Vol.3, No.9 pp.293~300 pISSN: 2287-5891 http://dx.doi.org/10.3745/KTCCS.2014.3.9.293

# An Enhanced DESYNC Scheme for Simple TDMA Systems in Single-Hop Wireless Ad-Hoc Networks

Sanghyun Hyun† · Jeyul Lee† · Dongmin Yang††

## ABSTRACT

TDMA(Time Division Multiple Access) is a channel access scheme for shared medium networks. The shared frequency is divided into multiple time slots, some of which are assigned to a user for communication. Techniques for TDMA can be categorized into two classes: synchronous and asynchronous. Synchronization is not suitable for small scale networks because it is complicated and requires additional equipments. In contrast, in DESYNC, a biologically-inspired algorithm, the synchronization can be easily achieved without a global clock or other infrastructure overhead. However, DESYNC spends a great deal of time to complete synchronization and does not guarantee the maximum time to synch completion. In this paper, we propose a lightweight synchronization scheme, C-DESYNC, which counts the number of participating nodes with *GP*(Global Packet) signal including the information about the starting time of a period. The proposed algorithm is mush simpler than the existing synchronization TDMA techniques in terms of cost-effective method and guarantees the maximum time to synch completion. Our simulation results show that C-DESYNC guarantees the completion of the synchronization process within only 3 periods regardless of the number of nodes.

**Keywords : DESYNC, TDMA, Ad-Hoc Network, Sensor Network**

# 단일홉 무선 애드혹 네트워크에서 단순 TDMA 시스템을 위한 DESYNC 알고리즘 개선 방안

현 상 현† · 이 제 율† · 양 동 민††

## 요 약

TDMA(Time Division Multiple Access)는 무선 네트워크에서 한정된 주파수 대역을 일정한 크기의 시간 단위인 슬롯으로 분할하고 사용자가 할당된 슬롯을 이용하여 통신할 수 있는 채널 접속 기술이다. TDMA에 사용되는 기술에 따라 동기와 비동기 방식으로 나눌 수 있다. TDMA의 동기화 과정은 복잡하고, 추가 장비가 필요할 수 있기 때문에 소규모 네트워크에 적합하지 않다. 반면, 비동기 방식의 DESYNC에서는 전역 클록(global clock)이나 기반 시설 도움 없이 동기화를 이룰 수 있다. 하지만 DESYNC는 동기화 완료하는 데 제법 시간이 걸리고, 소요되는 최대 지연 시간이 얼마인지 보장하지 못한다. 그래서 본 논문에서는 소규모 네트워크에 적합한 경량 동기화 기법인 C-DESYNC를 제안한다. C-DESYNC는 참가 하는 노드의 주기 시작 정보를 가지고 있는 *GP*(Global Packet) 신호와 노드들의 firing 개수를 이용하여 노드의 개수를 파악하고, 이 정보를 이용하여 동기화를 이룬다. 제시하는 알고리즘은 기존의 동기화 방식의 TDMA 기법에 비해 간단하여 비용 측면에서도 효율적이며, 동기화 완료시까지 걸리는 최대 지연시간을 보장한다. 시뮬레이션 결과를 통해서 C-DESYNC는 참가 노드 개수에 관계없이 오직 3 주기 내에 동기화 완료를 보장하는 것을 보여준다.

**키워드 : DESYNC, TDMA, 애드 혹 네트워크, 센서 네트워크**

## 1. 서 론

한 무선 이동 네트워크와 같이 한정된 채널 자원을 공유하여 다수의 노드들이 통신하는 무선 네트워크에서 효율적으로 통신하기 위해서는 다중접속(multiple access)기술이 필요하다. 다중 접속 기술은 방식에 따라 TDMA(Time Division Multiple Access), FDMA(Frequency Division Multiple Access), CDMA(Code Division Multiple Access),

※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(NRF-2012R1A1A1042703).
※ 본 논문은 미래창조과학부의 2013년 고용계약형 SW석사과정 지원사업을 지원받아 수행한 결과임(ITAHB3011301101700010001001000).
† 준 회 원 : 대전대학교 정보통신공학과 석사과정
†† 정 회 원 : 대전대학교 정보통신공학과 교수
Manuscript Received : July 8, 2014
First Revision : September 4, 2014
Accepted : September 4, 2014
* Corresponding Author : Dongmin Yang(dmyang@dju.kr)

OFDMA(Othogonal Frequency Division Multiple Access) 등이 있다. 본 논문에서는 다중접속 방식 중 TDMA 기법에 대해 다루고자 한다.

TDMA란 가용 대역폭을 시간구간인 슬롯으로 나누어 자신에게 할당된 슬롯에서 모든 대역폭을 이용하여 통신하는 다중 접속 기술이다. TDMA에서 사용자는 다른 사용자들과 시간을 달리하여 채널을 공유하기 때문에, 사용자에게 주어진 슬롯 동안 전체 주파수 대역을 사용할 수 있다. TDMA 기법은 동기화 방식과 비동기화 방식으로 나눌 수 있는데, 동기화 방식은 동기를 위한 방법이 복잡하고, 동기화를 위한 추가 장비가 필요하기 때문에 소규모 네트워크에 이용하기에 비효율적이다[1].

본 논문에서는 간단한 규칙을 이용하여 소규모 네트워크에 적합한 동기화 방식의 TDMA 기법을 제안하기 위해 DESYNC를 이용한다. DESYNC는 자연 현상에서 영감을 얻어 동기를 유지시키는 기법으로, 간단한 규칙을 이용해 주기적으로 공정한 이벤트를 발생시킨다. 주변 노드들은 이 이벤트에서 얻은 정보를 이용하여 동기를 유지하기 때문에 기존의 TDMA 기법과 달리 동기화를 위한 전역 클록(global clock)이나 기반시설이 필요하지 않고, 참여하는 노드의 수에 따라 자동으로 동기화를 유지할 수 있다[2, 3]. 또한 동기화를 위한 신호의 크기가 매우 작아 에너지 효율 측면에서도 매우 우수한 기법이다[4].

하지만 DESYNC는 모든 사용자가 균등하게 채널을 공유할 수 있는 동기화 완료가 되기까지 지연시간을 보장하지 않는 단점이 있다. 이 문제를 해결하기 위해 제안하는 알고리즘은 DESYNC의 firing 기법을 이용하여 전체 노드의 개수를 파악하고, 이 정보를 이용하여 노드의 개수에 상관없이 네트워크가 동기화 완료에 이르는 최대 지연 시간을 보장한다. 그리고 주기의 시작 정보를 가지고 있는 *GP*(Global Packet)을 이용하여 기존의 동기화 방식의 TDMA 기법에 비해 간단하여 비용 측면에서도 효율적이다.

## 2. 관련 연구 - DESYNC

DESYNC는 1-홉 형태의 무선 네트워크에서 비동기 방식으로 통신하기 위해 제안되었다. DESYNC는 기존의 TDMA 기법과 달리 동기화를 위한 신호나 추가 작업이 필요 없고, 네트워크에 참여하는 노드의 수에 관계없이 노드들 스스로 동기를 유지시킬 수 있다[2]. 또한 네트워크를 구성하는 노드들에 결함이 발생하여 노드가 삭제되거나, 새로운 노드가 추가되는 등 네트워크를 구성하는 요소가 변해도 스스로 네트워크를 유지한다[4].

DESYNC에서 각 노드들은 일정한 간격마다 주기적으로 펄스신호를 발생시킨다. 이 펄스신호를 firing이라고 하고, 노드는 이 firing 신호를 주기 $T$마다 한 번씩 자신의 전송 범위 내에 있는 주변 노드들에게 브로드캐스팅한다[5]. 네트워크를 구성하는 노드들이 $n$개 존재할 때 각각의 노드를 $N_i(0 \le i \le n-1)$라 한다. $\Phi_{N_i}(t)$는 $t$에 주기 $T$에서 $N_i$가 위치하고 있는 시각을 나타낸다. 여기서 $t$는 전체 시간을 의미하며, 배치되었을 때 0으로 설정되고, 시간 흐름에 따라 증가한다. 노드들은 배치된 시점에 따라 다른 $t$를 가지고 있다. $\Phi_{N_i}(t) \in [0,1]$일 때, $\Phi_{N_0}(t) = 0.75$이면 Fig. 1과 같이 $N_0$는 $T$에서 75%에 해당하는 지점에 위치한 것을 의미한다. $\Phi_{N_i}(t)$는 시간의 흐름에 따라 증가하며, 노드는 $\Phi_{N_i}(t) = 1$이 되면 firing하고 $\Phi_{N_i}(t) = 0$으로 초기화한다. 즉, $\Phi_{N_i}(t) = 0$과 $\Phi_{N_i}(t) = 1$은 시간상으로 동일하다. $T$에서 firing한 노드는 직전에 firing 한 노드와 직후에 firing 한 노드의 시간차를 이용하여 자신의 다음 firing 시각을 정한다. 본 논문에서 노드 $N_i$의 직전에 firing 한 노드와 직후에 firing 노드를 인접노드라고 한다.

노드의 시간차는 Fig. 1과 같이 표현되며 Equation (1)을 만족한다.

$$\triangle_{N_i}(t) = \Phi_{N_i}(t) - \Phi_{N_{i-1}}(t) \tag{1}$$

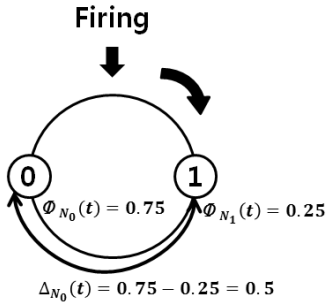


Fig. 1. Firing time difference of nodes

노드가 다음 주기에 firing할 시각은 다음 Equation (2)에 의해 결정된다.

$$\begin{aligned} \tilde{\phi}_{mid}(t) &= \frac{1}{2}\left[\tilde{\phi}_{N_{i-1}}(t) + \tilde{\phi}_{N_{i+1}}(t)\right] \\ &= \phi_i(t) + \frac{1}{2}(\widetilde{\triangle}_{i+1} - \widetilde{\triangle}_i) \end{aligned} \tag{2}$$

모든 인접노드의 시간차가 Equation (3)을 만족하면 네트워크는 동기화 완료가 된다.

$$\triangle^{*}_{N_{i-1}}(t) = \frac{1}{n} \tag{3}$$

즉, 모든 노드의 firing 간격이 일정해 지면 동기화 완료가 이루어진 것이다.

Fig. 2는 DESYNC의 수행 과정을 보여준다. Fig. 2에서 큰 원은 $T$를 의미하고, 작은 원은 노드를 의미한다. Fig. 2(A)는 DESYNC의 초기 상태를 보여준다. 노드 $N_1$, $N_2$,

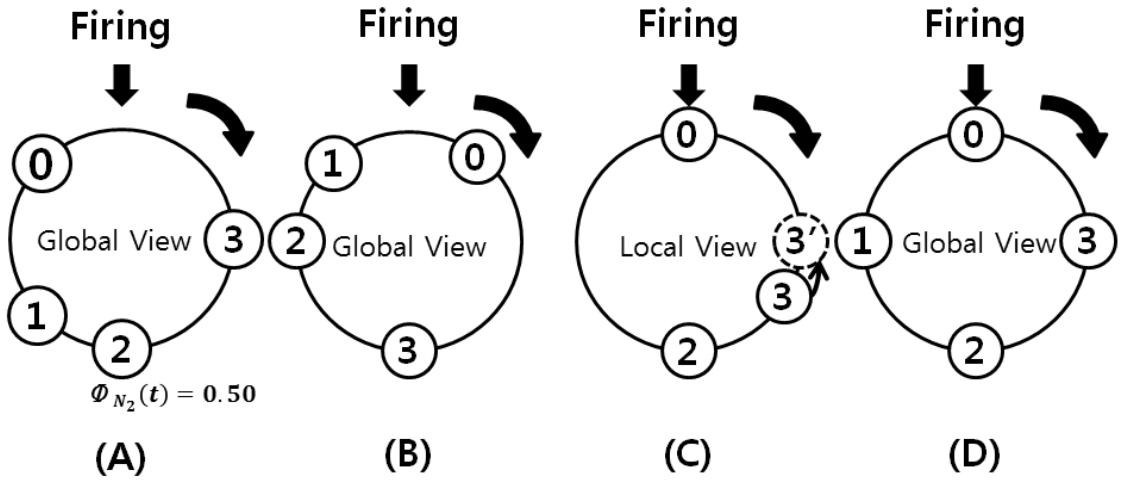


Fig. 2. DESYNC procedure

$N_3$, $N_4$가 초기에 위치하는 시각은 $T$상에 임의로 정해진다. 노드들은 시간의 흐름에 따라 큰 원 위에서 시계방향으로 회전한다고 가정하고, 12시에 위치하면 firing한다. Fig. 2(B)에서 다음에 firing할 노드는 $N_1$이다. 모든 노드들은 firing할 때 자신의 정보를 전송범위 내에 있는 노드들에게 전송한다. 그렇기 때문에 모든 노드들은 $GP$를 기준으로 $T$내에서 자신의 firing 시각을 기준으로 자신이 firing하기 직전에 firing한 노드와 직후에 firing한 노드의 정보를 얻을 수 있다. Fig. 2(C)는 노드 $N_3$의 Local View를 보여준다. 노드 $N_3$은 인접 노드의 firing 정보를 이용하여 자신이 다음에 firing할 위치를 정하게 된다. 노드 $N_3$의 인접노드는 노드 $N_0$과 노드 $N_2$이기 때문에 두 노드와의 시간차와 Equation (2)를 이용하여 $N_3$'을 구할 수 있다. 이렇게 여러 주기가 반복되면 모든 노드들의 간격이 Fig. 2(D)와 같이 일정해지고, 네트워크는 동기화 완료가 된다.

## 3. 시스템 모델

DESYNC는 기존의 동기화 방식의 TDMA 기법과 달리 추가적인 동기화 작업이 필요 없고, 간단한 규칙과 펄스 신호를 이용하여 동기를 유지하기 때문에 비용적인 측면에서 효율적이다[2]. 하지만 DESYNC는 네트워크가 동기화 완료가 되기까지 지연시간을 보장하지 않기 때문에 노드의 초기 firing 시각과, 노드의 개수에 따라 동기화 완료에 이르는 지연시간이 달라진다. 본 논문에서는 이러한 문제를 해결하기 위해 C-DESYNC를 제안하고자 한다. Table 1은 기존의 DESYNC를 수행하기 위한 노드의 기본정보와 C-DESYNC를 수행하기 위한 노드의 기본정보를 비교한 표를 나타낸다. Table 1에서 사용되는 DESYNC와 C-DESYNC의 모델 변수는 아래와 같다.

● $\triangle_{N_i}(t)$
: 자신이 firing하기 직전에 firing한 노드와의 firing시간차.
● $\triangle_{N_{i+1}}(t)$
: 자신이 firing하기 직후에 firing한 노드와의 firing시간차.
● $GP$(Global Packet)
: 노드주기의 시작을 의미하는 신호.
● $FT$(Firing Time)
: 자신이 firing하는 시간정보이며 초기에 $T$상에서 임의로 배치된다.($0 < FT \leq T$)
● $BFC$(Before Firing Count)
: GP신호를 기준으로 한 주기 $T$내에서 자신의 $FT$이전에 firing한 노드의 개수.
● $AFC$(After Firing Count)
: GP신호를 기준으로 한 주기 $T$내에서 자신의 $FT$이후에 firing한 노드의 개수.
● $GPR$(Global Packet Recent)
: $GP$를 수신한 시각.

기존의 DESYNC에서의 각 노드들은 $T$에서 인접한 노드의 firing 시간차 정보 $\triangle_{N_i}(t)$, $\triangle_{N_{i+1}}(t)$과 자신의 $FT$를 가지고 있어 주기마다 Equation (2)에 의해 갱신된다.

DESYNC는 인접 노드와의 시간차를 이용하여 동작되는 반면에 C-DESYNC는 네트워크를 구성하는 전체 노드들의

Table 1. Node information of DESYNC and C-DESYNC

| DESYNC : 노드의 기본정보 | | | | |
|---|---|---|---|---|
| $\triangle_{N_i}(t)$ | $FT$ ($\Phi_{N_i}(t)$) | $\triangle_{N_{i+1}}(t)$ | | |
| C-DESYNC : 노드의 기본정보 | | | | |
| $BFC$ | $FT$ ($\Phi_{N_i}(t)$) | $AFC$ | $GP$ | $GPR$ |

개수를 파악하여 동작한다. 그렇기 때문에 각 노드들은 자신이 firing 한 $FT$와 firing 하기 전후의 시간차 정보 $BFC$, $AFC$를 가지고 있고, 동기화를 위한 정보 $GP$, $GPR$을 가지고 있다.

C-DESYNC의 $FT$는 DESYNC와 마찬가지로 초기에 $T$에서 임의로 배치되고($0 < FT \leq T$), 주기마다 $BFC$와 $AFC$를 이용하여 전체 노드의 개수를 파악하고, Equation (4)에 의해 갱신된다.

$$\Phi_{N_i}^{'}(t) = T \times (BCF \times \frac{1}{ACF + BCF + 1}) \quad (4)$$

## 4. C-DESYNC

본 절에서는 DESYNC에서 동기화 완료가 되기까지 걸리는 지연시간을 보장하지 못하는 문제점을 개선하기 위해 C-DESYNC를 제안한다. DESYNC에서 각 노드들은 네트워크가 동기화 완료가 되기 위하여 인접노드의 시간차와 Equation (2)를 이용하였다. C-DESYNC에서 모든 노드들은 주기의 시작 정보를 가지고 있는 $GP$를 이용하여 동기화를 이룬다. 그리고 DESYNC의 firing 기법을 이용하여 $GP$를 기준으로 $T$내에서 자신의 $FT$를 기준으로 $BFC$와 $AFC$ 정보를 이용해 전체 노드의 개수를 파악하고, Equation (4)를 이용하여 노드들의 다음 $FT$를 정한다. C-DESYNC는 $BFC$를 이용하여 자신이 몇 번째로 firing 하는지를 알 수 있고, $BFC$와 $AFC$를 이용하여 전체 노드의 개수를 파악할 수 있다.

노드들은 초기에 배치되면 $GP$를 수신하기 위하여 초기에 배치되었을 때부터($t = 0$) $T$만큼의 시간이 지날 때까지 대기한다($t = T$). 만약 $T$시간 내에 $GP$를 수신하면 네트워크에 $LN$이 존재하고 있다고 판단하고 일반 노드와 같이 동작한다. 만약 $GP$를 받지 못한 노드는 네트워크에 $LN$이 존재하지 않다고 판단하고, $T$시간 이후에 제일 먼저 firing 하는 노드가 $LN$이 되어 $T$마다 한 번씩 $GP$를 전송하고, 일반 노드의 동작도 수행한다. $GP$신호는 노드의 주기 시작정보를 가지고 있어 이 신호를 받은 일반 노드들은 $LN$의 시간과 동기화를 이룬다. Fig. 3에서 Fig. 5는 C-DESYNC가 적용된 네트워크에서 노드들의 동작 알고리즘이다.

Fig. 3은 $LN$을 선출하기 위한 알고리즘이다. 초기에 노드가 네트워크에 참여하였을 때 노드는 $t = T$ 까지 기다리며, $GP$을 수신했는지 여부에 따라 자신이 일반 노드로 동작할지 $LN$으로 동작할지 결정한다. Fig. 4는 일반 노드가 $GP$를 기준으로 $T$ 내에서 자신의 $FT$를 기준으로 $BFC$와 $AFC$의 개수를 구한 후, 자신의 다음 $FT$를 결정하는 알고리즘이다. 만약 $t - GPR > T$이면 $LN$이 제거된 것으로 판단하고 Fig. 3의 알고리즘을 다시 수행한다. Fig. 5는 $LN$의 동작 알고리즘을 보여준다. $LN$은 $\Phi_{N_i}(t) = 1$가 되면 $GP$를 전송하고, 일반 노드와 같이 동작을 수행한다.

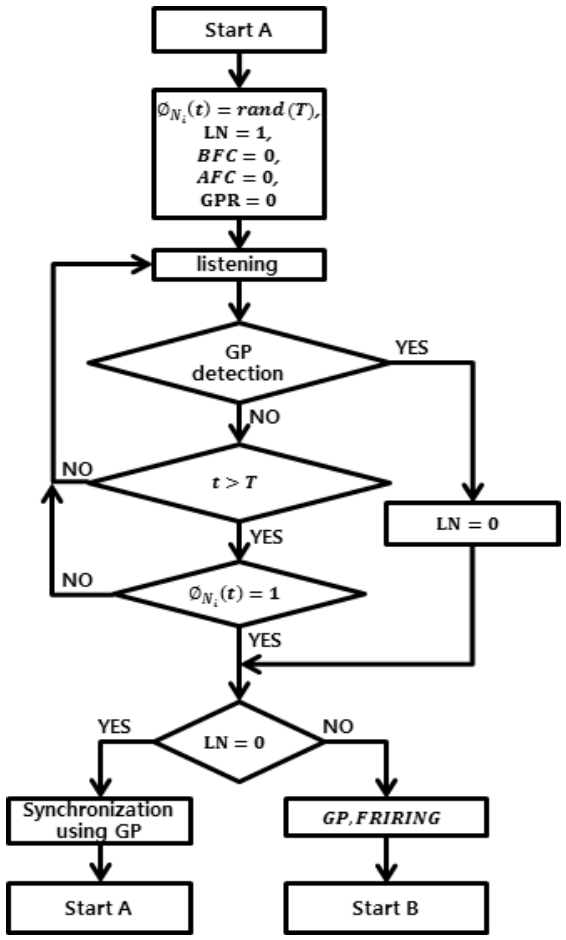


Fig. 3. LN selection algorithm

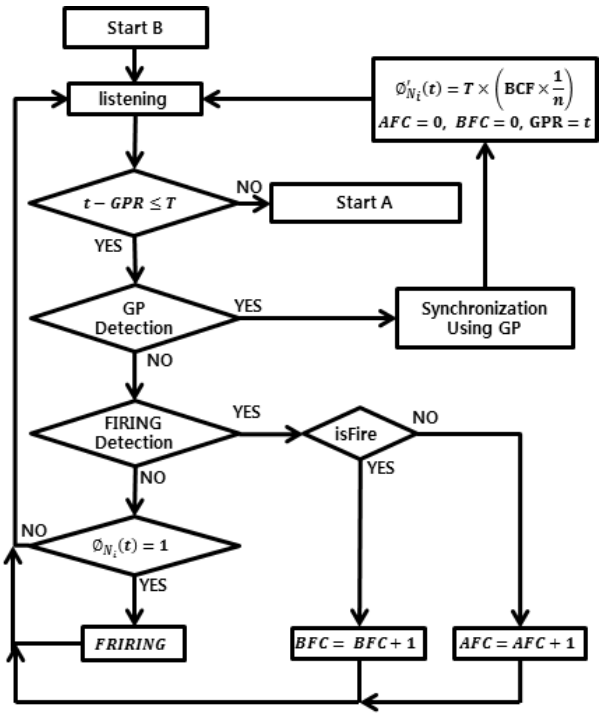


Fig. 4. Normal node algorithm

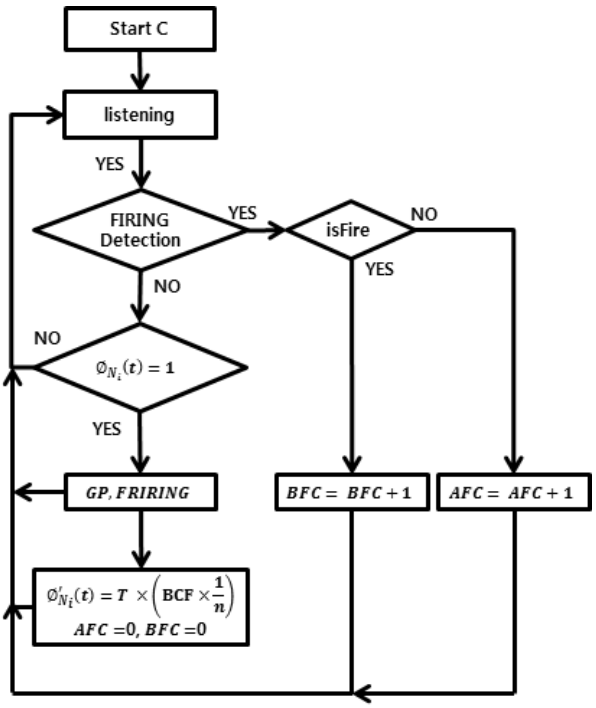


Fig. 5. Leader node algorithm

Fig. 6은 C-DESYNC의 수행 과정을 보여준다. Fig. 6에서 큰 원은 $T$를 의미하고, 작은 원은 노드를 의미한다. Fig. 6(A)는 C-DESYNC의 초기 상태를 보여준다. 노드 $N_1$, $N_2$, $N_3$, $N_4$의 초기위치는 $T$상에 임의로 정해진다. 모든 노드들은 초기에 배치되었을 때부터($t$=0) $T$만큼의 시간이 지날 때까지($t$=$T$) 아무런 수행을 하지 않으며, Fig. 3과 같이 동작한다. $T$시간 이내에 $GP$가 감지되면 네트워크에 $LN$이 있는 것으로 판단하고, Fig. 4에 따라 일반 노드로 동작하게 된다. 만약 $T$시간 이내에 $GP$신호를 감지하지 못한다면 네트워크에 $LN$이 없는 것으로 판단하고, Fig. 5에 따라 $LN$으로 동작하게 된다. 모든 노드들은 시간의 흐름에 따라 큰 원 위에서 시계 방향으로 회전한다고 가정하고, 12시에 위치하면 firing하게 된다. $LN$은 firing하기 전에 자신의 주기의 시작정보를 가지고 있는 $GP$를 전송한다. Fig. 6(A)에서 초기에 배치된 이후 $T$시간만큼 대기한 모든 노드들은 $GP$를 받지 못하였기 때문에 처음 firing하게 되는 노드인 $N_0$가 $LN_0$이 된다. Fig. 6(B)는 $LN_0$가 보내는 $GP$에 의해 모든 노드들의 주기 시작 시각이 동기화된 상태를 보여준다. $LN_0$는 주기의 시작을 맞추기 위해 $GP$를 보냄과 동시에 일반 노드의 동작도 수행한다. Fig. 6(C)에서 동기화된 모든 노드들은 $GP$를 기준으로 $T$동안 firing하는 모든 노드들을 $BFC$와 $AFC$로 저장한다. Fig. 6(C)에 파악한 전체 노드 개수와 Equation (4)를 이용하여, 모든 노드들은 다음 주기의 $FT$를 결정한다. 그 결과 모든 노드들의 간격은 Fig. 6(D)와 같이 일정해지고, 네트워크는 동기화가 완료된다.

## 5. C-DESYNC 동작 시나리오

Fig. 7에서 Fig. 11은 C-DESYNC가 적용된 네트워크에서 상황에 따른 동작 시나리오를 설명하고 있다. 동작 순서는 (1) -> (2) -> (3) … (10) 순으로 진행 된다.

Fig. 7은 초기 네트워크에 노드 $N_0$, $N_1$가 참여하였을 경우를 보여준다. Fig. 7의 1 Period에서 노드 $N_0$, $N_1$는 초기에 배치된 이후 $T$시간만큼 $LN$이 있는지 확인하기 위해 $GP$신호를 기다린다. 만약 $T$시간 이내에 $GP$신호가 감지되면, $N_0$, $N_1$는 네트워크에 $LN$이 있다고 인지하고 $GP$신호에 의해 주기를 초기화한다. 그렇지 않고 $N_0$, $N_1$가 $T$시간

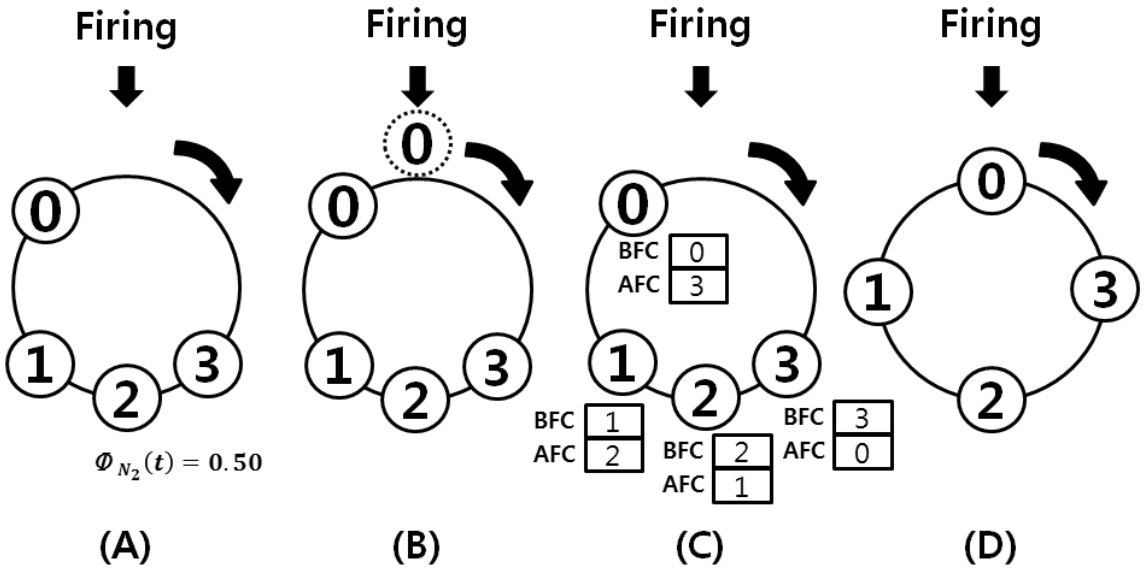


Fig. 6. C-DESYNC procedure

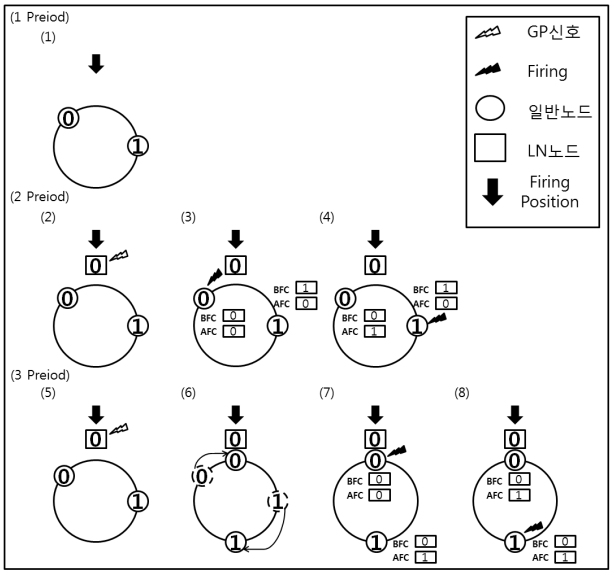


Fig. 7. LN selection in C-DESYNC

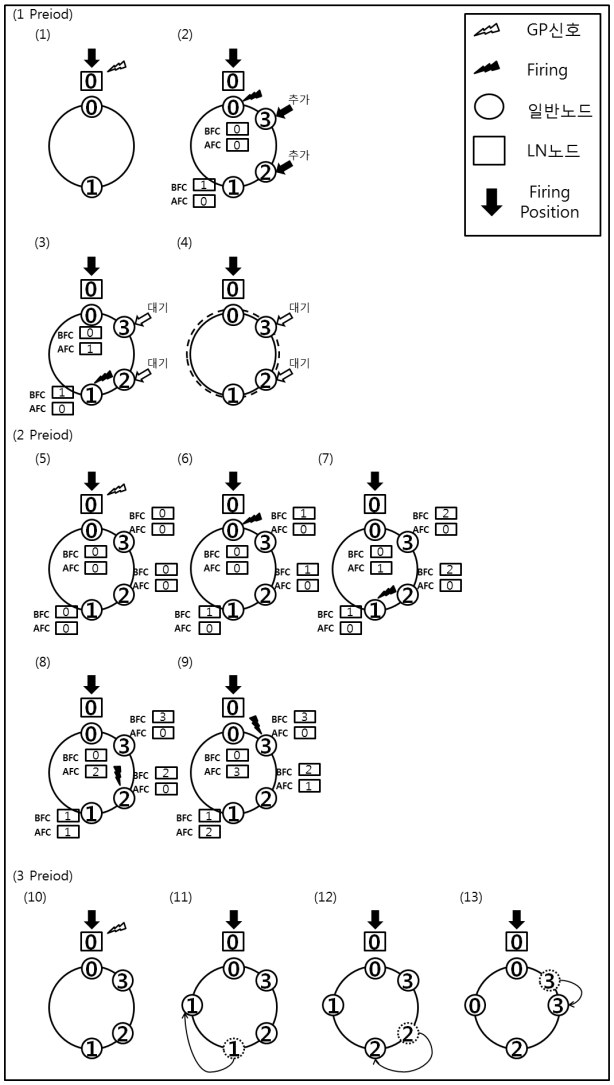


Fig. 8. Addition of two nodes in C-DESYNC

이내에 $GP$신호를 감지하지 못했다면 주변에 $LN$이 없는 것으로 판단하고, Fig. 7의 2 Period처럼 가장 먼저 firing하는 $N_0$가 $LN_0$이 되어 $GP$신호를 주기마다 전송하고, 일반 노드의 동작도 수행한다. 이때 $N_0$, $N_1$는 $GP$와 $T$ 내에서 자신의 $FT$를 기준으로 이전에 firing 하였으면 $BFC$를 증가시키고 이후에 firing 하였으면 $AFC$를 증가시킨다. Fig. 7의 3 Period 이후에는 전체 노드 개수와 Equation (4)를 이용해 $FT$를 정한다. Fig. 7의 3 Period에서 (5)와 (6)은 동시에 일어난다.

Fig. 8는 Fig. 7의 3 Period 이후에 $N_2$, $N_3$이 네트워크에 참여하였을 경우를 보여준다. Fig. 8의 1 Period에서 노드 $N_2$, $N_3$가 초기에 배치된 이후 $T$시간만큼 $LN$이 있는지 확인하기 위해 $GP$신호를 기다린다. Fig. 8의 2 Period에서 $T$ 내에서 노드 $N_2$, $N_3$가 $GP$신호를 인지하여 동기화되고, $T$내에서 자신의 $FT$를 기준으로 $BFC$와 $AFC$를 파악한다. Fig. 9의 3 Period 이후에는 전체 노드 개수와 Equation (4)를 이용해 노드 $N_0$, $N_1$, $N_2$, $N_3$는 $FT$를 정한다. Fig. 8의 3 Period에서 (10), (11), (12), (13)은 동시에 일어난다.

Fig. 9은 $LN_0$가 존재하는 네트워크에 참여하고 있던 노드 $N_0$, $N_1$, $N_2$, $N_3$ 중 $N_3$가 삭제되었을 경우를 보여준다.

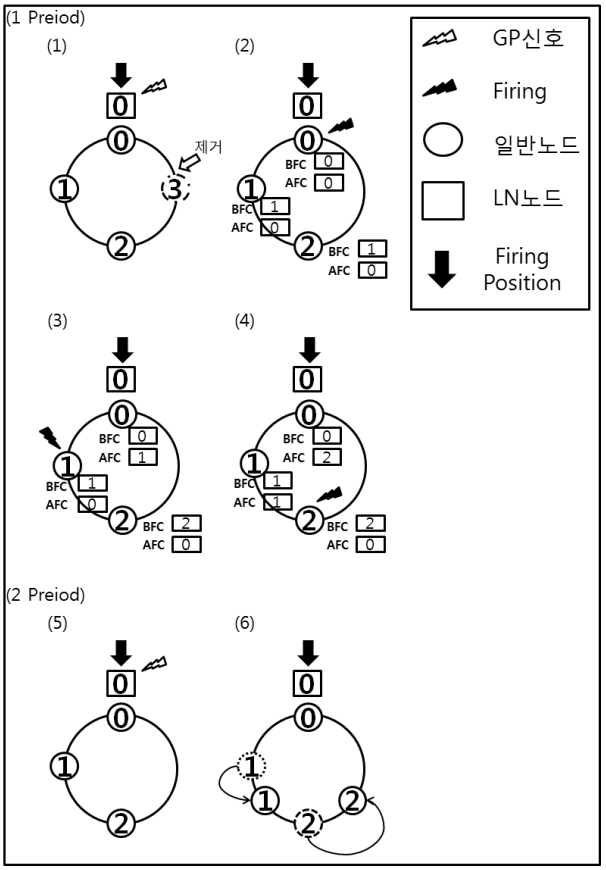


Fig. 9. Removal of a node in C-DEYSNC

Fig. 9의 1 Period에서 $N_3$가 삭제되어 나머지 세 개의 노드만 firing한다. Fig. 10의 2 Period에서 남은 노드들은 $GP$와 $T$내에서 자신의 $FT$를 기준으로 $BFC$와 $AFC$를 파악하고, Equation (4)를 이용해 다음 주기에서 $FT$를 정한다. Fig. 10의 2 Period에서 (5)와 (6)은 동시에 일어난다.

Fig. 10은 Fig. 9의 2 Period 이후에 네트워크에 참여하고 있던 $LN_0$가 삭제되었을 경우를 보여준다. Fig. 10의 1 Period에서 $N_1$노드와 $N_3$노드는 $GP$를 기준으로 $T$동안에 $GP$를 감지하지 못한 상태이다. Fig. 10의 2 Period는 $GP$를 기준으로 $T$ 내에서 $GP$를 인지하지 못한 노드 $N_1$과 $N_2$는 $LN$을 선출하기 위해 대기한다. 이때 먼저 firing하는 $N_1$이 $GP$를 보내며 $N_1$이 $LN$을 수행하며 일반 노드의 역할도 수행한다. 노드 $N_2$는 $LN_1$의 $GP$신호를 감지하여 동기화되고, 노드들은 $GP$를 기준으로 $T$ 내에서 자신의 $FT$를 기준으로 $BFC$와 $AFC$를 파악한다. Fig. 10의 3 Period에서 노드들은 파악된 전체노드의 개수와 Equation (4)를 이용해 다음 주기에서 $FT$를 정한다. Fig. 10의 3 Period에서 (5)와 (6)은 동시에 일어난다.

## 6. 실험 및 고찰

본 논문에서 제안한 C-DESYNC의 성능을 평가하기 위해 개발한 환경은 JAVA를 기반으로 자체 시뮬레이터를 개발하였다. 실험 방식은 에러율을 고려하지 않는 이상적인

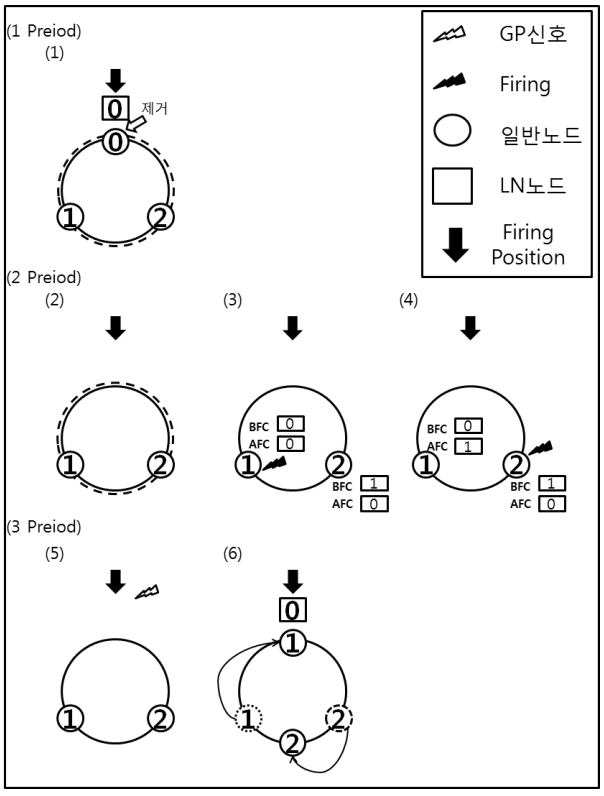


Fig. 10. Removal of LN in C-DESYNC

통신 채널 환경에서 DESYNC와 C-DESYNC를 비교하였다. 비교 성능 평가 요소는 네트워크가 동기화 완료까지 이르는 최대 지연시간인 MTTSC(Maximum Time To Synch Completion)을 측정하였다. 각 실험 결과는 $T=1{,}000$일 때, 1,000번씩 반복한 MTTSC의 평균을 구하였다.

Table 2. MTTSC of DESYNC and C-DESYNC

| 노드 | DESYNC | C-DESYNC |
|---|---|---|
| 5 | 34526 | 2101 |
| 10 | 52554 | 2091 |
| 20 | 89204 | 2047 |
| 40 | 166216 | 2024 |
| 100 | 209231 | 2010 |
| 200 | 234187 | 2005 |
| 250 | 250211 | 2004 |
| 500 | 359604 | 2001 |

Table 2는 DESYNC와 C-DESYNC를 적용한 네트워크에서 각각 노드의 개수가 10, 20, 40, 100, 200, 250, 500개 일 때 MTTSC를 비교한 표이다. 표에서 알 수 있듯이 DESYNC를 적용한 네트워크는 노드 개수에 따라 MTTSC가 증가한 것을 알 수 있다. 이에 반해 C-DESYNC를 적용한 네트워크는 노드의 개수에 관계없이 3주기 이내에 동기화 완료를 이루는 것을 알 수 있다.

Table 3. MTTSC at the removal of a node

| 노드 | DESYNC | C-DESYNC |
|---|---|---|
| 일반 노드 | 22548 | 3173 |
| 리더 노드 | | 5771 |

Table 3은 DESYNC와 C-DESYNC를 적용한 네트워크에서 각각 노드의 개수가 5개일 때 동기화 완료 후($T=2101$), 노드가 하나 삭제되었을 경우, 동기화 완료 상태가 다시 이루어지기까지의 MTTSC를 비교한 표이다. DESYNC를 적용한 네트워크에서는 하나의 노드가 삭제 된 이후에 다시 동기화 완료 상태가 이루어지기까지 추가적인 지연시간이 많이 필요하다. 반면에, C-DESYNC를 적용한 네트워크는 다음 주기에서 동기화 완료가 이루어진 것을 알 수 있다 ($T=3173$).

Table 4. MTTSC at the addition of a node

| 노드 | DESYNC | C-DESYNC |
|---|---|---|
| 일반 노드 | 24714 | 4195 |

Table 4는 DESYNC와 C-DESYNC를 적용한 네트워크에서 각각 노드의 개수가 4개일 때 동기화 완료 후($T=2101$), 노드가 하나 추가되었을 경우, 동기화 완료 상태가 다시 이루어지기까지의 MTTSC를 비교한 표이다. DESYNC를 적

용한 네트워크에서는 하나의 노드가 추가 된 이후에 다시 동기화 완료 상태가 이루어지기까지 추가적인 지연시간이 많이 필요하다. 하지만 C-DESYNC를 적용한 네트워크에서는 *GP*를 수신하기 위해 1주기를 대기한 이후에 다음 주기인 즉 2주기에 동기화 완료가 이루어지는 것을 알 수 있다 ($T$=4195).

## 7. 결 론

본 논문에서는 DESYNC를 이용하여 소규모 네트워크에 적합한 동기화 방식으로 TDMA 기법을 가능하게 하는 C-DESYNC를 제안하였다. C-DESYNC는 주기의 시작 정보를 가지고 있는 *GP*를 이용하여 동기를 유지하고, DESYNC의 firing 기법을 이용하여 전체 노드의 개수를 파악한다. 그리고 이 정보를 이용하여 네트워크가 동기화 완료 상태가 3주기 이내로 빠르게 이루어지도록 한다. 또한 DESYNC를 기반으로 하고 있기 때문에 네트워크를 구성하는 노드들에 결함이 발생하거나, 네트워크를 구성하는 요소가 변해도 스스로 네트워크를 유지할 수 있다. C-DESYNC는 복잡한 방식으로 동기를 유지해야 했던 기존의 동기화 방식과 달리 간단한 정보와 동작방식으로 동기를 유지시키기 때문에 비용측면에서 효율적이며, 동기화 완료 상태까지 걸리는 최대 지연 시간을 3주기 이내로 보장한다. 추후 C-DESYNC 기법을 2-홉 또는 그 이상의 다중 홉으로 확장하고 USRP 또는 무선 센서 노드 플랫폼에 C-DESYNC 및 확장 기법을 구현 및 개발하는 것이 필요하다.

## References

[1] Sivalingam, K. M., Agrawal, P., and Kishore, S. J., “A comparison of MAC protocols for wireless local networks based on battery power consumption”, INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Vol.1, 29 pp.150-157, Mar.-2, Apr., 1998.

[2] Degesys, J., Rose, I., Patel, A., and Nagpal, R., “DESYNC: Self-Organizing Desynchronization and TDMA on Wireless Sensor Networks”, Information Processing in Sensor Networks, 2007. IPSN 2007, 6th International Symposium on, pp.11-20, pp.25-27, April, 2007.

[3] Steven Strogatz, “The Emerging Science of Spontaneous order”, p.352, 2004.

[4] Patel, Ankit, Julius Degesys, and Radhika Nagpal, “Desynchronization: The theory of self-organizing algorithms for round-robin scheduling”, Self-Adaptive and Self-Organizing Systems, 2007. SASO'07, First International Conference on, IEEE, pp.87-96, 2007.

[5] Jeyul Lee, Sanghyun Hyun, and Dongmin Yang, “Asynchronous TDMA scheme using DESYNC in Wireless Networks”, the 40th Korea Information Processing Society Fall Conference Journal, Article 20, No. 2, 2013.11

현 상 현

e-mail : hsh8023@lycos.co.kr

2013년 대전대학교 정보통신공학과(학사)

2013년～현 재 대전대학교 정보통신공학과 석사과정

관심분야 : Sensor Networks, Mobile Applications

이 제 율

e-mail : backsujy@nate.com

2013년 대전대학교 정보통신공학과(학사)

2013년～현 재 대전대학교 정보통신공학과 석사과정

관심분야 : Wireless Networks, Internet of Things

양 동 민

e-mail : dmyang@dju.kr

2000년 POSTECH 컴퓨터공학과(학사)

2003년 POSTECH 컴퓨터공학과(석사)

2011년 POSTECH 컴퓨터공학과(박사)

현 재 대전대학교 정보통신공학과 교수

관심분야 : Wireless Mobile Communications, Cognitive Radio Networks, IoT