets, causing high end-to-end delays. On the other hand, as the duty cycle increases, the radio is powered on more frequently, thus mitigating delays while using up more battery power. Due to the ever-changing network conditions (periods of high and low traffic rates) and each sensor node's power status, the notion of having dynamic duty cycles is now an area of active research [16–18]. Thus, a neighbor discovery protocol must support duty cycles at a fine granularity in order for these new dynamic duty cycled schemes to come into fruition.

Neighbor discovery: Suppose two nodes a and b have schedules $\mathbf{s}_a$ and $\mathbf{s}_b$ of periods T_a and T_b, respectively. If $\exists t \in [0, \text{lcm}(T_a, T_b))$ such that $s_a^t = s_b^t = 1$ where $\text{lcm}(T_a, T_b)$ is the least common multiple of T_a and T_b, we say that:

- Nodes a and b can discover each other in slot t.
- Slot t is called a *discovery slot* between a and b.

Fig. 1 shows an example of two sensor nodes with neighbor discovery schedules $\mathbf{s}_a = \{0, 0, 0, 0, 0, 1\}$ and $\mathbf{s}_b =$

$\{0, 1, 0, 0, 0, 0, 0, 0, 1\}$, that have period lengths of $T_a = 6$ and $T_b = 9$ respectively. Node a is active on 1 slot within each period (6 slots) while node b is active on 2 slots within each period (9 slots). Thus the duty cycles of a and b are $d_a = \frac{1}{6} \approx 16.7\%$ and $d_b = \frac{2}{9} \approx 22.2\%$. In Fig. 1a, we see that for every period of 18 slots ($\text{lcm}(T_a, T_b) = 18$), nodes a and b discover each other in slot 17. However, as illustrated in Fig. 1b, when a one-slot clock drift occurs in node b, we have $rotate(\mathbf{s}_b, 1) = \{1, 0, 0, 0, 0, 0, 0, 1, 0\}$ and these two nodes can no longer discover each other.

Duty cycle granularity: Duty cycle is the percentage of one period in which a sensor/radio is active. Duty cycle granularity measures how small the duty cycle can be supported by a neighbor discovery protocol.

In a practical implementation, when two nodes are active in the same time slot, if both in reception mode, no discovery will happen; on the other hand, if both in transmission mode, collision may happen. Collision resolution mechanisms have been well studied [19,20]. If one is in reception mode while the other is in transmission mode, discovery happens.

3. A taxonomy of neighbor discovery protocols

In this section, we introduce a taxonomy of deterministic asynchronous neighbor discovery protocols. Through examining existing solutions to the neighbor discovery problem, we divide these protocols into two broad categories.

3.1. Why deterministic protocols

Many solutions have been proposed to solve the neighbor discovery problem. One of the earliest such solutions are the birthday protocols [21], which take upon a *probabilistic* approach to neighbor discovery. These protocols rely on the *birthday paradox*, which states that with as few as 23 people, the probability that two people have the same birthday exceeds $\frac{1}{2}$. As a non-deterministic protocol based upon probability, birthday protocols are heterogeneous and supports every duty cycle with the finest granularity. Following this, many more similar probabilistic protocols were also developed [22–25]. However, due to their probabilistic nature, these protocols fail to provide a guaranteed upper bound for neighbor discovery latency, which means that there is a chance for two neighbors to never discover each other.

To combat this insufficiency, *deterministic* protocols with worst case bounds for neighbor discovery were developed. The earlier deterministic protocols such as [14,26], and [27] all use the quorum concept. However, while these protocols are effective in guaranteeing neighbor discovery, they are generally lacking in duty cycle support. For example, [14] and [26] are homogeneous, meaning that they require all the nodes to have the same duty cycle. As a result, the co-primality based approach was developed with Disco [6] and U-Connect [7], although U-Connect is in some ways a hybrid approach using elements from both the quorum and co-primality paradigms.

3.2. Quorum vs. co-primality based protocols

The deterministic protocols for neighbor discovery can be largely classified into two major categories, *quorum* based protocols and *co-primality* based protocols.

3.2.1. Quorum-based protocols

Quorum-based protocols take advantage of geometry in a 2-dimensional array.

Bounded discovery delay: In the most original protocols like [14], time is arranged into an $m \times m$ matrix. Every node then chooses a row and a column for which to wake up. This ensures that regardless of any clock drifts, any two nodes would be able to wake up at the same time slot every m^2 time slots, thus guaranteeing an upper bound for neighbor discovery. However, this method only works if every node happens to use the same duty cycle. Lai et al. [27] improve upon this method by constructing *cyclic quorum system* and *grid quorum system* pairs, which allow for two different duty cycles to coexist and still ensure bounded neighbor discovery.

Example protocols: The current latest development in quorum-based protocols is Searchlight [4], which is able to support multiple duty cycles in the network. Searchlight essentially divides the duty cycle period into a $\frac{t}{2} \times t$ matrix, and uses a combination of *anchor* and *probing* slots to generate wake-up patterns. At the beginning of every t time slots is an anchor slot, and a probing slot occurs at random slots between the anchor slots. With this technique, Searchlight [4] shows that it is able to allow neighbor discovery among nodes with many different duty cycles.

3.2.2. Co-primality based protocols

A co-primality based neighbor discovery protocol is one in which

- Each node, say, node a, chooses a set of integers (not necessarily distinct)

  $N_a = \{n_1^a, n_2^a, n_3^a, \ldots, n_{|N_a|}^a\}$.
- For two distinct nodes a and b, N_a and N_b must satisfy the following *co-prime pair property*.

Definition 3. For two distinct nodes a and b under a co-primality based neighbor discovery protocol, there exists an integer in N_a that is co-prime to an integer in N_b—i.e., $\exists n_{i_0}^a \in N_a$ and $n_{j_0}^b \in N_b$ such that $n_{i_0}^a$ and $n_{j_0}^a$ are co-prime.

Node a's schedule $\mathbf{s}_a \triangleq \{s_a^t\}_{0 \le t < T_a}$ under this co-primality based protocol is

$$s_a^t = \begin{cases} 1 & t \text{ is divisible by some} n_i^a \in N_a \\ 0 & \text{otherwise} \end{cases}$$

The period length is $T_a = \text{lcm}(n_1^a, n_2^a, \ldots, n_{|N_a|}^a)$ and its duty cycle δ_a is

$$\delta_a = \sum_{1 \le i_1 \le |N_a|} \frac{1}{n_{i_1}^a} - \sum_{1 \le i_1 < i_2 \le |N_a|} \frac{1}{\text{lcm}(n_{i_1}^a, n_{i_2}^a)} \cdots + (-1)^{|N_a|+1} \frac{1}{\text{lcm}(n_1^a, n_2^a, n_3^a, \cdots, n_{|N_a|}^a)}.$$

Bounded discovery delay: By the Chinese remainder theorem (CRT) [15], we can obtain the following theorem.

Theorem 1. *A co-primality based neighbor discovery protocol can guarantee discovery for any two nodes for any amount clock drift if the associated integer sets of the nodes in this network satisfy the co-prime pair property. And the worst-case discovery delay is bounded by the product of the two smallest co-prime numbers, one from each set, i.e.:*

$$\min_{\gcd(n_i^a, n_j^b) = 1, 1 \le i \le N_a, 1 \le j \le N_b} \{n_i^a \cdot n_j^b\}.$$

Suppose the clock of node a is d time slots ahead of that of node b, i.e., node b's t^{th} time slot is the $(t+d)^{th}$ time slot of node a, where d is the clock drift, the following congruence system w.r.t. t applies:

$$\begin{cases} t \equiv 0 \pmod{n_i^a} & \text{for some } i = 1, 2, 3 \ldots, N_a \\ t + d \equiv 0 \pmod{n_j^b} & \text{for some } j = 1, 2, 3, \ldots, N_b \end{cases} \quad (1)$$

If t is a solution to Eq. (1), then node a will discover node b in node a's t-th time slot (i.e., node b's $(t+d)$-th time slot). By CRT, if n_i^a

and n_j^b are co-prime, there exists a solution $t \equiv t_0 \pmod{n_i^a n_j^b}$; i.e., the worst-case discovery delay is bounded by $n_i^a n_j^b$. The co-prime pair property guarantees the existence of such co-prime n_i^a and n_j^b. Thus the worst-case discovery delay is bounded by the minimum of the product of co-prime n_i^a and n_j^b, i.e.,

$$\min_{\gcd(n_i^a, n_j^b)=1, 1\le i\le N_a, 1\le j\le N_b} \{n_i^a \cdot n_j^b\}.$$

Example protocols: Disco [6], as such a co-primality based protocol, ensures co-primality by only using prime numbers as possible parameters. In Disco, each node chooses two distinct primes to create its wake-up schedule. For example, node a chooses two distinct primes p_1 and p_2 and node b chooses p_3 and p_4. Node a is active (wakes up) in the t-th time slot iff t is divisible by either p_1 or p_2 while node b is active in the t-th time slot iff t is divisible by either p_3 or p_4. Therefore, Disco can guarantee neighbor discovery for any two nodes for any amount of clock drift with a bounded discovery delay of

$$\min_{\gcd(p_i, p_j)=1, i=1,2, j=3,4} \{p_i \cdot p_j\}.$$

Again, this delay is the product of the two smallest co-prime numbers following from the CRT.

U-Connect [7] is a combination of Disco and the basic quorum-based protocol in that it restricts the dimensions of the square quorum matrix to be a prime number. In this way, if the duty cycles of the nodes happen to be the same, neighbors would discover one another via the quorum method. On the other hand, if they are different, the numbers chosen would be co-prime to each other and thus enabling neighbor discovery by Theorem 1.

More comprehensive surveys on neighbor discovery can be found in [28] and [29].

4. Hedis: optimizing quorum-based protocols

Hedis is an asynchronous periodic slot-based neighbor discovery protocol where each node picks its anchor and probing slots according to the elements of a quorum that is carefully selected in an $(n-1)$ by n matrix.

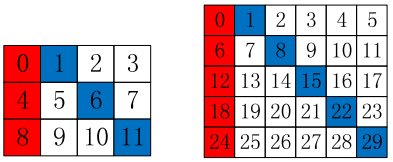
4.1. Design of the hedis schedule

For a node a that has a desired duty cycle δ, the period of its schedule under Hedis, $\mathbf{s}_a = \{s_a^t\}_{0\le t<n(n-1)}$, consists of $n(n-1)$ time slots, where the integer n is chosen such that $\frac{2}{n}$ comes as close to δ as possible (and we call n the *parameter* of this node). Under Hedis, its schedule is

$$s_a^t = \begin{cases} 1 & t = ni, (n+1)i+1 (i = 0, 1, 2, \ldots, n-2) \\ 0 & \text{otherwise} \end{cases},$$

where ni $(i = 0, 1, 2, \ldots, n-2)$ denotes the index of an *anchor* slot and $(n+1)i+1$ denotes the index of a *probing* slot. We use the color grid and white grid to represent $s_a^t = 1$ and $s_a^t = 0$ respectively in the rest of our paper.

Fig. 2 shows two example Hedis schedules when $n = 4, 6$, and the two schedules consist of $n(n-1) = 12, 30$ time slots, respectively. Each grid in the figure represents a time slot, and the integer inside a grid denotes its slot index, e.g., the grid with 0 inside denotes the 0th time slot in the schedule (note that a schedule starts from the 0th time slot). The red and blue slots represent the anchor and probing slots, during which the node wakes up. When $n = 4$, the duty cycle is $2/4 = 50\%$. The full schedules are depicted in Fig. 3, where the two nodes with different duty cycles can achieve successful neighbor discovery (overlap of colored slots between schedules of nodes a and b) for many times in every period. Next, we will show that Hedis can guarantee neighbor discovery for any two nodes of same-parity parameters (both odd or both even) with heterogeneous duty cycles for any amount of clock drift.



(a) $n = 4$ (b) $n = 6$

Fig. 2. Two example Hedis schedules.

4.2. Bounded discovery delay under hedis

We need a lemma first, as presented below. This lemma is reproduced from Theorem 2.9 on page 61 in [15].

Lemma 2. *Let m and n be positive integers. For any integers a and b, there exists an integer x such that*

$$x \equiv a \pmod{m} \tag{2}$$

and

$$x \equiv b \pmod{n} \tag{3}$$

if and only if

$$a \equiv b \pmod{\gcd(m, n)}.$$

If x is a solution of congruences (2) and (3), then the integer y is also a solution if and only if

$$x \equiv y \pmod{\mathrm{lcm}(m, n)}.$$

By Lemma 2, we further establish the following theorem.

Theorem 3. *Hedis guarantees neighbor discovery within bounded latency for any two nodes with the same-parity parameters n and m, given any amount of clock drift between their schedules. The average discovery latency is $O(nm)$.*

Proof. Nodes a and b are two arbitrarily given nodes, whose parameters are n and m, respectively. The periods of the Hedis schedules of nodes a and b are $T_a = n(n-1)$ and $T_b = m(m-1)$, respectively. We use d to denote the clock drift.

Without loss of generality, we study the following system of congruences with respect to t:

$$\begin{cases} t \equiv ni + d, (n+1)i + 1 + d \pmod{n(n-1)} \\ t \equiv mj, (m+1)j + 1 \pmod{m(m-1)}, \end{cases} \tag{4}$$

where $i \in [0, n-2]$, $j \in [0, m-2]$.

$t \equiv ni + d, (n+1)i + 1 + d \pmod{n(n-1)}$

$(i \in [0, n-2])$ is true iff $\exists i \in [0, n-2]$ such that it is true, and the same meaning for

$t \equiv mj, (m+1)j + 1 \pmod{m(m-1)}$

$(j \in [0, m-2])$.

There are a number of nm pairs of simultaneous congruences, which we divide into 4 groups: anchor-anchor, anchor-probing, probing-anchor and probing-probing groups. E.g., the anchor-probing group denotes the case where an anchor slot of node a overlaps a probing slot of node b. Note that if we find a solution that meets the requirements of any one of these congruences, we obtain a solution to Eq. (4).

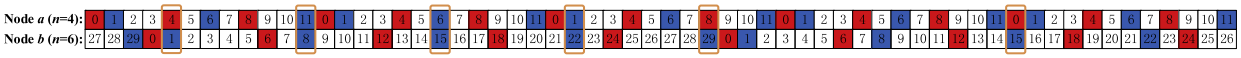


Fig. 3. Node a's schedule is 3-slot ahead of node b's schedule. The overlapped colored slots between their schedules represent the successful neighbor discovery.

Group 1: anchor-anchor: Consider the following system of congruences

$$\begin{cases} t \equiv ni + d \pmod{n(n-1)} & i \in [0, n-2] \\ t \equiv mj \pmod{m(m-1)} & j \in [0, m-2] \end{cases},$$

which is equivalent to

$$\begin{cases} t \equiv d \pmod{n} \\ t \equiv 0 \pmod{m} \end{cases} \tag{5}$$

By Lemma 2, Eq. (5) has a solution if and only if

$$\gcd(n, m) \mid d.$$

Group 2: anchor-probing: Consider the following system of congruences

$$\begin{cases} t \equiv ni + d \pmod{n(n-1)} & i \in [0, n-2] \\ t \equiv (m+1)j + 1 \pmod{m(m-1)} & j \in [0, m-2] \end{cases}, \tag{6}$$

which is equivalent to

$$\begin{cases} t \equiv d \pmod{n} \\ t \equiv (m+1)j + 1 \pmod{m(m-1)} & j \in [0, m-2] \end{cases} \tag{7}$$

By Lemma 2, Eq. (7) has a solution if and only if $\gcd(n, m(m-1)) \mid (m+1)j + 1 - d$ for some integer $j \in [0, m-2]$, i.e., the congruence with respect to j

$$(m+1)j \equiv d - 1 \pmod{\gcd(n, m(m-1))} \tag{8}$$

has a solution.

Group 3: probing-anchor: Consider the following system of congruences

$$\begin{cases} t \equiv (n+1)i + 1 + d \pmod{n(n-1)} & i \in [0, n-2] \\ t \equiv mj \pmod{m(m-1)} & j \in [0, m-2] \end{cases}, \tag{9}$$

which is equivalent to

$$\begin{cases} t \equiv (n+1)i + 1 + d \pmod{n(n-1)} & i \in [0, n-2] \\ t \equiv 0 \pmod{m} \end{cases} \tag{10}$$

By Lemma 2, Eq. 10 has a solution if and only if

$$\gcd(m, n(n-1)) \mid (n+1)i + 1 + d$$

for some integer $i \in [0, n-2]$, i.e., the congruence with respect to i

$$(n+1)i \equiv -d - 1 \pmod{\gcd(m, n(n-1))}$$

has a solution.

Group 4: probing-probing: Consider the following system of congruences

$$\begin{cases} t \equiv (n+1)i + 1 + d \pmod{n(n-1)} & i \in [0, n-2] \\ t \equiv (m+1)j + 1 \pmod{m(m-1)} & j \in [0, m-2] \end{cases} \tag{11}$$

By Lemma 2, Eq. 11 has a solution if and only if

$$\gcd(n(n-1), m(m-1)) \mid (n+1)i - (m+1)j + d$$

for some integer $i \in [0, n-2]$ and $j \in [0, m-2]$.

Now we begin to prove this theorem by cases.

Case 1: If $m > n$, the congruence system of anchor-probing (Group 2) is true. *Proof:* If $m > n$, we have $m - 1 \geq n \geq \gcd(n, m(m-1))$. And note that $\gcd(m+1, \gcd(n, m(m-1))) = \gcd(m+1, n, m(m-1)) = \gcd(m+1, 2, n) = 1$. This is because m and n are both odd or are both even. So one of $m+1$ and n are odd, and we have $\gcd(m+1, 2, n) = 1$. Therefore $(m+1)j$ $(j \in [0, m-2])$ runs over all congruence classes modulo $\gcd(n, m(m-1))$. Then Eq. (8) has at least $\lfloor (m-1)/\gcd(n, m(m-1)) \rfloor$ solutions and on average $(m-1)/\gcd(n, m(m-1))$ solutions. Hence Eq. (7) has at least $\lfloor (m-1)/\gcd(n, m(m-1)) \rfloor$ solutions and on average $(m-1)/\gcd(n, m(m-1))$ solutions modulo $\operatorname{lcm}(n, m(m-1))$. Therefore, the average discovery latency is $\frac{\operatorname{lcm}(n, m(m-1))}{(m-1)/\gcd(n, m(m-1))} = nm$.

Case 2: If $n > m$, the congruence system of probing-anchor (Group 3) is true. *Proof:* If $n > m$, similarly to case 1, we have Eq. (10) has at least $\lfloor (n-1)/\gcd(m, n(n-1)) \rfloor$ solutions and on average $(n-1)/\gcd(m, n(n-1))$. Hence Eq. (9) has at least $\lfloor (n-1)/\gcd(m, n(n-1)) \rfloor$ solutions and on average $(n-1)/\gcd(m, n(n-1))$ modulo $\operatorname{lcm}(m, n(n-1))$. Therefore, the average discovery latency is nm.

Case 3: If $n = m$, we consider the result of $d \bmod n$. If $d \equiv 0 \pmod{n}$, then $\gcd(n, m) = n \mid d$, and thus the anchor-anchor case (Group 1) is true and the average discovery latency is $O(nm)$. Now we concentrate on the case where $d \not\equiv 0 \pmod{n}$. Since $n = m$, Eq. (8) becomes

$$(n+1)j \equiv d - 1 \pmod{n}.$$

Since $(n+1)j = nj + j \equiv j \pmod{n}$, this is equivalent to

$$d \equiv j + 1 \pmod{n}.$$

For $j \in [0, n-2]$, $j+1$ runs over $[1, n-1]$. Because $d \not\equiv 0 \pmod{n}$, there exists a $j \in [0, n-2]$ that satisfies Eq. (8), and therefore the anchor-probing case (Group 2) is true. Similarly, the probing-anchor case (Group 3) is also true. And it is easy to check that the average discovery latency is $O(n^2)$, i.e., $O(nm)$. □

5. Todis: optimizing co-primality based protocols

Now we optimize the asynchronous co-primality based protocols, and propose Todis that exploits properties of consecutive odd integers for achieving co-primality.

As a co-primality based protocol, Todis creates wake-up schedules for the nodes based on multiples of numbers that are co-prime to each other. This ensures that any two given nodes would be able to wake up at the same time by the co-prime pair property as illustrated in Section 3, thus succeeding in neighbor discovery. Recall that Disco [6] guarantees this by simply using prime numbers as parameters, which limits the variety of parameters to choose from.

For two nodes a and b, we need to construct two sets of integers, N_a and N_b, that must satisfy the co-prime pair property. In our quest to find co-prime pairs, we observe that for two numbers to be co-prime, at least one of them must be odd. Thus, we explore the possibility of achieving co-primality using odd integers. We observe that given two odd integers a and b, if they are not co-prime, often times either "$a+2$ and b", or "a and $b+2$" is a co-prime pair. For example, if 15 and 21 are not co-prime, we are able to find that either "17 and 21", or "15 and 23" is a co-prime pair. Following this logic, we design our Todis protocol using sets of consecutive odd integers.

5.1. Design of the Todis schedule

Ideally, we want to construct a co-primality based protocol that requires the co-prime pair property (Definition 3) for the set of in-

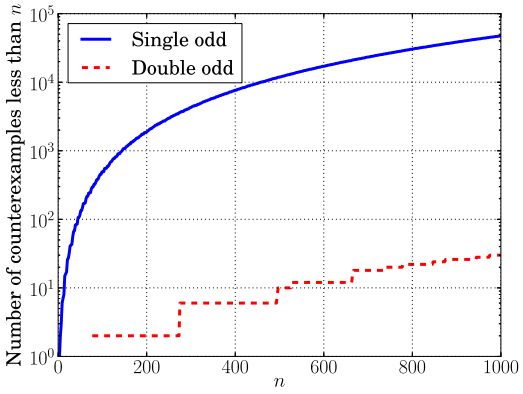


Fig. 4. The number of counterexamples less than n for the single-odd and double-odd scenarios. A single-odd instance is said to be less than n if the single odd number per se is less than n. A double-odd instance is said to be less than n if the two consecutive odd numbers are both less than n. An instance is called a counterexample if it does not satisfy the co-prime pair property (Definition 3). Note that the single-odd case results in a large number of counterexamples while the double-odd case leads to 30 counterexamples less than $n = 1000$. It is noteworthy that there is no counterexample for the triple- and quadruple-odd cases.

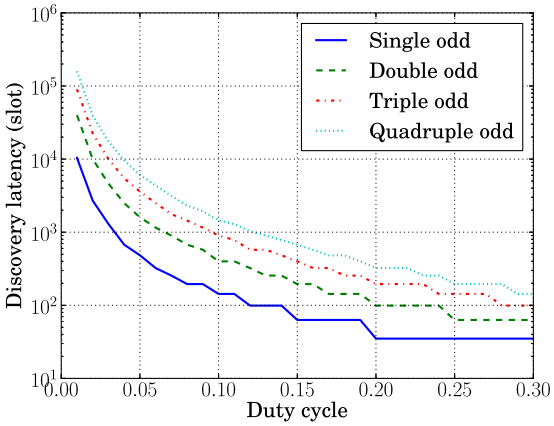


Fig. 5. The discovery delay (in time slots) for different duty cycles, under single-, double-, triple-odd and quadruple-odd scenarios. We observe that using more odd numbers in a set results in a larger discovery latency.

tegers associated with each node. In this subsection, we consider a family of candidate protocols termed the family of "n-tuple-odd protocols", including single-, double-, triple, quadruple-odd protocols, etc.

Recall that a co-primality based protocol associates each node, say, node a, with a set of integers, denoted by $N_a = \{n_1^a, n_2^a, n_3^a, \ldots, n_{|N_a|}^a\}$ (Section 3.2.2). An n-tuple-odd protocol associates each node with a set of n consecutive odd integers, e.g., {13, 15} is an instance under the double-odd protocol, {3, 5, 6} is an instance under the triple-odd protocol, and {11, 13, 15, 17} is an instance for the quadruple-odd protocol.

5.1.1. Co-prime pair property

We need to check whether an n-tuple-odd protocol satisfies the co-prime pair property.

Trying the single-odd protocol: First, we tried the single-odd protocol in which each node is assigned a single odd integer. However, there are a large number of odd integer pairs that are not co-prime: e.g., 3 and 9 are not co-prime, neither are 5 and 15. We call two odd integers a counterexample of the single-odd protocol if the two odd integers are not co-prime.

A counterexample of the single-odd protocol is said to be less than n if both odd numbers are less than n. Generally, a counterexample of the n-tuple-odd protocol is two sets of n consecutive odd numbers that violate the co-prime pair property. A counterexample of the n-tuple-odd protocol is said to be less than n if all odd integers in this counterexample is less than n.

Fig. 4 shows the number of counterexamples of the single-odd protocol under n when n varies from 0 to 1000. We can observe that the single-odd protocol results in a huge amount of counterexamples. There are approximately 4×10^4 counterexamples under 1000. Clearly, the single-odd protocol violates the co-prime pair property, and it cannot be used for creating the neighbor discovery wake-up schedules.

Trying the double-odd protocol: Then, we tried using two consecutive odd integers in N_a for each node a, i.e., $N_a = \{n, n+2\}$ where $n \geq 1$ and n is odd. Unfortunately, for given nodes a and b, there are many instances where the sets N_a and N_b do not satisfy the co-prime pair property for very small numbers (i.e., less than 100). For example, when $N_a = \{33, 35\}$ and $N_b = \{75, 77\}$, $\forall n_i^a \in N_a, n_j^b \in N_b$, we have $\gcd(n_i^a, n_j^b) > 1$.

Fig. 4 shows the number of counterexamples for the double-odd protocol under n. There are approximately 30 counterexamples under 1000. As aforementioned, the smallest counterexample is $N_a = \{33, 35\}$ and $N_b = \{75, 77\}$. The double-odd protocol fails to satisfy the co-prime pair property, which is also impractical for creating the neighbor discovery wake-up schedules.

Trying the triple- and quadruple-odd protocols: Different from the single- and double-odd protocols, triple- and quadruple-, as well as n-tuple-odd protocols for $n \geq 3$, have no counterexample under 1000. In other words, there is no counterexample for $n \geq 3$ within the scope of practical duty cycles—more precisely, the duty cycles that are greater than 0.00000187496. In practical networks, the duty cycle is usually much greater than 0.00000187496. We will elaborate on this issue in the sequel (Section 5.1.4).

5.1.2. Discovery delay

In this part, we examine the discovery delays for n-tuple-odd protocols.

Given an arbitrary duty cycle, we measure the discovery latency (in time slots) for single-, double-, triple-, and quadruple-odd protocols. The results are illustrated in Fig. 5. We observe that using more odd numbers in a set incurs a higher discovery latency for the n-tuple-odd neighbor discovery protocol. In fact, the discovery delay for the n-tuple-odd protocol is approximately $(n/\delta)^2$ time slots, where δ is the duty cycle.

5.1.3. Tradeoff between co-prime pair property and discovery delay

Based on the previous analysis on the co-prime pair property and the discovery delay, we can arrive at the following conclusions:

- The single- and double-odd protocols have a number of counterexamples within the scope of practical duty cycles. The n-tuple-odd protocols for $n \geq 3$ have no counterexample in practical networks (where the duty cycle is greater than 0.00000187496).
- The n-tuple-odd protocol for larger n incurs a larger discovery delay. To be precise, the discovery delay for the n-tuple-odd protocol is approximately $(n/\delta)^2$ time slots, where δ is the duty cycle.

Therefore, the triple-odd protocol, among the family of n-tuple-odd protocols, is the only one that satisfies the co-prime pair property for practical duty cycles and meanwhile achieves the lowest discovery delay. We present and analyze the triple-odd protocol, called *Todis* (*triple-odd based discovery*) hereinafter, which strikes a balanced tradeoff between the co-prime pair property and the discovery latency.

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26
27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53
54	55	56	57	58	59	60	61	62
63	64	65	66	67	68	69	70	71

Fig. 6. The first 71 time slots under Todis when $n = 15$ (i.e., the node chooses 13, 15 and 17). The node wakes up in slots 0, 13, 15, 17, 26, 30, 34, 39, 45, 51, 52, 60, 68, ….

5.1.4. Using sets of three consecutive integers in Todis

In Todis, we use three consecutive odd integers $n-2$, n and $n+2$ ($n \geq 3$) for constructing a wake-up schedule.

The co-prime pair property requires that at least one of the three consecutive odd integers that node a chooses (i.e., $n-2$, n and $n+2$) is co-prime w.r.t. one of the three integers that node b chooses (i.e., $m-2$, m and $m+2$).

Bounded discovery delay in practical networks: Generally, the triples consisting of three consecutive odd integers can also fail to satisfy the required co-prime pair property, as seen in counterexamples shown by the CRT. However, the two smallest sequences of odd integers in these counterexamples are $N_a = \{1600023, 1600025, 1600027\}$ and $N_b = \{2046915, 2046917, 2046919\}$. Such integers are too large to be chosen for creating a "practical" duty cycle anyway. For example, an n value larger than 1600023 would imply a duty cycle δ_a smaller than 0.00000187496. In practical applications, however, duty cycles are much greater than 0.00000187496. Therefore, any chosen sets N_a and N_b based on duty cycles would satisfy the co-prime pair property. By Theorem 1, Todis guarantees neighbor discovery with a delay bounded by

$$\min_{\gcd(n+i,m+j)=1, i,j=-2,0,2} \{(n+i)\cdot(m+j)\}.$$

A node a that has a desired duty cycle of δ may therefore choose an odd integer n such that

$$\frac{3(n^2-n-1)}{n(n^2-4)} \approx \frac{3}{n} = \hat{\delta}$$

is as close to δ as possible. We call n the *parameter* of node a.

Under Todis, its wake-up schedule is

$$s_a^t = \begin{cases} 1 & t \text{ is divisible by either } n-2, n, \text{or} n+2 \\ 0 & \text{otherwise} \end{cases},$$

with a period length of $(n-2)n(n+2)$ and a duty cycle of

$$\frac{1}{n-2}+\frac{1}{n}+\frac{1}{n+2}-\frac{1}{(n-2)n}-\frac{1}{n(n+2)}-\frac{1}{(n-2)(n+2)}$$
$$+\frac{1}{(n-2)n(n+2)} = \frac{3(n^2-n-1)}{n(n^2-4)}.$$

Fig. 6 shows the first 71 time slots under the Todis schedule when $n = 15$ (i.e., the node chooses 13, 15 and 17). Each grid in the figure represents a time slot, and the integer inside a grid denotes its slot index, e.g., the grid with 0 inside denotes the 0th time slot in the schedule (note that a schedule starts from the 0th time slot). The gray slots represent the active slots where the node wakes up. In this example, the duty cycle is $\frac{3\cdot(5^2-5-1)}{5\cdot(5^2-4)} \approx 18.9\%$.

5.2. Analysis of duty cycle granularity

Now we discuss the granularity of Todis in matching any desired duty cycle in practical applications. Suppose node a's desired duty cycle is δ_a, the relative error $\epsilon(\delta_a)$ between δ_a and its approximation $\hat{\delta}_a$ is defined by

$$\epsilon(\delta_a) = \frac{\left|\hat{\delta}_a - \delta_a\right|}{\delta_a}. \tag{12}$$

We want to mathematically estimate the upper bound of ϵ given δ_a, which we denote as $\hat{\epsilon}(\delta_a)$.

In Todis, node a needs to choose an odd integer n_a such that $\frac{3(n_a^2-n_a-1)}{n_a(n_a^2-4)}$ lies closest to δ_a, i.e.,

$$n_a = \arg\min_{n \text{ odd}} \left|\frac{3(n^2-n-1)}{n(n^2-4)} - \delta_a\right|.$$

Thus, the best approximation of the desired duty cycle δ_a is

$$\hat{\delta}_a = \frac{3(n_a^2-n_a-1)}{n_a(n_a^2-4)} \equiv \min_{n \text{ odd}} \left|\frac{3(n^2-n-1)}{n(n^2-4)} - \delta_a\right|.$$

Let $f(2k-1)$ and $f(2k+1)$ be two consecutive supported duty cycles, where $f(n) = \frac{3(n^2-n-1)}{n(n^2-4)}$. Relative error ϵ reaches a local maximum at $\delta_a = \frac{f(2k-1)+f(2k+1)}{2}$. Thus we obtain a quartic equation with respect to k

$$16\delta_a k^4 - 24k^3 + (12-40\delta_a)k^2 + 36k + 9\delta_a - 9 = 0. \tag{13}$$

By Eq. (13), we can obtain a solution $k = k(\delta_a)$ in complex radicals (the other three solutions are discarded). Then we have

$$\epsilon(\delta_a) \leq \hat{\epsilon}(\delta_a) \triangleq \frac{f(2k(\delta_a)-1)-\delta_a}{\delta_a},$$

where $\hat{\epsilon}(\delta_a)$ is also a complex expression in radicals with respect to δ_a.

Note that $\epsilon(\delta_a) = \hat{\epsilon}(\delta_a)$ iff $\delta_a = \frac{3(n^2-n-1)}{n(n^2-4)}$ for some odd integer n. We illustrate $\hat{\epsilon}(\delta_a)$ in Figs. 7 and 8 (see the "Estimation" lines), and we can observe that $\hat{\epsilon}(\delta_a)$ is a very tight upper bound for $\epsilon(\delta_a)$.

The upper bound function $\hat{\epsilon}(\delta_a)$ is an increasing function in [0, 1). In practical applications, δ_a is smaller than 20%, and thus ϵ is upper bounded by 6.71%, which is a very small relative error. Moreover, ϵ drops below 3.34% when $\delta_a \leq 10\%$. Asymptotically,

$$\hat{\epsilon}(\delta_a) \simeq \frac{2\delta_a}{\sqrt{9+4\delta_a^2}+3} \simeq \frac{1}{3}\delta_a$$

linearly approaches 0 as δ_a goes to 0. This property implies that the error decreases with the decline of the desired duty cycle.

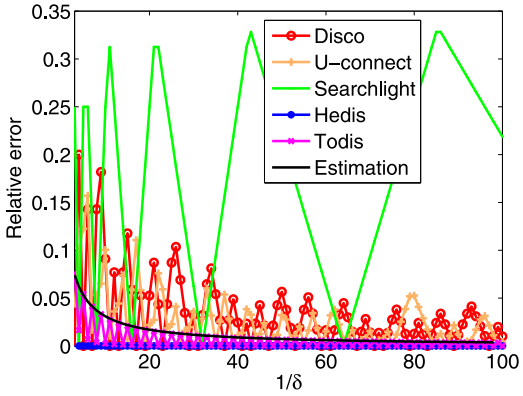


Fig. 7. Relative error vs. small duty cycle δ. The "Estimation" line is the theoretical upper bound estimation of relative error induced by Todis (see Section 5).

6. Performance evaluation

We compare Hedis and Todis against state-of-the-art neighbor discovery protocols of both the quorum-based and the co-primality

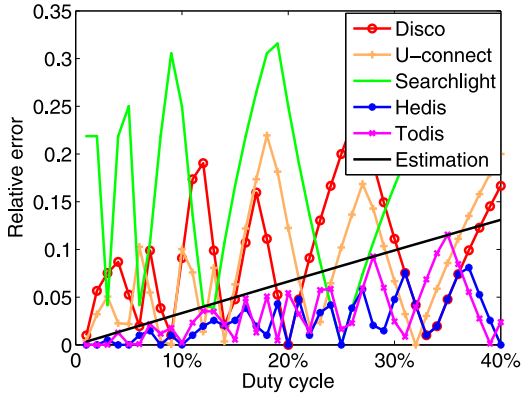


Fig. 8. Relative error vs. large duty cycle δ. The "Estimation" line is the theoretical upper bound estimation of relative error induced by Todis (see Section 5).

Table 1
Comparison of Hedis and Todis with existing neighbor discovery protocols.

Protocol name	Parameter restriction	Average dis. delay	Supported duty cycles
Disco	prime p_1, p_2	$O(\min\{p_1p_3, p_1p_4, p_2p_3, p_2p_4\})$	$\frac{1}{p_1}+\frac{1}{p_2}$
U-Connect	prime p_1 , p_2	$O(p_1p_2)$	$\frac{3p_1+1}{2p_1^2}$
Searchlight	power-multiple of t_1 , t_2	$O(t_1t_2)$	$\frac{2}{t_1^i}$
Hedis	same parity n, m	$O(nm)$	$\frac{2}{n}$
Todis	odd n, m	$O(nm)$	$\frac{3(n^2-n-1)}{n(n^2-4)} \approx \frac{3}{n}$

based varieties. These protocols include Disco [6] (co-primality based), Searchlight [4] (quorum-based), and U-Connect [7] (a combination of both). We evaluate the performances of these protocols using two metrics, namely the discovery latency and the duty cycle granularity.

- In Disco, each node chooses a pair of primes p_1 and p_2 to support duty cycles of the form $\frac{1}{p_1}+\frac{1}{p_2}$, and the worst-case discovery latency is $\min\{p_1p_3, p_1p_4, p_2p_3, p_2p_4\}$.
- In U-Connect, each node wakes up 1 time slot every p time slots and wakes up $\frac{p+1}{2}$ time slots every p^2 time slots. Therefore U-Connect supports duty cycles of the form $\frac{3p+1}{2p^2}$, and has the worst-case discovery latency of p_1p_2 if one node uses prime p_1 while another uses p_2. The dependence of Disco and U-Connect upon prime numbers greatly restricts their support of choices of duty cycle varieties.
- Searchlight requires that a node's parameter n_1 be a multiple or factor of its neighboring node's parameter n_2 to guarantee neighbor discovery. Therefore, in a network that implements Searchlight, the number that each node chooses must be a power-multiple of the smallest chosen number (i.e., 2, 4, 8, 16, or 3, 9, 27, 81, etc.), guaranteeing that any two nodes' numbers are multiples of each other. As a result, Searchlight only supports duty cycles of the form $\frac{2}{t^i}$, where t is an integer (i.e., the aforementioned smallest chosen number) and $i = 0, 1, 2, 3, \ldots$

Table 1 gives an overall theoretical comparison among these protocols. As the table shows, while the difference in discovery latency exists among these protocols, all of them perform on the order of the multiple of the principle parameters in the two participating nodes.

- Discovery latencies may be similar among the different protocols, because two nodes may choose similar parameters so as to match the desired duty cycle.
- In contrast, the metric of duty cycle granularity presents a different story. While all the parameters used in the protocols all have special restrictions due to protocol design, it is obvious that those for Hedis and Todis are the least stringent. For example, fewer than 2% of integers under 1000 are prime, while half of them are odd, giving Todis a much larger pool of numbers to choose from for its parameters as compared to Disco and U-Connect.

We confirm these theoretical results using simulations. We measure the relative errors each of the aforementioned protocols yields at differing duty cycles, as well as their discovery latencies in node pairs operating at various duty cycles.

6.1. Duty cycle granularity

The first set of simulations comparatively studies the supported duty cycles. We study two groups of duty cycles:

1. Small duty cycles $1 \le 1/\delta \le 100$, i.e., $\delta = 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots, \frac{1}{100}$;
2. Equispaced large duty cycles $0 \le \delta \le 1$, i.e., $\delta = 0\%, 1\%, 2\%, 3\%, 4\%, \ldots, 100\%$.

We use the metric called *the relative error* (defined in Eq. 12) to quantify the capability of supporting each studied duty cycle, which is denoted as

$$\epsilon \triangleq |\delta' - \delta|/\delta,$$

where δ' is the closest duty cycle that is supported by each simulated protocol, w.r.t. δ. Note that a smaller ϵ implies that the protocol provides more choices for energy conservation with a finer granularity of duty cycle control. For Searchlight, we let the smallest duty cycle unit be 1/2 to allow the finest duty cycle granularity.

Fig. 7 illustrates the results for small duty cycles, while Fig. 8 shows those of large duty cycles. These results provide us the following insights:

- Searchlight is inferior to the other protocols in supporting various duty cycles because it requires the duty cycle to be $\frac{2}{t^i}$, where t is a fixed integer and $i = 1, 2, 3, \ldots$. In this simulation, we use $t = 2$ to give Searchlight support for the duty cycles $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$. The relative error increases significantly as the desired duty cycle deviates away from the supported duty cycles (e.g., in Fig. 8, it has a peak at $37.5\% = \frac{1/2+1/4}{2}$, and 1/2 and 1/4 are supported duty cycles).
- The schedules in Disco and U-Connect are generated using prime numbers, which have a denser distribution than power-multiples. Thus, Disco and U-Connect perform better than Searchlight.
- Both Hedis and Todis greatly outperform all the other protocols, having very small relative errors. In fact, for small duty cycles, the relative errors from Hedis is nearly constantly zero (see Fig. 7). On the other hand, although Todis also performs well, its error rate obviously increases much faster than Hedis as the duty cycle δ increases.
- The theoretical "Estimation" lines for Todis (see Figs. 7 and 8) holds up well in that it follows the same pattern as Todis' actual error rates. This confirms our prior analysis in Section 5 of Todis' duty cycle granularity, where we estimated the upper bound of relative error for Todis.

6.2. Discovery latency in pairs

In this section, we study the discovery latencies of these protocols by simulations.

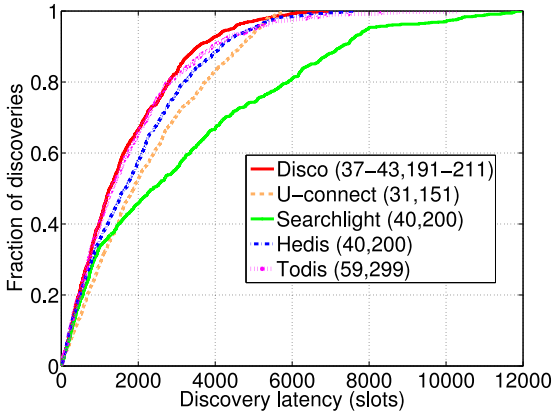


Fig. 9. CDF of discovery latency when each pair of nodes operate at duty cycles 1% and 5%, respectively. Numbers in the parentheses indicate the parameters of each corresponding protocol.

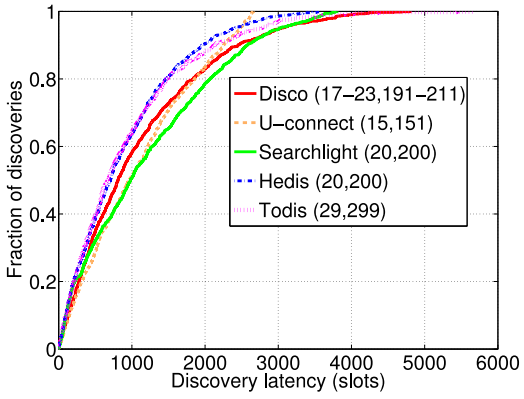


Fig. 10. CDF of discovery latency when each pair of nodes operate at duty cycles 1% and 10%, respectively. Numbers in the parentheses indicate the parameters of each corresponding protocol.

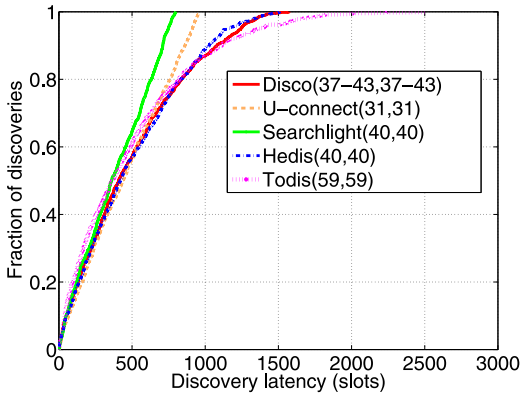


Fig. 11. CDF of discovery latency when each pair of nodes operate at the same duty cycle of 5%. Numbers in the parentheses indicate the parameters of each corresponding protocol.

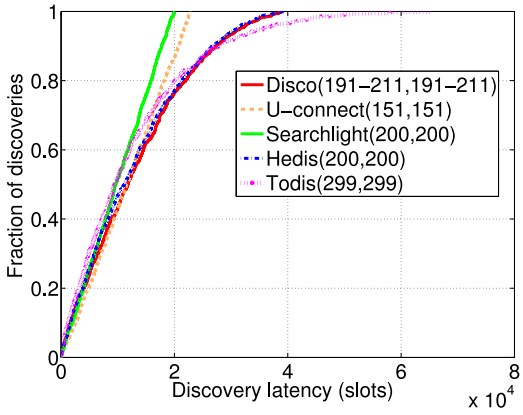


Fig. 12. CDF of discovery latency when each pair of nodes operate at the same duty cycle of 1%. Numbers in the parentheses indicate the parameters of each corresponding protocol.

6.2.1. Distribution of discovery latency

In this set of simulations, we take 1000 independent pairs of nodes and assign various duty cycles. In two instances, we compare the protocols' performance in heterogeneous discovery scenarios. We assign duty cycles of 1% and 5% to each respective node in the node pair in the first instance (see Fig. 9), and 1% and 10% in the second instance (see Fig. 10). We also compare the performance of the protocols in two homogeneous discovery scenarios, with each node in the node pair operating at the same duty cycles of 5% in the first scenario (see Fig. 11) and 1% in the second one (see Fig. 12).

Heterogeneous vs. homogeneous duty cycles: From these four cumulative distribution function graphs (CDFs), we see that overall, all of the protocols have comparative discovery latencies, with the odd exception of Searchlight in Fig. 9. Nonetheless, it must be noted that all 5 protocols presented were eventually successful in neighbor discovery for 100% of the pairs tested. These CDFs also show that Hedis is one of the few protocols that consistently perform above average in both the heterogenous and homogeneous neighbor discovery cases. For example, Figs. 11 and 12 indicate that Searchlight is the clear winner for discovery latency in the homogeneous case, but it does poorly in the heterogeneous cases, as seen in Figs. 9 and 10.

In addition, we see that for up to 90% of the CDF, Hedis and Todis are both near top performers, but the protocol with one of the smallest latencies in reaching 100% of the CDF in every case is U-Connect. We attribute this to the fact that U-Connect uses smaller values as its parameters, thus having a smaller upper bound in the worst case.

Similarly, we attribute Todis' consistent long tail in each CDF scenario to its larger parameters. Therefore, although it can quickly allow nodes to discover each other in most cases, seen in its quickly reaching 90% in the CDFs, it has the longest latency in the worst-case scenarios.

Hedis vs. Todis: These various simulations show that Hedis and Todis optimize the duty cycle granularity in both the quorum-based and the co-primality based neighbor discovery approaches, with Hedis having a finer granularity than Todis. Additionally, both protocols perform reasonably well in terms of discovery latency, wi th Todis having a larger worst case latency bound due to its larger parameters.

6.2.2. Impact of duty cycles in heterogeneous case

In this set of simulations, we investigate the impact of heterogeneous duty cycles. We take 1000 independent pairs of nodes where the ratio of the duty cycles of two nodes in a pair (denoted by γ) is fixed for each simulation; i.e., in each pair, the duty cycle of one node is δ and that of the other is $\gamma\delta$. In Figs. 13, 14 and 15, the ratio is set to be 1/2, 1/3 and 1/4, respectively.

We study the impact of δ (varying from 1% to 25%) upon the discovery latency under different neighbor discovery protocols. For example, in Fig. 13, we set the duty cycles of two nodes in a pair to be δ and $\delta/2$, and we can observe that Hedis and Todis have similar performance. Moreover, we can also see that with the duty cycle δ fixed, a smaller ratio γ (i.e., a smaller duty cycle for the other

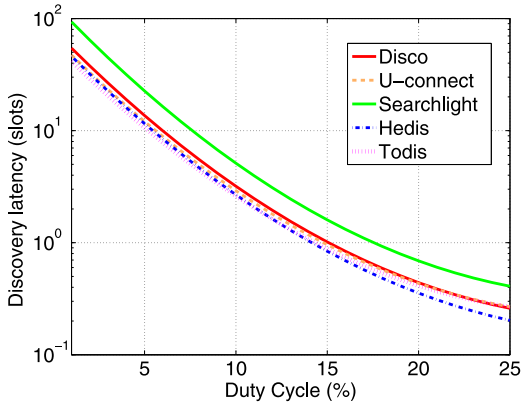


Fig. 13. Heterogeneous case: discovery delay vs. duty cycle δ, where δ and $\delta/2$ represent the duty cycles of two nodes in a pair.

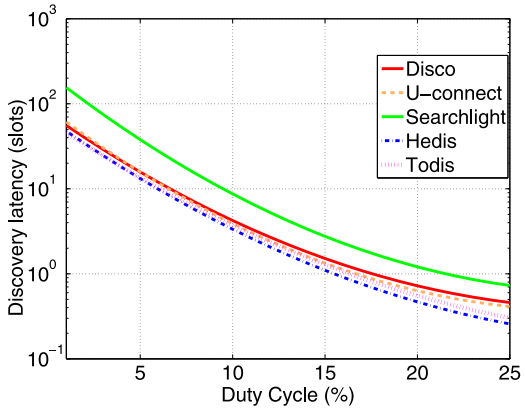


Fig. 14. Heterogeneous case: discovery delay vs. duty cycle δ, where δ and $\delta/3$ represent the duty cycles of two nodes in a pair.

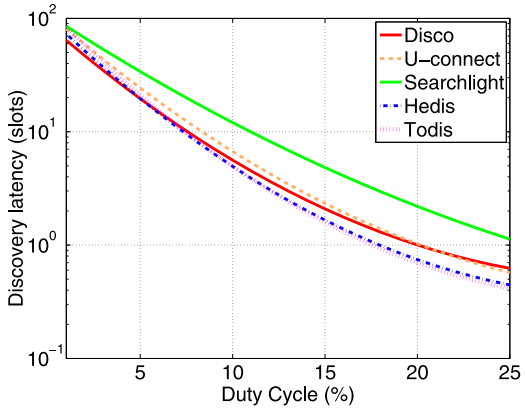


Fig. 15. Heterogeneous case: discovery delay vs. duty cycle δ, where δ and $\delta/4$ represent the duty cycles of two nodes in a pair.

node in a pair) results in a slight increase in discovery delay for most protocols except Searchlight (comparing Fig. 15 with Fig. 13).

6.3. Discovery latency in clusters

In this set of simulations, we study the discovery latency in clusters: when a new node joins a cluster, we measure the neighbor discovery latency when it discovers its 1st, 2nd, 3rd, 4th, and 5th neighbors, provided that different neighbor discovery protocols are employed.

Suppose that the duty cycle of the new node is δ_{new} and that the rest of the cluster nodes have a duty cycle of δ_{clu}. Figs. 16, 17, and 18 show the latency of finding the 1st, 2nd, 3rd, 4th, and 5th neighbors in box plots when $\delta_{new} = 1\%$ and $\delta_{clu} = 3\%$, $\delta_{new} = 5\%$ and $\delta_{clu} = 3\%$, and $\delta_{new} = 5\%$ and $\delta_{clu} = 9\%$, respectively.

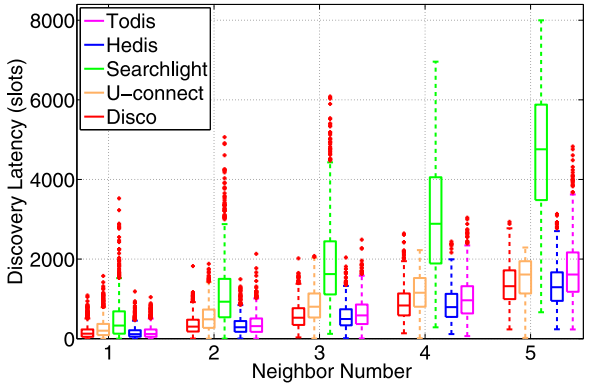


Fig. 16. Discovery latency when a new node joins the cluster and discovers its 1st, 2nd, 3rd, 4th, and 5th neighbors (regardless of their actual IDs), where the new node operates at a duty cycle of $\delta_{new} = 1\%$ while the rest of the cluster nodes operate at a duty cycle of $\delta_{clu} = 3\%$.

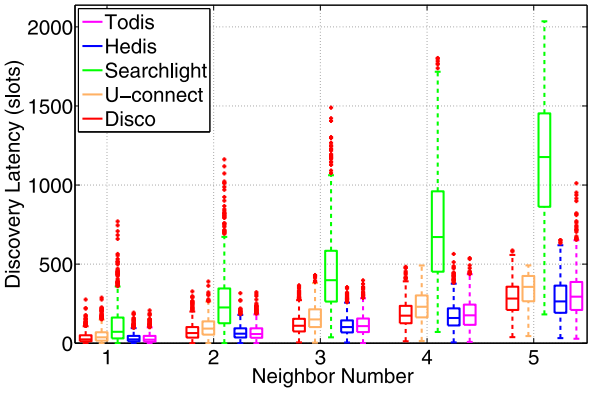


Fig. 17. Discovery latency when a new node joins the cluster and discovers its 1st, 2nd, 3rd, 4th, and 5th neighbors (regardless of their actual IDs), where the new node operates at a duty cycle of $\delta_{new} = 5\%$ while the rest of the cluster nodes operate at a duty cycle of $\delta_{clu} = 3\%$.

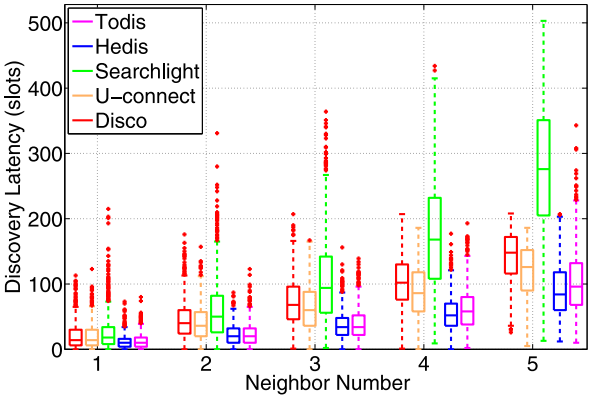


Fig. 18. Discovery latency when a new node joins the cluster and discovers its 1st, 2nd, 3rd, 4th, and 5th neighbors (regardless of their actual IDs), where the new node operates at a duty cycle of $\delta_{new} = 5\%$ while the rest of the cluster nodes operate at a duty cycle of $\delta_{clu} = 9\%$.

Impact of duty cycles on discovery latency: Figs. 16 and 17 both fix $\delta_{clu} = 3\%$ and have different values for δ_{new}. The duty cycle of the new node is $\delta_{new} = 1\%$ in Fig. 16 and it is $\delta_{new} = 5\%$ in Fig. 17. We observe that a larger duty cycle of the new node results in a smaller discovery latency of finding the neighbors.

Figs. 17 and 18 both fix $\delta_{new} = 9\%$ and have different values for δ_{clu}. The duty cycle of the rest of the cluster is $\delta_{clu} = 3\%$ in Fig. 17 and it is $\delta_{clu} = 9\%$ in Fig. 18. We observe that a larger duty cycle of the rest of the cluster reduces the discovery latency of finding the neighbors.

Figs. 16, 17, and 18 show that Hedis and Todis outperform all other protocols in finding the neighbors when a new node joins a cluster. We can also observe that: (1) Searchlight incurs the greatest discovery latency; (2) it depends on the values of the duty cycles whether Disco or U-Connect can achieve a better performance.

7. Testbed implementation

In this section, we evaluate the performance of Hedis and Toids by experiments over a real-world testbed. One of the major applications of neighbor discovery is to facilitate proximity-based communication between handheld devices like smartphones. We implemented Hedis, Todis and other protocols (U-connect, Disco, Searchligh) in a single Android app over smartphone devices (Xiaomi Mi-Note). The Mi-Note phone, a Android 6.0.1 smartphone manufactured by Xiaomi, which supports bluetooth low energy (BLE).

7.1. Implementation setup

Slot Duration: Earlier protocols for asynchronously discovering neighbors were all implemented on sensor nodes, allowing them to use small slots on the order of milliseconds [6] or even microseconds [7]. Later protocols were implemented over bluetooth of smartphones, which have a much larger slot duration due to the standard specification of bluetooth [4].

In our study, we implemented all protocols to use the BLE of Mi Note phones for neighbor discovery, since BLE is the off-the-shelf technology which has widely been equipped in many devices. Unlike the implemention over sensor radios, BLE have a non-negligible transition latency from the mode of sleep to that of transmit/receive. On the Mi Note, the time for BLE to start transmission and stop transmission is around 53 ms. The time for BLE to start scanning and stop scanning is around 13 ms. Because of this latency, we choose a slot duration of 2 s. Such a chosen slot duration (2 s) is sufficiently large, which leads to a longer discovery delay in our study compared with the implementation over sensor radios. Note that this is a limitation of the current protocol and hardware, and it is not because of our protocol design. For the same reason, the duty cycle assigned in the implementation study is greater than that in the simulation study.

Message sent and listened in an active slot: In the Android app we developed, we implement the message exchange between two nodes in an active slot. In neighbor discovery process, each node schedules active slots according to the desired duty cycle and the protocol it uses. Similar to other implemented testbed [4], in an active slot, a node needs to send "hello" messages to others, and meanwhile it listens over the channel in search of the hello messages sent by others, in order to discover a neighboring node. Specifically, during an active slot:

- *When to send hello messages*: In our implementation, hello messages containing the sender node's ID is sent at the very beginning and at the very end of the slot.
- *When to listen for hello messages*: Between the very beginning and the end of an active slot, the node continuously listens for hello messages sent from other nodes, without sending messages.
- *When is the pairwise neighbor discovery complete?*: When node 1 gets gets a hello message sent from node 2, we say node 1 discovers node 2, and it adds the ID of the sender to a list of discovered nodes. A pairwise neighbor discovery between nodes 1 and 2 is complete when the two nodes receive the hello messages sent from each other.

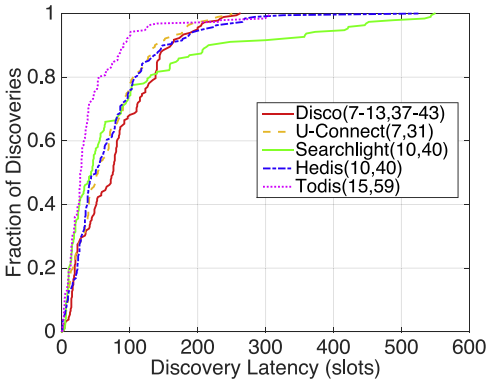


Fig. 19. Implementation in heterogeneous case: CDF of discovery latency at duty cycles 5% and 20%, respectively. Numbers in the parentheses indicate the parameters of each corresponding protocol.

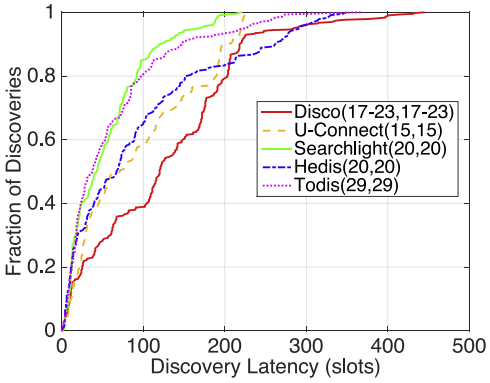


Fig. 20. Implementation in homogeneous case: CDF of discovery latency at the same duty cycle of 10%. Numbers in the parentheses indicate the parameters of each corresponding protocol.

Possible collision of hello messages: In experiment, we place all devices together in a large room of 20 m by 30 m, to avoid interference from neighboring bluetooth/WiFi devices. However, collision of hello messages may exist when multiple pairs send the messages simultaneously, which will affect the performance of certain protocols as discussed below.

7.2. Results

We implemented considered protocols on 8 Mi Note phones, and recorded the discovery latency with various duty cycles. To create the asynchronous clocks, we added a random number of slots before the neighbor discovery process starts on each device.

In two instances, we compare the performance of the protocols in heterogeneous scenarios. We assign duty cycles of 5% and 20% in the first instance (see Fig. 19), and 10% and 20% in the second instance (see Fig. 21). We also compare the performance of the protocols in two homogeneous discovery scenarios, with each phone operating at the same duty cycles of 10% in the first scenario (see Fig. 20) and 20% in the second one (see Fig. 22).

The overall trends in these experimental results agree with the simulation results. When the duty cycle is relatively large, i.e., 20% as shown in Fig. 21, Fig. 22, all phones can discover each other (100% in the CDF) within a short time; that is, all protocols perform well in these cases.

Searchlight performs differently in heterogeneous and homogeneous cases: Searchlight performs well in homogeneous

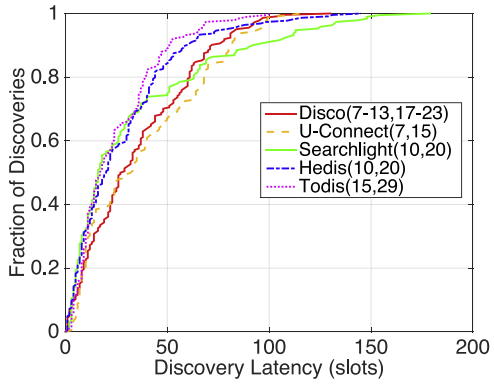


Fig. 21. Implementation in heterogeneous case: CDF of discovery latency at duty cycles 10% and 20%, respectively. Numbers in the parentheses indicate the parameters of each corresponding protocol.

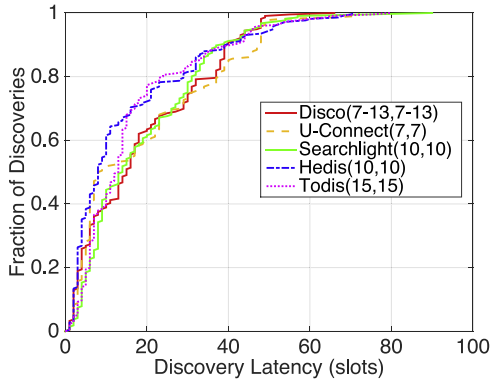


Fig. 22. Implementation in homogeneous case: CDF of discovery latency at the same duty cycle of 20%. Numbers in the parentheses indicate the parameters of each corresponding protocol.

cases (Fig. 20), while it has an inferior performance in heterogeneous cases (Fig. 19), which confirms the simulation results.

Todis excels in real-world implementation: Todis has the best performance in the heterogeneous cases (Figs. 19 and 21); meanwhile, it has the second best performance in the homogeneous cases. This is can be attributed to the fact that: in Todis, the active slots in a wake-up schedule are more evenly distributed than the active slots in the wake-up schedule of other protocols. This leads to less probability of collision of hello messages in the implementation. Hence, Todis excels in the discovery latency owing to less collision of hello messages.

8. Conclusion

In this paper, we explored the current two main approaches of designing an asynchronous heterogeneous neighbor discovery protocol with guaranteed latency upper bounds—the quorum-based and the co-primality based approaches. Using these two approaches we designed the Hedis and Todis neighbor discovery protocols, emphasizing on duty cycle granularity optimization for both. Hedis, as a quorum-based protocol, forms a $(n-1)\times n$ matrix of time slots and uses the anchor- probing slot method to ensure neighbor discovery. Todis, as a co-primality based protocol, uses sets of three consecutive odd integers to ensure co-primality and thus ensures neighbor discovery due to CRT. In the design of both protocols we proved their capability in ensuring acceptable upper bounds in discovery latency. Through analytical comparisons as well as simulations, we confirmed the fine-granularity of Hedis and Todis in duty cycle control compared with existing protocols. Hedis is able to support duty cycles in the form of $\frac{2}{n}$, while Todis can support duty cycles roughly in the form of $\frac{3}{n}$, allowing both protocols to effectively cover any practical duty cycle and thus prolong battery longevity.

We also showed in both our analysis and simulations that Hedis as a quorum-based protocol is similar with Todis as a co-primality based protocol in duty cycle granularity, while Todis is better than Hedis in discover latency. By being able to support duty cycles at such a fine granularity while still guaranteeing an acceptable discovery latency bound, Hedis truly paves the way for neighbor discovery in wireless sensor networks.

Acknowledgement

This work is partially supported by the National Key Research and Development Program no. 2017YFB0803302, and the National Natural Science Foundation of China under Grant nos. 61572051 and 61632017.

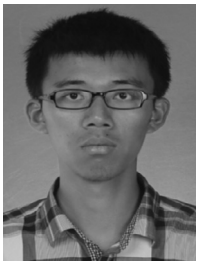
References

[1] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, A. T. Campbell, Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the Cenceme Application, in: ACM SenSys, 2008, pp. 337–350.
[2] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, C. Diot, Mobiclique: middleware for mobile social networking, in: Proceedings of the 2nd ACM Workshop on Online Social Networks, 2009, pp. 49–54.
[3] L.M. Feeney, M. Nilsson, Investigating the energy consumption of a wireless network interface in an ad hoc networking environment, in: IEEE INFOCOM, 2001, pp. 1548–1557.
[4] M. Bakht, M. Trower, R.H. Kravets, Searchlight: won't you be my neighbor? in: ACM MobiCom, 2012, pp. 185–196.
[5] L. Chen, K. Bian, M. Zheng, Heterogeneous multi-channel neighbor discovery for mobile sensing applications: theoretical foundation and protocol design, in: ACM MobiHoc, 2014, pp. 307–316.
[6] P. Dutta, D. Culler, Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications, in: ACM SenSys, 2008, pp. 71–84.
[7] A. Kandhalu, K. Lakshmanan, R.R. Rajkumar, U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol, in: ACM IPSN, 2010, pp. 350–361.
[8] D. Zhang, T. He, Y. Liu, Y. Gu, F. Ye, R.K. Ganti, H. Lei, Acc: generic on-demand accelerations for neighbor discovery in mobile applications, in: ACM SenSys, 2012, pp. 169–182.
[9] R. Zheng, J.C. Hou, L. Sha, Asynchronous wakeup for ad hoc networks, in: ACM MobiHoc, 2003, pp. 35–45.
[10] T. Meng, F. Wu, A. Li, G. Chen, N.H. Vaidya, On robust neighbor discovery in mobile wireless networks, in: CoNext, ACM, 2015, p. toappear.
[11] L. Chen, R. Fan, K. Bian, M. Gerla, T. Wang, X. Li, On heterogeneous neighbor discovery in wireless sensor networks, in: Computer Communications (INFOCOM), 2015 IEEE Conference on, IEEE, 2015, pp. 693–701.
[12] Y. Zhang, K. Bian, L. Chen, P. Zhou, X. Li, Dynamic slot-length control for reducing neighbor discovery latency in wireless sensor networks, in: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–6.
[13] K. Bian, Y. Zhang, P. Qiao, Z. Li, Fine-grained collision mitigation control for neighbor discovery in wireless sensor networks, in: IEEE/CIC International Conference on Communications in China, IEEE, 2017.
[14] Y.-C. Tseng, C.-S. Hsu, T.-Y. Hsieh, Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks, Comput. Netw. 43 (3) (2003) 317–337.
[15] M.B. Nathanson, Elementary Methods in Number Theory, 195, Springer, 2000.
[16] C.M. Vigorito, D. Ganesan, A.G. Barto, Adaptive control of duty cycling in energy-harvesting wireless sensor networks, in: Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07. 4th Annual IEEE Communications Society Conference on, IEEE, 2007, pp. 21–30.
[17] X. Wang, X. Wang, G. Xing, Y. Yao, Dynamic duty cycle control for end-to-end delay guarantees in wireless sensor networks, in: Quality of Service (IWQoS), 2010 18th International Workshop on, IEEE, 2010, pp. 1–9.
[18] H. Yoo, M. Shim, D. Kim, Dynamic duty-cycle scheduling schemes for energy-harvesting wireless sensor networks, Communications Letters, IEEE 16 (2) (2012) 202–204.
[19] I. Demirkol, C. Ersoy, F. Alagoz, et al., Mac protocols for wireless sensor networks: a survey, IEEE Commun. Mag. 44 (4) (2006) 115–121.
[20] W. Ye, J. Heidemann, D. Estrin, An energy-efficient mac protocol for wireless sensor networks, in: INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 3, IEEE, 2002, pp. 1567–1576.
[21] M.J. McGlynn, S.A. Borbash, Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks, in: ACM MobiHoc, 2001, pp. 137–145.
[22] S. Vasudevan, J. Kurose, D. Towsley, On neighbor discovery in wireless networks with directional antennas, in: IEEE INFOCOM, 2005, pp. 2502–2512.

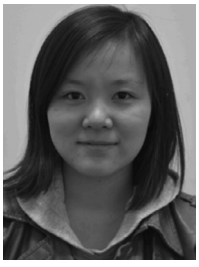
[23] S. Vasudevan, D. Towsley, D. Goeckel, R. Khalili, Neighbor discovery in wireless networks and the coupon collector's problem, in: Proceedings of the 15th annual international conference on Mobile computing and networking, ACM, 2009, pp. 181–192.
[24] W. Zeng, S. Vasudevan, X. Chen, B. Wang, A. Russell, W. Wei, Neighbor discovery in wireless networks with multipacket reception, in: ACM MobiHoc, 2011, pp. 3:1–3:10.
[25] Z. Zhang, B. Li, Neighbor discovery in mobile ad hoc self-configuring networks with directional antennas: algorithms and comparisons, IEEE Trans. Wireless Commun. 7 (5) (2008) 1540–1549.
[26] J.-R. Jiang, Y.-C. Tseng, C.-S. Hsu, T.-H. Lai, Quorum-based asynchronous power-saving protocols for ieee 802.11 ad hoc networks, Mobile Netw. Appl. 10 (1–2) (2005) 169–181.
[27] S. Lai, B. Ravindran, H. Cho, Heterogenous quorum-based wake-up scheduling in wireless sensor networks, IEEE Trans. Comput. 59 (11) (2010) 1562–1575.
[28] G. Anastasi, M. Conti, M. Di Francesco, A. Passarella, Energy conservation in wireless sensor networks: a survey, Ad Hoc Netw. 7 (3) (2009) 537–568.
[29] V. Galluzzi, T. Herman, Survey: discovery in wireless sensor networks, Int. J. Distrib. Sens. Netw. 8 (1) (2012) 271860.

Lin Chen received the B.S. degree in computer science from Peking University, Beijing, China, in 2014. He is currently working toward the Ph.D. degree with the Department of Electrical Engineering, Graduate School of Arts and Sciences, Yale University, New Haven, CT, USA. His research interests focus on wireless networks and network theory.

Ruolin Fan received the B.S. degree in computer science from University of California, Los Angeles, USA, in 2011. He is currently working toward the Ph.D. degree in Computer Science at University of California, Los Angeles, USA. His research interests focus on computer networks, particularly in wireless networks.



Yangbin Zhang received the B.S. degree in computer science from Tongji University, Shanghai, China, in 2015. He is currently working toward the M.S. degree with the Institute of Network Computing and Information Systems, School of Electronics Engineering and Computer Science, Peking University, Beijing, China. His research interests focus on wireless networks and network theory.



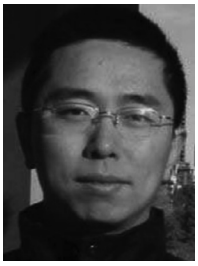
Shuyu Shi received the bachelor's degree from the University of Science and Technology of China in 2011, majoring in computer science. She has been working toward the PhD degree in the Department of Informatics, School of Multi- disciplinary Science, Graduate University for Advanced Studies since 2011.

Kaigui Bian (M'11) received the Ph.D. degree in computer engineering from the Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, in 2011. He is currently an Assistant Professor with the Institute of Network Computing and Information Systems, School of Electronics Engineering and Computer Science, Peking University, Beijing, China. His research interests include mobile computing, cognitive radio networks, network security, and privacy.

Lin Chen (S'07'M'10) received his B.E. degree in radio engineering from Southeast University, China, in 2002 and the Engineer Diploma from Telecom Paris Tech, Paris, France, in 2005. He also holds an M.S. degree of networking from the University of Paris 6. He currently works as an Associate Professor in the Department of Computer Science of the University of Paris-Sud. He serves as Chair of the IEEE Special Interest Group on Green and Sustainable Networking and Computing with Cognition and Cooperation, IEEE Technical Committee on Green Communications and Computing. His main research interests include modeling and control for wireless networks, distributed algorithm design, and game theory.

Pan Zhou (S'07'M'14) received the Ph.D. degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2011. He was a Senior Technical Member with Oracle Inc., Boston, MA, USA, from 2011 to 2013, where he was involved in Hadoop and distributed storage systems for big data analytics at Oracle cloud Platform. He is currently an Associate Professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China. His current research interests include communication and information networks, security and privacy, machine learning, and big data.

Mario Gerla (F'02) received the degree in engineering from Politecnico di Milano, Milano, Italy, and the Ph.D. degree from the University of California, Los Angeles (UCLA), CA, USA. From 1973 to 1976, he was with Network Analysis Corporation, New York, NY, USA, where he helped transfer ARPANET technology to government and commercial networks. In 1976, he joined UCLA, where he is currently a Professor of computer science. At UCLA, he was part of the team that developed the early ARPANET protocols under the guidance of Prof. L. Kleinrock. He has also designed and implemented net- work protocols, including ad hoc wireless clustering, multicast (ODMRP and CodeCast), and Internet transport (TCP Westwood). He has lead the $12M six-year ONR MINUTEMAN project, designing the nextgeneration scalable airborne Internet for tactical and homeland defense scenarios. He is currently leading two advanced wireless network projects under U.S. Army and IBM funding. His team is developing a vehicular testbed for safe navigation, urban sensing, and intelligent transport. A parallel research activity explores personal communications for cooperative networked medical monitoring (see http://www.cs.ucla.edu/NRL for recent publications).



Tao Wang (SM'11) received the PhD degree in computer science from Peking University, Beijing, China, in 2006. He is currently an associate professor with Peking University, Beijing, China. His research interests include computer architecture, reconfigurable logic, wireless network, and parallel computing. He is a senior member of the IEEE.

Xiaoming Li (SM'03) received the Ph.D. degree in Computer Science from Stevens Institute of Technology, Hoboken, NJ, USA, in 1986. He is currently a Professor with Peking University, Beijing, China. His research interests include web search and mining and online social network analysis. Dr. Li is an Editor of both Concurrency and Computation and Networking Science.