# Spear: A Practical Neighbor Discovery Framework for Wireless Sensor Networks

***Abstract*—Neighbor discovery is a crucial step in constructing wireless sensor networks. Various protocols have been proposed to minimize the discovery latency or to prolong the lifetime of sensors. However, none of them has addressed all the critical concerns coming from practical networks, including communication collisions, latency constraints and energy management. In this paper, we propose Spear, the first practical framework to meet these requirements for general neighbor discovery protocols. Spear offers two new methods to reduce communication collisions, thus boosting the discovery rate, which is defined as the number of discovered neighbors divided by all actual neighbors. Spear attends to latency constraints and facilitates timely adjustments to reduce the discovery latency. Spear realizes two practical energy management methods, which evidently can prolong the nodes' lifetime. Overall, Spear is compatible with existing discovery protocols without having to modify them, and it automatically improves the discovery results for these protocols. We implemented Spear and evaluated several notable neighbor discovery protocols: Spear greatly improves the discovery rate from $33.0\%$ to $99.2\%$, prolongs the lifetime up to $6.47$ times, and achieves up to $11.24$ times higher throughput for routing.**

## I. Introduction

With the ascent of Internet-of-Things [1], wireless sensor networks are increasingly being adopted for tracking and monitoring applications in various areas such as healthcaring, smart buildings, agricultural management and assisted living. For example, a wireless sensor network is deployed for agriculture information monitoring [19], and sensors are attached to inventory items in a large warehouse for object identification [11].

As a crucial process in constructing a wireless network, neighbor discovery, where sensor nodes try to find the existence of neighboring nodes within their communication range, has drawn much attention in the last decade or so [2], [5], [6], [8]–[12], [15]–[18], [20], [23]. Sensor nodes are powered by batteries, and most existing works focus on designing discovery schedules for a low *duty cycle* which is defined as the fraction of time the radio is turned on. However, despite decades of efforts, designing practical neighbor discovery protocols for real-life networks remains still a gigantic challenge, which must consider all the three critical factors below.

First of all, communication collisions happen when multiple nodes transmit on the same channel simultaneously. Second, practical applications require bounded latency constraints. For example, object detection needs to discover the neighbors for transmitting emergent information with very low latency. Third, to prolong the nodes' lifetime, the nodes need energy management methods to adjust the duty cycle during the discovery process. The three concerns are the major ones pointed out by previous works. Without addressing all these practical concerns, the nodes may fail to discover many neighbors or run out of energy quickly.

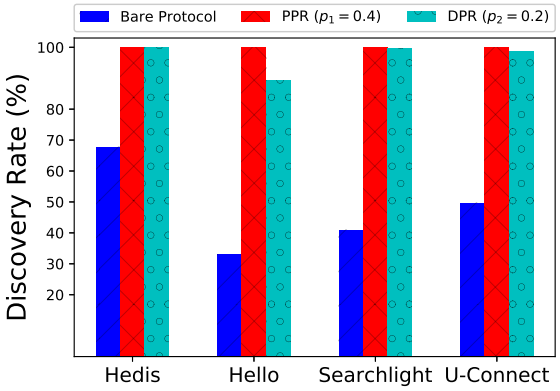



Fig. 1. Spear greatly improves discovery rates for four notable protocols. Discovery rate for each protocol is defined as the number of discovered neighbors divided by all actual neighbors. Bare protocols mean they do not run in Spear.

Unfortunately, no existing work has considered all the concerns in their protocols; actually most of them only consider one factor. Two major types of protocols exist, including probability-based and deterministic ones. Probability-based protocols turn on the radio with different probabilities to reduce communication collisions [12], [18], [23], but the discovery latency often varies significantly. Compared to probability-based protocols, deterministic protocols are much more popular in practice and are the main focus in this paper, because they have deterministic discovery schedule and the discovery latency is mostly stable [2], [5], [6], [9], [11], [15], [16], [20]. However, existing deterministic protocols mainly target at only two neighbors, where collisions cannot occur. As shown in Fig. 1, when these (bare) deterministic protocols (Hedis [5], Hello [16], Searchlight [2], and U-Connect [9]) are adopted in a network of $1000$ nodes, the discovery rates are only $33.0 \sim 65.9\%$ because of collisions.

In this paper, we propose **Spear**[1], a practical neighbor discovery framework that deals with all the critical factors mentioned above. The advantages of Spear are:

a) Spear creates two new methods, Pure Probability Reducing (PPR) and Decreased Probability Reducing (DPR), to reduce communication collisions among multiple

[1]Spear is a general, powerful weapon of ancient Hawaii.

nodes. Running in Spear, a deterministic protocol targeting at two neighbors can be automatically extended to support multiple nodes, while still keeping a stable discovery latency;

b) Spear introduces two methods to manage energy and one unified method to handle latency constraints; it prolongs the node's lifetime and enhances the availability for various applications;

c) Spear enables the quantitative analysis of various neighbor discovery protocols, and generates the optimal neighbor discovery schedule automatically.

We implemented Spear and evaluated several notable neighbor discovery protocols on $1000$ nodes. As shown in Fig. 1, when running PPR or DPR in Spear, the discovery rates for these protocols greatly increase from $33.0\%$ to $99.2\%$. By incorporating the energy management methods, nodes' lifetime can be extended up to $6.47$ times than that of running bare protocols. We have applied Spear to the wireless sensor network routing problem: compared to bare protocols, Spear incurs moderate time overhead on initiating routes, but it can achieve $11.24$ times higher throughput due to its much higher neighbor discovery rates.

The main contribution of Spear is automatically reducing communication collisions, promoting discovery rate, and prolonging lifetime for neighbor discovery, making the construction of wireless sensor networks much more effective and easier. Furthermore, Spear can be broadly applied to tackle various problems in wireless sensor networks.

The rest of the paper is organized as follows. We introduce some existing neighbor discovery protocols in the next section, and we introduce the preliminaries in Section III. We describe Spear in detail in Section IV. The methods to manage the node's energy and handle the latency requirements are proposed in Section V, and the methods to reduce communication collisions for any protocol are introduced in Section VI. We implemented Spear and evaluated several notable neighbor discovery protocols, the results are presented and discussed in Section VII. Finally, we conclude the paper in Section VIII.

## II. Related Works

The neighbor discovery problem in wireless sensor networks has been widely studied and the goal is to reduce the duty cycle or to reduce the latency of discovering the neighboring nodes. Generally speaking, there are two categories of neighbor discovery algorithms.

One category is *probability algorithms*, which utilize randomness to discover the neighbors. *Birthday protocol* [12] is one of the earliest algorithms that works on the *birthday paradox*, i.e. the probability that two people have the same birthday exceeds $\frac{1}{2}$ among 23 people. Following that, more smarter probabilistic algorithms were proposed [18], [23]. Although they have considered the discovery process among multiple nodes, they cannot guarantee an upper bound on the discovery latency between every two neighbors.

The other category is *deterministic algorithms*, which adopt some mathematic tools to ensure discovery between every two neighbors. The first tool is called *quorum system*: for any two intersected quorums, two neighboring nodes could choose any quorum in the system to design the discovery schedule and the discovery latency can be bounded in a short time. Many algorithms are related to the quorum system [5], [8]–[10], but only a few of them support asymmetric duty cycles of the nodes, such as Hedis [5]. Another important tool is *co-primality* where two co-prime numbers are chosen by the neighbors to design the discovery schedule, and they can discover each other within a bounded latency by the Chinese Remainder Theorem [14]. Some representative algorithms are Disco [6], U-Connect [9], and Todis [5].

Many neighbor discovery protocols assume that time is divided into slots of equal length and the nodes have aligned slots. Some works also study a general scenario that the slots are aligned between the users. They use *probe*, *beacon* or *anchor* to design the discovery protocols, and the representative algorithms are Searchlight [2], Hello [16] and Nihao [15]. There are also some other neighbor discovery protocols that are based on different techniques, such as combinatorial design [22], BlindDate [20], and Panda [11], the details of which we omit here. Among these algorithms, some of them only support *symmetric duty cycle* (the nodes select the same duty cycle), such as Quorum and Balanced Nihao [15], while others support asymmetric duty cycles (nodes select different duty cycles), such as Disco, U-Connect, Searchlight [2], Hello [16], Hedis and Todis [5].

To the best of our knowledge, most deterministic neighbor discovery algorithms are designed for two neighbors, with symmetric or asymmetric duty cycles. A few of them consider the energy management of each node and they ignore the communication collisions when they are extended for multiple nodes. Therefore, we propose a practical framework incorporating these issues and enable the existing deterministic protocols to be applicable for networks with multiple nodes.

## III. Preliminaries

### A. Sensor Node Model

Each sensor node $u_i$ has a distinguishable identifier $I_i$. Suppose node $u_i$ is powered by the battery it carries, we can denote the maximum power as $P_{max}$ and the remaining energy at time $t$ as $P_i(t)$. The node dies at time $t$ if $P_i(t) = 0$ (rechargeable battery is a possibility but not considered here).

Driven by different applications, each node can carry out many operations, such as sensing nearby information, transmitting data, etc. Among these operations, communications through the wireless channel dominate the energy consumption [11]. Therefore, we assume only two states $\{ON, OFF\}$ in this paper; $OFF$ means the node turns its radio off to save energy, while $ON$ means the node

TABLE I
NOTATIONS FOR NEIGHBOR DISCOVERY

| Notation | Description |
|---|---|
| $u_i$ | Sensor node $u_i$ |
| $I_i$ | Identifier of node $u_i$ |
| $P_{max}$ | The maximum energy of each sensor |
| $P_i(t)$ | The remaining energy of $u_i$ at time $t$ |
| $t_0$ | The length of each time slot |
| $p_n$ | The consumed energy to turn on the radio in each slot |
| $d_c$ | Communication range of each sensor |
| $t_i^s$ | Start time of node $u_i$ |
| $S_i$ | Neighbor discovery schedule of node $u_i$ |
| $s_i(t)$ | The schedule of node $u_i$ at time $t$ |
| $L(i,j)$ | Discovery latency between $u_i, u_j$ |
| $N_i$ | The set of neighbors of node $u_i$ |
| $L(i,N_i)$ | The latency for $u_i$ to discover all neighbors |
| $\theta_i(T_1,T_2)$ | Node $u_i$'s duty cycle between time $[T_1,T_2]$ |
| $t_i^e$ | The time node $u_i$ runs out of energy |
| $Lf_i$ | Lifetime of node $u_i$ |

turns the radio on for communication. We focus on the communications during neighbor discovery process.[2]

Suppose time is divided into slots of equal length $t_0$ which is sufficient for the nodes to establish a communication link on the channel. Denote the consumed energy in each time slot as $p_n$ if the node turns on the radio, and $p_f$ if the radio is off. In practical systems, switching the radio states also consumes energy, such as $p_{nf}$ (switching the radio from on to off) and $p_{fn}$ (switching the radio from off to on). In this paper, we adopt the common assumption $p_f = p_{nf} = p_{fn} = 0$, and prolonging the lifetime of the node is equivalent to reducing its percentage of time slots when the radio is on. The notations are also given in Table I.

### B. Communication Model

Considering a wireless sensor network that consists of $N$ nodes, $\{u_1, u_2, \ldots, u_N\}$. Suppose only one wireless channel is available for communication. When the nodes turn on their radios simultaneously, they can transmit information through the wireless channel. Denote the communication range of each node as $d_c$, and two nodes are called *neighbors* if their distance is no larger than $d_c$ (denote the distance of nodes $u_i, u_j$ as $d(u_i, u_j)$).

In real networks, whether one node can communicate with a neighboring node successfully is dependent on many factors, such as environment noises, the sending power energy, the path-loss exponent during transmission, beaconing, and handshaking; we simplify the process and assume two neighboring nodes can communicate if their distance is within $d_c$ and they both have turned on the radio.

In the practical networks, multiple nodes may turn on the radio simultaneously and they could cause communication collisions on the channel. For example, node $u_1$ has two neighbors $u_2, u_3$ and they all turn on the radio simultaneously. Suppose both $u_2, u_3$ send a message to $u_1$; then $u_1$ cannot decode the composited message correctly. Therefore, we say *communication collision* happens and node $u_1$ cannot find its neighbors.

[2]Before each node tries to communicate with others, it has to identify the neighboring nodes first. Therefore, we suppose the nodes can communicate when the discovery is successful.

### C. Neighbor Discovery

Neighbor discovery is the foundation of constructing wireless sensor networks. When the sensor nodes are deployed in the monitoring area, each node can only know its local information; the nodes have to find their neighboring nodes, and then the network can be established.

Suppose node $u_i$ starts at time $t_i^s$ and it tries to discover its neighbors by turning on the radio. In order to save energy, the node runs some pre-defined algorithms to generate a discovery schedule $S_i = \{s_i(t) | t \geq t_i^s\}$, where:

$$s_i(t) = \begin{cases} 0 & if\ u_i\ turns\ the\ radio\ OFF \\ 1 & if\ u_i\ turns\ the\ radio\ ON \end{cases}$$

The **neighbor discovery** problem between two neighboring nodes is defined as:

*Problem 1:* For two neighboring nodes $u_i$ and $u_j$, design the discovery schedules $S_i, S_j$ respectively such that there exists $T$ satisfying:

$$s_i(T) = s_j(T) = 1$$

Two nodes may start at different times, which is referred to as the *asynchronous* case in the literature, and the *discovery latency* is defined as:

*Definition 3.1:* The discovery latency between two neighboring nodes $u_i$ and $u_j$ is the time cost to turn on the radio simultaneously after they both have started:

$$L(i,j) = T - \max\{t_i^s, t_j^s\} \tag{1}$$

Considering the networks with multiple nodes, the neighbor discovery problem for each node $u_i$ is defined as:

*Problem 2:* For each node $u_i$, denote the set of neighboring nodes as $N_i = \{u_j | d(u_i, u_j) \leq d_c\}$. Design the discovery schedule for each node, such that there exists $T_{i,j}$ satisfying:

$$\begin{cases} s_i(T_{i,j}) = s_j(T_{i,j}) = 1 \\ s_k(T_{i,j}) = 0, \forall u_k \in N_i, u_k \neq u_j \end{cases}$$

Similarly, the *discovery latency* for node $u_i$ is defined as:

*Definition 3.2:* The discovery latency for node $u_i$ is the time cost to find all neighbors:

$$L(i,N_i) = \max_{u_k \in N_i} T_{i,k} - t_i^s \tag{2}$$

Most neighbor discovery algorithms generate their discovery schedules with regard to the *duty cycle* which is defined as:

*Definition 3.3:* The duty cycle of node $u_i$ between time $T_1, T_2$ ($T_1 < T_2$) is the percentage of time slots when $u_i$ turns on the radio:

$$\theta_i(T_1,T_2) = \frac{|\{s_i(t) = 1 | T_1 \leq t \leq T_2\}|}{T_2 - T_1}$$

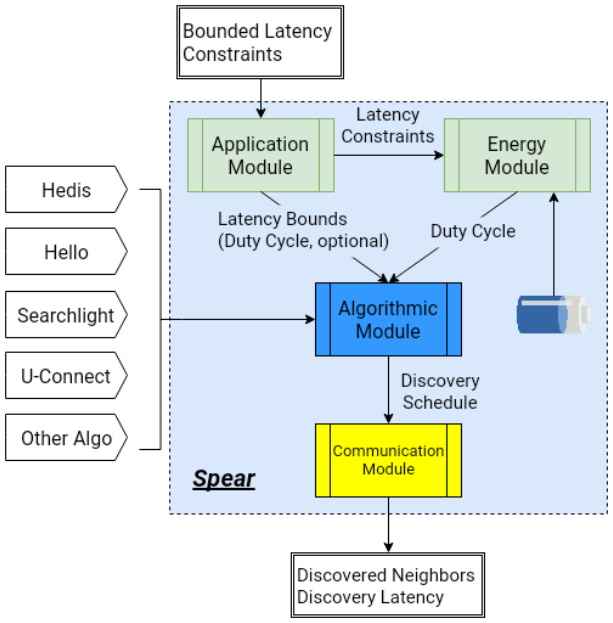



Fig. 2. Overview of Spear

The existing deterministic algorithms try to minimize the discovery latency between two neighbors for pre-defined duty cycles. Both symmetric and asymmetric duty cycles should be considered.

## IV. Spear: Neighbor Discovery Framework

### A. Framework Overview

As shown in Fig. 2, Spear consists of four modules: **energy module** is in charge of the node's energy management; **application module** collects various latency constraints from the applications; **algorithmic module** generates a neighbor discovery schedule by invoking the discovery protocols; and **communication module** is responsible for the communication with neighbors.

Spear accepts bounded latency constraints and remaining energy as the inputs, and the neighbor discovery algorithms (such as Hedis, Hello, Searchlight, and U-Connect in the figure) are plugged into the framework to generate the discovery schedule. Spear outputs the discovered neighbors and the corresponding discovery latency, which both are needed for constructing the network and achieving other functions.

### B. Energy Module

The energy module receives the remaining energy and the latency constraints from the application module as inputs. The output is the duty cycle which the node is to adopt. Two main functions are incorporated into the energy module: *computing the node's lifetime* and *adjusting the duty cycle*.

For any node $u_i$ which starts at time $t_i^s$, suppose the generated schedule is $S_i = \{s_i(t) | t \geq t_i^s\}$ and it runs out of energy at time $t_i^e$. We derive the following equation as:

$$\int_{t_i^s}^{t_i^e} s_i(t) \cdot p_n = P_{max} \tag{3}$$

Then, the lifetime of node $u_i$ (denoted as $Lf_i$) is:

$$Lf_i = t_i^e - t_i^s \tag{4}$$

If node $u_i$ turns the radio on all the time, Eqn. (3) can be rewritten as:

$$(t_i^e - t_i^s) \cdot p_n = P_{max}$$

and the lifetime is $Lf_i = \frac{P_{max}}{p_n}$, which is the minimum value.

To extend the lifetime, node $u_i$ turns on the radio for a fraction of the time. If node $u_i$ selects the duty cycle as a fixed value $\hat{\theta} \in (0, 1)$, rewrite Eqn. (3) as:

$$(t_i^e - t_i^s) \cdot \hat{\theta} \cdot p_n + (t_i^e - t_i^s) \cdot (1 - \hat{\theta}) \cdot p_f \approx P_{max}$$

We use '$\approx$' since the schedule may not be a complete cycle, but it makes very little difference, and we can just regard it as '='. As we assume $p_f = 0$, the lifetime of node $u_i$ is computed as:

$$Lf_i = \frac{P_{max}}{p_n \cdot \hat{\theta}} \tag{5}$$

Suppose node $u_i$ adjusts the duty cycle timely, denote the time that node $u_i$ changes the duty cycle as: $T_0 < T_1 < \ldots < T_m$ where $T_0 = t_i^s$ and $T_m < t_i^e$. For simplicity, denote $T_{m+1} = t_i^e$ and Eqn. (3) is rewritten as:

$$\begin{aligned} & \textstyle\sum_{k=0}^{m} (\int_{T_k}^{T_{k+1}} s_i(t) \cdot p_n) = P_{max} \\ \Rightarrow \quad & \textstyle\sum_{k=0}^{m} (T_{k+1} - T_k) \cdot \theta_i(T_k, T_{k+1}) \cdot p_n = P_{max} \end{aligned}$$

Since $\theta_i(T_k, T_{k+1})$ and $T_i, i \in [0, m]$ are known beforehand, $t_i^e = T_{m+1}$ is computed as:

$$t_i^e = \frac{\frac{P_{max}}{p_n} - \sum_{k=0}^{m-1} (T_{k+1} - T_k) \cdot \theta_i(T_k, T_{k+1})}{\theta_i(T_m, T_{m+1})} + T_m \tag{6}$$

Then, the lifetime can be computed. In Section V, we analyze the impact of the node's lifetime by different strategies that adjust the duty cycles.

### C. Application Module

Wireless sensor networks are used in many applications and there could be many different requirements for the nodes. For example, when a node detects an emergency, such as a very low temperature, a moving enemy, etc., it needs to inform the whole network quickly. We regard these requirements as latency constraints. That is, the node has to discover the neighbors within a bounded latency. Therefore, in order to send out the information quickly, the node has to increase the duty cycle and turn on the radio more frequently. The application module passes the latency constraints to the energy module and the algorithmic module. Two main functions are implemented in the module (for node $u_i$):

1) To collect latency constraints at time $t$, such that the discovery latency should be bounded within $\hat{L}_i(t)$;
2) to compute an appropriate duty cycle according to the latency constraint.

TABLE II
ALGORITHMS COMPARISON FOR TWO NEIGHBORS

| Algorithms | DC 1 | DC 2 | Latency | Asymmetric? |
|---|---|---|---|---|
| Quorum [17] | $\theta$ | $\theta$ | $\frac{4}{\theta^2}$ | No |
| LL-Optimal [22] | $\theta$ | $\theta$ | $\frac{1}{\theta^2}$ | No |
| Disco [6] | $\theta_1$ | $\theta_2$ | $\frac{4}{\theta_1\theta_2}$ | Yes |
| U-Connect [9] | $\theta_1$ | $\theta_2$ | $\frac{9}{4\theta_1\theta_2}$ | Yes |
| Searchlight [2] | $\theta_1$ | $\theta_2$ | $\frac{2}{\theta_1\theta_2}$ | Yes |
| C-Torus [4] | $\theta_1$ | $\theta_2$ | $\frac{9}{4\theta_1\theta_2}$ | Yes |
| BlindDate [20] | $\theta_1$ | $\theta_2$ | $\frac{9}{5\theta_1\theta_2}$ | Yes |
| Hedis [5] | $\theta_1$ | $\theta_2$ | $\frac{4}{\theta_1\theta_2}$ | Yes |
| Todis [5] | $\theta_1$ | $\theta_2$ | $\frac{9}{\theta_1\theta_2}$ | Yes |
| Hello [16] | $\theta_1$ | $\theta_2$ | $\frac{(c_1+1)(c_2+1)}{c_1c_2\theta_1\theta_2}$ | Yes |

**Remarks:**
1) 'DC' is short for duty cycle; we use $\theta$ for symmetric duty cycle and $\theta_1, \theta_2$ for asymmetric duty cycle;
2) Hello is a little different from other algorithms, where there are two parameters to choose;
3) some results of discovery latency are modified (or simplified) on the basis of symmetric analyses.

### D. Algorithmic Module

Once the duty cycle is adjusted by the energy module or the application module, the node has to invoke the algorithmic module to compute the discovery schedule for the coming time slots. The interface involves duty cycle and latency constraints as inputs, and outputs the discovery schedule. Notice that, Spear is designed for the practical networks and the nodes could adjust the duty cycle locally. Therefore, the implemented algorithms should be applicable for asymmetric duty cycles. We summarize the state-of-the-art algorithms by considering the relationship between duty cycles and discovery latency in Table II.

### E. Communication Module

The node carries out the operations according to the generated schedule by the algorithmic module. The target of the communication module is to discover the neighbors when collisions exist among multiple nodes. We summarize the two main functions that are implemented:

1) Discover the neighbors and record the neighbors' information, such as the identifier, the start time, and the duty cycle;
2) compute the corresponding discovery latency of the neighbors.

When we deploy existing algorithms for multiple nodes, communication collisions often occur and many nodes cannot discover their neighbors. In Section VI, we devise two new methods to reduce the collisions, and the algorithms modified by the methods can achieve good performances.

### F. Measurements

In the paper, we utilize three metrics to evaluate the algorithms. Considering each node $u_i$,

1) **Lifetime** $Lf_i$ reveals how long the node can survive;
2) **Discovery latency** $L(i, N_i)$ is the number of time slots ($t_0$) to discover all neighbors;
3) **Discovery rate** is the percentage of discovered neighbors in $N_i$ for a bounded latency.

Lifetime and discovery latency are commonly adopted in the existing works. Due to communication collisions, some nodes may not be able to find all neighbors, we introduce discovery rate for evaluation.

## V. METHODS OF ADJUSTING DUTY CYCLE

Neighbor discovery is affected by nodes' duty cycles. Existing works have designed efficient discovery schedules for fixed duty cycles, but few of them study how to adjust the duty cycle during a node's lifetime. In this section, we present several methods to adjust the duty cycle according to the remaining energy and the latency constraints.

### A. Energy Management Methods

As shown in Eqn. (5), node $u_i$'s lifetime is $Lf_i = \frac{P_{max}}{p_n \cdot \theta}$ if it sticks to duty cycle $\theta$ all the time. To extend the lifetime, it could reduce the duty cycle when the remaining energy is depleting. We propose two methods to adjust the duty cycle.

**Piece-wise Reducing (PWR) Method**: Generate $m$ different energy levels as $P_{max} = \hat{P}_1 > \hat{P}_2 > \ldots \hat{P}_m > 0$ and $m$ corresponding duty cycle levels as $\hat{\theta}_1 > \hat{\theta}_2 > \ldots > \hat{\theta}_m$ in advance. When the remaining energy drops down to $\hat{P}_j$, the node adjusts the duty cycle to $\hat{\theta}_j$. For simplicity, denote $\hat{P}_{m+1} = 0$ and node $u_i$ selects duty cycle at time $t$ as:

$$\theta_i(t) = \hat{\theta}_j, \ if \ P_i(t) \in (\hat{P}_{j+1}, \hat{P}_j] \tag{7}$$

The lifetime of node $u_i$ is computed as:

$$Lf_i \quad = \sum_{j=1}^{m} \frac{\hat{P}_j - \hat{P}_{j+1}}{p_n \cdot \hat{\theta}_j} \tag{8}$$

The PWR method can extend the node's lifetime compared to Eqn. (5), but it has to generate different levels of energy and duty cycles beforehand. We propose another method which is much easier to implement.

**Periodical Reducing (PDR) Method**: Node $u_i$ selects an initial duty cycle $\hat{\theta}_0$ when it starts (with energy $P_{max}$). The node adjusts the duty cycle every $\hat{T}$ time slots according to the remaining energy, where $\hat{T}$ is a fixed constant. Suppose the remaining energy at time $t$ is $P_i(t)$, the duty cycle is reduced as:

$$\frac{P_i(t)}{\theta_i(t)} = \frac{P_{max}}{\hat{\theta}_0} \tag{9}$$

When the remaining energy is very low ($P_i(t) \leq P_{min}$ where $P_{min}$ is a small constant), the node has to fix the duty cycle as $\hat{\theta}_{min} = \frac{P_{min}}{P_{max}} \cdot \hat{\theta}_0$. In order to compute the lifetime, it is necessary to compute the number of times that the node adjusts the duty cycle. Suppose after $m$ periods of length $\hat{T}$, the remaining energy is no larger than $P_{min}$, the following

equations are derived:

$$\begin{cases} \hat{P}_0 = P_{max}, & \hat{\theta}_0 = \hat{\theta}_0 \\ \hat{P}_1 = \hat{P}_0 - \hat{T} \cdot \hat{\theta}_0 \cdot p_n, & \hat{\theta}_1 = \frac{\hat{P}_1}{\hat{P}_0} \cdot \hat{\theta}_0 \\ \quad \vdots & \quad \vdots \\ \hat{P}_i = \hat{P}_{i-1} - \hat{T} \cdot \hat{\theta}_{i-1} \hat{p}_n, & \hat{\theta}_i = \frac{\hat{P}_i}{\hat{P}_0} \cdot \hat{\theta}_0 \\ \quad \vdots & \quad \vdots \\ \hat{P}_m = \hat{P}_{m-1} - \hat{T} \cdot \hat{\theta}_{m-1} \cdot p_n, & \hat{\theta}_m = \frac{P_{min}}{\hat{P}_0} \cdot \hat{\theta}_0 \end{cases} \tag{10}$$

Combine these to derive:

$$\begin{aligned} \hat{P}_m = \quad & \hat{P}_0[1 - \tbinom{m}{1}\frac{\hat{T}\hat{\theta}_0 p_n}{\hat{P}_0} + \tbinom{m}{2}(\frac{\hat{T}\hat{\theta}_0 p_n}{\hat{P}_0})^2 + \cdots \\ & + \cdots + (-1)^m \tbinom{m}{m}(\frac{\hat{T}\hat{\theta}_0 p_n}{\hat{P}_0})^m] \\ = \quad & \hat{P}_0(1 - \frac{\hat{T}\hat{\theta}_0 p_n}{\hat{P}_0})^m \end{aligned} \tag{11}$$

When $\hat{P}_m \leq P_{min}$, the number of periods is:

$$m \geq \frac{\log(P_{min}/P_{max})}{\log(1 - \hat{T}\hat{\theta}_0 p_n / P_{max})}$$

and the lifetime of node $u_i$ is:

$$Lf_i = m \cdot T + \frac{\hat{P}_m P_{max}}{P_{min}\hat{\theta}_0 p_n} \tag{12}$$

Both PWR and PDR methods could prolong the node's lifetime and we evaluate them in Section VII.

### B. Latency Constraints

When the applications have latency constraints, such as fast streaming or real time detection applications, the node has to increase the duty cycle in order to discover the neighbors in bounded time. However, discovery latency between two neighbors is determined by the chosen algorithm and the duty cycles of both nodes.

As listed in Table II, different algorithms lead to different discovery latencies. Take Disco [6] as an example. Given latency constraint $\hat{L}_i(t)$ at time $t$ for node $u_i$, it can check the recorded information of the discovered neighbors. Suppose one neighbor $u_j$'s duty cycle is $\hat{\theta}_j$, node $u_i$ has to increase its duty cycle as: $\hat{\theta}_i \geq \frac{4}{\hat{\theta}_j \hat{L}_i(t)}$. In order to discover all neighbors, node $u_i$ has to check the smallest duty cycle (denote it as $\hat{\theta}_m$) and it has to increase the duty cycle as $\hat{\theta}_i \geq \frac{4}{\hat{\theta}_m \hat{L}_i(t)}$. After satisfying the application requirements, node $u_i$ can then adjust the duty cycle by the remaining energy as described above. In order to reduce communication collisions, the duty cycle has to be larger.

Combining remaining energy and latency constraints, the duty cycle should be adjusted by both factors; that is, to design a function of adjusting the duty cycle as:

$$\theta_i(t) = f(P_i(t), \hat{L}_i(t)) \tag{13}$$

When there is no latency constraint, $\hat{L}_i(t) = +\infty$, PWR and PDR are two representative examples. In Spear, researchers could implement the interface for evaluating more functions which can timely adjust the duty cycle.

## VI. Methods of Reducing Collisions

In real communication scenarios, two neighboring nodes can communicate successfully only when they are not interfered by other nodes. If existing deterministic algorithms are extended to handle multiple nodes directly, communication collisions happen and most nodes cannot find the neighbors. In this section, we analyze the discovery probability and propose two methods to reduce the collisions.

### A. Discovery Probability under Collision

Considering two neighboring nodes $u_i, u_j$, denote the sets of each node's neighbors as $N_i, N_j$ respectively ($u_i \in N_j, u_j \in N_i$). Suppose nodes $u_i, u_j$ turn on their radio at time $t$, and denote the sets of neighbors that also turn on the radio as $\tilde{N}_i \subseteq N_i, \tilde{N}_j \subseteq N_j$. Since $u_j \in \tilde{N}_i, u_i \in \tilde{N}_j$, $u_i$ and $u_j$ can discover each other only when:

$$|\tilde{N}_i| = 1, |\tilde{N}_j| = 1$$

Denote the average duty cycle for node $u_k$ as $\theta_k$. We consider the scenario where node $u_k$ turns on the radio with probability $\theta_k$ independently in each time slot (expected situation). Then, on the basis of the event that nodes $u_i, u_j$ turn on the radio at time $t$, the probability for successful discovery is derived as:

$$\begin{aligned} & Pr(|\tilde{N}_i| = 1, |\tilde{N}_j| = 1) \\ \leq \quad & \min\{Pr(|\tilde{N}_i| = 1), Pr(|\tilde{N}_j| = 1)\} \\ = \quad & \min\{\textstyle\prod_{u_k \in N_i, k \neq j}(1 - \theta_k), \prod_{u_k \in N_j, k \neq i}(1 - \theta_k)\} \end{aligned}$$

If $\theta_k = 1\%$ and $\max\{|N_i|, |N_j|\} \geq 69$ (or $\theta_k = 10\%$ and $\max\{|N_i|, |N_j|\} \geq 7$), the probability of successful discovery is less than $1/2$. Therefore, the existing algorithms cannot be applied to multiple nodes directly. We adopt the idea of the probabilistic protocols to reduce the communication collisions, two efficient methods are proposed.

### B. Pure Probability Reducing Method

The **Pure Probability Reducing** $PPR$ Method works as follows. For any deterministic neighbor discovery algorithm $f$, denote the generated discovery schedule for node $u_i$ as $S_i = \{s_i(t) | t \geq t_i^s\}$. For any time $t$ that $s_i(t) = 1$, node $u_i$ turns on the radio with probability $p_1$ (a constant value in $(0, 1)$); that is, to generate a modified sequence $\tilde{S}_i = \{\tilde{s}_i(t) | t \geq t_i^s\}$ as:

$$\begin{cases} If\ s_i(t) = 0, & \tilde{s}_i(t) = 0 \\ If\ s_i(t) = 1, & \tilde{s}_i(t) = 1\ with\ probability\ p_1 \end{cases}$$

If two neighbors $u_i, u_j$ turn on the radio at time $t$, the expected probabilities of $|\tilde{N}_i| = 1$ and $|\tilde{N}_j| = 1$ are:

$$\begin{aligned} Pr(|\tilde{N}_i| = 1) \quad &= \textstyle\prod_{u_k \in N_i, k \neq j}(1 - p_1 \cdot \theta_k) \\ Pr(|\tilde{N}_j| = 1) \quad &= \textstyle\prod_{u_k \in N_j, k \neq i}(1 - p_1 \cdot \theta_k) \end{aligned}$$

If $p_1 = 0.5$, $\theta_k = 1\%$, the probability of successful discovery is less than $1/2$ when $\max\{|N_i|, |N_j|\} \geq 139$. By choosing different values of $p_1$, the performances could be different. We evaluate the sensitivity of $p_1$ in Section VII.

### C. Decreased Probability Reducing Method

The $PPR$ method can increase the discovery probability, but it is independent of the schedule itself. We present the **Decreased Probability Reducing** ($DPR$) method, which tries to coordinate any discovery schedule with the method. For the generated discovery schedule $S_i$ by any algorithm $f$, node $u_i$ should turn on the radio at time $t_1$ when $s_i(t_1) = 1$. Denote the next time slot that $u_i$ turns on the radio by schedule $S_i$ as $t_2$, i.e. $s_i(t_2) = 1, t_2 > t_1$. Modify the schedule of time slots $[t_1, t_2)$ as:

- Change $s_i(t) = 0$ for $t \in [t_1, t_2)$;
- increase $t^*$ from $t_1$ to $t_2 - 1$, if $s_i(t) = 0$ for all $t \in [t_1, t^*)$, set $s_i(t^*) = 1$ with probability $p_2 \cdot \frac{t_2 - t^*}{t_2 - t_1 + 1}$ where $p_2$ is a constant value in $(0, 1)$.

Node $u_i$ turns on the radio in the initial slot with probability $p_2 \cdot (1 - \frac{1}{t_2 - t_1 + 1})$, and it could reduce the probability of collisions. If $u_i$ does not turn on the radio at time $t_1$, it decreases the probability and attempts to turn on the radio in the next slot, $t_1 + 1$. This process does not finish until $u_i$ turns on the radio in any slot within $[t_1, t_2)$, or it keeps the radio off for all of them.

Overall, both PPR and DPR are designed to support deterministic neighbor discovery protocols. Unlike probabilistic-based protocols, the neighbor discovery schedules computed by both PPR and DPR are based on the schedules generated by deterministic protocols. Therefore, a deterministic protocol running with PPR or DPR can achieve a stable discovery latency (confirmed in Section VII-A).

## VII. Evaluations

We have implemented Spear in C++, which includes the interfaces between different modules, important functions for computing in each module, and measurements to evaluate the performances. We implemented four state-of-the-art algorithms including Hedis [5], Hello [16], Searchlight [2] and U-Connect [9] (we also implemented Quorum [17], but the result is only presented in Fig. 9 since it is inapplicable when the users' duty cycle are different), and run these algorithms in a cluster with 9 servers, each equipped with an Intel Xeon 2.6GHz CPU with 24 hyper-threading cores, 64GB memory and 1T SSD. The basic settings in the simulations are: $d_c = 50m$, $P_{max} = 100000, p_n = 1$ and $t_0 = 20ms$. We choose three scenarios for comparison:

1) Discovery in a star network. The central node $u_c$ has $|N_c|$ neighbors in the star network. A neighboring node selects the duty cycle randomly within $[0.1, 0.5]$, while $u_c$'s duty cycle ($\theta_c$) is set to different figures;
2) Discovery among $N = 1000$ nodes. The area is set as a rectangle of size $1000 \times 1000m^2$, and the node's coordinates are generated randomly. Each node selects the duty cycle randomly within $[0.1, 0.5]$.
3) Discovery between two neighbors. Spear enables the evaluation for existing protocols and generates the best schedule for fixed duty cycles.

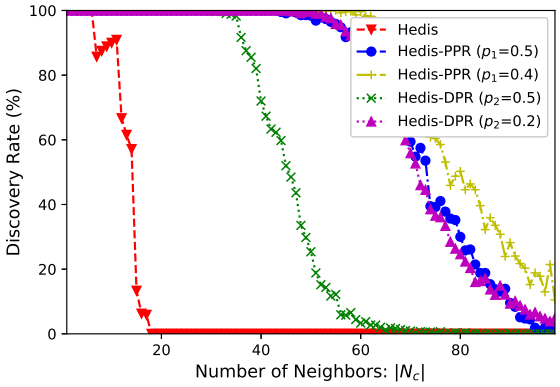
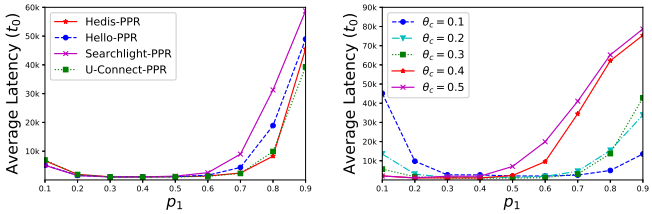



Fig. 3. Spear increases discovery rate by incorporating PPR and DPR.




(a) $\theta_c$=0.3 (b) U-Connect-PPR

Fig. 4. Sensitivity of $p_1$ in the PPR method.

We evaluated average discovery latency, lifetime, or percentage of discovery under different settings, and we describe the detailed parameters for each figure. The start time of any node is generated randomly within $[0, 1000]$ and the results are based on 1000 separate runs.

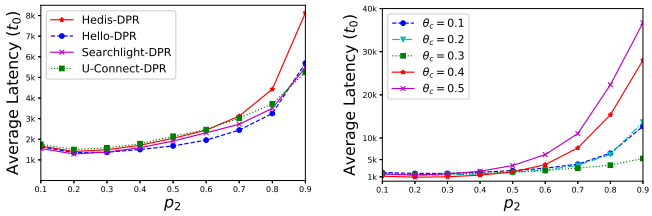
### A. Increasing Discovery Rate

In the network with multiple nodes, communication collisions could affect the discovery results. We evaluate the performance of the proposed collision reducing methods (PPR and DPR in Section VI), and compare them with the bare (not running in Spear) algorithms.

**Number of neighbors**. In a star network, the central node $u_c$ has $|N_c|$ neighbors and it selects duty cycle $\theta_c = 0.3$. We select Hedis as the example and set $p_1 = p_2 = 0.5$ for PPR and DPR. Fig. 3 shows the discovery rate (y-axis) of $u_c$ within 100000 time slots when $|N_c|$ (x-axis) increases from 1 to 100. From the figure, (bare) Hedis cannot discover all neighbors when $|N_c| \geq 7$, and it cannot find even one neighbor when $|N_c| \geq 18$. Modified by PPR and DPR, all neighbors can be discovered when $|N_c| \leq 43$ and $|N_c| \leq 33$ respectively. We also evaluate the performance of PPR and DPR at $p_1 = 0.4$ and $p_2 = 0.2$, and they outperform the methods when $p_1 = p_2 = 0.5$.

**Sensitivity of** $p_1$. In a star network, the central node $u_c$ has $|N_c| = 20$ neighbors[3] and we evaluate PPR's performance under different values of $p_1$. As shown in Fig. 4(a), $\theta_c$ is set to 0.3 and the average discovery latency (y-axis) of different algorithms are changed by different values of $p_1$ (x-axis). When $p_1 \in [0.3, 0.4]$, the performance is better. In

[3]We set $|N_c| = 20$ since the bare algorithms fail to find any neighbor.




(a) $\theta_c$=0.3 (b) U-Connect-DPR

Fig. 5. Sensitivity of $p_2$ in the DPR method.

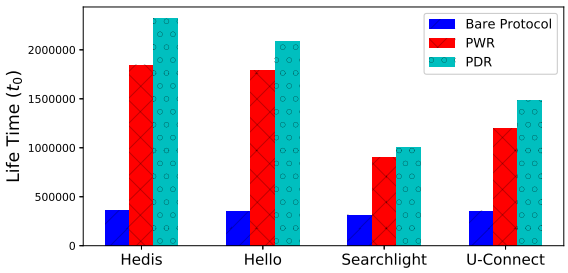
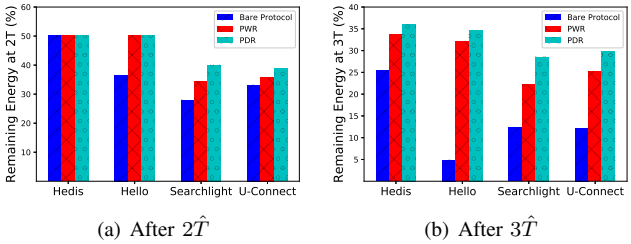



Fig. 6. Spear prolongs the lifetime with PWR and PDR.




(a) After $2\hat{T}$ (b) After $3\hat{T}$

Fig. 7. Spear saves more energy by incorporating PWR and PDR compared to bare algorithms.

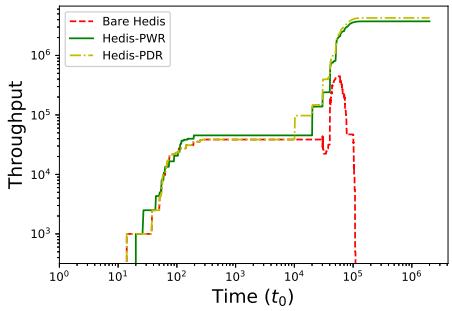



Fig. 8. Spear improves higher throughput for routing (both x-axis and y-axis are in log scale).

Fig. 4(b), we select U-Connect as the example and set $\theta_c$ as $0.1, 0.2, 0.3, 0.4, 0.5$ respectively. The average discovery latency is changed by different values of $p_1$ (x-axis), and the performance is also better when $p_1$ approaches $[0.3, 0.4]$. Overall, our discovery latency is stable when $p_1 \leq 0.6$.

**Sensitivity of** $p_2$. In a star network, the central node $u_c$ has $|N_c| = 20$ neighbors and the DPR's performance is evaluated under different values of $p_2$. As shown in Fig. 5(a), $\theta_c$ is set to 0.3 and the average discovery latency (y-axis) of different algorithms are changed by different values of $p_2$ (x-axis). When $p_2$ is close to 0.2, the performance is better. In Fig. 5(b), we select U-Connect for different $\theta_c$ $(0.1, 0.2, 0.3, 0.4, 0.5$ respectively). The average discovery latency (y-axis) is changed by different values of $p_2$ (x-axis), and the performance is better when $p_2 \approx 0.2$. Overall, our discovery latency is stable when $p_2 \leq 0.7$.

1000 **nodes**. In a randomly generated network with $N = 1000$ nodes, we evaluate the performances of different algorithms. As shown in Fig. 1, modified by PPR ($p_1 = 0.4$) and DPR ($p_2 = 0.2$), the discovery rates (y-axis) are much larger than the bare algorithms. Especially for Hello, the discovery rate is only $33.0\%$, while PPR and DPR could greatly increase the rate to $99.2\%, 95.5\%$ respectively.

### *B. Prolonging Lifetime*

The two methods (PWR and PDR) that can extend a node's lifetime are also implemented in Spear. We evaluate the performance in a star network where the central node $u_c$ has $|N_c| = 20$ neighbors. In PWR, $m = 30$ levels of remaining energy and corresponding duty cycles are generated in advance. Due to the page limit, we cannot list these values here. In PDR, $\hat{\theta}_0$ is set to 0.3, $P_{min} = 200$, and the node adjusts the duty cycle every $\hat{T} = 100000$ time slots.

**Lifetime**. The lifetime of $u_c$ is illustrated in Fig. 6 for different algorithms, both PWR and PDR prolong the lifetime significantly. For example, the lifetimes of PWD and PDR are $5.15$ and $6.47$ times longer than bare Hedis respectively.

**Percentage of Remaining Energy**. We show the percentage of remaining energy (y-axis) after $2\hat{T}$ and $3\hat{T}$ time slots in Fig. 7. After $2\hat{T}$ time slots, PWR and PDR are just a little better than bare Hedis protocol (see Fig. 7(a)), while the difference becomes much larger after $3\hat{T}$ time slots (see Fig. 7(b)). By incorporating PWR and PDR, Spear can save power and extend the lifetime significantly.

**Overhead and Benefit on routing**. In each time slot, every node relays 500KB packets to the discovered neighbors; the throughput of the network is the total amount of transmitted bytes in each slot [13]. We compare the throughput of the network constructed by bare Hedis and networks constructed by Hedis running with PWR and PDR respectively. Fig. 8 shows the results. When running bare Hedis, nodes run out of energy quickly and the throughput decreases dramatically. In contrast, Hedis-PWR and Hedis-PDR extend both the lifetime of nodes by more than 10 times and they discover many more neighbors. Therefore, the throughput of Hedis-PWR and Hedis-PDR could even be $11.24$ and $9.80$ times higher than the level of bare Hedis respectively. However, Hedis-PWR and Hedis-PDR take more time to initiate routes and to reach peak throughput. The reason is that PWR and PDR adaptively lower Hedis' duty cycle to save energy, thus they incur higher discovery latency than that of bare Hedis. This overhead is moderate and worthwhile because Spear greatly improves throughput and lifetime. Other protocols running in Spear have similar overhead.

### *C. Improving Discovery for Two Neighbors*

Spear enables the evaluation of neighbor discovery protocols supporting symmetric and asymmetric duty cycles, which facilitates the generation of the optimal schedule.

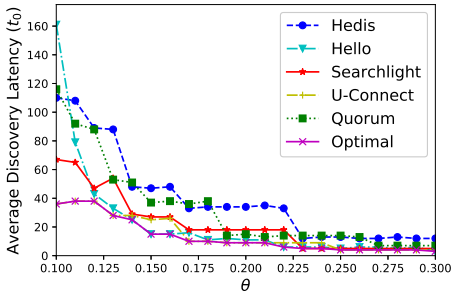



Fig. 9. Spear enables the evaluation of the neighbor discovery algorithms for symmetric duty cycle and generates the optimal schedule automatically.

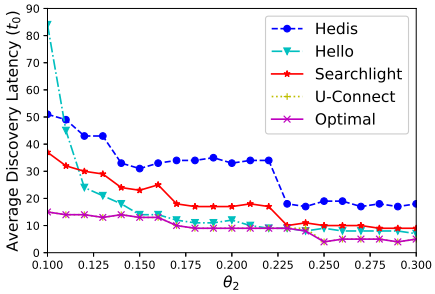



Fig. 10. Spear enables the evaluation of the neighbor discovery algorithms for asymmetric duty cycle and generates the optimal schedule automatically.

**Symmetric duty cycle**: suppose two neighbors select the same duty cycle $\theta$ that increases from 0.1 to 0.3, we evaluate the average discovery latency in Fig. 9. As shown in the figure, the discovery latency (y-axis) of all algorithms decreases as $\theta$ (x-axis) increases, and they have similar performances. This is because the discovery latency is proportional to $\frac{1}{\theta^2}$ and the trends of these curves match the analysis. Finally, the optimal schedule can be generated automatically; for example U-Connect is selected when $\theta \in [0.1, 0.13]$, while Hello is selected when $\theta \in [0.13, 0.16]$.

**Asymmetric duty cycle**: suppose one node's duty cycle is fixed as $\theta_1 = 0.2$ and the other node's duty cycle $\theta_2$ increases from 0.1 to 0.3. Since Quorum is inapplicable for asymmetric duty cycles, we compare the other four algorithms. As shown in Fig. 10, the average discovery latency (y-axis) decreases as $\theta_2$ (x-axis) increases, and the decreasing trends are much more gently than Fig. 10. This is because the discovery latency is proportional to $\frac{1}{\theta_2}$ when $\theta_1$ is a constant.

### D. Effectiveness of Spear Components

In Spear, the communication module evidently increases the discovery rate compared to the bare protocols, as demonstrated by the evaluations in Section VII-A. The energy module significantly extends a node's lifetime, as described in Section VII-B. Though we did not evaluate the application module separately, the evaluations targeting at discovery latency and discovery rate imply that Spear could adjust the related parameters (such as duty cycle, $p_1$ of PPR, and $p_2$ of DPR) to reduce the discovery latency. The algorithmic module computes an optimal schedule automatically as shown in Section VII-C.

## VIII. Conclusion

We have presented Spear, the first practical framework for general neighbor discovery protocols in wireless sensor networks. Extensive evaluations have shown that Spear can greatly increase the discovery rate and extend the nodes' lifetime. Spear has the potential to be applied to tackling broad problems in wireless sensor networks.

## References


[1] L. Atzori, A. Lera, and G. Morabito. The Internet of Things: A Survey. In *Computer Networks*, 54(15), pp. 2787-2805, 2010.

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

[3] K. Bian, J.-M. Park, and R. Chen. A Quorum-Based Framework for Establishing Control Channels in Dynamic Spectrum Access Networks. In *Mobicom*, 2009.

[4] L. Chen, B. Yan, J. Zhang, Neighbor Discovery Algorithm in Mobile Low Duty Cycle WSNs. In *Journal of Software*, 2014.

[5] L. Chen, R. Fan, K. Bian, L. Chen, M. Gerla, T. Wang, and X. Li. On Heterogeneous Neighbor Discovery in Wireless Sensor Networks. In *INFOCOM*, 2015.

[6] P. Dutta, and D. Culler. Practical Asynchronous Neighbor Discovery and Rendezvous for Mobile Sensing Applications. In *SenSys*, pp. 71-84, 2008.

[7] Z. Gu, Q.-S. Hua, Y. Wang, and F. C.M. Lau. Reducing Information Gathering Latency through Mobile Aerial Sensor Network. In *INFOCOM*, 2013.

[8] J.-R. Jiang, Y.-C. Tseng, C.-S. Hsu, and T.-H. Lai. Quorum-based Asynchronous Power-Saving Protocols for IEEE 802.11 Ad Hoc Networks, *Mobile Networks and Applications*, 10(1-2): 169-181, 2005.

[9] A. Kandhalu, K. Lakshmanan, and R. R. Rajkumar. U-Connect: A Lower Latency Energy-Efficient Asynchronous Neighbor Discovery Protocol. In *IPSN*, 2010.

[10] S. Lai, B. Ravindran, and H. Cho. Heterogeneous Quorum-based Wake-up Scheduling in Wireless Sensor Networks. *IEEE Transaction on Computers*, 59(11): 1562-1575, 2010.

[11] R. Margolies, G. Grebla, T. Chen, D. Rubenstein, and G. Zussman. Panda: Neighbor Discovery on a Power Harvesting Budget. In *INFOCOM*, 2016.

[12] M. J. McGlynn, and S. A. Borbash. Birthday Protocols for Low Energy Deployment and Flexible Neighbor Discovery in Ad Hoc Wireless Networks. In *MobiHoc*, 2001.

[13] G. Miao, J. Zander, K. W. Sung, and B. Slimane. Fundamentals of Mobile Data Networks. Combridge University Press, 2016.

[14] M. B. Nathanson. Elementary Methods in Number Theory. Vol. 195, Springer, 2000.

[15] Talk More Listen Less: Energy-Efficient Neighbor Discovery in Wireless Sensor Networks. In *INFOCOM*, 2016.

[16] W. Sun, Z. Yang, X. Zhang, and Y. Liu. Hello: A Generic Flexible Protocol for Neighbor Discovery. In *INFOCOM*, 2014.

[17] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh. Power-Saving Protocols for IEEE 802.11-based Multi-Hop Ad Hoc Networks. In *Computer Networks*, 43(3), pp. 317-337, 2003.

[18] S. Vasudevan, D. Towsley, D. Goeckel, and R. Khalili. Neighbor Discovery in Wireless Networks and the Coupon Collector's Problem. In *MobiCom*, 2009.

[19] Y. Wang, Y. Wang, X. Qi, L. Xu, J. Chen, and G. Wang. L3SN: A Level-Based, Large-Scale, Longevous Sensor Network System for Agriculture Information Monitoring. In *Wireless Sensor Networks*, 2010.

[20] K. Wang, X. Mao, and Y. Liu. BlindDate: A Neighbor Discovery Protocol. In *TPDS*, 26(4), pp. 949-959, 2015.

[21] X. Xu, J. Luo, and Q. Zhang. Delay Tolerance Event Collections in Sensor Networks with Mobile Sink. In *INFOCOM*, 2010.

[22] R. Zheng, J. C. Hou, and L. Sha. Asynchronous Wakeup for Ad Hoc Networks. In *MobiHoc*, 2003.

[23] W. Zeng, S. Vasudevan, X. Chen, B. Wang, A. Russel, and W. Wei. Neighbor Discovery in Wireless Networks with Multipacket Reception. In *MobiHoc*, 2011.