sustainability

MDPI

Article

Asymmetric Block Design-Based Neighbor Discovery Protocol in Sensor Networks

Sangil Choi [1] and Gangman Yi [2,*]

[1] Department of Computer Science, Swaziland Christian University, P.O. Box A624 Swazi Plaza, Mbabane H101, Swaziland; sgilchoi@gmail.com

[2] Department of Computer Science & Engineering, Gangneung-Wonju National University, Wonju 220-711, Korea

* Correspondence: gangman@cs.gwnu.ac.kr

Academic Editors: James Park, Han-Chieh Chao and Marc A. Rosen
Received: 22 March 2016; Accepted: 25 April 2016; Published: 29 April 2016

Abstract: Neighbor discovery is one of the emerging research areas in a wireless sensor network. After sensors are distributed, neighbor discovery is the first process to set up a communication channel with neighboring sensors. This paper proposes a new block design–based asymmetric neighbor discovery protocol for sensor networks. We borrow the concept of combinatorial block designs for our block combination scheme for neighbor discovery. First, we introduce an asymmetric neighbor discovery problem and define a target research question. Second, we propose a new asymmetric block design–based neighbor discovery protocol and explain how it works. Third, we analyze the worst-case neighbor discovery latency numerically between our protocol and some well-known protocols in the literature, and compare and evaluate the performance between the proposed protocol and others. Our protocol reveals that the worst-case latency is much lower than that of *Disco* and *U-Connect*. Finally, we conclude that the minimum number of slots per a neighbor schedule shows the lowest discovery time in terms of discovery latency and energy consumption.

Keywords: sustainable wireless sensor networks; neighbor discovery protocol; design theory; balanced incomplete block design

1. Introduction

The main energy source of wireless sensors is usually a battery. In addition, it is well known that it is very difficult to replace or recharge the batteries of sensors after they are deployed. Hence, most recent studies of the wireless sensor network (WSN) area focus on energy consumed by wireless sensors [1–6]. Most wireless sensors operate in an *ad hoc* mode, which means they do not have allocated IP addresses. After they are deployed in specific areas, there is no location information indicating where their neighbors are located for communication. In other words, until they know and obtain the information of their neighbors, there is no way for them to communicate with each other. This is the reason why finding neighbors becomes one of the critical issues in WSNs. Without a proper neighbor discovery mechanism, wireless nodes might waste a lot of energy to set up a communication channel with others.

A variety of neighbor discovery protocols have been developed in order to address the neighbor discovery problem. A probabilistic neighbor discovery protocol, the *Birthday* protocol [7], has been proposed for asynchronous neighbor discovery based on discovery probabilities. A quorum-based neighbor discovery protocol has been introduced in multi-hop *ad hoc* networks [8,9]. A node arbitrarily chooses one column and one row of entries for neighbor discovery in a given two-dimensional $m \times m$ array. While *Disco* [10] selects two different prime numbers, adapting the Chinese Remainder Theorem, *U-Connect* [11] only uses a single prime number for asynchronous neighbor discovery. Lastly,

Zheng *et al.* [12] employ the concept of combinatorial block designs to create neighbor discovery schedules. Their solution only deals with a symmetric operation when the duty cycle of all sensor nodes is uniform in the network.

Especially, in this paper, we propose an asymmetric block design–based neighbor discovery protocol. By applying the concept of block design theory to the neighbor discovery problem, we suggest an energy-efficient neighbor discovery mechanism. Some previously proposed neighbor discovery protocols have also adapted the block design. However, they cannot support an asymmetric operation. In this scenario, the duty cycles of nodes can be different. Some nodes operate at a lower duty cycle based on their energy expenditure level, but others perform their duty at a higher duty cycle because they have to carry and relay urgent packets to the base station. This scenario is more realistic and general than the symmetric operation. The basic idea behind the proposed technique 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 the proposed protocol with existing asymmetric neighbor discovery protocols.

This paper is organized as follows: Section 2 reviews related works in neighbor discovery. Section 3 introduces an asymmetric neighbor discovery problem. Section 4 details how the proposed neighbor discovery protocol can solve the problem and summarizes our key contributions. In Section 5, we compare the performance of the proposed scheme with that of other protocols in the literature. Finally, we conclude the paper in Section 6.

2. Related Work

The 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 [7] 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 on 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 in discovery probabilities. In addition, this probabilistic approach cannot guarantee that each sensor node always meets the other within a certain amount of time.

Tseng *et al.* [8,9] introduce a *Quorum*-based protocol for multi-hop *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 two-fold challenge: (1) it is possible that two different schedules overlap frequently based on the selection of an arbitrary row and column of entries; and (2) two 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.* [12] apply the concept of combinatorial block designs using difference sets [13–15] 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 provide the 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* [11], 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 neighbor discovery schedules.

Disco [12] 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 the *Quorum* and *Birthday* protocols for asymmetric scenarios and are 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* [11], uses a single prime number, p. Instead of just waking up only one slot in every p slots, the nodes also wake up $\frac{p+1}{2}$ consecutive slots every p^2 slots. 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, e.g., below 1%, the number of contiguous active slots grows considerably and worst-case discovery latency increases greatly.

Searchlight [16] presents a beacon scheduling mechanism. In this scheme, each node sends two beacons at the beginning and the end of the slot. The node remains in listening mode in between beacons. It shows much better performance in a symmetric operation, but needs to be improved in an asymmetric scenario. In [17], they adopt a code-based formulation of the neighbor discovery problem using a perfect difference set and design *Diff-Code* for the symmetric case. In addition, by extending *Diff-Code* to *ADiff-Code*, they try to deal with asymmetric neighbor discovery. In [18], they formulate a heterogeneous multi-channel neighbor discovery problem and establish a theoretical framework of the problem. Furthermore, they propose a novel multi-channel discovery protocol called *Mc-Dis* (Multi-channel Discovery).

3. Asymmetric Neighbor Discovery Problem

Most wireless sensor devices usually support two power modes: *active* and *power-saving (sleep)* to save their battery power [19,20]. 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, these nodes should be in the active mode at the same time. By simply synchronizing the timer of wireless sensors through exchanging periodic control packets, sensor nodes can meet each other within a certain amount of time. Diverse time synchronization mechanisms [21–24] are developed to address the neighbor discovery problem. However, this approach has a huge overhead for exchanging control messages between nodes at a regular interval of time. It is well known that periodic wireless message exchange is one of the primary sources of battery drainage [25]. Therefore, this technique is not appropriate for ultra-low power operations in a sensor network.

Instead of using time synchronization schemes, 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.

3.1. Neighbor Discovery Schedule

Wireless sensors keep changing their power mode in order to save their energy from inactive (*sleep*) to active, and *vice versa*, based on their own power-saving policy. To save more energy, we can make a schedule in which a node might turn off its radio most of its lifetime and only turn it on shortly. In this policy, we may guarantee the energy conservation of nodes. However, it might be difficult to guarantee that the energy-saving policy ensures wireless sensors keep communicating with their neighbors properly. From a different point of view, active and frequent communication might be more important than energy conservation in some special circumstances. We could set up a new policy in which sensors might turn on their radios frequently for easy communication with their neighbors. Therefore, we need a clearly defined time schedule that explains how many times a node wakes up and sleeps within a pre-defined cycle. As we discussed in the previous paragraph, a wireless node can stay in the active or power-saving mode. It is possible to represent these two modes with a binary number, where number '1' expresses an active mode and '0' denotes a power-saving mode. By using this simple binary notation, we can define a *schedule* as follows:

Definition 1: A schedule S is a sequence of 0 and 1 representing the power-saving and active modes, respectively. An active mode is a state where a sensor turns on its radio and is ready to send or receive data packets and a sleep mode is a state where the sensor turns off its radio and only senses and collects some environmental data. In S, the binary value '0' represents the power-saving mode and the value '1' denotes the active mode.

According to different power-saving policies of sensor networks, each policy can decide how many times nodes turn their radios on and find their neighbors during each active mode of nodes. For instance, while some numbers of nodes need to wake up frequently, others sleep most of their lifetime and wake up once in a while. In other words, there should be a duty cycle that contains how many active and power-saving modes comprise one schedule in a given time interval based on the power-saving policy of sensor nodes. The following is the definition of a *duty cycle*:

Definition 2: A duty cycle is the percentage of the ratio of the number of active modes over the total number of active and power-saving modes per a given time interval. 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 modes and T is the total number of active and power-saving modes in the schedule.

We need the following two definitions regarding a combinatorial block design. The following definitions came from [15].

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, and
(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 $v > k \geqslant 2$. 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:

(A) $|X| = v$,
(B) Each block contains exactly k points, and
(C) Every pair of distinct points is contained in exactly λ blocks.

A (7, 3, 1)-BIBD is one of the typical BIBDs in the case of $\lambda = 1$. In the *(7, 3, 1)*-BIBD, X = *{1, 2, 3, 4, 5, 6, 7}*, and A = *{124, 235, 346, 457, 561, 672, 713}*.

We can illustrate a neighbor discovery schedule (NDS) using 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 power-saving modes. Figure 1 shows an example of the NDS. In this example, this schedule consists of seven numbers of active and power-saving modes and the duty cycle of the NDS is about 43% ($\frac{3}{7} \times 100$ %).

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.

3.2. Asymmetric Operation

The duty cycles of all wireless sensors in the network could be the same, which means each sensor has the same number of active and power-saving modes within its duty cycles. We call this scenario a *symmetric* operation. We can consider a more realistic situation in which all sensors do not have to have the same duty cycle. Some nodes operate at a lower duty cycle based on their energy expenditure level, but others perform their duty at a higher duty cycle because they have to carry and relay urgent packets to the base station. This scenario is more realistic and general than the symmetric operation. In an *asymmetric* operation, the duty cycles of nodes can be different.

In this paper, we address the following research question that is closely related to the asymmetric neighbor discovery problem: How can we enable two neighboring nodes to communicate with each other under the asymmetric scenario?

4. Asymmetric Block Designed–Based Neighbor Discovery Mechanism

In a symmetric operation, all nodes in the network have the same duty cycle. This assumption is somewhat limited in a general wireless network environment. For example, wireless sensor nodes are resource-constrained and battery-oriented so that they require low duty cycles. On the other hand, powered-on devices can maintain higher duty cycles compared to wireless sensors. Therefore, there are a number of nodes operating with different duty cycles under asymmetric scenarios. The main research interest of this chapter is to provide a fine solution to the asymmetric neighbor discovery problem using a technique called block design combination.

4.1. Block Design Combination Scheme

The basic concept under this combination scheme is combining two block designs. There are three steps to combining two block designs: (1) choose two target block designs; (2) replace each active slot of one design (*base*) with the entire blocks of the other (*replacement*); and (3) generate a new block design for neighbor discovery. Our proposed method can virtually construct a neighbor discovery schedule with almost any duty cycles by selecting a proper set of previously well-known block designs. Figure 2 explains the combining process of two block designs. Note that you can refer to [26] if you want to know the combining process in detail.

Theorem 1 (Theorem 1.2.1 in [27]*)*: If (X, A) is a symmetric design with parameters (v, k, λ), then any two distinct blocks have exactly λ points in common.

Theorem 2: If two distinct schedules s_i and s_j are mapped to 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 the (X, A) is a symmetric design with parameters (v, k, λ), we can write the 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 to two different blocks in A, respectively. Let two blocks in A be B_i and B_j. According to Theorem 1, 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 to B_i and B_j, have λ common active time slots.

Theorem 2 shows that two arbitrary NDSs have common active time slots when we apply the symmetric BIBD to the design of NDSs using our block design combination scheme. Consequently, the proposed block design combination scheme combines two BIBDs and preserves the properties of the block design so that we can guarantee that two schedules produced by our combination scheme have at least one common active slot without any time synchronization processes.

Algorithm: Combination of Block Designs

```
1: procedure GENERATING A NEW NEIGHBOR DISCOVERY DESIGN
2:     Φ ← (v1, k1, λ1) − BIBD
3:     Ψ ← (v2, k2, λ2) − BIBD
4:     ∅ ← a sleep schedule
5:     for all p ∈ Φ do
6:         replace p by Ψ
7:     end for
8:     for all q ∉ Φ do
9:         replace q by ∅
10:    end for
11: end procedure
```

Figure 2. An algorithm of combining two block designs.

4.2. Problem of Symmetric Block Designs

In an asymmetric environment, two neighboring nodes could have different duty cycles. *Disco* [10] and *U-Connect* [11] can support this environment. However, there are a number of situations where symmetric BIBDs cannot assist an asymmetric operation. Let us assume that one node uses a *(7, 3, 1)*-BIBD and the other adapts a *(21, 5, 1)*-BIBD. Figure 3 represents discovery schedules from these two different designs. As seen from this figure, there are no common active slots between the two BIBDs. The duty cycle of a *(7, 3, 1)*-BIBD is about 43% and that of a (*21, 5, 1*)-BIBD is approximately 24%. We cannot guarantee that two arbitrary symmetric block designs have at least one common active slot. In other words, it is impossible to apply the concept of block designs to an asymmetric neighbor discovery problem.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
(7,3,1)	1	1	0	1	0	0	0	1	1	0	1	0	0	0	1	1	0	1	0	0	0
(21,5,1)	0	0	1	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0

Figure 3. Discovery schedules from a *(7, 3, 1)* and *(21, 5, 1)* design.

4.3. Proposed Asymmetric Neighbor Discovery Protocol

The block combination scheme can be used to solve an asymmetric neighbor problem. The primary characteristic of the proposed method is that an active slot of the *base* design is replaced by the *replacement* design. In other words, each active slot of the *base* design contains the *replacement* design. From this characteristic of the block combination technique, we can find the key to the asymmetric neighbor discovery problem.

Definition 5: A *sleep schedule* is a square matrix of order n with all 0's.

Definition 6: Let $\mathcal{S}$ be a (v_1, k_1, 1)-BIBD and $\mathcal{T}$ 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 3: Let A be a (v_a, k_a, 1)-BIBD with a duty cycle d_a, B be a (v_b, k_b, 1)-BIBD with a duty cycle d_b, and C be a (v_c, k_c, 1)-BIBD with a duty cycle d_c. If C is made from A $\oplus$ B then there is at least one common active slot between B and C.

Proof. According to Definition 6, if we combine two BIBDs, A and B, then all active slots of A are replaced by B. Let C be $A \oplus B$. The duty cycles d_b of B are $\frac{k_b}{v_b} \times 100\%$ and d_c of C are $\frac{k_a \times k_b}{v_a \times v_b} \times 100\%$, where $d_b \neq d_c$. According to Theorem 2, any two schedules, s_i and s_j ($i \neq j$) from either B or C, have λ active time slots in common. Therefore, there is at least one common active slot between B and C because $B \subset C$.

Let us assume that we have two groups of wireless sensor nodes and make the wireless network operate in the asymmetric scenario for neighbor discovery. The duty cycle of the first group is about 67% and that of the second group is about 29%. For the 67% duty cycle, the *(3, 2, 1)*-BIBD can be assigned. We call this block design B in this section. To assign a 29% duty cycle to the second group, it is possible to apply the proposed combination scheme to the construction of discovery schedules. Using B and the block combination technique, we can create a new block design, C, for the second group. Because C's duty cycle is 29%, we need a block design A with about a 43% duty cycle. One of the best candidates is the *(7, 3, 1)*-BIBD. Therefore, we perform the $A \oplus B$ operation to combine A and B, and produce C with a 29% duty cycle (43 % $\times$ 67 % $\approx$ 29 %). Finally, we can assign C to the second group for neighbor discovery. Figure 4 represents block designs A and B and Figure 5 shows the combined block design C (= $A \oplus B$).

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

A

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

B

Figure 4. Block designs *A* (*7, 3, 1*) and *B* (*3, 2, 1*).

Let us think about the following discovery schedules to show that there is at least one common active slot between B and C. As we mentioned before, block designs can be used to create an NDS. Hence, one of the blocks from B can be assigned to the first group of sensors and one of the blocks from C can also be assigned to the second group. Then we can set up the duty cycles for two groups. In this situation, we simply assign the first row (block) from B and C to the first and second group for discovery schedules, respectively. Figure 6 illustrates the assignment of discovery schedules. As seen from this picture, we know that there is a six-fold overlap between the two schedules. In the case of the first group, its schedule repeats every three time slots because the length of the total slots of B is

three. In addition, there are two active slots per cycle in the schedule of the first group. In Figure 6, there is only one cycle of the schedule of the second group. This schedule consists of 21 total slots and six active slots. Furthermore, every active slot of A is converted to B during the combining process. Therefore, C has the same wake-up and sleep pattern as B. This is the reason why these two different discovery schedules have at least one more overlap in this example. Consequently, we guarantee that there is at least one common active slot in these two different discovery schedules with different duty cycles. Finally, our proposed block combination scheme can solve the asymmetric neighbor discovery problem using Theorem 3.

Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	1	1	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
2	0	1	1	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
3	1	0	1	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
4	0	0	0	1	1	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
5	0	0	0	0	1	1	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0
6	0	0	0	1	0	1	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	1	1	0	0	0	0
8	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	1	1	0	0	0
9	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	1	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	1	1	0
11	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	1	1
12	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	1	0	1
13	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0
14	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0
15	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0
16	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0
17	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1
18	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1
19	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0
20	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1
21	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1

Figure 5. Combined block design C (= $A \oplus B$).

First Group:	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0
Second Group:	1	1	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0

Figure 6. Discovery schedule assignment for two sensor groups.

5. Performance Evaluation

We only consider three different neighbor discovery protocols to evaluate the performance in the asymmetric scenario—*the Proposed Protocol*, *Disco*, and *U-Connect*—because these three protocols can support an asymmetric operation. An asymmetric ratio R [11] is used in *U-Connect* to measure the asymmetric performance. R is defined as follows:

$$R = \frac{Duty\ cycle\ of\ a\ higher\ duty\ cycle\ node}{Duty\ cycle\ of\ a\ lower\ duty\ cycle\ node} \tag{2}$$

For example, if a duty cycle of a lower duty cycle node is 2% and that of a higher duty cycle node is 10%, then R is 5. In this section, we borrow this performance metric to compare the performance of three protocols in the asymmetric environment. Table 1 shows the parameter settings for simulation in the asymmetric scenario. Basically, *U-Connect* uses one prime number in the symmetric operation, but needs to use two prime numbers in the asymmetric case.

Table 1. Parameter settings for simulation under an asymmetric scenario.

Protocol	R = 1 (DC = 10%)	R = 2 (DC = 10% and 5%)	R = 5 (DC = 10% and 2%)	R = 10 (DC = 10% and 1%)
U-Connect	P = 13	P_1 = 13, P_2 = 29	P_1 = 13, P_2 = 73	P_1 = 13, P_2 = 149
Disco	P_1 = 17, P_2 = 23	P_1 = 17, P_2 = 23 P_3 = 37, P_4 = 43	P_1 = 17, P_2 = 23 P_3 = 93, P_4 = 103	P_1 = 17, P_2 = 23 P_3 = 197, P_4 = 199
Proposed	*(147, 15)*	*(147, 15)* *(511, 27)*	*(147, 15)* *(2821, 60)*	*(147, 15)* *(10431, 112)*

For performance evaluation, 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:

Energy Consumption: The amount of battery power consumed by all sensors in the network during the neighbor discovery process.

Discovery Latency: The total elapsed time that all sensors in the network spend during the neighbor discovery process.

Figure 7 illustrates the discovery latency of three different neighbor discovery protocols (NDPs) under an asymmetric scenario ranging from the asymmetric ratio 1 to 10. These simulation results are averaged over the results of 10 random simulation runs in order to compute the average discovery latency of each discovery protocol. As shown in Figure 7, a relatively lower asymmetric ratio such as 1 or 2 shows similar discovery latency for all three protocols. However, in the case of 5 and 10, the graph shows dissimilar trends. First, *Proposed* shows the lowest discovery time in terms of discovery latency. This is because, as we have already discussed before, the total number of slots of the *Proposed* protocol is the smallest among these NDPs. Although all three NDPs run in the same environment, the design of discovery schedules ultimately plays an important role with the worst-case discovery latency.

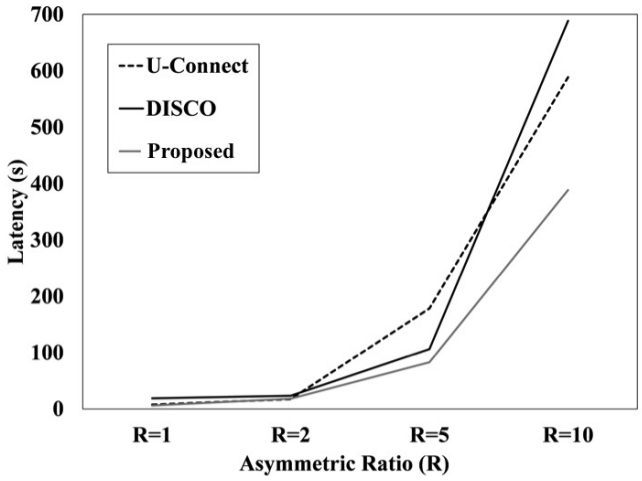

Figure 7. Discovery latency in an asymmetric scenario.

We also measure the amount of energy consumed by each sensor using different discovery protocols. As expected, the patterns of the communication energy expenditure of the three NDPs shown in Figure 8 are quite similar to those of the latency distributions shown in Figure 7. Similarly, the total number of slots significantly affects the energy consumption. Therefore, we conclude that there is a close relationship between discovery latency and the energy consumption in the neighbor discovery problem in asymmetric environments.

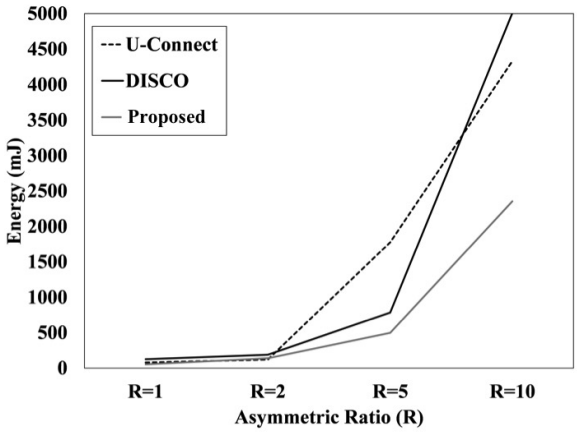

Figure 8. Energy consumption in an asymmetric scenario.

6. Conclusions

We discuss the problem of a symmetric solution for neighbor discovery and introduce a different assumption in the neighbor discovery problem in this paper. In a symmetric operation, there is an assumption that all nodes operate with the same duty cycle. Under an asymmetric scenario, there are a number of groups maintaining different duty cycles. Previous studies of the discovery protocol have also dealt with the asymmetric case.

The problem with using symmetric block designs in an asymmetric environment is that we cannot guarantee at least one rendezvous point between two discovery schedules. That is why *Combinatorial* cannot support an asymmetric operation. However, our block combination technique can solve this problem by controlling the combination of two block designs.

We conduct simulation experiments to show that our approach can support the asymmetric operation for neighbor discovery. There are two important measurements: the discovery latency and energy consumption. We have already shown that the length of the total slot of the discovery schedule impacts the performance of the discovery protocol. Our simulation results also agree with these results for the asymmetric scenario. We finally conclude that designing the minimum number of total slots in a discovery schedule is one of the most important factors in the neighbor discovery problem.

Author Contributions: All authors contributed equally to this work. Sangil Choi studied the concept of Balanced Incomplete Block design, applied to neighbor discovery, designed the block combination scheme, and wrote the paper. Gangman Yi designed and performed experiments, analyzed data and wrote the paper.

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

References

1. Meng, T.; Wu, F.; Yang, Z.; Chen, G.; Vasilakos, A.V. Spatial Resuability-Aware Routing in Multi-Hop Wireless Networks. *IEEE Trans. Comput.* **2015**, *65*, 244–255. [CrossRef]
2. Chen, L.; Li, Y.; Vasilakos, A.V. Oblivious Neighbor Discovery for Wireless Devices with Directional Antennas. In Proceedings of the IEEE INFOCOM, San Francisco, CA, USA, 10–15 April 2016.
3. Yao, Y.; Cao, Q.; Vasilakos, A.V. EDAL: An Energy-Efficient, Delay-Aware, and Lifetime-Balancing Data Collection Protocol for Heterogeneous Wireless Sensor Networks. *IEEE/ACM Trans. Networking* **2014**, *23*, 810–823. [CrossRef]
4. Bhuiyan, M.Z.A.; Vasilakos, A.V. Local Area Prediction-Based Mobile Target Tracking in Wireless Sensor Networks. *IEEE Trans. Comput.* **2014**, *64*, 1968–1982. [CrossRef]
5. Zhang, X.M.; Zhang, Y.; Yan, F.; Vasilakos, A.V. Interference-Based Topology Control Algorithm for Delay-Constrained Mobile Ad Hoc Networks. *IEEE Trans. Mob. Comput.* **2014**, *14*, 742–754. [CrossRef]
6. Xiao, Y.; Peng, M.; Gibson, J.; Xie, G.G.; Du, D.-Z.; Vasilakos, A.V. Tight Performance Bounds of Multihop Fair Access for MAC Protocols in Wireless Sensor Networks and Underwater Sensor Networks. *IEEE Trans. Mob. Comput* **2011**, *11*, 1538–1554. [CrossRef]

7. 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; pp. 137–145.
8. Jiang, J.; Tseng, Y.; Hsu, C.; Lai, T. Quorum-based asynchronous power-saving protocols for IEEE 802.11 Ad Hoc networks. In Proceedings of the 2003 International Conference on Parallel Processing, Kaohsiung, Taiwan, 6–9 October 2003; Volume 10, pp. 257–264.
9. Tseng, Y.-C.; Hsu, C.-S.; Hsieh, T.-Y. Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. In Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, New York, NY, USA, 23–27 June 2002; pp. 200–209.
10. 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, 4–7 November 2008; pp. 71–84.
11. Kandhalu, A.; Lakshmanan, K.; Rajkumar, R. 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, Stockholm, Sweden, 12–15 April 2010; pp. 350–361.
12. 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; pp. 35–45.
13. Anderson, I. *Combinatorial Designs and Tournaments*; Oxford University Press: Oxford, UK, 1988.
14. Colbourn, C.J.; Dinitz, J.H. *The CRC Handbook of Combinatorial Designs*; CRC Press: Boca Raton, FL, USA, 1996.
15. Stinson, D.R. *Combinatorial Designs: Constructions and Analysis*; Springer: Berlin, Germany, 2004.
16. 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.
17. Meng, T.; Wu, F.; Chen, G. On designing neighbor discovery protocols: A code-based approach. In Proceedings of the IEEE Conference on Computer Communications, Toronto, ON, Canada, 27 April–2 May 2014.
18. Chen, L.; Bian, K.; Zheng, M. Heterogeneous multi-channel neighbor discovery for mobile sensing applications: Theoretical Foundation and Protocol Design. In Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking & Computing, Philadelphia, PA, USA, 11–14 August 2014.
19. Chen, B.; Jamieson, K.; Balakrishnan, H.; Morris, R. Span: An energy-efficient coordination algorithm for topology maintenance in Ad Hoc wireless networks. In Proceedings of the 7th annual international conference on Mobile computing and networking, Rome, Italy, 16–21 July 2001; Volume 8, pp. 481–494.
20. Chiasserini, C.; Rao, R.R. A distributed power management policy for wireless ad hoc networks. In Proceedings of the Wireless Communications and Networking Confernce, Chicago, IL, USA, 23–28 September 2000; pp. 1209–1213.
21. Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. A survey on sensor networks. *IEEE Commun. Mag.* **2002**, *40*, 102–114. [CrossRef]
22. Han, K.; Luo, J.; Liu, Y.; Vasilakos, A.V. Algorithm design for data communications in duty-cycled wireless sensor networks: A survey. *IEEE Commun. Mag.* **2013**, *51*, 107–113. [CrossRef]
23. Huang, P.; Xiao, L.; Soltani, S.; Mutka, M.W.; Xi, N. The evolution of MAC protocols in wireless sensor networks: A survey. *IEEE Commun. Surv. Tut.* **2013**, *15*, 101–120. [CrossRef]
24. Sun, W.; Yang, Z.; Zhang, X.; Liu, Y. Energy-efficient neighbor discovery in mobile Ad Hoc and wireless sensor networks: A survey. *IEEE Commun. Surv. Tut.* **2014**, *16*, 1448–1459. [CrossRef]
25. Hill, J.; Szewczyk, R.; Woo, A.; Hollar, S.; Culler, D.; Pister, K. System architecture directions for networked sensors. In Proceedings of the Ninth International Conference on Architectural Support for Programming Language and Operating System, Cambridge, MA, USA, 13–15 November 2000; pp. 93–104.
26. Choi, S.; Lee, W.; Song, T.; Yoon, J. Block Design-based Asynchronous Neighbor Discovery Protocol for Wireless Sensor Networks. *J. Sens.* **2015**. [CrossRef]
27. Godsil, C. Combinatorial Design Theory, 2010. Available online: http://www.math.uwaterloo.ca/~kpurbhoo/co634/Designs.pdf (accessed on 29 April 2016).

© 2016 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).