Hindawi Publishing Corporation
Journal of Sensors
Volume 2015, Article ID 951652, 12 pages
http://dx.doi.org/10.1155/2015/951652

Research Article

Block Design-Based Asynchronous Neighbor Discovery Protocol for Wireless Sensor Networks

Sangil Choi,[1] Wooksik Lee,[2] Teukseob Song,[3] and Jong-Hoon Youn[4]

[1]*College of Information Science and Technology, University of Nebraska at Omaha, 6001 Dodge Street, Omaha, NE 68182, USA*
[2]*Department of Computer Science, Kyonggi University, Suwon 443-760, Republic of Korea*
[3]*Division of Convergence Computer and Media, Mokwon University, Daejeon 302-729, Republic of Korea*
[4]*Department of Computer Science, University of Nebraska at Omaha, 6001 Dodge Street, Omaha, NE 68182, USA*

Correspondence should be addressed to Teukseob Song; teukseob@mokwon.ac.kr

Received 2 March 2015; Accepted 11 June 2015

Academic Editor: Jianhua Tong

Neighbor discovery is a significant research topic in wireless sensor networks. After wireless sensor devices are deployed in specific areas, they attempt to determine neighbors within their communication range. This paper proposes a new Block design-based Asynchronous Neighbor Discovery protocol for sensor networks called *BAND*. We borrow the concept of combinatorial block designs for neighbor discovery. First, we summarize a practical challenge and difficulty of using the original block designs. To address this challenge, we create a new block generation technique for neighbor discovery schedules and provide a mathematical proof of the proposed concept. A key aspect of the proposed protocol is that it combines two block designs in order to construct a new block for neighbor discovery. We analyze the worst-case neighbor discovery latency numerically between our protocol and some well-known protocols in the literature. Our protocol reveals that the worst-case latency is much lower than others. Finally, we evaluate the performance of *BAND* and existing representative protocols through the simulation study. The results of our simulation study show that the average and maximum latency of *BAND* is about 40% lower than that of existing protocols. Furthermore, *BAND* spends approximately 30% less energy than others during the neighbor discovery process.

1. Introduction

Wireless sensor networks (WSNs) have been widely used for collecting or monitoring environmental data in areas of interest. After sensors are deployed, they initially try to find their neighbors within their communication range for exchanging information and maintaining network connectivity. The issue of neighbor discovery has been actively discussed for over a decade since the introduction of WSN applications. This topic is one of the critical research issues in WSNs for the following reasons. First, if sensors cannot find neighbors within a certain amount of time, they are unattended or useless. Second, if this situation continues, the entire network can be partitioned or disconnected.

A variety of asynchronous neighbor discovery protocols have been developed in order to address a neighbor discovery problem. A probabilistic neighbor discovery protocol, *Birthday* protocol [1], has been proposed for asynchronous neighbor discovery based on discovery probabilities. A Quorum-based neighbor discovery protocol has been introduced in multihop ad hoc networks [2, 3]. A node arbitrarily chooses one column and one row of entries for neighbor discovery in a given two-dimensional $m \times m$ array. By using two prime numbers, a new asynchronous neighbor discovery approach was proposed in [4, 5]. While *Disco* [4] selects two different prime numbers adapting the Chinese Remainder Theorem, *U-Connect* [5] only uses a single prime number for asynchronous neighbor discovery. Lastly, Zheng et al. [6] employ the concept of combinatorial block designs to create neighbor discovery schedules. Their solution deals with a symmetric operation when the duty cycle of all sensor nodes is uniform in the network. After finding a block design with a desired duty cycle, we can simply employ the block design to create neighbor discovery schedules for switching between the power-saving and active modes.

However, there is a practical challenge of simply borrowing the idea of combinatorial block designs for neighbor discovery. These designs are usually constructed by algebraic methods, but there is no unified construction method. So, each block design has been constructed in an ad hoc manner. Also, various heuristic-based search algorithms have been tried, but the success of these methods has been limited. In particular, the construction of block designs with a really long block length is extremely hard due to the vast size of the search space and the complexity of search algorithms.

The most recent research on neighbor discovery focuses on low duty cycles, about 1% or below. In many WSN monitoring applications, sensors are expected to work for months or even years before replacing their battery, and hence ultralow duty cycle protocols (e.g., 0.1%) are desired.

Therefore, this paper proposes a new asynchronous neighbor discovery protocol called *BAND*. The basic idea behind *BAND* is combining two existing block designs to produce a new block design while preserving the desired properties of the original block designs. By using a small set of existing block designs, it is possible to create neighbor discovery schedules with a given duty cycle. We also compare the performance of *BAND* with existing asynchronous neighbor discovery protocols through a simulation study with a popular WSN simulation tool called TOSSIM [7].

The main contributions of this paper are (1) a new block construction mechanism by combining two block designs, (2) a mathematical verification of the proposed construction scheme, and (3) development of an ultralow duty neighbor discovery protocol using block designs called *BAND*.

This paper is organized as follows: Section 2 reviews related works in asynchronous neighbor discovery. Section 3 introduces an asynchronous neighbor discovery problem, the relationship between neighbor discovery schedules and block designs, and a practical challenge in applying block designs to the problem of neighbor discovery. Section 4 details how we combine two block designs and summarizes our key contributions. In Section 5, we evaluate and measure the discovery latency and energy efficiency of *BAND* and compare the performance of the proposed scheme with that of other protocols in the literatures by using a sensor network simulation tool. Finally, we conclude the paper in Section 6.

2. Related Work

A neighbor discovery problem has been studied through a variety of ways. The main purpose of neighbor discovery protocols is to find neighboring nodes as soon as possible while consuming the minimum battery power in the network. Eventually, the solution of the neighbor discovery problem extends the network life by reducing the energy expenditure of all sensor nodes in a sensor network.

The *Birthday* protocol [1] has been proposed for asynchronous neighbor discovery in static ad hoc networks. This protocol focuses on power savings while sensor nodes are deployed in the network and energy-efficient neighbor discovery after the deployment using a scheme in which nodes wake up, listen, transmit, or sleep with different probabilities. The authors conclude that neighbor discovery would be useful in static ad hoc networks. Probabilistic neighbor discovery produces aperiodic and nondeterministic discovery latencies and long tails on discovery probabilities. In addition, this probabilistic approach cannot guarantee that each sensor node always meets each other within a certain amount of time.

Tseng et al. [2, 3] introduce a *Quorum*-based protocol for multihop ad hoc networks. The *Quorum* divides the total number of time slots into a set of m^2 contiguous intervals, m being a global parameter. The m^2 intervals are represented as a two-dimensional $m \times m$ array in a row-major manner. A node arbitrarily selects one column and one row of entries to transmit and receive data packets. Thus, a total of $2m - 1$ slots are selected from a given array of size $m \times m$ slots. The *Quorum* protocol has a twofold challenge: (1) it is possible that two different schedules overlap frequently based on the selection of arbitrary row and column of entries and (2) another pair of schedules may not have frequent overlapping. In the first case, two neighboring nodes can find each other relatively quickly, and discovery latency and energy consumption are very low. However, in the second case, the *Quorum*-based approach leads to long discovery latency, and the performance of this protocol is worse than other protocols.

Zheng et al. [6] apply the concept of combinatorial block designs using difference sets [8–10] to the asynchronous neighbor discovery problem. Their solution addresses the symmetric duty cycle operation when the duty cycle of all sensor nodes is uniform in the network. They propose an optimal design for neighbor discovery schedules using a block design. They conclude that, for asymmetric duty cycles, their approach reduces to an NP-complete minimum vertex cover problem requiring a centralized solution. In *U-Connect* [5], the authors provide a theoretical formulation for measuring the performance of neighbor discovery protocols called the *power-latency product*. According to this metric, the combinatorial method has the best performance with respect to the worst-case latency for neighbor discovery. However, the disadvantage of the Combinatorial protocol is that there is no unified block construction method for generating neighbor discovery schedules.

Disco [4] uses two different prime numbers (p_1 and p_2) for asynchronous neighbor discovery. This protocol adapts the Chinese Remainder Theorem, where the nodes select a pair of prime numbers such that the sum of their reciprocals is equal to the desired duty cycle. The nodes stay awake in the slot if the current slot number is a multiple of the selected prime numbers. The worst-case discovery latency is the product of the minimum of the primes selected by the nodes. The authors suggest the use of balanced primes (the difference between two prime numbers of each node is minimal) for symmetric discoveries and unbalanced primes (where the difference is maximal) for asymmetric operations. This characteristic presents deterministic discovery latencies that are much better than *Quorum* and *Birthday* protocols for asymmetric scenarios and similar to *Quorum* in the symmetric case. The typical problem of *Disco* is the selection of proper prime numbers in both symmetric and asymmetric

operations. If nodes pick the same balanced primes in asymmetric situations or unbalanced numbers in symmetric scenarios, the worst-case discovery latency worsens.

A more recent deterministic approach, *U-Connect* [5], uses a single prime number, p. Instead of just waking up only one slot in every p slot, the nodes also wake up $(p + 1)/2$ consecutive slots every p^2 slot. The worst-case latency for *U-Connect* is p^2 which is similar to *Disco*. However, for the *energy-latency product*, a metric proposed by the authors to evaluate the energy-efficiency of asynchronous neighbor discovery protocols, the performance of *U-Connect* is better than that of *Disco* in the symmetric case. However, *U-Connect* also shows an uneven distribution of active slots because a large number of active slots are located at the beginning of the discovery schedules. In particular, at low duty cycles, for example, below 1%, the number of contiguous active slots grows considerably and worst-case discovery latency increases greatly.

3. Asynchronous Neighbor Discovery Problem

Most wireless sensor devices usually support two power modes: *active* and *power-saving (sleep)* to save their battery power [11, 12]. While sensors can communicate with each other in an active mode, they turn their radio off in a power-saving mode. To find neighboring nodes in this environment, sensors should be in the active mode at the same time. After discovering a neighbor, the clock synchronization is maintained by exchanging periodic control packets and sensor nodes can meet each other within a certain amount of time. Diverse time synchronization mechanisms [13–16] are developed to address the neighbor discovery problem. However, this approach has a huge overhead for exchanging control messages between nodes at regular intervals of time. It is well known that periodic wireless message exchange is one of the primary sources of battery drainage [17]. Therefore, this technique is not appropriate for ultralow power operations in a sensor network.

Instead of maintaining time synchronization by exchanging periodic messages, each sensor could follow its own sleep and wake-up schedule, switching its modes. If two different schedules have at least one rendezvous time slot within a certain period of time, we can guarantee that these two nodes will find each other unless there is collision or wireless error. In this study, we address the following research questions that are closely related to the asynchronous neighbor discovery problem:

(1) How can we design neighbor discovery schedules?

(2) How can we enable two discovery schedules to have a common active slot without any synchronization techniques?

3.1. Neighbor Discovery Schedules. In this section, we explain how to represent a neighbor discovery schedule (NDS). We assume the sensor can stay in one of the following states during the neighbor discovery process: *active* or *power-saving* (*sleep*) mode. It is possible to illustrate these two modes with a binary number, where the number "1" expresses an active mode and "0" denotes a sleep mode. By using this simple binary notation, we can define a schedule and duty cycle as follows.

Slot index	1	2	3	4	5	6	7
mode	1	1	0	1	0	0	0

Figure 1: An illustration of a neighbor discovery schedule.

Definition 1. A *schedule* S is a sequence of 0 and 1 representing the sleep and active modes, respectively. The *Active mode (an active slot)* is a state where a wireless sensor turns its radio on and is ready to send or receive data packets, and the *sleep mode (a sleep slot)* is a state where the sensor turns its radio off and only senses and collects some environmental data. In S, a binary value "0" represents the sleep mode and the value "1" denotes the active mode.

Definition 2. A *duty cycle* is the percentage of the ratio of the number of active slots over the total number of slots per a given schedule. As a formula, a duty cycle D can be expressed as

$$D = \frac{A}{T} \times 100\%, \tag{1}$$

where A is the number of active slots and T is the total number of slots in the schedule.

We can illustrate an NDS based on Definitions 1 and 2. According to Definition 1, there are only two modes in the schedule. Therefore, we can say that the NDS is the series of binary numbers representing active and sleep modes. Figure 1 shows an example of the NDS. In this example, this schedule consists of seven time slots and the duty cycle of the NDS is $(3/7) \times 100\% \approx 43\%$.

3.2. Relationship between Neighbor Discovery Schedules and Combinatorial Block Designs. The concept of block designs can be applied to two research questions that we mentioned before. We borrow a combinatorial block design to address these questions. A block design is a set of nonempty subsets (called *blocks*) whose elements are selected to meet a specific set of properties. This design theory is widely applied to numerous applications such as experimental design, geometry, software testing, and cryptography. In particular, a Balanced Incomplete Block Design (BIBD) [8–10] is a well-studied experimental design that has various desirable features from a statistical perspective. First of all, we introduce some definitions and theorems related to BIBD. According to [10], a design is defined as follows.

Definition 3. A *design* is a pair (X, A) such that the following properties are satisfied:

(1) X is a set of elements called points.

(2) A is a collection (i.e., multiset) of nonempty subsets of X called blocks.

Definition 4. Let v, k, and λ be positive integers such that $v > k \geq 2$. A (v, k, λ)-*Balanced Incomplete Block Design* (which we abbreviate to (v, k, λ)-*BIBD*) is a design (X, A) such that the following properties are satisfied:

(1) Consider $|X| = v$.

(2) Each block contains exactly k points.

(3) Every pair of distinct points is contained in exactly λ blocks.

A (7, 3, 1)-BIBD is one of the typical BIBDs in case of $\lambda = 1$. In the (7, 3, 1)-BIBD,

$$X = \{1, 2, 3, 4, 5, 6, 7\},$$
$$A = \{124, 235, 346, 457, 561, 672, 713\}.$$

There is one more example of BIBDs. In a (7, 3, 2)-BIBD,

$$X = \{1, 2, 3, 4, 5, 6, 7\},$$
$$A = \{123, 145, 167, 246, 257, 347, 356, 123, 147, 156, 245, 267, 346, 357\}.$$

Here, we write blocks in the form *abc* instead of $\{a, b, c\}$. In the expression of (7, 3, 1)-BIBD, each block consists of three points satisfying the second property of BIBD. For example, a block 124 contains elements 1, 2, and 4. On the other hand, it does not have elements 3, 5, 6, and 7.

Theorem 5 (Theorem 1.8 in [10]). *In a (v, k, λ)-BIBD, every point occurs in exactly*

$$r = \frac{\lambda (v - 1)}{k - 1} \tag{2}$$

blocks.

Theorem 6 (Theorem 1.9 in [10]). *A (v, k, λ)-BIBD has exactly*

$$b = \frac{vr}{k} = \frac{\lambda \left(v^2 - v\right)}{k^2 - k} \tag{3}$$

blocks.

In the theory of combinatorial designs, Theorems 5 and 6 show that each point in X is the member of r blocks and a given BIBD has b blocks. If the number of blocks is the same as the number of points, that is, in the case of $b = v$, they define a *symmetric BIBD* in a combinatorial design theory. The symmetric BIBD has the property that $b = v$ if and only if $r = k$ [10]. In the previous examples, the (7, 3, 1)-BIBD is the symmetric BIBD because v is the same as b, but the (7, 3, 2)-BIBD is not the symmetric BIBD because v is different from b.

In [18], according to Theorem 2.1.2, it has been proved that if a given (v, k, λ)-BIBD is a symmetric BIBD then randomly selected two blocks have exactly common λ points. This theorem represents a fundamental property showing that two arbitrary discovery schedules could have common active time slots when we apply the symmetric BIBD to the asynchronous neighbor discovery problem. Therefore, we introduce this theorem here again.

Block	1	2	3	4	5	6	7
124	1	1	0	1	0	0	0
235	0	1	1	0	1	0	0
346	0	0	1	1	0	1	0
457	0	0	0	1	1	0	1
156	1	0	0	0	1	1	0
267	0	1	0	0	0	1	1
137	1	0	1	0	0	0	1

Figure 2: The binary expression of blocks of (7, 3, 1)-BIBD.

Theorem 7 (Theorem 2.1.2 in [18]). *If (X, A) is a symmetric design with parameters (v, k, λ), then any two distinct blocks have exactly λ points in common.*

For each point p_i in X, p_i may or may not be a member of each block B_i in A. We can use a binary number to express the status of p_i with the following notation:

$$p_i = \begin{cases} 1, & \text{if } p_i \in B_i, \\ 0, & \text{if } p_i \notin B_i. \end{cases} \tag{4}$$

With this notation, it is possible to rewrite all the blocks in A. For instance, a block 235 can be replaced by (0110100) with a binary number. The (7, 3, 1)-BIBD is likely to be illustrated graphically as shown in Figure 2 using the above-mentioned approach. This binary expression of blocks perfectly matches with the design of NDSs. Hence, we can randomly assign one of the blocks to each sensor as a discovery schedule.

Theorem 8 shows that arbitrary two NDSs have common active time slots when we apply the symmetric BIBD to the design of NDSs with (4).

Theorem 8. *If two distinct schedules s_i and s_j are mapped onto the (X, A) symmetric design with parameters (v, k, λ), then the two schedules, s_i and s_j, have λ active time slots in common.*

Proof. Since (X, A) is a symmetric design with parameters (v, k, λ), we can write X and A as follows:

$$X = \{p_1, p_2, \ldots, p_v\}, \text{ and}$$
$$A = \{B_i \mid B_i \subset X,\ |B_i| = k\}.$$

Two distinct schedules s_i and s_j are mapped onto two different blocks in A. Let two blocks in A be B_i and B_j respectively. By Theorem 7, there are λ common points between B_i and B_j. Hence, we can write $|B_i \cap B_j| = \lambda$. This means that s_i and s_j, which are mapped onto B_i and B_j, have λ common active time slots. □

In general, we cannot guarantee the existence of a (v, k, λ)-BIBD for some v, k, λ combinations, hence the simplest solution is using the existing BIBDs for developing

NDSs. It has been proved that if q is a power of a prime, there exists a $(q^2 + q + 1,\ q + 1,\ 1)$-BIBD under the special case of $\lambda = 1$ [8]. Zheng et al. [6] only focused on this kind of BIBD in their research. The $(q^2 + q + 1,\ q + 1,\ 1)$-BIBD satisfies the following equation based on Theorems 5 and 6:

$$b = \frac{\left(q^2 + q + 1\right)(q + 1)}{q + 1} = q^2 + q + 1. \tag{5}$$

Equation (5) shows that if q is a power of prime then $(q^2 + q + 1,\ q + 1,\ 1)$-BIBD is a symmetric BIBD. Therefore, our research only concentrates on the symmetric BIBD to generate NDSs in this paper.

The (7, 3, 1)-BIBD in Figure 2 is the case when q is 2. By selecting a proper power of a prime q, we would be able to create a number of discovery schedules with different duty cycles. Although this approach looks like a feasible method for designing NDSs, there is a practical challenge in employing a $(v, k, 1)$-BIBD directly.

3.3. Practical Challenge. The practical challenge of using a block design for asynchronous neighbor discovery is that there is no unified block construction method for generating a set of blocks that satisfies the desired properties specified in Definition 4. In a typical WSN application, we need to create a set of discovery schedules with a low duty cycle in order to enhance energy efficiency. In order to achieve a low duty cycle, we need to choose a relatively big prime number to generate a set of blocks. However, choosing a big prime number makes the number of blocks get bigger. Consequently, choosing big prime numbers might significantly affect the computational complexity of generating the blocks that satisfy all the properties. It might not be feasible to find a BIBD within a certain amount of time when extreme low duty cycles are given.

4. Block Construction Mechanism

It is difficult to directly apply BIBDs to construct discovery schedules due to the practical challenge discussed in the previous section. Hence, we need to develop a computationally efficient NDS generation scheme to resolve the asynchronous neighbor discovery problem. In this section, we elaborate on a new NDS construction scheme and its properties.

A basic concept under this construction technique is combining two block designs. To simplify the discussion, we only consider the symmetric BIBDs and use simple pregenerated block designs such as a (7, 3, 1)-design.

Definition 9. Let v, k, and λ be positive integers such that $v > k \geq 2$. A (v, k, λ)-*Neighbor Discovery Design* (which we abbreviate to (v, k, λ)-*NDD*) is a design (X, A) such that the following properties are satisfied:

(1) Consider $|X| = v$.

(2) Each block contains exactly k active time slots.

(3) Every pair of distinct blocks contains at least λ common active time slots.

Slot	1	2	3	4
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0

Slot	1	2	3
1	1	1	0
2	0	1	1
3	1	0	1

Figure 3: (4, 3, 2)- and (3, 2, 1)-designs.

Definition 10. A *sleep schedule* is a square matrix of order n with all 0's.

In this section, we introduce a new block construction scheme that generates a (v, k, λ)-*NDD* given in Definition 9 by combining two BIBDs.

4.1. Combination Process of Two Block Designs. Here we explain how to combine two designs in detail. For simplicity, two simple designs are used for demonstration purposes. Note that we created these two designs only for the purpose of illustrating the combination process. In addition, we use the terminology "point" and "slot" interchangeably in the previous sections. From this point, the "slot" is a representative terminology in the following sections.

Step 1. Choose two block designs.

First, we can select any two well-known block designs for the proposed method. We have two block designs: $A = (v_a, k_a, \lambda_a)$-*BIBD* and $B = (v_b, k_b, \lambda_b)$-*BIBD*. For the sake of simplicity, we use (4, 3, 2)- and (3, 2, 1)-designs to demonstrate the idea. Figure 3 shows (4, 3, 2)- and (3, 2, 1)-designs. Here, we call the (4, 3, 2)-design a *base* design and (3, 2, 1)-design a *replacement* design.

Step 2. Replace each active slot of the *base* design by the entire blocks of the *replacement* design and every sleep slot of the *base* design by a *sleep schedule* of order v_b.

In Step 2, for every active slot in the *base* design, we replace it by the entire blocks of the *replacement* design, the (3, 2, 1)-design. In addition, the sleep slots are replaced by a *sleep schedule* in Definition 10. Let us assume that we change each active slot from the (4, 3, 2)-design into the entire blocks from the (3, 2, 1)-design.

Step 3. Generate a new block design for neighbor discovery.

Finally, we can get a $(v_a \times v_b,\ k_a \times k_b,\ \lambda_a \times \lambda_b)$-*NDD* by combining A and B in this step. Figure 4 illustrates the resulting (12, 6, 2)-*NDD* by combining the (3, 2, 1) and (4, 3, 2)-*BIBD*.

By using the method in Step 2, we can construct a set of new NDSs with a different duty cycle. These combined schedules make the neighbor discovery problem easy to solve. We can generate a new NDS by combining any two block designs. If there exists a BIBD with a desired duty cycle, we can just use the BIBD for the neighbor discovery. However, due to the lack of a unified algorithm of generating BIBDs, it is not always possible to generate an NDS with a certain duty

Slot	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	1	1	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	1	1
3	0	0	0	1	0	1	1	0	1	1	0	1
4	1	1	0	0	0	0	1	1	0	1	1	0
5	0	1	1	0	0	0	0	1	1	0	1	1
6	1	0	1	0	0	0	1	0	1	1	0	1
7	1	1	0	1	1	0	0	0	0	1	1	0
8	0	1	1	0	1	1	0	0	0	0	1	1
9	1	0	1	1	0	1	0	0	0	1	0	1
10	1	1	0	1	1	0	1	1	0	0	0	0
11	0	1	1	0	1	1	0	1	1	0	0	0
12	1	0	1	1	0	1	1	0	1	0	0	0

Figure 4: Combination between (4, 3, 2)- and (3, 2, 1)-designs.

cycle. Therefore, the proposed block construction scheme gives us more options. For instance, the proposed method can virtually construct an NDD with almost any duty cycles by selecting a proper set of previously known block designs.

4.2. Verification of the Properties of the Proposed Design. In this section, we prove that the NDS constructed by merging two block designs still has the same properties as shown in Definition 4. The only difference between BIBD and NDD is the number of common active slots. The proposed design technique guarantees that the newly generated NDS still has the property of original designs.

Definition 11. Let $\mathcal{S}$ be a $(v_1, k_1, 1)$-*BIBD* and let $\mathcal{T}$ be a $(v_2, k_2, 1)$-*BIBD*. $\mathcal{S} \oplus \mathcal{T}$ indicates that all active slots in $\mathcal{S}$ are replaced by $\mathcal{T}$ and all sleep slots are changed by a *sleep schedule* with size of $v_2 \times v_2$.

Theorem 12. *Let $\mathcal{S}$ be a $(v_1, k_1, 1)$-BIBD and let $\mathcal{T}$ be a $(v_2, k_2, 1)$-BIBD. Then, $\mathcal{S} \oplus \mathcal{T}$ is a $(v_1 \times v_2, k_1 \times k_2, 1)$-NDD.*

Proof. $\mathcal{S}$ is a BIBD, so we can say that $\mathcal{S} = \{S_i \mid S_i \text{ is a schedule}\}$. $\mathcal{T}$ is also a BIBD; then $\mathcal{T} = \{T_i \mid T_i \text{ is a schedule}\}$. Let $\mathcal{W} = \mathcal{S} \oplus \mathcal{T}$ and $\mathcal{W} = \{W_i \mid 1 \le |W_i| \le v_1 v_2\}$ is a $(v_3, k_3, 1)$-NDD, where $v_3 = v_1 v_2$ and $k_3 = k_1 k_2$. That is, for any pair of schedules $W_i, W_j \in \mathcal{W}$, where $i \ne j$, $\exists 1 \le h \le v_1 v_2$ such that $\omega_{ih} = \omega_{jh} = 1$ (active slot). By the operation of $\mathcal{S} \oplus \mathcal{T}$, each active slot in $\mathcal{S}$ is replaced by $\mathcal{T}$. In addition, every sleep slot in $\mathcal{S}$ is changed by a *sleep schedule* with the size of $v_2 \times v_2$. Hence, we show that the total number of points in $\mathcal{W}$ is $v_1 \times v_2$ and each block in $\mathcal{W}$ contains exactly $k_1 \times k_2$ active time slots. These two properties exactly match with the first and second properties in Definition 9. Finally, we prove that every pair of distinct blocks contains at least λ common active time slots. We can represent $\mathcal{W}$ with the notation of the matrix as follows:

$$\mathcal{W} = \begin{bmatrix} \omega_{(1,1)} & \omega_{(1,2)} & \cdots & \omega_{(1,v_1v_2)} \\ \omega_{(2,1)} & \omega_{(2,2)} & \cdots & \omega_{(2,v_1v_2)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{(v_1v_2,1)} & \omega_{(v_1v_2,2)} & \cdots & \omega_{(v_1v_2,v_1v_2)} \end{bmatrix}.$$

Matrix $\mathcal{W}$ is a $v_1 v_2 \times v_1 v_2$ matrix and each element of $\mathcal{W}$ represents an active or a sleep slot. Schedules W_i and W_j represent one of the entire rows in $\mathcal{W}$. $\mathcal{S}$ is a BIBD, so $\mathcal{S}$ has a schedule S_i. Each S_i has at least one active slot s_{im}, where $1 \le m \le v_1 v_2$. By Definition 11, s_{im} is replaced by $\mathcal{T}$.

Case 1. For all index $i \ne j$ in $\mathcal{W}$,

$$i, j \in [v_2 \times (\alpha - 1) + 1, \ldots, v_2 \times (\alpha - 1) + v_2],$$

where $1 \le \alpha \le v_1$. W_i and W_j have at least one common active slot such that $\omega_{ih} = \omega_{jh} = 1$ by $\mathcal{T}$.

Case 2. For all $\alpha \ne \beta$ and index $i \ne j$ in $\mathcal{W}$,

$$i \in [v_2 \times (\alpha - 1) + 1, \ldots, v_2 \times (\alpha - 1) + v_2],$$
$$j \in [v_2 \times (\beta - 1) + 1, \ldots, v_2 \times (\beta - 1) + v_2],$$

where $1 \le \alpha, \beta \le v_1$, and W_i and W_j have at least one common active slot such that $\omega_{ih} = \omega_{jh} = 1$ by $\mathcal{S}$.

By Cases 1 and 2, schedules W_i and W_j in $\mathcal{W}$ have always at least λ common active slot. Therefore, $\mathcal{W}$ is a $(v_1 \times v_2,\ k_1 \times k_2,\ 1)$-*NDD*. □

Corollary 13. *Let $\mathcal{S} \oplus \mathcal{T}$ be an NDD. Then, $\mathcal{T} \oplus \mathcal{S}$ is also an NDD.*

Proof. By Theorem 12, for arbitrary BIBDs $\mathcal{S}$ and $\mathcal{T}$, $\mathcal{S} \oplus \mathcal{T}$ is a NDD. Hence, $\mathcal{T} \oplus \mathcal{S}$ is also an NDD by the proof of Theorem 12. □

Theorem 14. *Let $\mathcal{S}$ and $\mathcal{T}$ be given by Theorem 12. The duty cycle of $\mathcal{S} \oplus \mathcal{T}$ is*

$$\frac{k_1 \times k_2}{v_1 \times v_2} \times 100\%. \tag{6}$$

Proof. Let $\mathcal{S}$ be a $(v_1, k_1, 1)$-BIBD and let $\mathcal{T}$ be a $(v_2, k_2, 1)$-BIBD. By Definition 2, the duty cycle of $\mathcal{S}$ is $(k_1/v_1) \times 100\%$ and that of $\mathcal{T}$ is $(k_2/v_2) \times 100\%$. By Definition 11, all active slots in $\mathcal{S}$ are replaced by $\mathcal{T}$ and all sleep slots in $\mathcal{S}$ are changed by a *sleep schedule* with size of $v_2 \times v_2$ in the operation of $\mathcal{S} \oplus \mathcal{T}$. Hence, the total number of slots per cycle in $\mathcal{S} \oplus \mathcal{T}$ is $v_1 \times v_2$ and the number of active slots is $k_1 \times k_2$. Therefore, the duty cycle of $\mathcal{S} \oplus \mathcal{T}$ is

$$\frac{k_1 \times k_2}{v_1 \times v_2} \times 100\%. \tag{7}$$

□

Property 1. Let k be the number of active slots and let v be the total number of slots in a (v, k, λ)-NDD. The following property is satisfied:

$$\lim_{n \to \infty} \left(\frac{k}{v}\right)^n = 0, \tag{8}$$

where n is the number of combinations.

Proof. By Definition 4, the size of v is always greater than that of k. Hence, $|k/v| < 1$. The bigger the size of v is, the smaller $|k/v|$ is. Therefore, $\lim_{n \to \infty} (k/v)^n = 0$. □

Table 1: Worst-case discovery latency requirement.

Protocol	Worst-case latency
Combinatorial [6]	$q^2 + q + 1$
Quorum [2, 3]	m^2
U-Connect [5]	p^2
Disco [4]	$p_1 \cdot p_2$
BAND	$({q_1}^2 + q_1 + 1)({q_2}^2 + q_2 + 1)$

According to previous studies of neighbor discovery protocol (NDP), it is significantly important to maintain a low duty cycle (e.g., 1%) to prolong the network lifetime. We can significantly reduce battery consumption by putting sensors in a sleep mode most of the time and waking them up less frequently to communicate with their neighbors.

4.3. Advantages of Combining of Neighbor Discovery Designs. One of the most significant advantages of the new block construction mechanism for neighbor discovery is that we can easily generate the NDS with a given duty cycle. This flexible feature is really important in the neighbor discovery problem using block designs. The characteristics of *BAND* can overcome a practical challenge of using BIBD as we discussed in Section 3.3.

Secondly, the proposed technique is extremely simple. Given a certain duty cycle, it is not easy to find a BIBD satisfying all the properties for neighbor discovery. Although a number of heuristic-based search algorithms have been designed, they only work for very limited cases. However, our approach only takes $O(n^2)$ execution time to combine two block designs, where n is the total number of slots of a given base design.

Finally, the proposed method can be used recursively. If we cannot generate an NDS with a desired duty cycle by combining two known block designs, we can repeatedly apply the proposed $\oplus$ operation using any NDSs until we obtain the discovery schedules with the target duty cycle.

4.4. Worst-Case Latency Analysis. Let us analyze the worst-case performance of *BAND* and other NDPs in the literature in this subsection. Once the target duty cycle is given, it is easy to calculate the worst-case discovery latency of each NDP. The worst-case latency analysis is important because it indicates how many time slots should be required to find neighbors in the worst-case scenario. In general, the total number of slots required for each sensor to find all its neighbors directly affects the performance of NDPs. Therefore, the worst-case discovery latency is one of the key performance metrics in WSNs. Table 1 displays the equations of the worst-case latency for each NDP.

Table 2 shows the length of NDS selected for each NDP with duty cycles varying from 10% to 1%. This table clearly shows that the length of the NDS constructed for *BAND* is closest to that of *Combinatorial* for the same duty cycle. However, the length of other NDPs selected for *Quorum*, *U-Connect*, or *Disco* is much larger than that of *BAND*. In particular, with 1% of duty cycle, the length of the selected

Table 2: Total number of slots of different neighbor discovery protocols.

Protocol	10%	5%	2%	1%
Combinatorial	91	381	2451	9507
Quorum	361	1521	9801	39601
U-Connect	169	841	5329	22201
Disco	391	1591	9991	39203
BAND	147	511	2821	10431

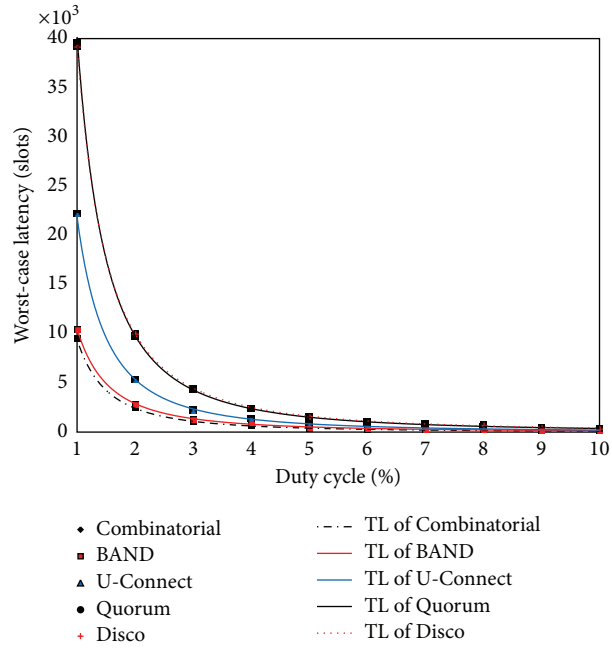

Figure 5: Worst-case latency with low duty cycles.

NDS of both *Quorum* and *Disco* is approximately 4 times greater than that of *BAND*. Note that the length of NDSs is strongly related to the performance of NDPs in terms of the maximum discovery latency. This observation reveals that, even for the same duty cycle, the worst-case latency guarantee varies significantly based on the length of five NDSs.

Figure 5 illustrates the worst-case latency for five NDPs with different low duty cycles ranging from 1% to 10%. Each NDP has its own trend line for latency. This graph shows that *Combinatorial* and *BAND* have a similar pattern and they outperform other NDPs in terms of the worst-case latency. Roughly, these two NDPs demonstrate the worst-case performance 2 times better than *U-Connect* and 4 times better than *Quorum* and *Disco*. Furthermore, we analyze the worst-case performance in the situations with extremely low duty cycles ranging from 0.1% to 0.9%. Note that, in the case of *Combinatorial*, we cannot guarantee that there is a known set of blocks for the given ultralow duty cycles, although the existence of these BIBDs is proven [8]. If there is no appropriate block construction mechanism for given ultralow duty cycles, we must consider $\binom{q^2+q+1}{q+1} \times \binom{q^2+q+1}{2}$ combinations in the worst-case to find all the blocks of

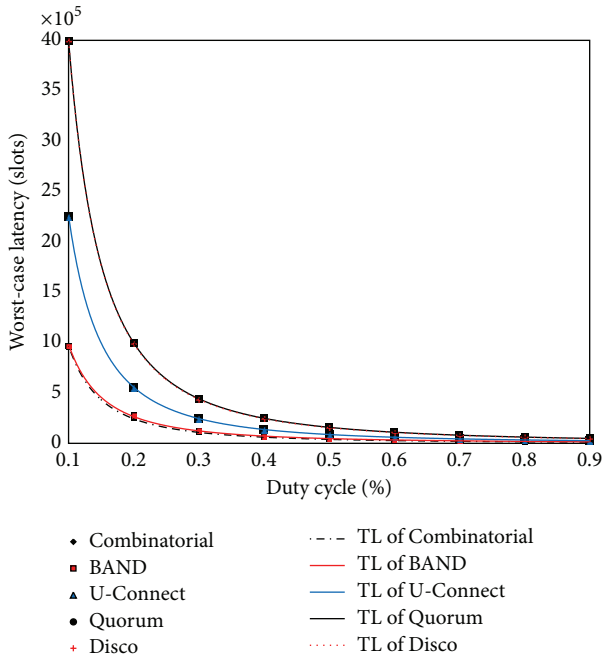

Figure 6: Worst-case latency with ultralow duty cycles.

a $(q^2 + q + 1,\ q + 1,\ 1)$-BIBD using a brute force method. However, the proposed approach provides a simple block construction mechanism that can generate NDSs for virtually any ultralow duty cycles within $O((q_a{}^2+q_a+1)\times(q_b{}^2+q_b+1))$ execution time since each time slot in a $(q_a{}^2+q_a+1,\ q_a+1,\ 1)$-BIBD is replaced by a $(q_b{}^2 + q_b + 1,\ q_b + 1,\ 1)$-BIBD itself. Therefore, Figure 6 just plots the length of slots required to achieve the target duty cycle for *Combinatorial*. However, we do not know how to construct a set of blocks for the *Combinatorial* protocol. For other NDPs, the construction of NDSs is straightforward. The trend lines of *Combinatorial* and *BAND* in Figure 6 demonstrate quite similar patterns.

5. Simulation Study

In this section, we evaluate the discovery latency and energy efficiency of *BAND* and compare its performance with four representative protocols in the literature: *Quorum*, *Combinatorial*, *Disco*, and *U-Connect*. Previously, we have already shown that *BAND* has an outstanding performance in the worst-case latency analysis. Therefore, a performance comparison of *BAND* with other protocols focuses on measuring the average-case performance through this simulation study. While evaluating the performance of *BAND*, we focus on the following two criteria: energy consumption and discovery latency. Energy consumption and discovery latency are defined as follows in this simulation study.

(i) *Energy consumption*: the amount of battery power consumed by all sensors in the network during the neighbor discovery process.

Table 3: Simulation configuration.

Parameter	Configuration
Topology	Random
Network size	100 × 100 m
Number of nodes	50 nodes
Transmission range	20–30 m
MAC protocol	CSMA/CA
Radio module	CC2420 radio
Length of time slot	15 ms

Table 4: Parameters of the energy model.

Parameter	Value
Transmit 0 dBm	17.5 mA
Transmit −25 dBm	8.5 mA
Receive	19.7 mA
CPU active	8 mA

(ii) *Discovery latency*: the total elapsed time that all sensors in the network spend during the neighbor discovery process.

For the simulation study, the proposed approach as well as the four NDPs listed above is implemented with a nesC language [19] for TOSSIM, an efficient power simulation tool for TinyOS applications. TinyOS [20] is one of the most popular application development platforms for WSNs. TinyOS has been widely adapted for performance evaluation, experimental studies, and real sensor applications. TOSSIM can give us the estimation of the energy expenditure of each sensor node based on a given energy model [21].

5.1. Simulation Model. Table 3 shows the configuration parameters of the simulation model for this study. In this simulation environment, a total of 50 sensor devices are randomly deployed within a field of 100 × 100 meters. A MAC protocol for simulation study follows a CSMA/CA manner. We set up CC2420 radio as the radio module for communication. The CC2420 module is commonly used in a number of real sensor network applications. The length of each slot in a discovery schedule is 15 milliseconds (ms). For other simulation parameters such as channel and radio models, we only employ a general link-layer model proposed by the Autonomous Networks Research Group at University of Southern California. For detailed information regarding all parameters, refer to the description of the simulation environments in [22].

5.2. Energy Model. We calculate the energy consumption of sensor nodes by using the parameters in Table 4. The values are taken from the datasheet of CC2420 [21]. Then, we embed a PowerTOSSIM [23–25] model into TOSSIM. This model automatically computes the energy consumed by each sensor node, and we calculate the average energy consumption of the entire network for each neighbor discovery algorithm.

Table 5: Parameter settings of different neighbor discovery protocols.

Protocol	10%	5%	2%	1%
Combinatorial	(*91*, *10*)	(*381*, *20*)	(*2451*, *50*)	(*9507*, *98*)
Quorum	$m = 19$	$m = 39$	$m = 99$	$m = 199$
U-Connect	$p = 13$	$p = 29$	$p = 73$	$p = 149$
Disco	$p_1 = 17$, $p_2 = 23$	$p_1 = 37$, $p_2 = 43$	$p_1 = 93$, $p_2 = 103$	$p_1 = 197$, $p_2 = 199$
BAND	(*147*, *15*)	(*511*, *27*)	(*2821*, *60*)	(*10431*, *112*)

5.3. Parameter Settings for Different Neighbor Discovery Algorithms. Table 5 shows different parameter settings for five asynchronous neighbor discovery techniques, *Combinatorial*, *Quorum*, *U-Connect*, *Disco*, and *BAND*, based on various duty cycles. For the simulation study, these parameters were used for each discovery algorithm to calculate discovery latency and energy consumption. For the construction of schedules in the *Combinatorial* algorithm, we refer to the book of combinatorial designs [9] and select proper designs according to a given duty cycle. *Quorum* is composed of a two-dimensional $m \times m$ array in a row-major manner. In *U-Connect*, a proper prime number p was selected with respect to different duty cycles. *Disco* uses two different prime numbers, p_1 and p_2. According to the recommendation given by the authors of *Disco* for the simulation, we pick two large prime numbers of approximately equal size. For the proposed mechanism, we only produce NDD designs that satisfy the target duty cycles.

5.4. Discovery Latency. We evaluate the performance of five NDPs with respect to the discovery latency here. In this section, we introduce two different discovery latencies used for our simulation study. We assume that each node might have one more neighbor within its communication range in the simulated network topology. In addition, each node in the network uses the number of its neighbors to calculate the following latency metrics:

(i) *Average discovery latency (AVG Latency)*: The average discovery latency is defined as the time a sensor takes to discover all its neighbors divided by the number of its neighbors. For a sensor with n neighbors, let $latency_i$ be the discovery latency of the ith neighbor

$$AVG\ Latency = \frac{latency_1 + latency_2 + \ldots + latency_n}{n}. \quad (9)$$

(iii) *Maximum discovery latency (MAX Latency)*: the discovery time in which a sensor node finds its last neighbor. This is closely related to the worst-case discovery latency.

In general, a wireless sensor does not have any information about the number of its neighbors. However, in this simulation, we assume every node knows the number of its neighbors just for the performance evaluation of given neighbor discovery algorithms. To make the simulation fair to every node in the network, we also assume that every node begins its discovery schedule at the same time. Without this assumption, early starters always have higher discovery latency than late nodes. Furthermore, we create and inject a random shifting factor into our simulation scenario to make the scheduling of each node asynchronous. Thus, each node has a different discovery schedule and the schedule is shifted randomly at the beginning of the simulation.

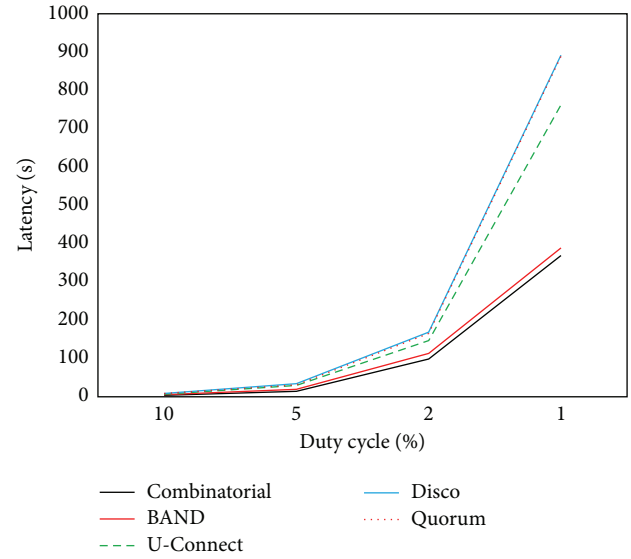

Figure 7: MAX Latency of five neighbor discovery protocols.

Figure 7 illustrates the MAX Latency of five different NDPs from 10% to 1% duty cycles. These simulation results are averaged over the results of 30 random simulation runs for computing the average MAX Latency of each discovery protocol. As shown in Figure 7, the relatively higher duty cycle scenarios such as 10% or 5% show similar MAX Latency values for all five protocols. However, in the case of 2% and 1% duty cycles, the graph shows distinctive trends. First, the *Combinatorial* protocol shows the best performance in terms of the MAX Latency. This is because, as we have already shown in Table 5, the total number of slots of the *Combinatorial* protocol is the smallest among the set of NDPs. Although all five NDPs run at the same duty cycle, the design of discovery schedules ultimately plays an important role with the worst-case discovery latency. This simulation result perfectly matches with the worst-case latency analysis in Section 4.4. In addition, both *Combinatorial* and *BAND* have almost the same trends through 10% to 1% duty cycles because the differences of total slots between these two techniques are trivial. Similarly, the results of *Quorum* and *Disco* are approximately the same due to the same reason.

We also measure the AVG Latency of five NDP schemes. Measuring the average-case performance is the primary purpose of this simulation study. If a certain sensor node has more than one neighbor, it keeps finding its neighbor until all neighbors are found. Figure 8 shows the AVG Latency of five techniques. As depicted in Figure 8, the AVG Latency

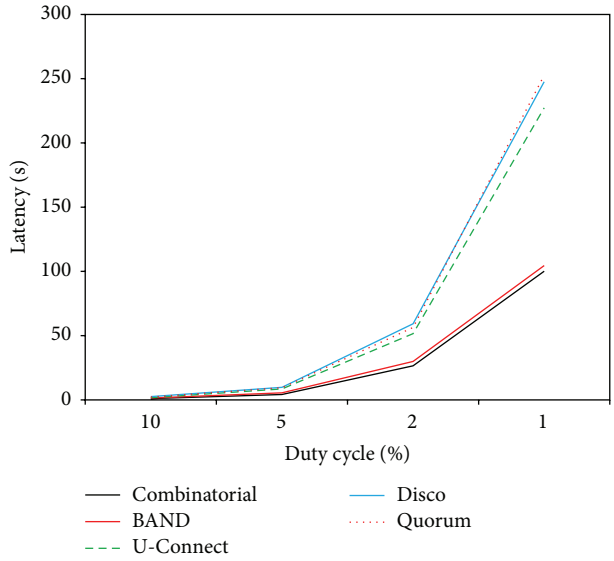

Figure 8: AVG Latency of five neighbor discovery protocols.

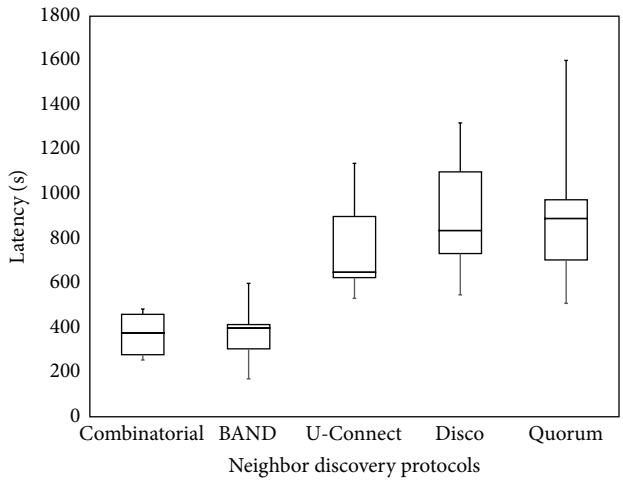

Figure 9: Distributions of MAX Latency of 1% duty cycle.

shows nearly the same pattern shown in Figure 7. The AVG Latency values are clearly separated from a duty cycle of 2%. The pattern of three protocols, *U-Connect*, *Quorum*, and *Disco*, differs from *Combinatorial* as well as *BAND*. Based on these experimental results, we conclude that the length of NDSs also influences the AVG Latency.

Figure 9 demonstrates the range of the MAX Latency of five NDPs at 1% duty cycle. This box plot represents a minimum, 1st quartile, median, 3rd quartile, and maximum of the MAX Latency. As can be seen in Figure 9, the median value of the MAX Latency of the *Combinatorial* is similar to that of *BAND*. Also, the minimum value of the proposed algorithm is slightly better than that of the *Combinatorial*. From these experimental results, we note that the overall discovery latency of *BAND* is similar to that of the *Combinatorial*. The largest value of the MAX Latency was from *Quorum* (see the long tail) in Figure 9. The NDSs of *Quorum* show

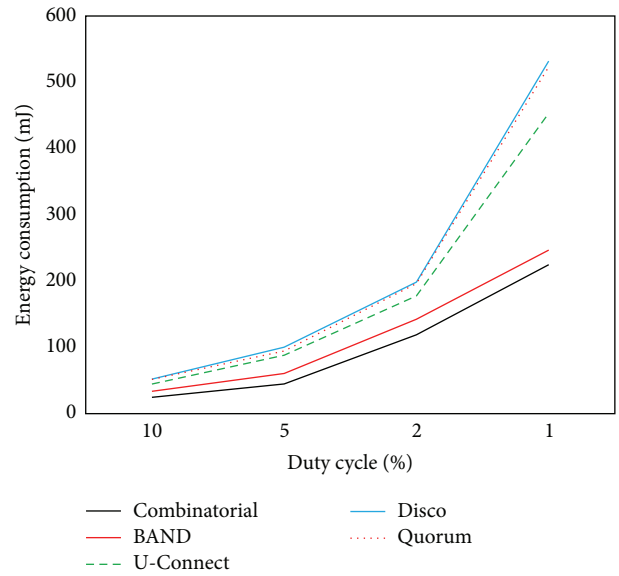

Figure 10: Energy consumption of five neighbor discovery protocols.

a consecutive wake-up pattern in a certain portion of the time slots followed by a lengthy sleep period. It causes a wide range of the discovery latency distribution since two nodes in the same communication range may have an infrequent overlapped wake-up pattern in their discovery schedules, in particular at 1% duty cycle.

5.5. Power Consumption. In this section, we measure the amount of energy consumed by each sensor during neighbor discovery. The PowerTOSSIM model was adapted for measuring the energy expenditure of CPU, sensing modules, EEPROM, ADC, LED, and a radio interface (CC2420 module) of wireless sensor devices. After the completion of each simulation scenario, PowerTOSSIM produces the energy consumption of these components. We have implemented a TinyOS application to measure the amount of battery power consumed by each sensor, and then TOSSIM runs this application to provide the energy expenditure of all devices based on the PowerTOSSIM model.

The PowerTOSSIM first measures the amount of energy consumed by each radio interface for all sensors and aggregates the energy consumption of the individual radio interfaces in order to compute the average energy expenditure of the entire network. As expected, the patterns of the communication energy expenditure of five NDPs shown in Figure 10 are quite similar to those of latency distributions shown in Figures 7 and 8.

Figure 11 depicts the distribution of energy consumption of five NDPs at 1% duty cycle. From this graph, we can explain why we get the result shown in Figure 10. Obviously, the total number of slots considerably impacts the energy consumption. In addition, from Figure 11, we observe that two box plots in Figures 9 and 11 also show similar patterns. Therefore, we conclude that there is a tight relationship

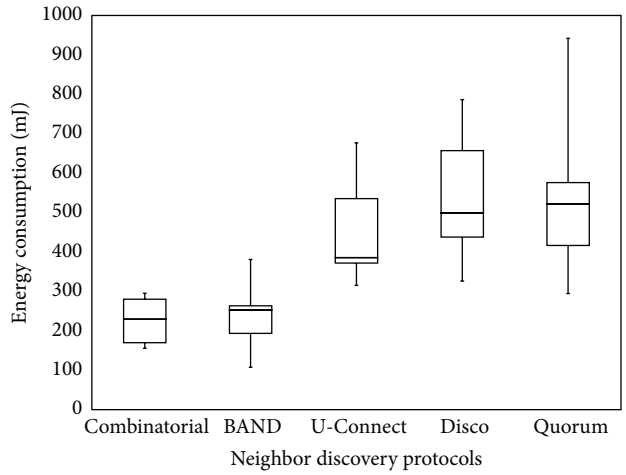

Figure 11: Distributions of energy consumption of 1% duty cycle.

between the MAX Latency and the energy consumption of NDPs.

6. Conclusion

In this paper, we propose a new block construction technique of NDSs for asynchronous neighbor discovery. The proposed mechanism combines two block designs to produce a new discovery schedule with an application-specific duty cycle. By merging two block designs, we generate discovery schedules with the desired properties of BIBDs. Furthermore, we compare the worst-case discovery latency of five NDPs with the same duty cycle. In general, given a duty cycle, *BAND* shows significantly lower discovery latency than *Quorum*, *Disco*, and *U-Connect*.

In the simulation study, we focus on two major criteria of neighbor discovery problems: energy consumption and discovery latency. Related studies of neighbor discovery also deal with these metrics in their performance evaluation. We have implemented four representative NDPs in the literature and *BAND* with a network simulation tool called TOSSIM and have adapted a PowerTOSSIM model for energy expenditure. The results of our simulation study demonstrate that there is a strong relationship between energy consumption and maximum discovery latency. In addition, we find that the total number of points of NDSs mainly affects the performance of discovery protocols. The performance of the proposed algorithm is much better than that of *Quorum*, *Disco*, and *U-Connect*, especially at a low duty cycle like 1%. Based on the simulation results, we conclude that making the total slots of discovery schedules small while maintaining at least one overlap of the sensor's active points with all neighbors is very critical for latency as well as energy expenditure.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2014-017928).

References

[1] M. J. McGlynn and S. A. Borbash, "Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks," in *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc '01)*, pp. 137–145, October 2001.

[2] 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*, vol. 10, no. 1-2, pp. 169–181, 2005.

[3] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, vol. 1, pp. 200–209, IEEE, June 2002.

[4] P. Dutta and D. Culler, "Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications," in *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys '08)*, pp. 71–83, November 2008.

[5] A. Kandhalu, K. Lakshmanan, and R. Rajkumar, "U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '10)*, pp. 350–361, April 2010.

[6] R. Zheng, J. C. Hou, and L. Sha, "Asynchronous wakeup for ad hoc networks," in *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc '03)*, pp. 35–45, June 2003.

[7] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 126–137, November 2003.

[8] I. Anderson, *Combinatorial Designs and Tournaments*, chapter 2, Oxford University Press, Oxford, UK, 1988.

[9] C. J. Colbourn and J. H. Dinitz, *The CRC Handbook of Combinatorial Designs*, CRC Press, New York, NY, USA, 1996.

[10] D. R. Stinson, *Combinatorial Designs: Constructions and Analysis*, Springer, New York, NY, USA, 2004.

[11] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Wireless Networks*, vol. 8, no. 5, pp. 481–494, 2002.

[12] C. F. Chiasserini and R. R. Rao, "A distributed power management policy for wireless ad hoc networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, pp. 1209–1213, September 2000.

[13] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–105, 2002.

[14] K. Han, J. Luo, Y. Liu, and A. Vasilakos, "Algorithm design for data communications in duty-cycled wireless sensor networks: a survey," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 107–113, 2013.

[15] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi, "The evolution of MAC protocols in wireless sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 101–120, 2013.

[16] W. Sun, Z. Yang, X. Zhang, and Y. Liu, "Energy-efficient neighbor discovery in mobile Ad Hoc and wireless sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1448–1459, 2014.

[17] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *Proceedings of the 9th Internatinal Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '00)*, pp. 93–104, ACM, November 2000.

[18] C. Godsil, "Combinatorial Design Theory," 2010, http://www.math.uwaterloo.ca/~kpurbhoo/winter2012-co634/Designs.pdf.

[19] D. Gay, P. Levis, R. von Behren et al., "The nesC language: a holistic approach to networked embedded systems," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '03)*, pp. 1–11, ACM, San Diego, Calif, USA, June 2003.

[20] http://www.tinyos.net/.

[21] CC2420 Datasheet, http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf.

[22] http://www.tinyos.net/tinyos-2.x/doc/html/tutorial/usc-topologies.html.

[23] E. Perla, A. Ó. Catháin, R. S. Carbajo, M. Huggard, and C. Mc Goldrick, "PowerTOSSIM Z: realistic energy modelling for wireless sensor network environments," in *Proceedings of the 3rd ACM International Workshop on Performance Monitoring, Measurement, and Evaluation of Heterogeneous Wireless and Wired Networks (PM2HW2N '08)*, pp. 35–42, October 2008.

[24] T. V. Prabhakar, S. Venkatesh, M. S. Sujay, J. Kuri, and K. Praveen, "Simulation blocks for TOSSIM-T2," in *Proceedings of the 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, pp. 17–23, Bengaluru, India, January 2008.

[25] V. Shnayder, M. Hempstead, B.-R. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 188–200, November 2004.