sensors

MDPI

Article

A Practical Neighbor Discovery Framework for Wireless Sensor Networks

Zhaoquan Gu [1,*], Yuexuan Wang [2,3,*], Wei Shi [4], Zhihong Tian [1,*] and Kui Ren [5] and Francis C.M. Lau [3]

1 Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China
2 College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China
3 Department of Computer Science, The University of Hong Kong, Hong Kong, China; fcmlau@csis.hku.hk
4 School of Information Technology, Carleton University, Ottawa, ON K1S5B6, Canada; wei.shi@carleton.ca
5 Institute of Cyberspace Research, Zhejiang University, Hangzhou 310027, China; kuiren@zju.edu.cn
* Correspondence: zqgu@gzhu.edu.cn (Z.G.); amywang@zju.edu.cn (Y.W.); tianzhihong@gzhu.edu.cn (Z.T.)

Received: 3 April 2019; Accepted: 17 April 2019; Published: 20 April 2019

Abstract: Neighbor discovery is a crucial operation frequently executed throughout the life cycle of a Wireless Sensor Network (WSN). Various protocols have been proposed to minimize the discovery latency or to prolong the lifetime of sensors. However, none of them have addressed that all the critical concerns stemming from real WSNs, including communication collisions, latency constraints and energy consumption limitations. In this paper, we propose Spear, the first practical neighbor discovery framework to meet all these requirements. Spear offers two new methods to reduce communication collisions, thus boosting the discovery rate of existing neighbor discovery protocols. Spear also takes into consideration latency constraints and facilitates timely adjustments in order to reduce the discovery latency. Spear offers two practical energy management methods that evidently prolong the lifetime of sensor nodes. Most importantly, Spear automatically improves the discovery results of existing discovery protocols, on which no modification is required. Beyond reporting details of different Spear modules, we also present experiment evaluations on several notable neighbor discovery protocols. Results show that Spear greatly improves the discovery rate from 33.0% to 99.2%, and prolongs the sensor nodes lifetime up to 6.47 times.

Keywords: neighbor discovery; wireless sensor networks; communication collision; latency; energy consumption

1. Introduction

With the ascent of Internet-of-Things (IoT) [1–7], wireless sensor networks (WSNs) are increasingly being adopted for tracking and monitoring applications in various areas such as health-caring, smart buildings, agricultural management and assisted living [8–10]. For example, WSN has been deployed for agriculture information monitoring [11], and sensors can be attached to inventory items in a large warehouse for object identification [12].

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 [12–27]. Sensor nodes are powered by batteries. To minimize the overall energy consumption, most existing work focuses on designing discovery schedules that maintain a low *duty cycle*. However, despite decades of efforts, designing practical neighbor discovery protocols for real-life WSNs remains a big challenge, especially when considering all three critical factors as pointed out by previous works:

1. Collisions happen when multiple nodes transmit on the same communication channel simultaneously.
2. Bounded latency is required for time-sensitive applications. For example, object detection applications require discovering the neighbors for transmitting emergent information with very low latency.
3. Prolonged lifetime of sensor nodes. In order to reduce the energy consumption and prolong the lifetime of sensor nodes, affective energy management methods that can dynamically adjust nodes' duty cycles during the discovery process are crucial.

Neglecting all these practical concerns, sensor nodes will likely fail to discover all neighbors before running out of energy.

Unfortunately, no existing work has considered all of the above-mentioned concerns in one setting. Typically, only one factor is considered at a time. Among all solutions, two major types of protocols exist: probability-based and deterministic ones. Probability-based protocols turn on the radio with different probabilities to reduce communication collisions [20,25,27–29], but the discovery latency often varies significantly. Compared to probability-based protocols, deterministic protocols are much more popular in practice. This approach is also the main focus of this paper because with a deterministic discovery schedule the discovery latency is mostly stable [12,13,15,16,18,21,22,26,30,31]. However, existing deterministic protocols mainly target at only two neighbors, where collision often does not occur. As shown in Figure 1, when these (bare) deterministic protocols (Hedis [15], Hello [22], Searchlight [13], and U-Connect [18]) are adopted in a network of 1000 nodes, the discovery rates are only 33.0–65.9% because of collisions.

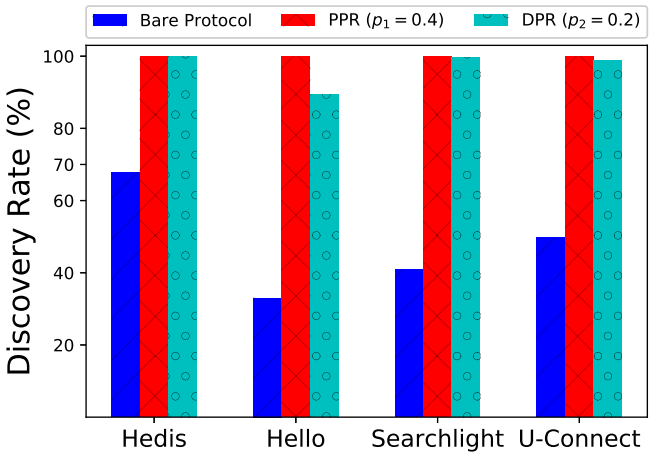

Figure 1. Spear greatly improves discovery rates for four notable protocols. Discovery rate for each protocol is defined as the number of discovered neighbors divided by all actual neighbors. Bare protocols mean they do not run in Spear.

In this paper, we propose **Spear** (Spear is a general, powerful ancient weapon), 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 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 Figure 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.

The main contributions of Spear are automatic reduction of communication collisions, improvement of discovery rate, and prolonging lifetime for neighbor discovery, which together make the construction of WSNs much more effective and easier. Furthermore, Spear can be broadly applied to tackle various problems in WSNs.

The rest of the paper is organized as follows. We introduce relevant neighbor discovery protocols in Section 2, and the preliminaries in Section 3. We describe Spear in detail in Section 4. Methods that manage node energy and handle the latency requirements are presented in Section 5, and the methods to reduce communication collisions for an arbitrary neighbor discovery protocol are introduced in Section 6. We implemented Spear and evaluated several notable neighbor discovery protocols; the results are presented and discussed in Section 7. Finally, we conclude the paper in Section 8.

2. Related Works

The neighbor discovery problem in WSNs 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 in a short expected time. *Birthday protocol* [20] 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. In the birthday protocol, each node transmits with probability $p \in [0, 1]$ and listens on the channel with probability $1 - p$ in each time slot independently; this protocol ensures that the nodes can discover the neighbors with high probability, but it cannot deduce a bounded discovery time. Aloha-like protocol [28] assumes each node is awake in each slot with probability p_w and an awake node transmits with probability p_t and listens with probability p_l; one can derive the expected time to discover all neighbors with this protocol, but cannot guarantee successful discovery for the worst case situation. Following that, more smarter probabilistic algorithms were proposed [25,27,29], but they cannot guarantee an upper bound on the discovery latency among the nodes.

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 [15,17–19,30], but only a few of them support asymmetric duty cycles of the nodes, such as Hedis [15]. 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 [32]. Some representative algorithms are Disco [16], U-Connect [18], and Todis [15].

In short, probabilistic algorithms cannot guarantee two neighboring nodes discover each other in a finite time, but they assure that the discovery can be successful with a high probability in an expected discovery time. In contrast, deterministic algorithms can ensure the discovery process between the nodes, and they can limit the maximum discovery latency to within a bounded time.

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 [13], Hello [22] and Nihao [21]. There are also some other neighbor discovery protocols that are based on different techniques, such as combinatorial design [33], BlindDate [26], and Panda [12], 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 [21], while others support asymmetric duty cycles (nodes select different duty cycles), such as Disco, U-Connect, Searchlight [13], Hello [22], Hedis and Todis [15].

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.

3. Preliminaries

3.1. 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 [12]. 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 turns the radio on for communication. We focus on the communications during neighbor discovery process. Before each node tries to communicate with others, it has to identify the neighboring nodes first. Therefore, we suppose the nodes can communicate only when the discovery is successful.

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's radio is turned on, and p_f if the radio is turned off. Notice that a node may send a beacon message for discovering neighbors, listen on channels, exchange information, etc. Different operations may consume different energy when the radio is ON; we assume p_n is the average energy in a time slot for simplicity. 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). As shown in Figure 2, a node has two states $\{ON, OFF\}$ and it switches states in each time slot. When a node switches from OFF to ON as $F \to N$, the consumed energy is p_{fn}; when it switches from ON to OFF as $N \to F$, the consumed energy is p_{nf}; when it keeps state ON as $N \to N$, the (average) consumed energy in the slot is p_n; when it keeps state OFF as $F \to F$, the consumed energy is p_f. 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 1.

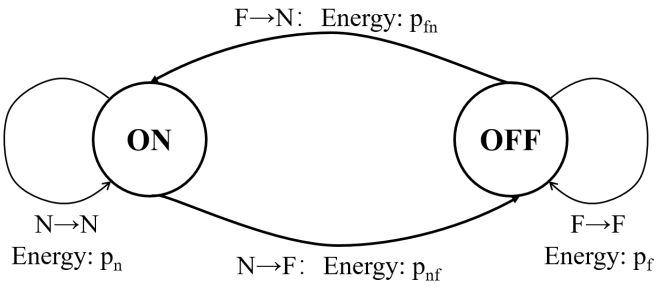

Figure 2. The finite state machine (FSM) of a sensor node's states.

Table 1. 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

3.2. Communication Model

Considering a WSN 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. The signal-to-interference-plus-noise ratio (SINR) model is a realistic model that captures the collision among multiple transmissions [34]. It is also shown that the SINR model can be converted to the communication graph model that two nodes are neighbors if their distance is within the communication range. Therefore, we simplify the process and assume that 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.

3.3. Neighbor Discovery

Neighbor discovery is the foundation of constructing WSNs. 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 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 network with multiple nodes, the neighbor discovery problem for 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 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. *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:*

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

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.

4. Spear: Neighbor Discovery Framework

4.1. Framework Overview

As shown in Figure 3, 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.

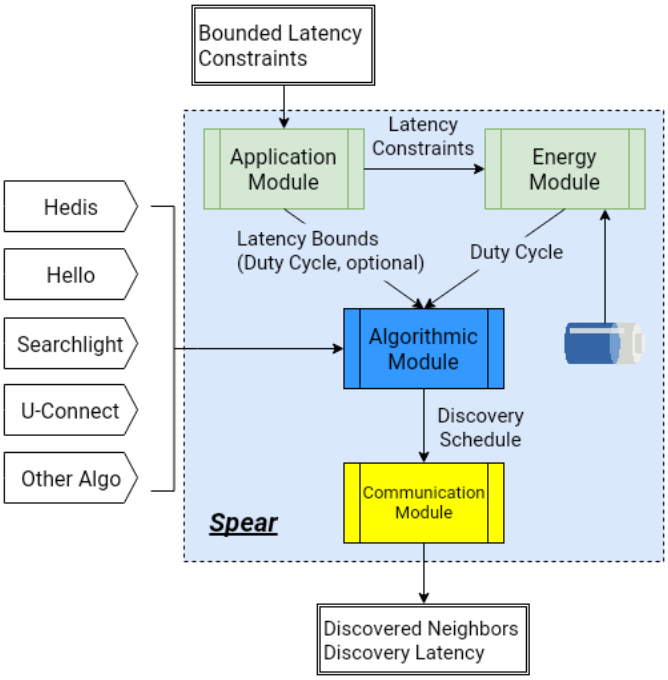

Figure 3. Overview of Spear.

4.2. 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 have the following equation:

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

Then, the lifetime of node u_i (denoted as Lf_i) is $Lf_i = t_i^e - t_i^s$.

Notice that we assume a sensor node has only two states $\{ON, OFF\}$ in Section 3.1. The generated schedule $S_i = \{s_i(t)\}$ contains a sensor state of each time slot t; specifically, $s_i(t) = 0$ if the state of the node is OFF while $s_i(t) = 1$ is the state of the node is ON. We define two identification functions as $f_{n2f}(t) = 1$ if and only if $s_i(t-1) = 1, s_i(t) = 0$, $f_{f2n}(t) = 1$ if and only if $s_i(t-1) = 0, s_i(t) = 1$. The first one implies that the node switches its state from ON to OFF, while the other one implies that the node switches its state from OFF to ON. Combining these, a complete energy formulation should be

$$\int_{t_i^s}^{t_i^e} s_i(t) \cdot p_n + (1 - s_i(t)) \cdot p_f + f_{n2f}(t) \cdot p_{nf} + f_{f2n}(t) \cdot p_{fn} = P_{max}.$$

Since we assume the consumed energy of state OFF and switching states are zero, i.e., $p_f = p_{nf} = p_{fn} = 0$, we derive the simplified formulation as Equation (3).

If node u_i turns the radio on all the time, Equation (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 $\hat{\theta} \in (0, 1)$, we can rewrite Equation (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:

$$Lf_i = \frac{P_{max}}{p_n \cdot \hat{\theta}}. \tag{4}$$

Suppose that node u_i adjusts the duty cycle timely. Denote the time that 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 Equation (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:

$$t_i^e = \frac{\frac{P_{max}}{p_n} - \sum_{k=0}^{m-1} (T_{k+1} - T_k) \cdot \theta_i(T_k, T_{k+1})}{\theta_i(T_m, T_{m+1})} + T_m. \tag{5}$$

Then, the lifetime can be computed. In Section 5, we analyze the impact of the node's lifetime by different strategies that adjust the duty cycles.

4.3. Application Module

WSNs 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.

4.4. 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 2.

Table 2. Algorithms comparison for two neighbors.

Algorithms	DC 1	DC 2	Latency	Asymmetric?
Quorum [24]	θ	θ	$\frac{4}{\theta^2}$	No
LL-Optimal [33]	θ	θ	$\frac{1}{\theta^2}$	No
Disco [16]	θ_1	θ_2	$\frac{4}{\theta_1\theta_2}$	Yes
U-Connect [18]	θ_1	θ_2	$\frac{9}{4\theta_1\theta_2}$	Yes
Searchlight [13]	θ_1	θ_2	$\frac{2}{\theta_1\theta_2}$	Yes
C-Torus [35]	θ_1	θ_2	$\frac{9}{4\theta_1\theta_2}$	Yes
BlindDate [26]	θ_1	θ_2	$\frac{9}{5\theta_1\theta_2}$	Yes
Hedis [15]	θ_1	θ_2	$\frac{4}{\theta_1\theta_2}$	Yes
Todis [15]	θ_1	θ_2	$\frac{9}{\theta_1\theta_2}$	Yes
Hello [22]	θ_1	θ_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 θ for symmetric duty cycle and θ_1, θ_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.

4.5. 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 6, we devise two new methods to reduce the collisions, and the algorithms modified by the methods can achieve good performances.

4.6. Measurements

In this 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, and so we introduce discovery rate for evaluation.

5. 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.

5.1. Energy Management Methods

As shown in Equation (4), node u_i's lifetime is $Lf_i = \frac{P_{max}}{p_n \cdot \theta}$ if it sticks to duty cycle θ 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:

$$\theta_i(t) = \hat{\theta}_j, \; if \; P_i(t) \in (\hat{P}_{j+1}, \hat{P}_j]. \tag{6}$$

The lifetime of node u_i is computed as:

$$Lf_i \quad = \sum_{j=1}^{m} \frac{\hat{P}_j - \hat{P}_{j+1}}{p_n \cdot \hat{\theta}_j}. \tag{7}$$

The PWR method can extend the node's lifetime as compared to Equation (4), 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. Supposing that the remaining energy at time t is $P_i(t)$, the duty cycle is reduced as:

$$\frac{P_i(t)}{\theta_i(t)} = \frac{P_{max}}{\hat{\theta}_0}. \tag{8}$$

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}, and the following equations are derived:

$$\begin{cases} \hat{P}_0 = P_{max}, & \hat{\theta}_0 = \hat{\theta}_0, \\ \hat{P}_1 = \hat{P}_0 - \hat{T} \cdot \hat{\theta}_0 \cdot p_n, & \hat{\theta}_1 = \frac{\hat{P}_1}{\hat{P}_0} \cdot \hat{\theta}_0, \\ \quad \vdots & \quad \vdots \\ \hat{P}_i = \hat{P}_{i-1} - \hat{T} \cdot \hat{\theta}_{i-1} \hat{p}_n, & \hat{\theta}_i = \frac{\hat{P}_i}{\hat{P}_0} \cdot \hat{\theta}_0, \\ \quad \vdots & \quad \vdots \\ \hat{P}_m = \hat{P}_{m-1} - \hat{T} \cdot \hat{\theta}_{m-1} \cdot p_n, & \hat{\theta}_m = \frac{P_{min}}{\hat{P}_0} \cdot \hat{\theta}_0. \end{cases} \tag{9}$$

Combine these to give:

$$\begin{aligned} \hat{P}_m = \quad & \hat{P}_0[1 - \tbinom{m}{1}\frac{\hat{T}\hat{\theta}_0 p_n}{\hat{P}_0} + \tbinom{m}{2}(\frac{\hat{T}\hat{\theta}_0 p_n}{\hat{P}_0})^2 + \cdots + (-1)^m \tbinom{m}{m}(\frac{\hat{T}\hat{\theta}_0 p_n}{\hat{P}_0})^m]. \\ = \quad & \hat{P}_0(1 - \frac{\hat{T}\hat{\theta}_0 p_n}{\hat{P}_0})^m. \end{aligned} \tag{10}$$

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:

$$Lf_i = m \cdot T + \frac{\hat{P}_m P_{max}}{P_{min} \hat{\theta}_0 p_n}. \tag{11}$$

Both PWR and PDR methods could prolong the node's lifetime and we evaluate them in Section 7.

5.2. 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 2, different algorithms lead to different discovery latencies. Take Disco [16] 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:

$$\theta_i(t) = f(P_i(t), \hat{L}_i(t)). \tag{12}$$

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.

6. 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.

6.1. 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 that 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 θ_k. We consider the scenario where node u_k turns on the radio with probability θ_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:

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

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.

6.2. Pure Probability Reducing (PPR) Method

The 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:

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

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 performance could be different. We evaluate the sensitivity of p_1 in Section 7.

6.3. Decreased Probability Reducing (DPR) Method

The *PPR* method can increase the discovery probability, but it is independent of the schedule itself. We present the *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 7.1).

7. 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 [15], Hello [22], Searchlight [13] and U-Connect [18] (we also implemented Quorum [24], but the result is only presented in Figure 9 since it is inapplicable when the users' duty cycle are different), and run these algorithms in a cluster with nine servers, each equipped with an Intel Xeon 2.6 GHz CPU (central processing unit) with 24 hyper-threading cores, 64 GB memory and 1T SSD (solid-state disk). The basic settings in the simulations are: $d_c = 50$ m, $P_{max} = 100,000$, $p_n = 1$ and $t_0 = 20$ ms. 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 (θ_c) is set to different figures.
(2) Discovery among $N = 1000$ nodes. The area is set as a rectangle of size 1000×1000 m^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.

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. The detailed parameters are described in Table 3.

Table 3. Parameters of the evaluations.

Figures	Parameters			
All figures	$d_c = 50$ m	$P_{max} = 100{,}000$	$p_n = 1$	$t_0 = 20$ ms
Figure 4	$\|N_c\| \in [1, 100]$	$\theta_c = 0.3$	$p_1 = 0.5$	$p_2 = 0.5$
Figure 5	$\|N_c\| = 20$	$\theta_c = 0.3$	$p_1 \in [0.1, 0.9]$	
Figure 6	$\|N_c\| = 20$	$\theta_c = 0.3$	$p_2 \in [0.1, 0.9]$	
Figure 7	$\|N_c\| = 20$	$\hat{\theta}_0 = 0.3$	$\hat{T} = 100{,}000$	
Figure 8	$\|N_c\| = 20$	$\hat{\theta}_0 = 0.3$	$\hat{T} = 100{,}000$	
Figure 9	$N = 2$	$\theta_1 \in [0.1, 0.3]$	$\theta_2 \in [0.1, 0.3]$	

7.1. 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 6), 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. Figure 4 shows the discovery rate (y-axis) of u_c within $100,000$ 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 $p_1 = p_2 = 0.5$.

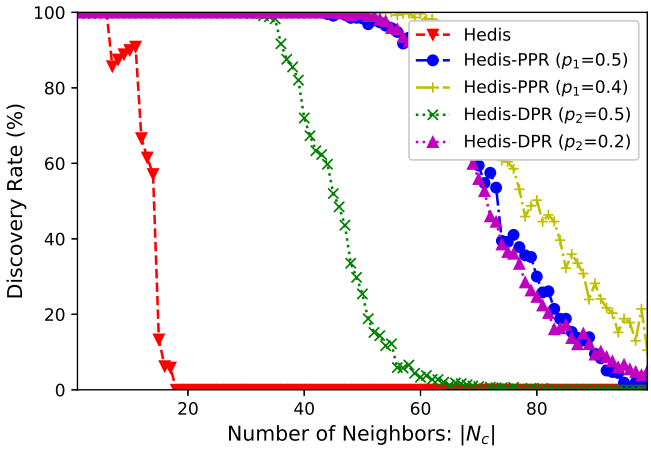

Figure 4. Spear increases discovery rate by incorporating PPR and DPR.

Sensitivity of p_1. In a star network, the central node u_c has $|N_c| = 20$ neighbors (we set $|N_c| = 20$ since the bare algorithms fail to find any neighbor) and we evaluate PPR's performance under different values of p_1. As shown in Figure 5a, θ_c is set to 0.3 and the average discovery latency (y-axis) of different algorithms is changed by different values of p_1 (x-axis). When $p_1 \in [0.3, 0.4]$, the performance is better. In Figure 5b, we select U-Connect as the example and set θ_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 $p_1 \leq 0.6$.

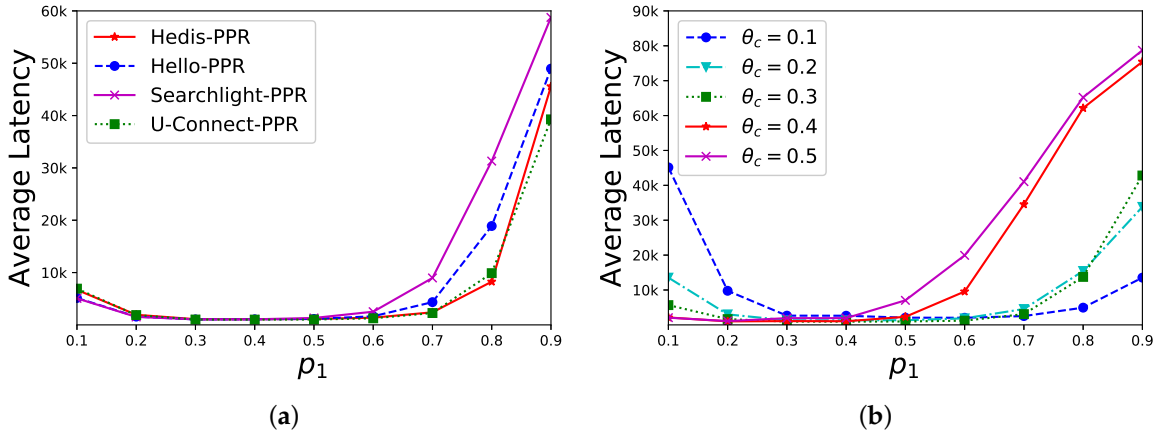

Figure 5. Sensitivity of p_1 in the PPR method. (**a**) θ_c = 0.3; (**b**) U-Connect-PPR.

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 Figure 6a, θ_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 Figure 6b, we select U-Connect for different θ_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 $p_2 \leq 0.7$.

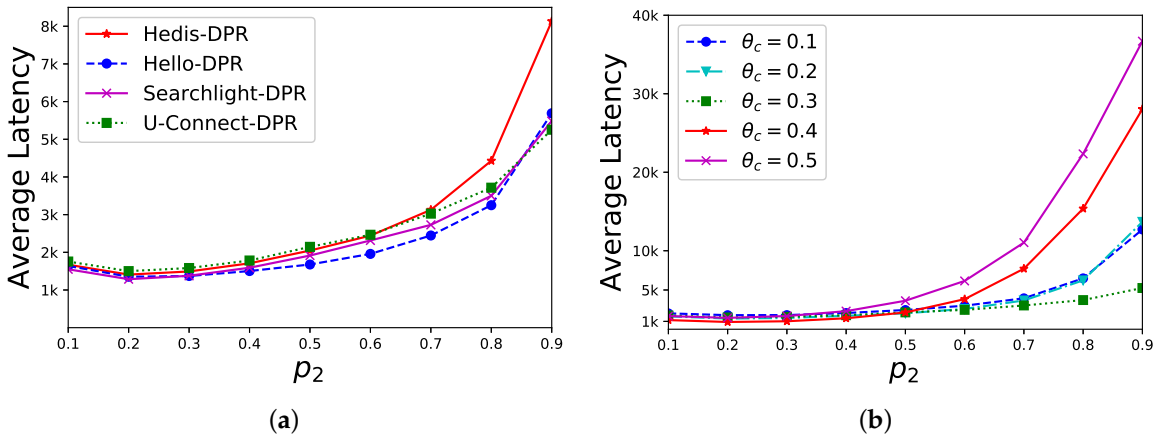

Figure 6. Sensitivity of p_2 in the DPR method. (**a**) θ_c = 0.3; (**b**) U-Connect-DPR.

1000 nodes. In a randomly generated network with $N = 1000$ nodes, we evaluate the performances of different algorithms. As shown in Figure 1, modified by PPR ($p_1 = 0.4$) and DPR ($p_2 = 0.2$), the discovery rates (y-axis) are much larger than those of 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.

7.2. 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. In PDR, $\hat{\theta}_0$ is set to 0.3, $P_{min} = 200$, and the node adjusts the duty cycle every $\hat{T} = 100,000$ time slots.

Lifetime. The lifetime of u_c is illustrated in Figure 7 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.

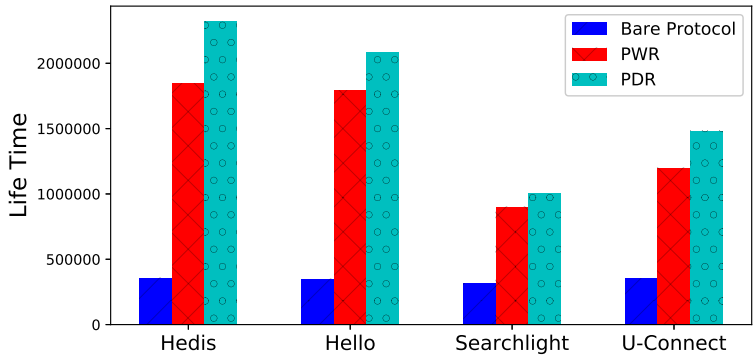

Figure 7. Spear prolongs the lifetime with PWR and PDR.

Percentage of Remaining Energy. We show the percentage of remaining energy (y-axis) after $2\hat{T}$ and $3\hat{T}$ time slots in Figure 8. After $2\hat{T}$ time slots, PWR and PDR are just a little better than the bare Hedis protocol (see Figure 8a), while the difference becomes much larger after $3\hat{T}$ time slots (see Figure 8b). By incorporating PWR and PDR, Spear can save power and extend the lifetime significantly.

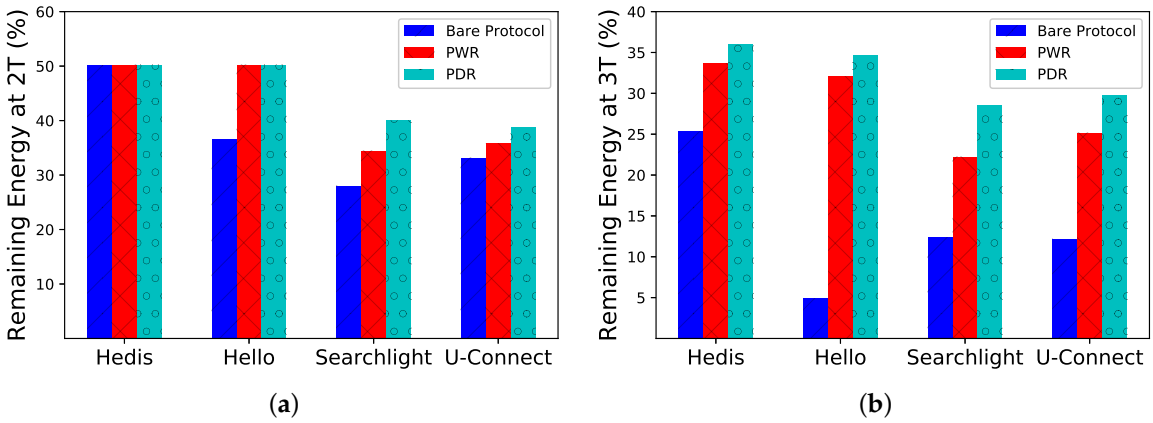

Figure 8. Spear saves more energy by incorporating PWR and PDR compared to bare algorithms. (**a**) After $2\hat{T}$; (**b**) After $3\hat{T}$.

7.3. 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.

Symmetric duty cycle: suppose two neighbors select the same duty cycle θ that increases from 0.1 to 0.3, we evaluate the average discovery latency in Figure 9a. As shown in the figure, the discovery latency (y-axis) of all algorithms decreases as θ (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 θ_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 Figure 9b, the average discovery latency (y-axis) decreases as θ_2 (x-axis) increases, and the decreasing trends are much more gentle than in Figure 9a. This is because the discovery latency is proportional to $\frac{1}{\theta_2}$ when θ_1 is a constant.

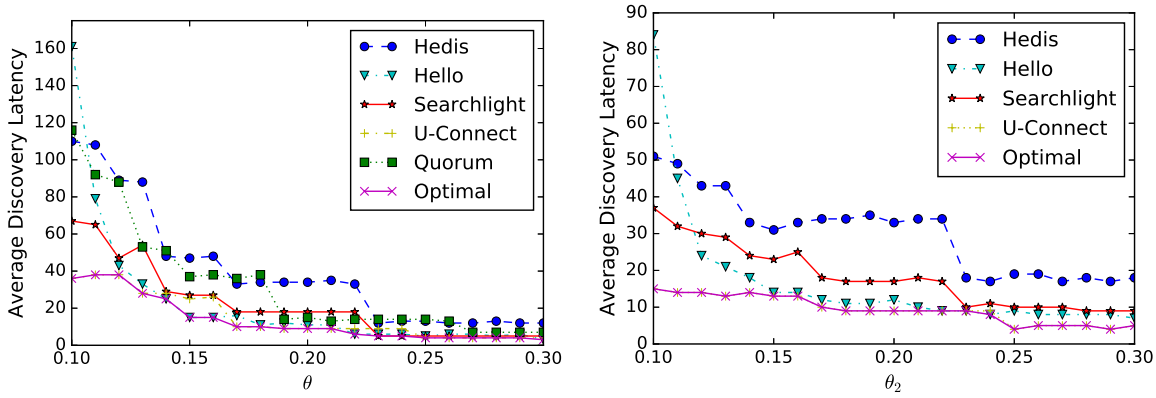

Figure 9. Spear enables the evaluation of the neighbor discovery algorithms for symmetric and asymmetric duty cycles and generates the optimal schedule automatically. (**a**) Symmetric duty cycle; (**b**) asymmetric duty cycle.

7.4. Effectiveness of Spear Components

In Spear, the communication module evidently increases the discovery rate as compared to the bare protocols, as demonstrated by the evaluations in Section 7.1. The energy module significantly extends a node's lifetime, as described in Section 7.2. 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 7.3.

8. Conclusions

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

In the future, we would like to explore the connection between some standard MAC (media access control) protocols for reducing collisions and neighbor discovery protocols, such as the Carrier Sense multiple Access/Collision Avoidance (CSMA/CA) protocol. We would also compare the performance of the proposed methods with the standards and bring these intuitive ideas to design efficient neighbor discovery protocols. Furthermore, we would evaluate the proposed framework with some well known IoT operation systems such as RiOT and Contiki [7], and conduct experiments on a real deployed WSN.

Author Contributions: Z.G. proposed the framework and the methods, and wrote the paper; Y.W. performed the evaluations of various protocols; W.S. improved the system model; Z.T. designed the framework and its modules; K.R. contributed evaluations tools; F.C.M.L. analyzed the latest protocols and polished the paper.

Funding: This work is supported in part by the National Key R&D Program of China 2018YEB1004003, China grants U1636215.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Atzori, L.; Lera, A.; Morabito, G. The Internet of Things: A Survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]
2. Tan, Q.; Gao, Y.; Shi, J.; Wang, X.; Fang, B.; Tian, Z. Towards a Comprehensive Insight into the Eclipse Attacks of Tor Hidden Services. *IEEE Internet Things J.* **2018**. [CrossRef]

3. Tian, Z.; Cui, Y.; An, L.; Su, S.; Yin, X.; Yin, L.; Cui, X. A Real-Time Correlation of Host-Level Events in Cyber Range Service for Smart Campus. *IEEE Access* **2018**, *6*, 35355–35364. [CrossRef]
4. Tian, Z.; Shi, W.; Wang, Y.; Zhu, C.; Du, X.; Su, S.; Sun, Y.; Guizani, N. Real Time Lateral Movement Detection based on Evidence Reasoning Network for Edge Computing Environment. *IEEE Trans. Ind. Inform.* **2019**. [CrossRef]
5. Tian, Z.; Li, M.; Qiu, M.; Sun, Y.; Su, S. Block-DES: A Secure Digital Evidence System using Blockchain. *Inf. Sci.* **2019**, *491*, 151–165. [CrossRef]
6. Zikria, Y.B.; Yu, H.; Afzal, M.K.; Rehmani, M.H.; Hahm, O. Internet of Things (IoT): Operating System, Applications and Protocols Design, and Validation Techniques. *Futuer Gener. Comput. Syst.* **2018**, *88*, 699–706. [CrossRef]
7. Zikria, Y.B.; Afzal, M.K.; Ishmanov, F.; Kim, S.W.; Yu, H. A Survey on Routing Protocols Supported by the Contiki Internet of Things Operating System. *Futuer Gener. Comput. Syst.* **2018**, *82*, 200–219. [CrossRef]
8. Gu, Z.; Hua, Q.-S.; Wang, Y.; Lau, F.C.M. Reducing Information Gathering Latency through Mobile Aerial Sensor Network. In Proceedings of the 2013 Proceedings IEEE INFOCOM, Turin, Italy, 14–19 April 2013.
9. Miao, G.; Zander, J.; Sung, K.W.; Slimane, B. *Fundamentals of Mobile Data Networks*; Combridge University Press: Combridge, UK, 2016.
10. Xu, X.; Luo, J.; Zhang, Q. Delay Tolerance Event Collections in Sensor Networks with Mobile Sink. In Proceedings of the 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010.
11. Wang, Y.; Wang, Y.; Qi, X.; Xu, L.; Chen, J.; Wang, G. L3SN: A Level-Based, Large-Scale, Longevous Sensor Network System for Agriculture Information Monitoring. *Wirel. Sens. Netw.* **2010**, *2*, 655–660. [CrossRef]
12. Margolies, R.; Grebla, G.; Chen, T.; Rubenstein, D.; Zussman, G. Panda: Neighbor Discovery on a Power Harvesting Budget. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 3606–3619. [CrossRef]
13. Bakht, M.; Trower, M.; Kravets, R.H. Searchlight: Won't you be my neighbor? In Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, Istanbul, Turkey, 22–26 August 2012.
14. Bian, K.; Park, J.-M.; Chen, R. A Quorum-Based Framework for Establishing Control Channels in Dynamic Spectrum Access Networks. In Proceedings of the 15th Annual International Conference on Mobile Computing and Networking, Beijing, China, 20–25 September 2009.
15. Chen, L.; Fan, R.; Bian, K.; Chen, L.; Gerla, M.; Wang, T.; Li, X. On Heterogeneous Neighbor Discovery in Wireless Sensor Networks. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015.
16. Dutta, P.; Culler, D. Practical Asynchronous Neighbor Discovery and Rendezvous for Mobile Sensing Applications. In Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, Raleigh, NC, USA, 5–7 November 2008; pp. 71–84.
17. Jiang, J.-R.; Tseng, Y.-C.; Hsu, C.-S.; Lai, T.-H. Quorum-based Asynchronous Power-Saving Protocols for IEEE 802.11 Ad Hoc Networks. *Mob. Netw. Appl.* **2005**, *10*, 169–181. [CrossRef]
18. Kandhalu, A.; Lakshmanan, K.; Rajkumar, R.R. U-Connect: A Lower Latency Energy-Efficient Asynchronous Neighbor Discovery Protocol. In Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, Stockholm, Sweden, 12–16 April 2010.
19. Lai, S.; Ravindran, B.; Cho, H. Heterogeneous Quorum-based Wake-up Scheduling in Wireless Sensor Networks. *IEEE Trans. Comput.* **2010**, *59*, 1562–1575. [CrossRef]
20. McGlynn, M.J.; Borbash, S.A. Birthday Protocols for Low Energy Deployment and Flexible Neighbor Discovery in Ad Hoc Wireless Networks. In Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing, Long Beach, CA, USA, 4–5 October 2001.
21. Qiu, Y.; Li, S.; Xu, X.; Li, Z. Talk More Listen Less: Energy-Efficient Neighbor Discovery in Wireless Sensor Networks. In Proceedings of the IEEE INFOCOM 2016–The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016.
22. Sun, W.; Yang, Z.; Zhang, X.; Liu, Y. Hello: A Generic Flexible Protocol for Neighbor Discovery. In Proceedings of the IEEE INFOCOM 2014—IEEE Conference on Computer Communications, Toronto, ON, Canada, 27 April–2 May 2014.
23. Tian, Z.; Su, S.; Shi, W.; Du, X.; Guizani, M.; Yu, X. A Data-driven Model for Future Internet Route Decision Modeling. *Future Gener. Comput. Syst.* **2019**, *95*, 212–220. [CrossRef]

24. Tseng, Y.-C.; Hsu, C.-S.; Hsieh, T.-Y. Power-Saving Protocols for IEEE 802.11-based Multi-Hop Ad Hoc Networks. *Comput. Netw.* **2003**, *43*, 317–337. [CrossRef]
25. Vasudevan, S.; Towsley, D.; Goeckel, D.; Khalili, R. Neighbor Discovery in Wireless Networks and the Coupon Collector's Problem. In Proceedings of the 15th Annual International Conference on Mobile Computing and Networking, Beijing, China, 20–25 September 2009.
26. Wang, K.; Mao, X.; Liu, Y. BlindDate: A Neighbor Discovery Protocol. *TPDS* **2015**, *26*, 949–959. [CrossRef]
27. Zeng, W.; Vasudevan, S.; Chen, X.; Wang, B.; Russel, A.; Wei, W. Neighbor Discovery in Wireless Networks with Multipacket Reception. In Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing, Paris, France, 17–19 May 2011.
28. Sun, G.; Wu, F.; Chen, G. Neighbor Discovery in Low Duty Cycle Wireless Sensor Networks with Mutipacket Reception. In Proceedings of the IEEE International Conference on Parallel and Distributed Systems, Singapore, 17–19 December 2012.
29. Zanella, A.; Bazzi, A.; Pasolini, G.; Masini, B.M. On the Impact of Routing Strategies on the Interference of Ad hoc Wireless Networks. *IEEE Trans. Commun.* **2013**, *61*, 4322–4333. [CrossRef]
30. Chen, P.; Chen, Y.; Gao, S.; Niu, Q.; Gu, J. Efficient group-based discovery for wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2017**. [CrossRef]
31. Montero, S.; Gozalvez, J.; Sepulcre, M. Neighbor Discovery for Industrial Wireless Sensor Networks with Mobile Nodes. *Comput. Commun.* **2017**, *111*, 41–55. [CrossRef]
32. Nathanson, M.B. *Elementary Methods in Number Theory*; Springer: Berlin/Heidelberg, Germany, 2000; Volume 195.
33. Zheng, R.; Hou, J.C.; Sha, L. Asynchronous Wakeup for Ad Hoc Networks. In Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing, Annapolis, MD, USA, 1–3 June 2003.
34. Halldorsson, M.M.; Wang, Y.; Yu, D. Leveraging Multiple Channels in Ad Hoc Networks. In Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, San Sebastián, Spain, 21–23 July 2015.
35. Chen, L.; Yan, B.; Zhang, J. Neighbor Discovery Algorithm in Mobile Low Duty Cycle WSNs. *J. Softw.* **2014**, *25*, 1352–1368.