PD-DESYNC: Practical and Deterministic Desynchronization in Wireless Sensor Networks

Sang-Hyun Hyun[1], Geon Kim[2], and Dongmin Yang[2*]
[1] SOFTITECH Co., Ltd., Jukdong-ro, Yuseong-gu, Daejeon, Republic of Korea
[e-mail: shhyun@sit21c.com]
[2] Graduate School of Archives and Records Management, Chonbuk National University, Baekje-daero, Jeonju-si, Jeollabuk-do, Republic of Korea
[e-mail: godardkim@jbnu.ac.kr (G. K.), dmyang@jbnu.ac.kr (D. Y.)]
*Corresponding author: Dongmin Yang (dmyang@jbnu.ac.kr)

Received March 20, 2018; revised June 15, 2018; revised August 9, 2018; accepted February 12, 2019; published August 31, 2019

Abstract

Distributive desynchronization algorithms based on pulse-coupled oscillator (PCO) models have been proposed for achieving collision-free wireless transmissions. These algorithms do not depend on a global clock or infrastructure overheads. Moreover, they gradually converge to fair time-division multiple access (TDMA) scheduling by broadcasting a periodic pulse signal (called a 'firing') and adjusting the next firing time based on firings from other nodes. The time required to achieve constant spacing between phase neighbors is estimated in a closed form or via stochastic modeling. However, because these algorithms cannot guarantee the completion of desynchronization in a short and bounded timeframe, they are not practical. Motivated by the limitations of these methods, we propose a practical solution called PD-DESYNC that provides a short and deterministic convergence time using a flag firing to indicate the beginning of a cycle. We demonstrate that the proposed method guarantees the completion of desynchronization within three cycles, regardless of the number of nodes. Through extensive simulations and experiments, we confirm that PD-DESYNC not only outperforms other algorithms in terms of convergence time but also is a practical solution.

Keywords: Desynchronization, Distributed algorithms, Wireless MAC protocol, Time division multiple access, Wireless Sensor Networks

1. Introduction

Wireless sensor networks (WSNs) have been used widely in different domains and have recently been further highlighted owing to a rapidly increasing interest in the Internet of Things (IoT). However, networking among mobile nodes remains considerably restricted because of the scarcity of resources for the nodes and the heaviness from layering stacks of standards. For short-range wireless communications, medium access control (MAC) protocols can be placed into two categories: contention-based and contention-free MAC protocols. Carrier-sense multiple access with collision avoidance (CSMA/CA) is a contention-based MAC protocol for accessing an available shared network medium for data transmission. In this domain, a node verifies if a channel is idle and able to transmit data. If the channel is busy, the node holds the transmission. When the channel is not idle, nodes contend for shared network media using a binary exponential backoff algorithm. If the nodes recognize the shared network channel as idle and simultaneously transmit data, a collision occurs. Thus, CSMA/CA is a simple and adaptive protocol that operates well for small numbers of nodes and variable traffic; however, it experiences significant latency and message loss owing to competition and collisions between data packets. Many TDMA-based wireless sensor MAC protocols [1–4] have been introduced. TDMA is a contention-free MAC protocol where time is divided into frames that include several time slots, and each node can communicate with the others in its assigned slot. Furthermore, predefined and dedicated time slots provide a collision-free environment for data communication. In general, TDMA-based MAC protocols are more efficient than CSMA/CA-based protocols in terms of energy efficiency and bandwidth utilization, regardless of traffic load. Moreover, time synchronization is the essential and most challenging issue in TDMA-based MAC protocols. The synchronization among the nodes is generally achieved through a central unit, such as a base station or an access point. However, such centralized MAC scheduling is not suitable for WSNs, where all nodes are homogeneous and resource-constrained devices.

Desynchronization is an attractive primitive for WSN networks in the sense that it can achieve fair TDMA scheduling via a simple, distributed rule without a centralized coordinator. In the majority of desynchronization schemes [5–7], nodes are modeled by pulse-coupled oscillators (PCOs) that broadcast individually in a periodic manner. PCOs were designed from synchronization that was inspired by the naturally occurring biological synchronization required for heartbeats among fireflies [8–11]. All nodes, interconnected through direct wireless links, periodically broadcast a pulse signal (called a ‘firing’) message during every firing cycle, T. Then, they update each with their own next firing times based on the firing messages of the other nodes. This process continues until the firing of all nodes is evenly distributed throughout T, where T can be given according to data rate, number of participating nodes, wireless channel, and guard interval, among other characteristics.

Convergence time (CT) is defined as the time difference between the instant when the network topology changes (creation, addition, removal) and the instant when all nodes achieve their convergent states. Whereas the existing PCO-based algorithms consider the convergence rate or speed based on progressive characteristics, we use a deterministic metric. Further, CT is a key factor in terms of bandwidth efficiency, energy consumption, and practical deployment in [5, 7, 8, 12, 13, 14]. Thus, CT, a function of the number of firing cycles to achieve the

convergence state, is derived in a closed form with regard to the rate of convergence in [5, 6, 10, 14] and is stochastically estimated in [15]. However, although *CT* can be conjectured, it requires a significant amount of time to achieve the convergence state. Thus, it cannot be guaranteed within a specific number of firing cycles and is not yet practical for real-world applications.

In this paper, we propose a practical and deterministic desynchronization scheme, PD-DESYNC, that provides deterministic *CT* in three firing cycles, regardless of the number of nodes. In previous work in this domain, such as [8, 14], it was assumed that the number of participating nodes was known and the algorithms were presented based on this information. However, extra complexity is required for each node to count participants; if any information regarding the nodes is known in advance, the algorithms can be significantly improved. Thus, obtaining information regarding the organization of the nodes is considered another important research domain. To the best of our knowledge, no desynchronization algorithms currently discuss or present a method to accomplish this.

Hence, all the nodes of PD-DESYNC periodically broadcast a pulse signal in the same manner as PCO-based techniques. Each node recalculates and updates the firing time in the next firing cycle based on the firing messages from the other nodes in the current cycle. When the flag firing (FF) being broadcast by a randomly selected leader node is considered the beginning point of a cycle, each node can count the firing messages during *T*. The next firing time is then determined by the number of nodes and order of broadcasted firings. There is no additional equipment or cost required for counting.

We conducted theoretical analysis and extensive simulations, and further verified the performance of PD-DESYNC through practical experiments on TinyOS-based Telos sensors. To evaluate the efficiency of the algorithms, *CT* (the time required to complete desynchronization) was used as a metric. To clarify the performance evaluations, we compared PD-DESYNC with DESYNC [5, 6] and anchored DESYNC [14].

The remainder of this paper is organized as follows. In Section 2, we briefly review related work. Section 3 presents PD-DESYNC, a practical and deterministic desynchronization method. Section 4 compares the performance of the algorithms with respect to *CT* through simulation results and implementation. Section 5 provides our concluding remarks.

2. Related Works

2.1 DESYNC [5–7]

DESYNC is considered to be a general framework for distributed algorithms to achieve the desynchronization required by TDMA. Nodes are modeled by PCOs in [8–11], which were designed for cardiac and firefly synchronization. These algorithms assume that (i) all *n* nodes can communicate with each other in a single channel, (ii) each node is modeled by an oscillator with the same fundamental frequency as period *T*, and (iii) there is no oscillator clock drift. Thus, the state of a node can be represented by the phase of its oscillator. Without loss of generality, it is convenient to assume that the fundamental frequency is one and the phase is in [0,1].

When a node reaches the end of its cycle, it fires and resets its phase to zero. This firing also notifies all the other nodes that it is beginning a new cycle. Next, it waits for the next node to fire and jumps to a new phase according to a certain function. This jumping function uses only the firing information of the node fired immediately before it and the node fired immediately after it. DESYNC achieves desynchronization (i.e., the phases of the n nodes are evenly spaced) if the new phase of each jump in a node is moved towards an estimated midpoint of the phases of two neighboring nodes. However, the *CT* of the DESYNC is only conjectured to be $O(n^2)$, which is nondeterministic.

2.2 Anchored DESYNC [14]

In anchored DESYNC (A-DESYNC), except for the fact that a single special node cannot adjust its own clock because it is already synchronized, the phase adjustment procedure is the same as that of DESYNC [5, 6]. Moreover, A-DESYNC is a centralized scheduling algorithm because it is assumed that each node knows the total number of nodes. Maintaining one node with fixed beaconing (i.e., an anchored node) allows for faster convergence to TDMA [15]. However, without suggesting how to select the anchored node, it is assumed that the anchored node is determined in advance or by a certain scheme. Thus, the *CT* of A-DESYNC is conjectured to be $O(n^2 \ln(n/\varepsilon))$, which is nondeterministic.

2.3 PCO-based DESYNC [8]

PCO-based desynchronization (PCO-DESYNC) with inhibitory coupling achieves round-robin scheduling by limiting the listening interval; every node updates its own phase after it receives a firing from a previous phase neighbor within the listening interval. It exhibits a logarithmic complexity, $O(\log(n))$, which is nondeterministic. It is also a centralized algorithm in the sense that it uses the total number of nodes; however, it does not explain how this number is obtained.

2.4 Fast DESYNC [16, 17]

In [16], by formalizing a well-established desynchronization algorithm as a gradient descent method for solving an optimization problem, a new upper bound of $O(\sqrt{n/\varepsilon})$, is established on the number of iterations required to achieve convergence. Using Nesterov's Accelerated Gradient Descent, a new algorithm that converges to the steady network state more quickly can be obtained. Moreover, it has been extended into a version of decentralized multichannel coordination. However, the *CT* is estimated as a convergence rate, which is nondeterministic.

A summary of representative DESYNC-based algorithms is provided in **Table 1**.

Table 1. Summary of representative DESYNC-based algorithms

Algorithms	Centralized or Distributed	Provide Bounded *CT*	Use total number of nodes	Provide how to determine total number of nodes
DESYNC	Distributed	NO	NO	-
A-DESYNC	Centralized	NO	YES	NO
PCO-DESYNC	Centralized	NO	YES	NO
Fast DESYNC	Distributed	YES	NO	-
PD-DESYNC	Distributed	YES	YES	YES

3. PD-DESYNC: PRACTICAL AND DETERMINISTIC DESYNCHRONIZATION

In this section, we first describe the PCO framework [9–11]. Using this framework, PD-DESYNC algorithms are then explained in detail. Notations are summarized in **Table 2**.

Table 2. Notations in PCO framework

Notation	Description
$\phi_i(t)$	phase of N_i at time t, $\phi_i(t) \in [0,1]$
T	firing cycle
n	total number of nodes
N_i	i-th node in T (i=0, ..., n-1)
τ_i	count-up timer of N_i from zero to T
$C_{i,AF}$	firing counter of N_i after its own firing between two consecutive FFs
$C_{i,BF}$	firing counter of N_i before its own firing between two consecutive FFs
NN	normal node
FN	flag node
FF	flag firing
hasFired	variable indicating if N_i has fired between two consecutive FFs
CT	convergence time

3.1 PD-DESYNC Framework

As in [5–7], we consider the desynchronization problem as a fully connected graph of n nodes; i.e., all n nodes are able to communicate with each other. Each node is modeled by an oscillator with frequency $T = 1$, where T is a firing cycle and there is no oscillator clock drift. Let $\phi_i(t) \in [0, 1]$ be the phase of N_i at time t, $i = 0, ..., n$-1. Upon reaching $\phi_i(t)=1$, N_i fires, indicating the termination of its cycle to the other nodes. Upon firing, the node resets $\phi_i(t+)$ to zero. Each node moves around a ring in a clockwise direction with period T (**Fig. 1**). Whenever a node reaches the top of a ring, it fires; all other nodes can detect this firing and record their times to adjust their own phases.

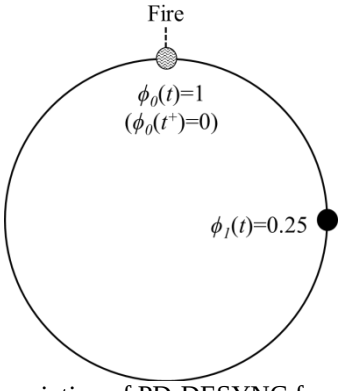

Fig. 1. Description of PD-DESYNC framework

3.2 PD-DESYNC Algorithm

The goal of the proposed PD-DESYNC algorithm is to adjust the phase of each node such that the phases of the n nodes can be evenly spaced within a short and deterministic CT. To achieve this goal, each node receives the total number of nodes (n) by counting the firings of the other nodes during T, and locates its own phase for the next cycle using the order of firings and number of nodes. The flag node (FN), which is chosen through the initial procedure, broadcasts an FF every T time unit. By counting the firings between two consecutive FFs, each node can obtain the total number of participants. Following this, N_i sets its own current phase, $\phi_i(t)$ using a function of n and two firing counters $C_{i,BF}$ and $C_{i,AF}$, where $C_{i,BF}$ and $C_{i,AF}$ are the firing counter before and after its own firing between two consecutive FFs, respectively. Every node acts as either an FN or a normal node (NN). The only difference is that when $\phi_i(t)$=1, an FN broadcasts not only a firing but also an FF. We outline the PD-DESYNC algorithm consisting of these three procedures following.

A) Initial procedure: Through the initialization procedure, a node can determine if it is an FN or NN. If a node does not detect an FF or a firing before its own firing, the node is an FN; otherwise, it is an NN. The initialization procedure of a node is described below (**Table 3** and **Fig. 2**).

Table 3. Initial procedure

Step	Action
(A.1)	N_i initializes $\tau_i = 0$, $C_{i,AF} = 0$, $C_{i,BF} = 0$, and *hasFired* = *false*.
(A.2)–(A.3)	If N_i detects an FF before τ_i expires, it acts as an NN. Otherwise, N_i is an FN of which $\phi_i(t)$ first reaches one.
(A.4)	N_i becomes an NN. It selects a random value in [0,1] and adjusts $\phi_i(t)$ to the value.

(A.5)	When τ_i expires, N_i selects a random value in [0,1] and adjusts $\phi_i(t)$ to the value.
(A.6)	If N_i detects an FF before $\phi_i(t) = 1$, it acts as an NN.
(A.7)–(A.8)	If $\phi_i(t) = 1$ without detecting any firing, it becomes an FN and broadcasts an FF and a firing.

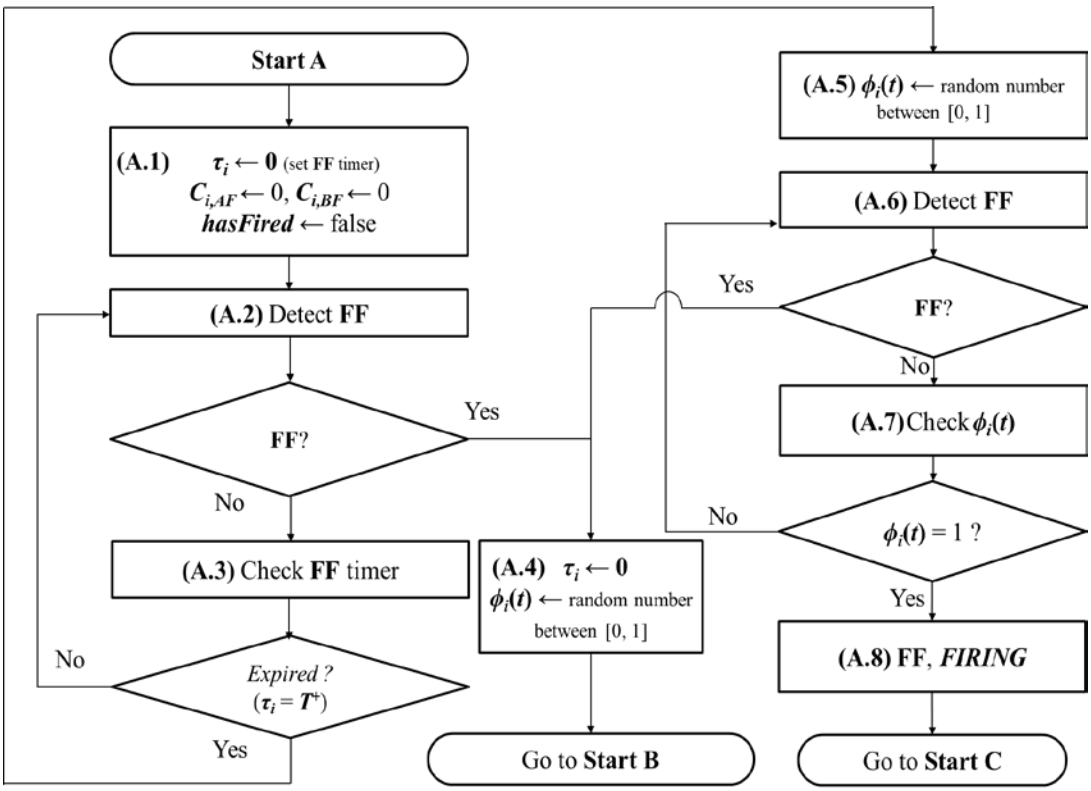

Fig. 2. Initial procedure of a node

The initial procedure is an important process of the PD-DESYNC algorithm, where each node becomes either an FN or NN. N_i initializes $\tau_i = 0$, a count-up timer that counts from zero towards T. If it detects an FF before the expiration of the FF timer (τ_i), N_i becomes an NN. If the FF timer expires, it becomes a candidate for an FN and sets $\phi_i(t)$ to a random number in [0,1]. The FF timers of the other nodes can expire simultaneously. If all of them become FNs and immediately broadcast FFs and firings, a collision must occur (refer to Section 4.1). Therefore, the N_i whose FF timer has expired selects a random number as an initial value of $\phi_i(t)$. The first node whose $\phi_i(t)$ reaches one becomes the FN and broadcasts an FF and a firing within every T period. The other nodes hearing the FF act as NNs and broadcast a firing whenever $\phi_i(t)$ reaches one.

For ease of understanding of the initial procedure, let us suppose that there are n = 4 nodes. **Fig. 3** illustrates the timeline of the initial procedure. Initially, the nodes are turned on in the sequence N_0, N_1, N_2, and N_3, and start their own FF timer. When the FF timer (τ_0) of N_0 expires first, N_0 becomes a candidate for an FN and sets $\phi_0(t)$ to 0.7, which is randomly generated in [0,1]. When $\phi_0(t)$ = 1, it becomes an FN and broadcasts an FF and a firing within every T period. The FF timer (τ_1) of N_1 also expires; it becomes a candidate for an FN and sets $\phi_1(t)$ to 0.3. However, when $\phi_1(t)$ reaches 0.5, N_1 hears an FF and a firing from N_0, becomes an NN, and resets $\phi_1(t)$ to 0.5. If $\phi_1(t)$ was set to a value close to one and reached one before $\phi_0(t)$, N_1 would become an FN instead of N_0. Both N_2 and N_3 become NNs because they detect an FF from N_0 before expiration of their FF timers (τ_2 and τ_3). Further, $\phi_1(t)$ and $\phi_1(t)$ are set to 0.8 and 0.2, respectively.

From the moment that N_0 becomes an FN, firing cycles begin. Within every T time unit, FN (N_0) broadcasts an FF and a firing and the NNs (N_1, N_2, N_3) execute a firing. NNs adjust their $\phi_i(t)$ (i = 1, 2, 3) by counting n, $C_{i,BF}$, and $C_{i,AF}$ and computing $\phi_i(t)$ in the next firing cycle.

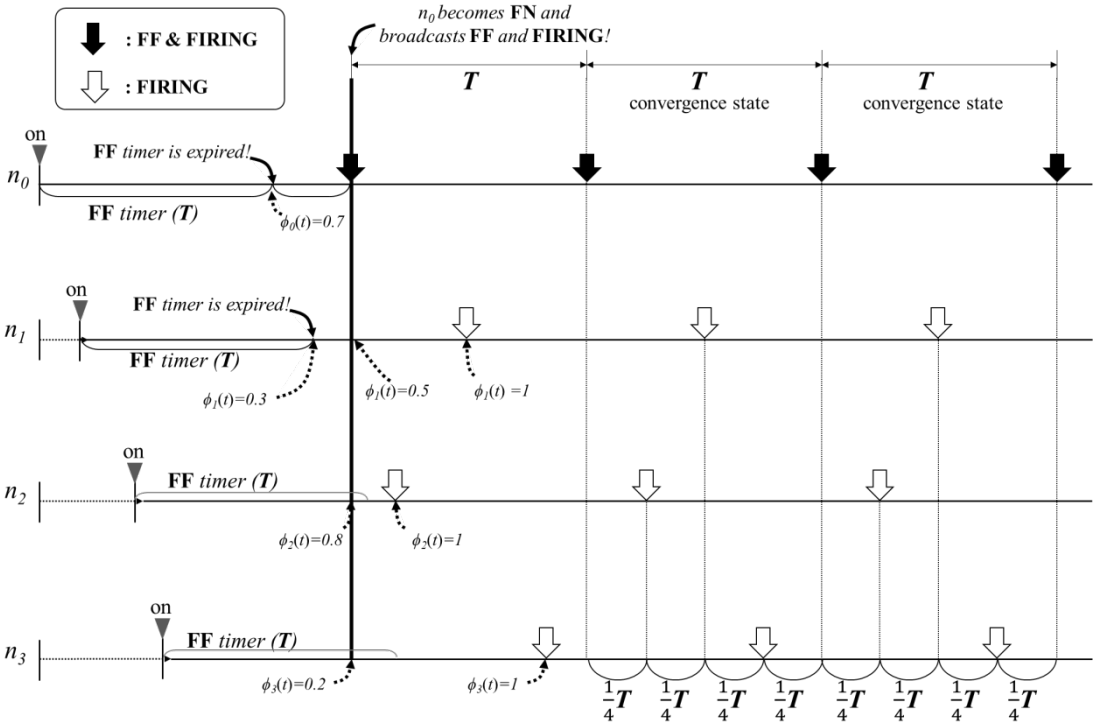

Fig. 3. Timeline of initial procedure (n = 4)

B) NN Procedure: Each NN executes three processes: The first is to detect FFs or firings, the second is to verify if $\phi_i(t)$ reaches one, and the third is to verify if τ_i has expired. The initialization procedure of a node is presented in **Table 4** and **Fig. 4**.

$$\phi_i(t) = T\left(1 - \frac{C_{i,BF}}{C_{i,AF}+C_{i,BF}+1}\right) \quad (1)$$

Table 4. *NN* Procedure

Step	Action
(B.1)–(B.3)	When N_i detects a firing, it increases $C_{i,BF}$ before its own firing and $C_{i,AF}$ after its own firing by one.
(B.4)	When N_i detects an FF, it adjusts $\phi_i(t)$ by the phase function (1), reinitializes $\tau_i = 0$, $C_{i,AF} = 0$, $C_{i,BF} = 0$, and *hasFired* = *false*.
(B.5)–(B.6)	When $\phi_i(t) = 1$, N_i broadcasts a firing and is set to *hasFired* = *true*. If a collision occurs,
(B.7)	If τ_i has expired, N_i assumes that there is no FN. To determine a new FF, N_i performs the initial procedure of Section 3.2.B) again.

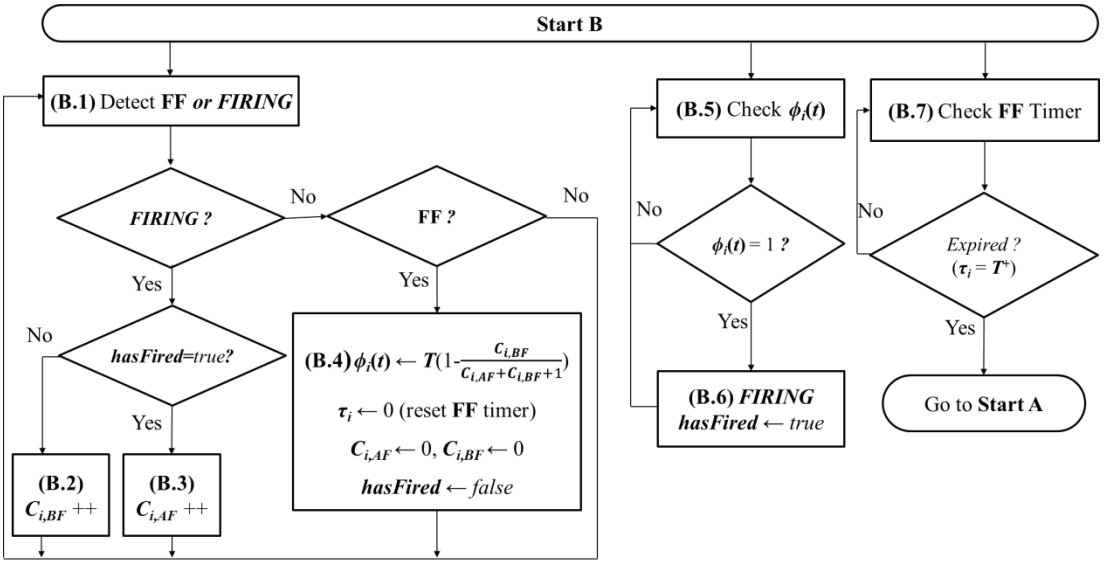

Fig. 4. Procedure of a Normal Node

C) FN Procedure: The FN procedure is simple, as indicated in the **Table 5** and **Fig. 5**.

Table 5. FN Procedure

Step	Action
(C.1)–(C.2)	When $\phi_i(t) = 1$, N_i broadcasts an FF and its own firing.
(C.3)	If N_i detects an FF before broadcasting of its own FF, it becomes an NN.

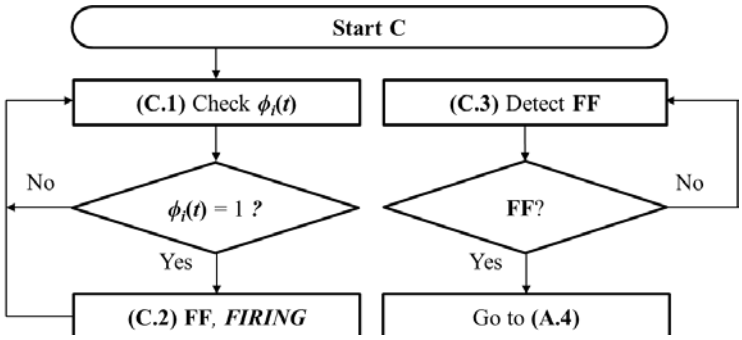

Fig. 5. Flag Node Procedure

4. Performance Evaluation

PD-DESYNC is a deterministic solution. Regardless of the number of nodes (n), desynchronization is completed within three firing cycles. It is sufficiently straightforward to demonstrate that the *CT* is bounded by only three firing cycles; every entering node performs the initial procedure. It requires $2T$ at the maximum degree, which is a worst-case scenario where the FF timer set to T has expired and all competitive nodes select values close to T in a random backoff process for FN. Next, after a second T, each node knows the total number of nodes and precisely adjusts its own phase based on that information; it requires up to $3T$ to achieve desynchronization.

To clarify the performance evaluation, we compared PD-DESYNC with DESYNC [5–7] and A-DESYNC [15]. To confirm that PD-DESYNC converges to desynchronization within three firing cycles, we conducted extensive simulations and practical implementations, where *CT* was used a metric.

- Convergence Time (*CT*): *CT* is defined as the time difference between the moment when the network topology changes (creation, addition, removal) and the time when all nodes achieve the convergent state. Whereas the existing PCO-based algorithms consider the convergence rate or speed based on progressive characteristics, we use a deterministic metric.

4.1 Addressing Collisions

In the PD-DESYNC algorithm, the method of selecting an FN and determining the time to broadcast an FF or firing is based on a timer and randomness. A random number is set in $\phi_i(t)$ for the FN and NN to broadcast an FF or firing. A collision could occur, and therefore, countermeasures are required to prevent this situation.

A major collision problem occurs when multiple nodes become FNs and broadcast an FF simultaneously. Let us call this ‘FF collision’. This could occur when all nodes are powered up simultaneously or the FN disappears from the network in the desynchronization state. If the FN disappears in the desynchronization state, the FF timers of the nodes expire simultaneously. At this stage, if multiple nodes become FNs and broadcast an FF simultaneously, an FF collision occurs, affecting the operation of PD-DESYNC. To minimize the FF collision problem, when the FF timer expires, $\phi_i(t)$ is set to a random number in [0,1]; N_i with the greatest value becomes the FN. This process is similar to CDMA/CA used in IEEE 802.11 DCF: when a node waits to become free, after the DIFS(DCF Interframe Space) interval, it

chooses a random backoff counter in the contention window and decrements until the backoff counter is zero before attempting to transmit.

Although the collision problem caused by the timer has been mitigated, the probability of FF collision by randomness continues to exist. If a node that is unrelated to an FF collision exists, this node cannot hear the FF and determines that there is no FN. During the FF collision, the node becomes the FN and broadcasts an FF. The nodes that are directly involved in the FF collision hear the FF and become NNs. Thus, there exists only one FN in the network. If there is no node unrelated to the FF collision, a collision occurs during data transfer and fails to deliver the data. At this point, the nodes involved in the collision can recognize the situation and avoid collision by repeating the procedure from the initial step.

After the FN is determined, two or more of the remaining nodes set $\phi_i(t)$ to the same random number causing another collision problem. Because their $\phi_i(t)$s reach one and they broadcast FFs at the same time, a collision can occur. This is called 'firing collision' because firing collision occurs repeatedly at each T, and eventually fails to transmit the data. Subsequently, a collision can be avoided by repeating the initial procedure. This is similar to the collision resulting from selecting the same random backoff counters in the contention period of IEEE 802.11 DCF. In the case of firing collision, the FF can recognize a firing collision before transmitting the data, and if it broadcasts a message to reset $\phi_i(t)$ to the NNs, the collision can be prevented before the data is transmitted. However, this method has the disadvantage of adding a new message and process to the nodes.

4.2 Determination of T

T is determined by the network application requirement and network environment factors. Here, the assumed network application requirement is that all participating nodes of the network must have at least one transmission opportunity within T. First, the number of network participating nodes is estimated. This number can be calculated stochastically or it can be given deterministically. If it is calculated stochastically, one or more transmission opportunities can be stochastically guaranteed. For example, if n is a normal distribution with $n \sim N(\mu, \sigma^2)$, a transmission opportunity can be guaranteed for a maximum $\lfloor 1.29\sigma + \mu \rfloor$ nodes with a probability of 90%. This is because $P(n < 1.29\sigma + \mu) = P(Z < 1.29) = 0.9015$, where $Z = (n - \mu) / \sigma$. If the number is given deterministically, one or more transmission opportunities are always guaranteed. For example, in a military operation, assume a squad, which is a sub-subunit led by a non-commissioned officer, always consists of 7–12 soldiers. Here, if the number of the participant nodes is set to 12, the participating nodes can be guaranteed of at least one transmission opportunity.

After the number of participating members is set, T can be calculated by the network environment factors such as data rate, wireless channel, and guard interval. IEEE 802.11n [18] is assumed here. Each node has a guard interval (L_{GI}) for transmission reliability and a firing interval (L_F) before data transmission. FN has L_{GI}, L_F and the FF interval (L_{FF}). The transmission time of one PPDU (PHY Protocol Data Unit) is $L_{PPDU\_SIFS\_ACK}$. **Fig. 6** displays a scenario of n nodes during T; **Fig. 7** displays the components of $L_{PPDU\_SIFS\_ACK}$. T must satisfy (2) because T must be greater than the time for at least n nodes to transmit data once.

$$T > L_{FF} + n\,(L_{GI} + L_F + L_{PPDU\_SIFS\_ACK}) \tag{2}$$

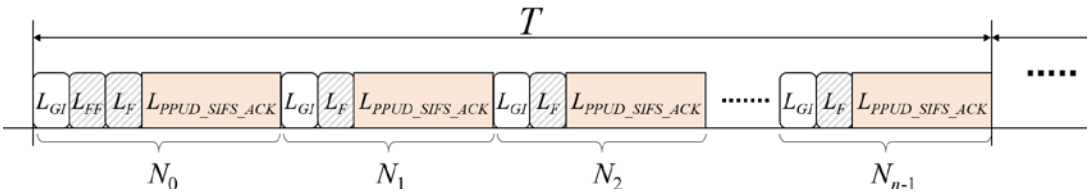

Fig. 6. Transmission scenario of n nodes in T

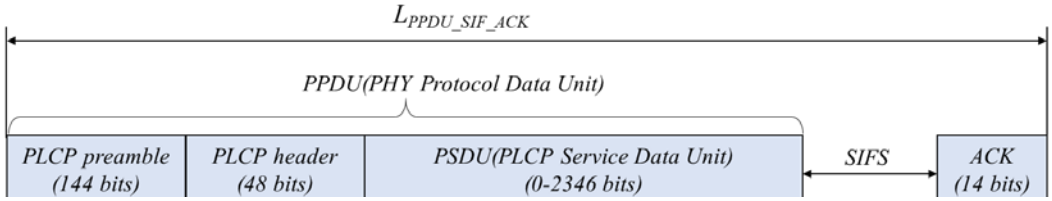

Fig. 7. PPDU

Table 6. Parameters of IEEE 802.11n

Parameter	Description	Value
data rate	data rate when MCS (Modulation Coding Scheme) Index = 5	52,000,000 *bps*
L_{GI}	guard interval (in seconds)	0.8 *us*
$L_{PPDU\_SIFS\_ACK}$	interval of long PPDU (PHY Protocol Data Unit) + SIFS + ACK	261.17 *us*
L_F	firing interval (in seconds) = type (4 bits) + node ID (48 bits)	52 *us* (52 bits)
L_{FF}	flag firing interval (in seconds) = type (4 bits) + node ID (48sbits)	52 *us* (52 bits)

Table 6 displays the IEEE 802.11n parameter values for calculating T in a real environment. The *data rate*, L_{GI}, and $L_{PPDU\_SIFS\_ACK}$ values are referenced from [18, 19]; L_F and L_{FF} are referenced from [20]. The PLCP (Physical Layer Convergence Procedure) preamble and PLCP header are transmitted at 1 Mbps for reliable transmission; other fields are transmitted at 52 Mbps, and SIFS is 20 *us*. If n is set to 12, then $T > 3819.64$ *us* = 52 *us* + 12(0.8 *us*+52 *us* + 261.17 *us*). After the determination of T, the efficiency of the slot utilization can be significantly improved using the firing offset adjustment technique in [20].

4.3 Analytical Results

To demonstrate that desynchronization was performed within 3T, we analyzed the timeline of the proposed desynchronization scheme under the worst-case circumstances. If network topology change occurred (such as node addition or node removal for an existing FN), all nodes exchanged firings with each other during a firing cycle after receiving an FF from the FN. Following this, desynchronization was achieved at the next firing cycle.

The process of adding n_4 in the second (2nd) firing cycle is detailed in **Fig. 8**, where four nodes (N_0,…,N_3) were in the convergence state in the first (1st) firing cycle. Next, N_4 was added into the network in the second (2nd) firing cycle, and detected an FF before the FF timer (T) expired. In the third (3rd) firing cycle, five nodes exchanged firings. When all five nodes received FFs at the start of the fourth (4th) firing cycle, they recalculated their own firing times by (1) and reached the convergence state. Thus, desynchronization was achieved within 2T (<3T). Whereas it requires 2T to complete desynchronization from the addition of a node, there is only one T (the third (3rd) firing cycle) where TDMA fails.

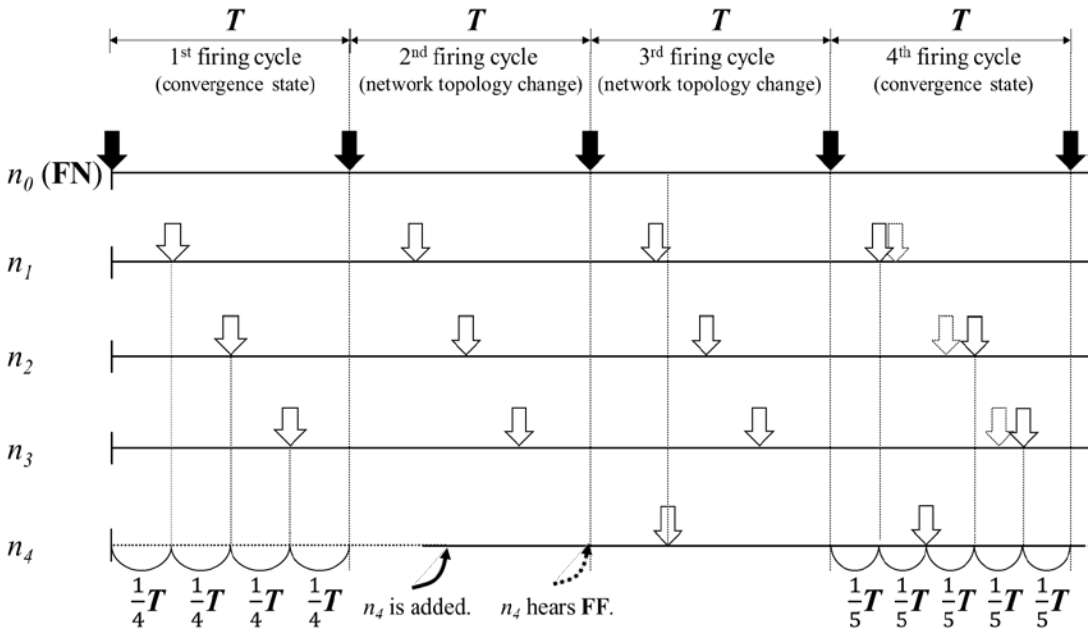

Fig. 8. Timeline in node addition

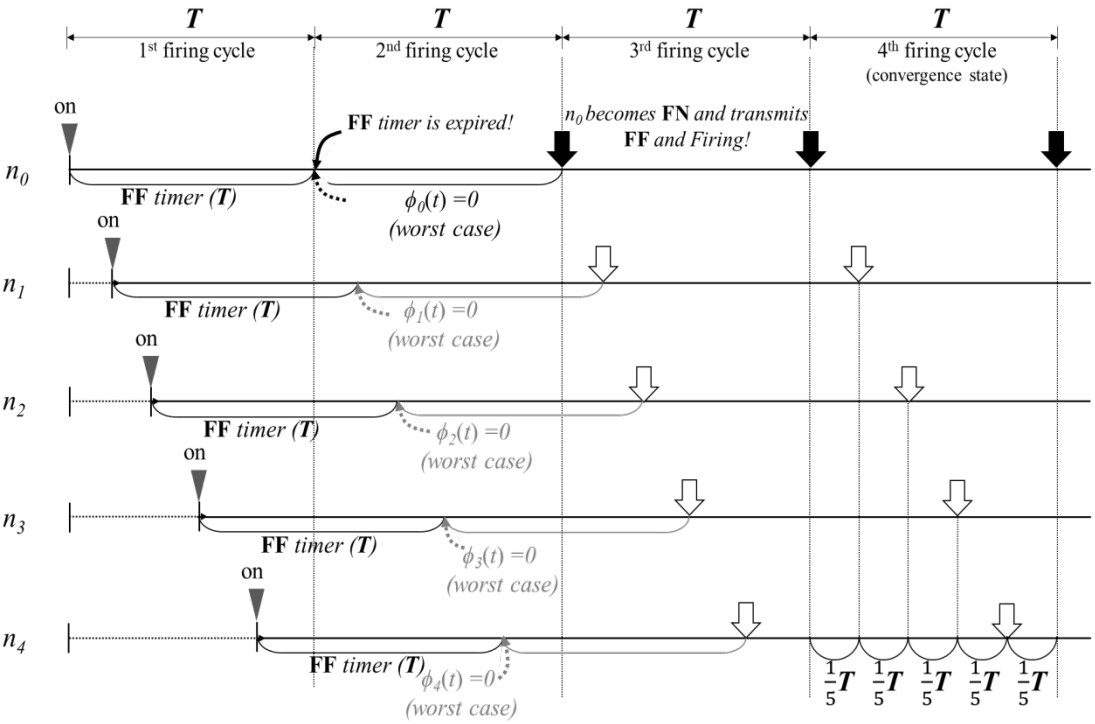

Fig. 9. Worst-case timeline in node addition

The worst-case scenario occurs when there is no FN in the network because all nodes power on or the FN disappears; the process of selecting an FN consumes additional time. This procedure is described from the moment when the five nodes power up to the moment when they converge to desynchronization (**Fig. 9**). For ease of presentation, the indices of nodes are numbered in the ascending order of their turn-on times. In the first (1^{st}) firing cycle, all nodes powered on at different times and then set FF timers. At the beginning of the second (2^{nd}) firing cycle, the FF timer of N_0 expired. To create a worst-case scenario, $\phi_i(t)$ (i = 0,…,3) was set to zero. When $\phi_i(t)$ = 1, N_0 became an FN and broadcasted an FF and a firing. The N_i (i = 1, …, 3) that received the FF from the FN, then became an NN and transmitted a firing when $\phi_i(t)$ = 1. When N_0 transmitted an FF and a firing at the beginning of the fourth (4^{th}) firing cycle, all nodes recalculated their own firing times by (1), and then converged to desynchronization. Thus, desynchronization can be achieved within 3T, even in a worst-case scenario.

4.4 Simulation Results

All current proposed methods in this domain are biologically inspired self-maintaining schemes for collision-free TDMA scheduling in a single-hop, single-channel network. In related work, convergence rates are used as a metric because the time required to converge to desynchronization is unbounded. Thus, we evaluated convergence time, which is a deterministic metric. We used the averages for the *CT*s of DESYNC and A-DESYNC and of the worst-case scenario for PD-DESYNC. The averaged *CT*s of DESYNC and A-DESYNC and the worst-case scenario *CT* of PD-DESYNC are displayed in **Fig. 10**.

We developed our event-driven simulator to perform extensive simulation tests. To ensure that all our results were comparable, we used the same test scenario for all simulations. All results were averaged after 3,000 iterations, and T was set to 1,000 ms. Finally, n varied from five to 50. Without loss of generality, we measured the *CT* as the total number of firing cycles from the cycle where the topology changed to the cycle immediately before the first desynchronization was achieved. We considered three cases: (i) initial network configuration, (ii) node addition, and (iii) node removal. In case (i), several nodes configured a network; in case (ii), one node (NN or FN) joined the network; in case (iii), one node (NN or FN) left the network.

1) **Initial network creation:** The existing algorithms consider the following situation to be a general case. As a group of nodes boots up and periodically broadcasts firings, the nodes are desynchronized, either gradually or deterministically, by their own algorithms. The *CT*s of the three schemes at the initial creation of a network are depicted in **Fig. 10**. Whereas the *CT*s of DESYNC and A-DESYNC increased rapidly as n increased, PD-DESYNC converged to a steady state within three firing cycles (regardless of n).

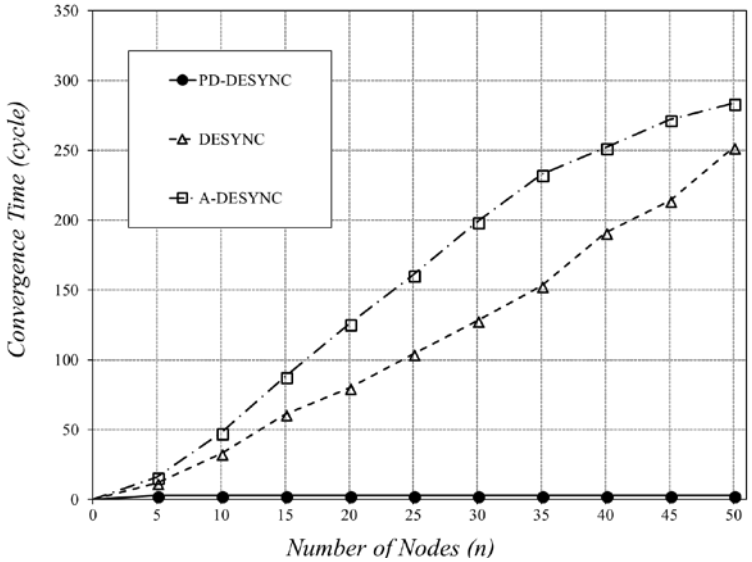

Fig. 10. *CT*s for DESYNC, A-DESYNC, and PD-DESYNC at initial creation of network, $n \in [5,50]$

2) Addition of a node: In this scenario, the network was assumed to be initially desynchronized. A new node joined the network, which was then re-desynchronized by its own algorithm. As indicated in **Fig. 11**, whereas the *CT*s of DESYNC and A-DESYNC increased rapidly as n increased, PD-DESYNC converged to a steady state within two firing cycles regardless of n. Because a random backoff process for FN selection was omitted, the *CT* decreased by one.

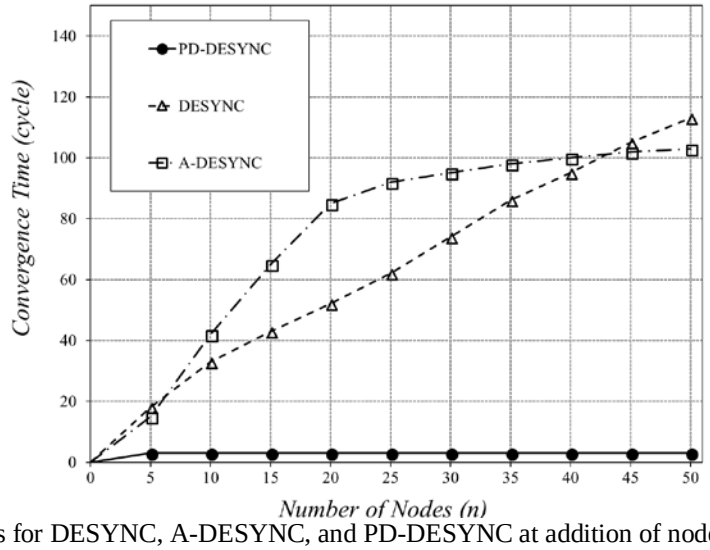

Fig. 11. *CT*s for DESYNC, A-DESYNC, and PD-DESYNC at addition of node, $n \in [5,50]$

3) Removal of a node: In this scenario, the network was also assumed to be initially desynchronized. One node left the network, which was then re-desynchronized by its own algorithm. In PD-DESYNC, two cases are possible: (i) removal of an NN or (ii) removal of an FN. Hence, **Fig. 12** displays the *CT*s of DESYNC, A-DESYNC, PD-DESYNC

(NN), and PD-DESYNC (FN). Whereas the *CT*s of DESYNC and A-DESYNC increased rapidly as n increased, PD-DESYNC (NN) and PD-DESYNC (FN) converged to a steady state within two and three firing cycles (regardless of n), respectively. When the FN leaves, three steps of the expiration of an FN timer, random backoff process, and counting of n follow. However, immediately after an NN leaves during the firing cycle, every node counts and obtains $n' = n - 1$. Furthermore, at the next firing cycle, all nodes can adjust their own phases according to the phase function (1).

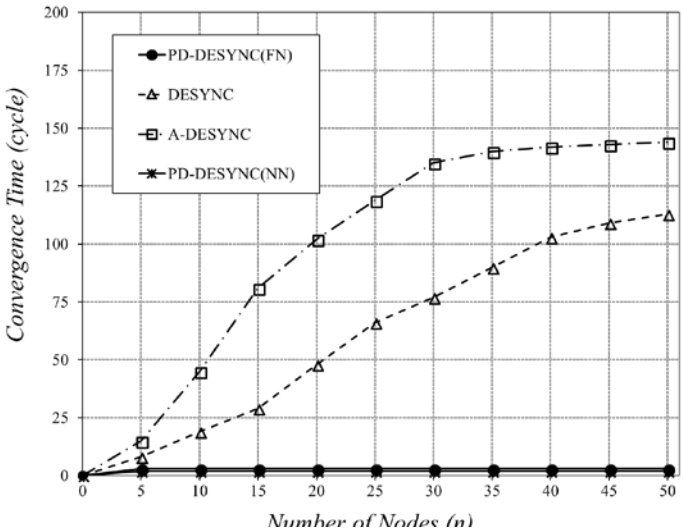

Fig. 12. *CT*s for DESYNC, A-DESYNC, PD-DESYNC(NN) and PD-DESYNC(FN) at the addition of a node, $n \in [5,50]$

4.5 Experimental Results with Kmotes

To validate the practicality of PD-DESYNC, we implemented PD-DESYNC programmed with NesC on TinyOS-based Kmote sensor motes, which is a clone of TelosB developed by the Korea Electronics Technology Institute. We measured the *CT*s of PD-DESYNC on the testbed with five motes for all experiments with a fixed parameter: T = 6000 *ms*. To evaluate the effect of the addition and removal of a node, we conducted three classes of experiments: creation of the network, addition of a node, and removal of a node. Our demonstration clip can be played on YouTube (refer to [21]).

Fig. 13 displays a typical run with five nodes. Each point represents the relative firing time when the firing time of FN was assumed to be zero within a single firing cycle.

In the first (1st) round, every node reset its own FF timer and waited for the FF without broadcasting a firing. In the second (2nd) round, N_0 won the backoff competition, became an FN, broadcasted an FF, and fired. Other nodes that received the FF broadcasted a firing when $\phi_i(t) = 1$ while detecting and counting other firings. In the third (3rd) round, as desynchronization was achieved, all nodes broadcasted their firings with equally spacing around the ring.

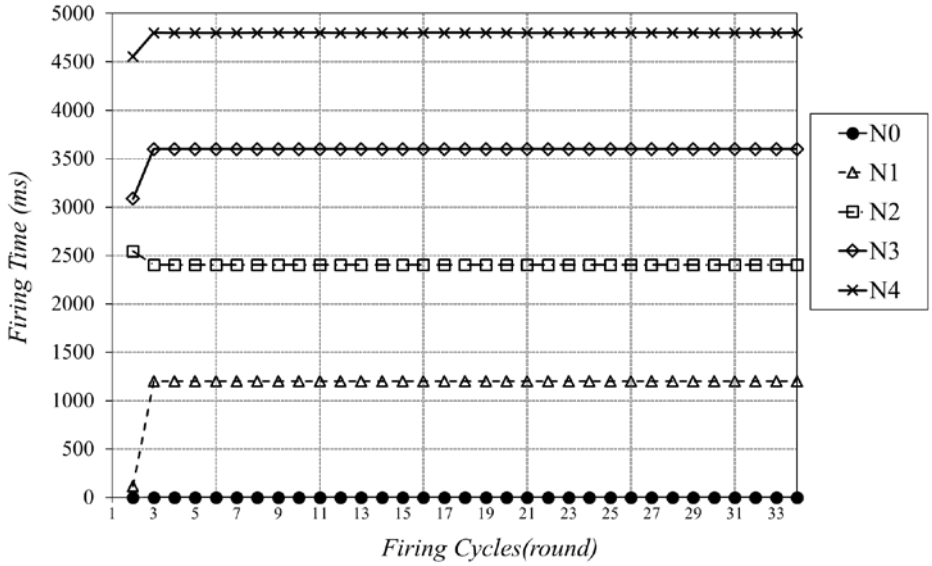

Fig. 13. Experiment with 5 Kmotes

Fig. 14 displays *CT*s at the initial network creation, the addition of a node, and removal of a node. When measuring *CT* at the initial network creation, we used the node that is first powered on because it was a probable worst case. In the initial network creation, the *CT* was measured from the starting cycle of the first booted node to the prior cycle where the value of $\phi_i(t)$ was calculated or recalculated. In both the addition and removal of a node, the *CT* was measured from the last cycle when the number of firings was unchanged to the prior cycle where the value of $\phi_i(t)$ was recalculated. Each result was averaged after 30 iterations. The *CT* at network creation was approximately two (<3); *CT*s at the addition and removal of a node were approximately one (<3). These results demonstrate that the proposed PD-DESYNC is a practical, deterministic, and effective desynchronization scheme.

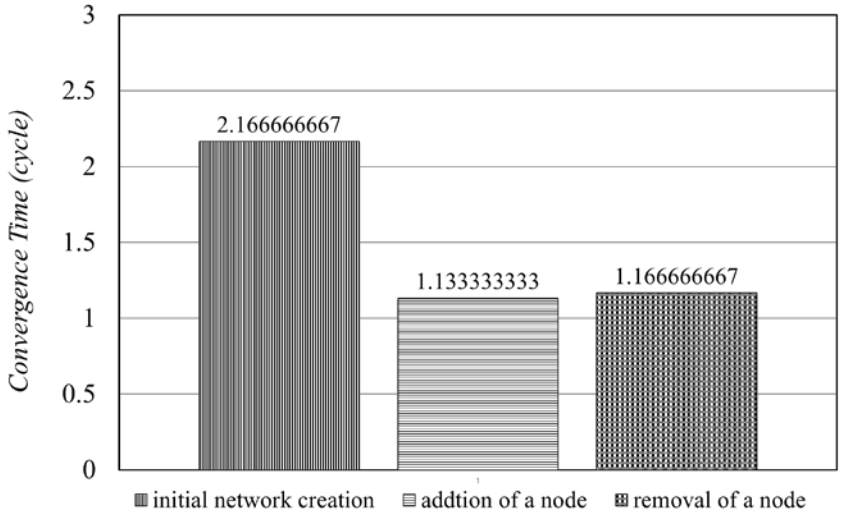

Fig. 14. Measurements of *CT*s with 5 Kmotes

5. Conclusion

Existing DESYNC-based algorithms cannot guarantee the completion of desynchronization in a short time. Thus, they are not practical for real, time-divided communication systems. To the best of our knowledge, we are the first to propose a deterministic desynchronization algorithm that converges to desynchronization within three firing cycles. The proposed PD-DESYNC consists of three procedures: initialization, NN, and FN procedures. Whereas other algorithms [8, 14] only assume that the total number of nodes is known, we have proposed a simple algorithm that obtains the total number of nodes based on the FFs broadcast by the FN, which was selected by chance in a random backoff competition. Finally, extensive, event-driven computer simulations demonstrated that the proposed PD-DESYNC outperforms DEYSNC and A-DEYSNC. Through our implementation and experiments, we can verify the efficacy of PD-DESYNC. In the near future, we plan to study two extensions to PD-DESYNC.

(1) Energy-efficient algorithms: Operations such as the counting of firings, broadcasting of FF, and the electing of FN into PD-DESYNC have been added to PD-DESYNC. These operations reduce the network lifetime as they consume a considerable amount of energy. In particular, the operation for listening constantly to hear firings is energy-intensive. One method to improve this issue and increase the network lifetime is to reduce the number of firings. The existing DESYNC must continue to fire because it does not know when the convergence state is achieved. However, PD-DESYNC can guarantee attaining the desynchronization state within $3T$. Therefore, if a stability state where the number of total nodes is constantly maintained is achieved, a sleep mode without firing for a certain period of time can be applied to PD-DESYNC. To implement this, the conditions for setting, maintaining, and releasing the sleep mode must be studied based on criteria for determining the stability state.

(2) Multi-hop case: Research on extending DESYNC to a multi-hop topology is a considerably challenging issue. In [7] and [22], the network topology always achieves the same desynchronization state in specific situations including the Hamiltonian cycle. However, in other network topologies, if the firing start time of the nodes is different, it is difficult to predict the time allocated for each node because each node attains a different desynchronization state; thus, there are situations where we cannot obtain sufficient capacity. Further, although the desynchronization state is achieved, a collision can occur owing to the hidden node problem. Therefore, research using information sharing on two-hop neighbor nodes has been proposed [23]. We have been investigating how to solve the hidden node problem by sharing information between two-hop neighbor nodes in a two-hop case or using FN as a cluster header without sharing information, and plan to extend this to a multi-hop case.

Acknowledgements

This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2016S1A5B8913575). This research was supported by “Research Base Construction Fund Support Program” funded by Chonbuk National University in 2017.

References

[1] F. Turati, M. Cesana, L. Campelli, "SPARE MAC Enhanced: A Dynamic TDMA Protocol for Wireless Sensor Networks," in *Proc. of IEEE GLOBECOM 2009, Hilton Hawaiian Village Honolulu, HI, USA*, 30 Nov – 04 Dec 2009. Article (CrossRef Link).

[2] S. Cho, K. Kanuri, J.-W. Cho, J.-Y. Lee, S.-D. June, "Dynamic Energy Efficient TDMA-based MAC Protocol for Wireless Sensor Networks," in *Proc. of ICAS-ISNS 2005*, 2005. Article (CrossRef Link).

[3] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: a Hybrid MAC for Wireless Sensor Networks," in *Procs. of SenSys 2005*, 90-101, 2005. Article (CrossRef Link).

[4] A. Barroso, U. Roedig, C. Sreenan, "/spl mu/- MAC: An energy efficient medium access control for wireless sensor networks," in *Procs. of The 2nd EWSN 2005*, pp. 70-80, 2005. Article (CrossRef Link).

[5] J. Degesys, I. Rose, A. Patel, R. Nagpal, "Desync: Self-organizing desynchronization and TDMA on wireless sensor networks," in *Proc. of IPSN 2007*, pp. 11-20, 2007. Article (CrossRef Link).

[6] A. Patel, J. Degesys, R. Nagpal, "Desynchronization: the theory of self-organizing algorithms for round-robin scheduling," in *Proc. of SASO 07*, pp. 87–96, 2007.

[7] J. Degesys and R. Nagpal, "Towards Desynchronization of Multi-hop Topologies," in *Proc. of SASO 08*, pp. 129–138, 2008. Article (CrossRef Link).

[8] R. Pagliari, Y.-W. Hong, A. Scaglione, "Bio-inspired algorithms for decentralized round-robin and proportional fair scheduling," *IEEE Journals on Selected Areas in Communications*, vol. 28, no. 4, pp. 564–575, 2010. Article (CrossRef Link).

[9] R. E. Mirollo and S. H. Strogatz, "Synchronization of pulse-coupled biological oscillators," *SIAM Journal on Applied Mathematics*, vol. 50, no. 6, pp. 1645–1662, 1990. Article (CrossRef Link).

[10] R. Leidenfrost and W. Elmenreich, "Firefly clock synchronization in an 802.15.4 wireless network," *Eurasip J. Embed. Syst.*, vol. 2009, Article id. 18585, pp. 17, 2009.

[11] R. Pagliari, Y.-W. Hong, and A. Scaglione, "Bio-inspired algorithms for decentralized round-robin and proportional fair scheduling," *IEEE Journal on Selected Areas in Communications: Special Issue on Bio-Inspired Networking*, vol. 28, no. 4, pp. 564–575, 2010. Article (CrossRef Link).

[12] D. Buranapanichkit and Y. Andreopoulos, "Distributed time frequency division multiple access protocol for wireless sensor networks," *IEEE Wireless Communications Letters*, vol. 1, no. 5, pp. 440–443, 2012. Article (CrossRef Link).

[13] H. Besbes, G. Smart, D. Buranapanichkit, C. Kloukinas, Y. Andreopoulos, "Analytic conditions for energy neutrality in uniformly formed wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 10, pp. 4916–4931, 2013. Article (CrossRef Link).

[14] C.-M. Lien, S.-H. Chang, C.-S. Chang, D.-S. Lee, "Anchored Desynchronization," in *Proc. of IEEE INFOCOM 2012*, pp. 2966–2970, 2012. Article (CrossRef Link).

[15] D. Buranapanichkit, N. Deligiannis, Y. Andreopoulos, "Convergence of Desynchronization Primitives in Wireless Sensor Networks: A Stochastic Modeling Approach," *IEEE Transactions on Signal Processing*, vol. 63, no. 1, pp. 221–233, 2015. Article (CrossRef Link).

[16] N. Deligiannis, F.C.J. Mota, G. Smart, Y. Andreopoulos, "Fast Desynchronization for Decentralized Multichannel Medium Access Control," *IEEE Transactions on Communications*, vol. 63, no. 9, pp. 3336–3349, 2015. Article (CrossRef Link).

[17] N. Deligiannis, F.C.J. Mota, G. Smart, "Decentralized multichannel medium access control: viewing desynchronization as a convex optimization method," in *Proc. of ACM IPSN* 2015, 13-24, 2015. Article (CrossRef Link).

[18] IEEE 802.11 Working Group, "IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," in *Proc. of IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp.1-2793, 2016. Article (CrossRef Link).

[19] D. Skordoulis, Q. Ni, H.-H. Chen, A. P. Stephens, C. Liu, A. Jamalipour, "IEEE 802.11n MAC Frame Aggregation Mechanisms for Next-Generation High-Throughput WLANs," *IEEE Wireless Communications*, vol. 15, no. 1, pp. 40–47, Feb. 2008. Article (CrossRef Link).
[20] K. Kim, S.-H. Shin, B.-H. Roh, "Firing Offset Adjustment of Bio-Inspired DESYNC-TDMA to Improve Slot Utilization Performances in Wireless Sensor Networks," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 3, pp. 1492–1509, Mar. 2017. Article (CrossRef Link).
[21] YouTube, "PD-DESYNC implementation and demonstration," Available Online: https://www.youtube.com/watch?v=b8VTOTXmxmE.
[22] A. Motskin, T. Roughgarden, P. Skraba, L. Guibas, "Lightweight coloring and desynchronization for networks," in *Proc. of IEEE INFOCOM 2009*, pp.2382-2391, 2009. Article (CrossRef Link).
[23] Y. Kim, J. Lee, E. Kong, U. You, C. Lee, H. Choi, M. Han, B. Roh, C. Park, M. Hoh, H. Choi, "Bio-inspired Resource Allocation for Multi-hop Networks," *The Journal of Korean Institute of Communications and Information Sciences*, vol. 40, no. 10, pp. 2035-2046, 2015. Article (CrossRef Link)

Sanghyun Hyun received his B.S. and M.S. degrees in Information and Communications Engineering from Daejeon University, Daejeon, Korea in 2013 and 2015, respectively. He is currently an associate consultant of Global Business Divisio at SOFTITECH, Daejeon, Korea. His research interests are system integration & consulting, ad hoc networks, wireless sensor networks, and IoT.

Geon Kim received B.S. degree from Chonbuk National University, Korea in 1991, M.S. degrees from University de Paris XII and VIII, France in 1999, and Ph. D. degree from University of Paris I Pantheon-Sorbonne, France in 2002. He is currently an Associate Professor at Graduate School of Archives and Records Management, Chonbuk National University. His current research interests include digital archives management, digital audiovisual archives, and bigdata.

Dongmin Yang received B.S., M.S., and Ph.D. degrees in Computer Science and Engineering from the POSTECH, Korea in 2000, 2003 and 2011, respectively. From Sep. 2009 to Sep. 2011, he was with Samsung Electronics Company, Korea, as a Senior Engineer. From Sep. 2011 to Sep. 2017, he was an Assistant Professor at the Department of Electronics, Information & Communications Engineering, Daejeon University. He is currently an Assistant Professor at Graduate School of Archives and Records Management, Chonbuk National University. His current research interests include archives & records information security, and MANET.