DOI: 10.3745/KIPSTC.2011.18C.2.119

플러딩 라우팅 프로토콜과 양방향 LTS를 결합한 센서 노드의 시간 동기화 기법

신 재 혁[†] · 오 현 수[†] · 전 중 남[††]

요 약

일반적으로 센서 네트워크에서는 라우팅 트리를 구축한 후에 시간 동기화 작업을 따로 진행하였다. 그 때문에 패킷교환의 횟수가 늘어나고 전력의 소모를 유발한다. 본 논문에서는 라우팅 트리 구축과정에서 교환하는 정방향과 역방향의 패킷에 LTS(Lightweight Time Synchronization) 알고리즘 연산에 필요한 정보를 추가하여 플러딩 라우팅 트리 구축 알고리즘과 시간 동기화 과정을 결합한 알고리즘을 제안한다. 또한, 일정한 라운드 시간을 사용하여 클럭 휨으로 인한 시간 오류를 보정하였다. 제안하는 알고리즘이 TSRA(Time Synchronization Routing Algorithm) 방식보다 센서 노드들 간의 시간을 더욱 정교하게 동기화한다는 것을 NS2 시뮬레이터를 통해서 증명하였다.

키워드 : 센서 네트워크, 시간 동기화, 라우팅 트리

A Time Synchronization Protocol of Sensor Nodes Combining Flooding-Routing Protocol with Bidirectional LTS

Jaehyuck Shin[†] · Hyunsu Oh[†] · Joongnam Jeon[††]

ABSTRACT

In wireless sensor networks Time synchronization used to be performed after routing tree is constructed. It results in increasing the number of packets and energy consumption. In this paper, we propose a time synchronization algorithm combined with flooding routing tree construction algorithm, which applies LTS (Lightweight Time Synchronization) information packed into the forwarding and backward routing packets. Furthermore, the proposed algorithm compensates the time error due to clock drift using the round time with fixed period. We prove that the proposed algorithm could synchronize the time of among sensor nodes more accurately compared to TSRA (Time Synchronization Routing Algorithm) using NS2 simulation tool.

Keywords : Sensor Network, Time Synchronization, Routing Tree

1. 서 론

무선 센서 네트워크(Wireless Sensor Network)는 다수의 센서 노드들이 지그비(zigbee)나 블루투스(bluetooth)와 같은 무선 통신 방식을 통해 서로 연결된 네트워크이다. 센서 노드들은 컴퓨팅 능력과 무선 통신 능력을 갖추고 있으며, 자연 환경과 같은 넓은 공간에 배치되어 자율적으로 네트워크를 형성하고 센서로부터 획득한 정보를 기지국(base station)으로 전달한다. 무선 센서 네트워크는 원격 감시 및 제어, 자연현상 연구, 그리고 국방 관련 연구 등에 활용된다.

무선 센서 네트워크를 구성하는 각 노드는 주변의 정보를 센싱하고 전달하는 역할이 주어진다. 일반적으로 센서 노드들은 TDMA(Time-Division Multiple Access) 방식을 사용하여 서로 통신한다. 따라서 독립적으로 동작하는 센서 노드들의 시간을 기지국과 정확하게 동기화할 필요가 있다. 노드들은 획득한 정보의 시간을 측정하기 위해서도 기지국과의 시간 동기화는 필수적이다.

노드들은 독립적으로 클럭을 사용하기 때문에 미세한 클럭 주기의 차이인 클럭 휨(clock drift) 현상이 존재한다. 노드들이 동일한 주파수의 클럭을 사용하더라도 클럭 휨에 의하여 장시간 경과 후 서로 간의 시간 오차가 발생할 수 있다. 센서 노드의 수명을 연장할 목적으로 무선 센서 네트워크는 주기적으로 라우팅 트리를 재구축하며 이 때 클럭 휨에 의하여 노드들의 시간이 달라질 수 있다. 따라서 센서

※ 이 논문은 2008년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(NO.R01-2008-000-20485-0).
† 준 회 원 : 충북대학교 컴퓨터과학과 석사과정
†† 종신회원 : 충북대학교 전자정보대학 교수(교신저자)
논문접수 : 2011년 1월 24일
수 정 일 : 1차 2011년 3월 22일
심사완료 : 2011년 3월 23일

노드들은 라우팅 트리를 재구축할 때마다 주기적으로 시간을 동기화할 필요가 있다.

기존의 TPSN(Time-sync Protocol for Sensor Network)[2] 방식은 센서 네트워크의 시간 동기화를 위하여 LTS를[1] 적용한 알고리즘이다. 다만 라우팅 트리 구축 후에 기지국부터 하위 계층으로 시간 동기화를 진행하기 때문에 동기화 과정에서 요구되는 패킷 교환 횟수가 많다. RTAF(Routing-Tree construct Algorithm by single Flooding)[3] 알고리즘은 플러딩 라우팅 프로토콜의 역방향 정보를 활용하여 한 번의 플러딩으로 네트워크의 라우팅 트리를 구축하는 알고리즘이다. TSRA(Time Synchronization Routing Algorithm)[4] 알고리즘은 플러딩 라우팅 프로토콜에 시간 동기화 정보를 포함하여 라우팅 트리 구축과 시간 동기화를 동시에 실행하여 효율성을 높였다.

본 논문에서 제안하는 TSRA-BL 알고리즘은 라우팅 트리 구축 과정에서 부모 노드가 자식 노드에게 전송하는 전방향 패킷뿐만 아니라 자식 노드가 부모 노드에게 응답하는 역방향 패킷에도 시간 동기화 과정을 포함하여 보다 정확한 시간을 동기화하도록 TSRA 방식을 개선한 것이다. 또한 시간 동기 패킷을 비콘 신호로 사용해서 기존 TSRA에서 클럭 휨 현상으로 발생하는 누적 시간 오차 현상을 보정하였다. TSRA-BL 알고리즘을 사용하면 라우팅 트리 구축을 완료함과 동시에 보다 정확하게 시간을 동기화할 수 있다.

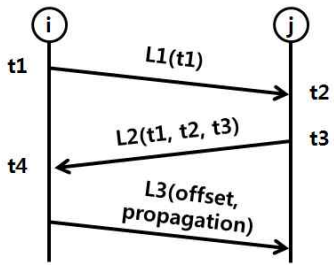
2. 관련 연구

센서 네트워크는 패킷의 전달을 위한 라우팅 트리 구축 과정과 노드들의 시간을 동기화 하는 과정이 필요하다. 일반적으로 센서 네트워크의 라우팅 프로토콜은 평면 기반 라우팅, 위치 기반 라우팅, 그리고 계층 기반 라우팅 프로토콜로 구분할 수 있다[7].

플러딩 프로토콜은 계층 기반 라우팅에 속하며 전파 범위에 있는 모든 노드에 패킷을 전달하는 라우팅 프로토콜이다. 플러딩 프로토콜을 사용하는 노드가 비콘 패킷을 수신하면 이것을 방송(broadcast)하여 근접한 모든 노드에게 전달한다. 패킷은 지정된 홉 수가 될 때까지 모든 가능한 경로를 통하여 전달한다. 기지국에서 라우팅 트리를 구축하기 위하여 플러딩 프로토콜을 사용한다.

RTAF (Routing-Tree construct Algorithm by single Flooding)는 플러딩 프로토콜을 기본으로 한 라우팅 트리 구축 알고리즘이다. 근접한 모든 노드에게 방송하는 정방향 패킷과, 되돌아오는 역방향 패킷을 이용하여 트리를 구축한다. RTAF 는 역방향 패킷을 활용함으로써 한번의 플러딩으로 라우팅 트리를 구축한다.

시간 동기화 알고리즘에는 시간을 동기화 하는 두 노드간 시간 차이를 구하는 LTS 알고리즘, 수신 노드들 간의 시간을 동기화 하는 RBS(Reference Broadcast Synchronization) 알고리즘[5], 송신자-수신자간의 시간 동기화 방법을 사용하는 TPSN, Tiny-Sync[6] 등이 있다.



(그림 1) LTS 시간 동기화 기법

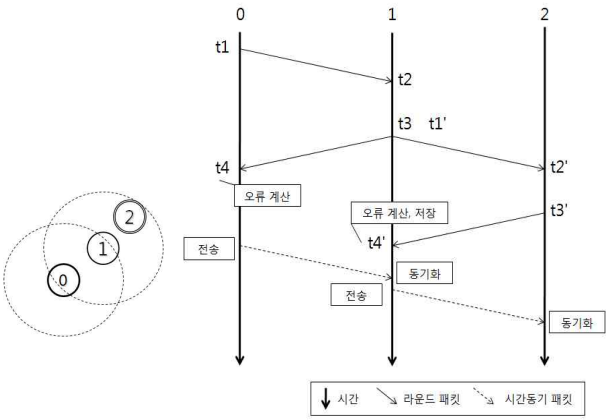
$$\text{Offset} = \frac{(t2 - t4 - t1 + t3)}{2}$$

$$\text{Propagation} = \frac{(t4 - t3) + (t2 - t1)}{2} \quad \text{---------- (1)}$$

LTS 기법은 두 노드간 시간 차이와 전송지연 시간을 계산하는 방법이다. 두 노드가 (그림 1)과 같이 시간 정보가 포함된 패킷을 주고 받는다. 기준 노드는 식(1)에 의하여 시간 차이와 전송지연 시간을 구한다. TPSN은 LTS 기법을 사용한 무선 센서 네트워크 시간 동기화 알고리즘이다. TPSN 알고리즘은 라우팅 트리 구축 단계와, 시간 동기화 과정을 순차적으로 진행하여 네트워크를 동기화한다. RBS는 싱크 노드가 보낸 패킷을 수신하는 노드들 간에 시간 동기화를 진행하는 방법으로, 수신 노드들은 싱크 노드가 보낸 시간을 서로 주고받음으로써 서로 간의 시간을 맞춘다. RBS는 센서 네트워크를 구성하는 전체 노드들에 대한 시간 동기화는 다루지 않는다.

Tiny-Sync는 두 노드들 간의 오프셋과 클럭 휨 현상을 고려하여 시간을 동기화하는 방법으로 동기화에 사용되는 데이터 샘플을 통해 동기화를 수행한다. 이 방법도 역시 라우팅 트리를 구축한 후에 두 노드들 간의 시간 차이를 구하기 위하여 LTS를 사용한다. LTS 방법으로 전송하는 샘플 데이터의 수가 많을수록 오프셋과 클럭 휨을 정확하게 보정할 수 있지만 데이터를 저장하는 저장 공간을 많이 소비하는 단점이 있다. 단점을 해결하기 위해 Mini-Sync는 알고리즘의 복잡도는 증가하지만 적은 샘플 데이터를 이용하여 공간의 낭비를 줄이는 방법도 함께 제시하였다.

TSRA는 단일 플러딩 라우팅 프로토콜을 사용하여 토폴로지 구성과 시간 동기화를 동시에 이루어 패킷의 교환을 최소화하여 보다 효율적으로 무선 센서 네트워크의 라우팅 트리 구축과 시간 동기화를 구현한 알고리즘이다. TSRA는 크게 라운드 패킷과 시간 동기 패킷 두 종류를 사용하여 시간 동기화한다. 라우팅 트리를 구축하는 과정에서 플러딩 라우팅 프로토콜의 정방향 패킷 뿐만 아니라 역방향 패킷도 활용하여 각 패킷에 시간정보를 포함 시킨다. 라우팅 과정에서 주고받는 시간정보들로 부모노드에 해당하는 노드가 LTS 연산을 수행하여 자식노드에게 시간 동기 패킷에 보정 시간을 보내면 자식노드는 그 값으로 자신의 시간을 수정하는 방식이다.



(그림 2) TSRA 시간 동기 기법

3. TSRA-BL (Time Synchronization Routing Algorithm - Bidirection LTS)

3.1 이론적 배경

TSRA-BL은 무선 센서 네트워크에서 라우팅 트리 구축과 시간 동기화를 동시에 수행하는 TSRA를 기반으로 설계되었다. TSRA 알고리즘은 제한적인 자원을 갖고 있는 센서 네트워크의 특성을 고려하여 패킷의 교환 횟수를 줄여서 라우팅 트리 구축과 시간 동기화에 요구되는 시간을 단축시키고, 그에 따라 전력 소모를 최소화 하는 방법이다. 플러딩 방식에서의 역방향 패킷을 활용하여 라우팅 트리 구축과 시간 동기화를 통합한 TSRA 방식에 양방향 LTS 연산을 채택하여 그 정밀도를 높였다.

다수의 센서 노드들은 각각의 하드웨어 특성에 따라 클럭의 주기가 모두 동일하지 않은 현상이 있는데 그것을 클럭 휨(clock drift)이라 한다. TSRA 알고리즘에서는 이러한 클럭 휨 현상을 해결하지 못하여 클럭 휨에 대한 오차가 계속 누적되는 결과를 보였다. TSRA-BL 알고리즘은 시간 동기 패킷을 비콘 신호로 활용하여 클럭 휨을 계산하는 기능을 포함하고 있으며, 이에 따라 두 번째 라운드부터 클럭 휨에 의한 시간 오차를 보정하는 능력을 갖는다.

TSRA-BL은 라우팅 트리 구조를 만들고 시간을 동기화하기 위해 라운드 패킷과 시간 동기 패킷을 사용한다. 라운드 패킷은 노드들의 부모-자식 관계를 파악하여 라우팅 트리를 구축하기 위해 사용되고, 시간 동기 패킷은 시간 동기화 과정을 위해 사용된다. 라운드 패킷의 구조는 <표 1>과 같고, 시간 동기 패킷의 구조는 <표 2>와 같다.

3.2 TSRA-BL 구축 알고리즘

본 논문에서 제안하는 TSRA-BL은 TSRA를 기반으로 설계하였다. 기존 TSRA 방식과의 차이는 두 노드의 시간 차이를 어떤 노드가 계산하는가에 있다. TSRA 알고리즘은 트리 구조에서의 부모와 자식 노드간 시간 차이를 부모 노드가 중심이 되어 해결한다. 부모 노드는 시간 동기가 이루어진 시점에서 LTS 방식으로 시간 차이를 계산하여 자식

〈표 1〉 라운드 패킷 구조

Entry	Type	Size	설명
ID	nsaddr_t	2	노드 식별번호
PID	nsaddr_t	2	부모 노드의 식별번호
Round	uint16_t	2	라운드 번호
Hop	uint16_t	2	트리구조 내에서의 차수
Recv_ts	double	8	라운드 패킷을 받은 시간
Send_ts	double	8	라운드 패킷을 보낸 시간

〈표 2〉 시간 동기 패킷 구조

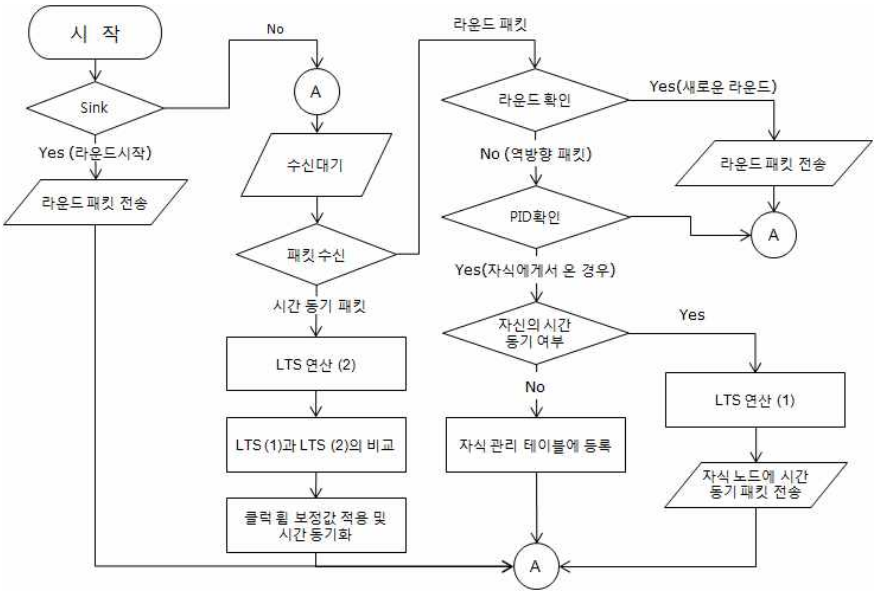
Entry	Type	Size	설명
PTYPE	Char	2	패킷 타입
ID	nsaddr_t	2	노드 식별번호
Offset	double	8	두 노드 간 시간차
Propagation	double	8	전송 지연 시간
ParentOffset	double	8	부모노드의 Offset
Send_ts1	double	8	부모 노드에게 라운드 패킷을 보낸 시간
Recv_ts	double	8	부모 노드가 자식노드에게 라운드 패킷을 받은 시간
Send_ts2	double	8	시간 동기 패킷을 자식노드에게 보낸 시간

노드에게 그 결과를 보내고, 자식 노드는 부모 노드에게 받은 시간 오류 값을 적용하여 노드간 시간 동기화를 이루는 방식이다. TSRA-BL은 자식 노드가 부모에게 받은 시간차 값 외에 스스로 LTS 연산을 수행함으로써, 부모 노드가 보내준 시간 차이와 자신이 계산한 시간 차이를 가질 수 있다. 이 두 가지 정보를 활용함으로써 시간차 계산에 포함되는 오차를 줄일 수 있는 가능성이 높아진다. (그림 3)은 TSRA-BL 알고리즘의 과정을 흐름도로 표현하였다.

TSRA-BL 알고리즘은 기지국 시간의 기준이 되는 기지국 노드에서 시작한다. 기지국 노드가 주변의 노드들에게 라운드 패킷을 전송한다.

일반 노드들은 수신 대기 상태에서 패킷을 수신하게 되면 라운드 패킷인지 시간 동기 패킷인지 판단하고 그에 해당하는 임무를 수행한다. 노드가 받은 패킷이 라운드 패킷일 경우에는 자신의 라운드와 패킷에 표기된 라운드를 비교하여 새로운 라운드인지 판단한다. 새로운 라운드일 경우에는 LTS 연산을 위한 시간정보 등의 패킷 내용을 채워 주변의 노드에게 전송한다. 새로운 라운드가 아닌 경우에는 역방향 패킷으로 판단하고 패킷의 PID를 확인한다. 만약 PID가 자신의 ID와 같지 않을 경우에는 다시 수신 대기 상태가 된다. 자식 노드에게서 역방향 패킷을 수신한 경우에는 자신이 시간 동기가 완료 되었는지를 먼저 판단하여 시간 동기가 되기 전에 자식에게서 패킷을 받으면 자식 관리 테이블에 먼저 등록한다. 시간 동기화가 완료 되었을 때는 LTS 연산을 수행하여 자식 관리 테이블에 등록된 노드 중에 시간 동기 패킷을 전송하지 않은 자식노드에게 전송한다.

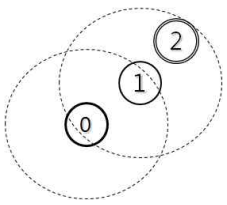
시간 동기 패킷을 수신한 노드는 패킷의 시간정보를 이용해서 LTS 연산을 한번더 수행한다. 부모노드가 연산해준



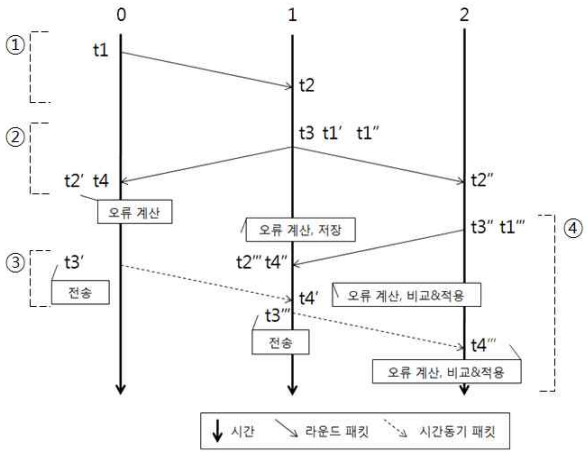
(그림 3) TSRA-BL 알고리즘

결과 값과 자신이 연산한 결과 값을 부모노드의 시간에 대입, 비교하여 둘 중 더욱 정밀한 값을 선택하여 시간을 동기화하는 오프셋으로 사용하게 된다. 시간 동기화의 마지막 단계는 시간 동기 패킷을 비콘 신호로 사용하여 라운드별 시간을 기준으로 클럭 휨에 대한 오류를 보상하여 시간을 적용한다.

위와 같은 과정을 네트워크 내의 모든 노드가 시간 동기화 될 때까지 반복한다. (그림 4)와 (그림 5)로 위 알고리즘의 간단한 적용의 예를 들어 설명하였다.



(그림 4) 노드배치의 예



(그림 5) TSRA-BL 시간 동기화 기법

$$Candidate1 = | Parent\_ts - (Current\_ts - Offset\_on\_Packet) |$$

$$Candidate2 = | Parent\_ts - (Current\_ts - Result\_of\_2^{nd}LTS) |$$

----------(2)

(그림 4)와 같이 배치된 노드들은 (그림 5)와 같은 단계를 거쳐 라우팅 트리와 시간을 동기화한다. (그림 5)에서 0번 노드는 기지국 노드로써 네트워크의 시간을 동기화하는 기준이 된다. 0번 노드와 근접한 1번 노드와 1번 노드에 근접한 2번 노드가 토폴로지를 구성하여 0번 노드의 시간으로 동기화를 진행한다.

(그림 5)의 ① : 0번 노드는 자신의 로컬 시간을 담아 라운드 패킷을 방송한다. 시간 동기가 되지 않은 1번 노드는 0번이 보낸 라운드 패킷을 받게 되고 자신의 라운드와 비교하여 최신 라운드 일 때 해당 라운드 패킷을 받은 시간을 기록한다.

(그림 5)의 ② : 1번 노드는 그 과정에서 얻은 두 시간과 자신의 부모노드 ID와 새로운 라운드 패킷을 보내는 시간을 포함 시켜 방송한다. 이때 1번 노드가 보낸 새로운 라운드 패킷은 근접한 0번 노드와 2번 노드에게 전달된다. 패킷을 수신한 0번 노드는 라운드 확인을 통하여 역방향 패킷인지를 먼저 파악하고, 패킷에 기록된 부모ID를 확인하여 자신의 ID와 같을 경우에는 패킷을 받은 시간과 패킷 내에 포함된 세 개의 시간을 사용하여 LTS 연산을 수행한다.

(그림 5)의 ③ : 앞서 계산된 오차와 역방향 패킷의 송수신 시간, 그리고 시간 동기 패킷을 1번 노드에게 보내는 시간을 포함하여 다시 1번 노드에게 보낸다. 시간 동기 패킷을 수신한 1번 노드는 패킷에 포함된 시간 정보들을 토대로 LTS 연산을 수행한다. 그렇게 되면 1번 노드는 스스로 연산한 시간 오차와, 0번 노드가 계산해서 시간 동기 패킷에

포함시켜 보내준 오차 두 가지 결과를 갖는다. 이때 이 두 가지 값을 수식(2)와 같은 연산을 통하여 둘 중에 0번 노드의 시간과의 차이가 적은 값을 선택하고, 시간 동기 패킷을 비콘 신호로 인식하여 해당 라운드의 기준이 되는 시간과 패킷에 포함된 전송 지연시간 정보를 기준으로 자신의 시간을 수정한다.

(그림 5)의 ③ : 1번 노드가 보낸 새로운 라운드 패킷은 근접한 노드인 2번 노드가 수신하게 되고 위와 같은 과정을 거친다. 하지만 1번 노드가 시간 동기가 되어있지 않은 상황에서 2번 노드가 보낸 역방향 패킷을 받게 되면 먼저 2번 노드를 자신의 자식노드로 인식을 하고, 시간 값을 따로 저장 해 두었다가 1번 노드 자신이 시간 동기가 되었을 때 비로소 LTS 연산을 하여 시간 오차를 2번 노드에게 전송한다.

기준이 되는 0번 노드는 일정한 라운드 갱신 시간을 두어 새로운 라운드를 수행하게 되고, 네트워크 내의 모든 노드가 기준 시간으로 동기화 되는 시점까지 위와 같은 과정을 연쇄적으로 반복한다.

3.3 TSRA와의 비교

본 논문에서 제안하는 TSRA-BL 알고리즘은 TSRA 알고리즘을 토대로 하여 TSRA의 장점들을 모두 포함하고 있으며, 양방향 LTS 방식과 클럭 휨 보정을 통해 TSRA보다 정밀한 시간 동기화를 이룰 수 있다.

동기화된 노드와 비동기 노드 간의 시간차를 구하고 그 관계를 형성하는 상황을 예로 들겠다. TSRA는 동기화된 노드에서 LTS 방식을 통해 두 노드 간 시간차를 계산한다. 동기화된 노드는 계산한 시간차 값을 비동기 노드에게 전달하고, 비동기 노드는 동기 노드에게서 받은 시간차를 자신의 로컬 시간에 적용하여 동기를 이룬다. TSRA-BL은 동기화된 노드와 비동기 노드, 두 노드에서 각각 LTS 연산을 수행한다. 그 과정에서 얻어진 두 개의 시간차 데이터 중 더욱 정밀한 시간 동기화를 이룰 수 있는 값을 선택하여 적용한다. 또한 시간 동기 패킷을 비콘 패킷으로 사용하여 클럭 휨 현상을 수정하여 기존 TSRA 방식 보다 더욱 정밀하게 시간을 동기화 할 수 있다.

4. 시뮬레이션 및 성능 분석

4.1 시뮬레이션 환경

시뮬레이션은 Windows XP 32bit OS를 기반으로 VMware를 설치하여 가상머신을 생성 후, LINUX 커널 2.6의 Fedora 8 배포판 OS를 설치하여 진행하였다. 시뮬레이션 툴은 NS-2.33을 사용하였고, 언어는 C++, OTCL(Object Tool Command Language)를 이용하여 제안하는 TSRA-BL과 비교 대상인 TSRA를 구현하였다.

NS2는 센서 네트워크를 시뮬레이션 하기 위해 노드와 링크(Link)를 설정하는데, 이 때 각 노드마다 부여받은 특정한 기능을 수행하기 위한 에이전트를 결정하고 이벤트를 설정하는 과정을 수행해야 한다. 네트워크 노드와 각 에이전트의 내부 기능 기술은 C++언어로 작성하고, 이벤트를 설정을 위해서 NS2 용 스크립트 언어인 OTCL로 작성하였다.

4.2 환경 설정

무선 센서 네트워크를 시뮬레이션 하기 위해서는 노드를 배치해야 한다. 본 실험에서는 총 100 개의 노드를 격자형과 랜덤한 위치에 초기 배치시켜 실험을 진행하였다. NS2 시뮬레이터에서 정의한 무선 센서네트워크의 특성에 따라 매 라운드마다 라우팅 트리 구성이 변할 수 있다. 전력 소비량 기준은 버클리 모트 전력 소모량 측정표[8]를 참고하였다.

TSRA-BL 알고리즘은 200초의 라운드 시간 간격으로 토폴로지 구성과 시간 동기화를 수행한다. 100개의 노드들에게는 서로 다른 로컬 시간을 갖게 하기 위해서, 싱크노드를 제외하고 모두 -10초에서 10초 사이의 랜덤한 오차 값을 주었다. 노드들은 개별적으로 존재하기 때문에 클럭의 생성 주기가 모두 다를 수밖에 없다. 그러한 환경을 만들기 위해 각 노드마다 -0.001%부터 0.001%초까지 랜덤하게 클럭 휨 값을 설정하였다. 총 10 라운드로 진행하며 각 경우의 수마다 2,000초 동안의 네트워크에서 수행되어지는 모든 과정을 수치화하여 측정하였다.

NS2 시뮬레이터에 적용한 환경변수는 <표 3>에 나타내었다.

〈표 3〉 NS2 의 실험 환경 변수

NS2의 실험 환경 변수	
Mote	MICA2 - ATmega128
Channel Type	Channel/Wireless Channel
Network Interfaces	Phy/WirelessPhy/802_15_4
Mac Protocol	Mac/802_15_4
Energy Consumption rate (CC1000)	rxPower 0.3
	txPower 0.4
Offset	-10 ~ 10 sec (Random)
Nodes clock_drift	-0.001 ~ 0.001 sec (Random)

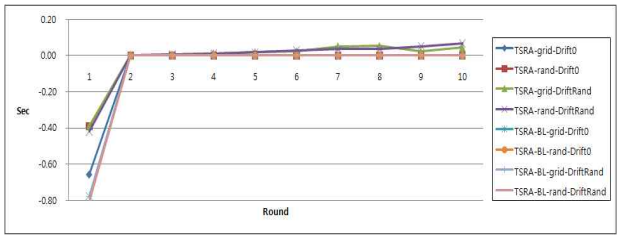
본 논문의 타당성을 검증하기 위한 실험에서 얻고자 하는 성능 분석 지표는 다음과 같다.

- 라운드별 초기 시간 오차의 분포 : 각 노드들의 시간을 달리 해주기 위해 초기에 설정한 오차 값들이, TSRA-BL 알고리즘을 통해 수정되는지에 대해 검증한다.
- 수정시간 적용 후 시간편차 : TSRA-BL 을 통해 얻은 수정시간을 비동기노드의 로컬 시간에 적용하였을 때의 결과로 양방향 LTS 방식의 성능을 검증한다.
- 라운드별 에너지 잔량 : 시간 동기화가 완료된 시점에서의 에너지 잔량을 측정하여, 한 라운드를 진행할 때 소모하는 전력량을 알아본다.

4.3 실험 결과 및 분석

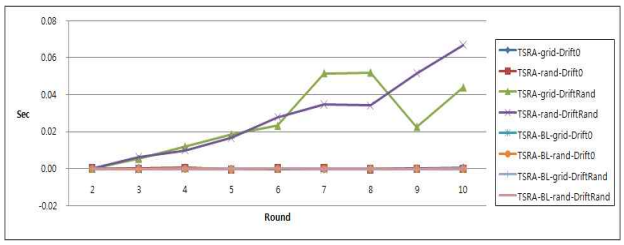
(1) 라운드별 초기 시간 오차의 분포

본 논문의 결과를 시뮬레이션 하기 위해 각 노드에 초기 시간 오차를 임의로 설정하였다. TSRA와 TSRA-BL 두 알고리즘의 비교를 위해 노드 배치 방식과 클럭 휨 값을 다르게 적용하여 총 여덟 가지의 경우의 결과를 볼 수 있도록 하였다.



(그림 6) 라운드별 초기 오차

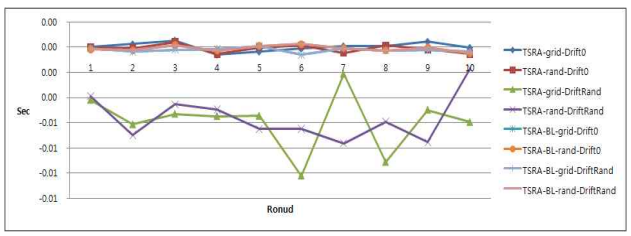
(그림 6)은 각 라운드별 초기 오차 값이다. 1라운드가 시작될 때 랜덤하게 설정 해준 시간 오차는 첫 라운드가 지난 후 수정된 것을 볼 수 있다. 2라운드 이후의 그래프는 수의 표현 범위가 100분의 1초 단위 이하고 낮기 때문에 2라운드 이후의 상황은 (그림 7)에 세밀하게 표현하였다.



(그림 7) 2라운드 이후의 라운드별 초기 오차

클럭 휨 값을 랜덤하게 설정하였을 때 TSRA 알고리즘에서는 클럭 휨 값이 라운드를 진행하면서 계속 누적되어 시간 오차가 조금씩 커지는 모습을 볼 수 있다. 반면 TSRA-BL은 시간 동기 패킷을 비콘 신호로 사용한 방법을 통해 클럭 휨 값을 보정하여 시간 오차가 누적되지 않고 미세한 시간 차이를 갖는 것을 볼 수 있다.

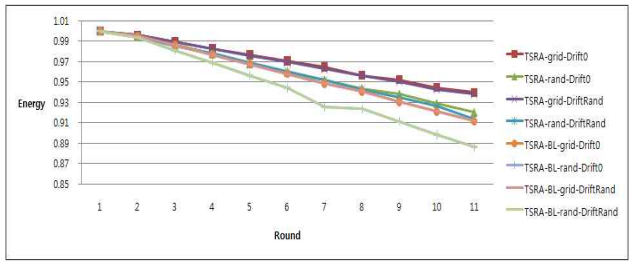
(2) 수정시간 적용 후 시간편차



(그림 8) 수정시간 적용 후 시간 편차

(그림 8)은 TSRA와 TSRA-BL 방식으로 각각 시간 동기화를 진행한 후의 시간 차이를 보여준다. 클럭 휨 값을 0으로 설정한 실험에서는 두 방법 모두 약 만분의 일초 단위의 시간차이를 보였다. 클럭 휨 값을 랜덤하게 설정한 실험에서 TSRA 방식은 시간 오차 수정 후에도 비교적 큰 시간 오차를 보이는 반면 TSRA-BL은 클럭 휨 값을 0으로 설정한 실험과 오차 범위가 크게 다르지 않다는 것을 확인할 수 있다.

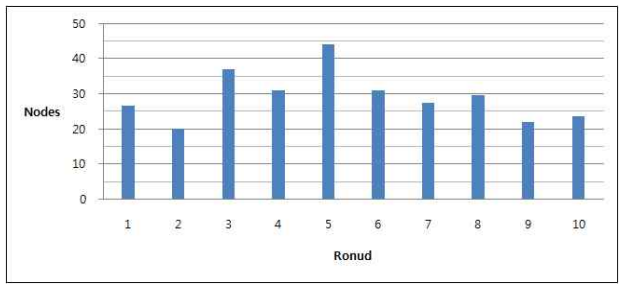
(3) 라운드별 에너지 잔량



(그림 9) 라운드별 에너지 잔량

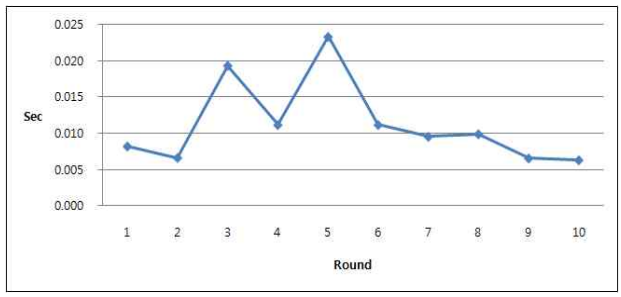
(그림 9)는 라운드별로 시간 동기화가 이루어진 시점의 에너지 잔량을 나타낸다. TSRA-BL 방식이 기존 TSRA보다 에너지를 더 소모하는 결과를 볼 수 있는데, 이는 기능의 추가로 인한 패킷 크기의 증가와 양방향 LTS 연산, 클럭 휨 값 연산 등의 추가 작업을 수행하기 때문이다.

(4) TSRA 와 TSRA-BL 의 비교



(그림 10) 개선된 노드 수

(그림 10)은 TSRA-BL 방식이 부모 노드에서 계산한 값을 그대로 사용하지 않고, 비동기 노드 자체에서 계산한 값을 선택한 횟수를 보여 준다. 총 100개의 노드로 구성된 네트워크에서 약 25%의 확률로 비동기 노드의 연산을 선택하는 결과를 볼 수 있다. 이는 단방향 LTS 연산을 수행하였을 때보다 더욱 정밀한 값을 획득할 수 있는 기회를 얻는 것을 말한다.



(그림 11) 라운드별 개선 시간 총합

(그림 11)은 TSRA 알고리즘을 통해 시간 동기화를 이룬 후의 결과를 기준 값으로 하여, TSRA-BL 방식의 시간 동기화 진행 결과가 얼마만큼의 시간 향상을 보이는지를 나타내는 그래프이다. 각 라운드를 진행 했을 때, 해당 라운드에서 TSRA 대비 향상을 이룬 시간을 합산하여 보여준다. TSRA-BL 방식을 사용했을 때, 기존의 TSRA를 사용했을 때 보다 전체 누적 평균 0.0112초 정도의 시간만큼 더욱 정밀한 시간 동기화를 이루고 있다는 것을 확인할 수 있다. 3라운드와 5라운드에서 개선시간의 차이를 보이는 것은 라운드마다 라우팅 트리의 구성이 조금씩 변하게 되고 전달지연시간이 노드마다 다르기 때문에 부모노드가 계산한 시간 오차 보다 자식노드가 계산한 값을 선택하는 경우가 일정하지 않을 수 있다.

5. 결 론

무선 센서 네트워크 분야에서 네트워크의 시간 동기화 기법과 라우팅 트리 구축 기법은 매우 중요한 의미를 가진다. 센서 네트워크의 특성상 제한적인 컴퓨팅 환경과 전력량 때문에 이에 대한 연구가 활발히 진행되고 있다. 그동안 무선 센서 네트워크에서의 동기화 알고리즘들은 네트워크 구조를 만들기 위한 과정과 시간 동기화가 별도로 구현되어 있어서 다수의 패킷 교환이 필요했다.

본 논문에서는 양방향 LTS 를 수행하여 라우팅 트리 구축과 보다 정밀한 시간 동기화를 이룰 수 있는 TSRA-BL 알고리즘을 제안하였다. 두 노드 간 시간차를 구할 때 시간 동기화 된 한쪽 노드에서만 연산하는 것이 아니라 동기화 되지 않은 노드에서도 연산을 하여 두 개의 차이 값 중 보다 정밀한 시간차를 선택하여 적용하였다. 또한 시간 동기 패킷을 비콘 패킷으로 사용하여 클럭 휨을 보정하도록 설계하여 시뮬레이션을 통해 동기화 과정을 검증하였다.

성능 분석 및 평가는 기존 TSRA 알고리즘과의 비교 실험을 통해 진행하였다. 동기화 과정을 마친 후 교정된 시간과 에너지 잔량을 측정하여 시간 동기화 정확도와 소모하는 에너지의 양을 측정하였다. 교정된 시간 측정을 통해 TSRA-BL 알고리즘이 클럭 휨으로 인한 시간 오차를 수정하고 있음을 확인 할 수 있었다. TSRA-BL 알고리즘이 기존 TSRA 알고리즘 보다 에너지 소비가 큰 것은 패킷 크기의 증가와 양방향 LTS, 그리고 클럭 휨 값 수정을 위한 연산들이 추가되었기 때문이다. 양방향 LTS를 통한 선택적 방법의 효율성은 동일한 조건에서 단방향 LTS 방식으로 얻은 결과와 비교하여 검증하였다. 양방향 LTS 연산을 통해 선택적으로 차이 값을 적용한 사례는 평균적으로 약 25%였으며, 그에 따라 기존 TSRA 알고리즘 보다 총 0.0112초 더 정밀하게 시간을 동기화할 수 있었다.

본 연구를 통해 TSRA-BL은 TSRA 알고리즘의 패킷 교환 횟수를 유지 하면서, 더욱 정밀한 시간 동기화를 이룰 수 있음을 확인할 수 있었다.

향후 연구과제로 실제 센서 노드에 적용하여 광범위한 필드에서 평가해야 한다. 본 연구의 검증으로 실시한 시뮬레이션 환경에서 발견하지 못한 문제점이 실제 환경에서의 노드 적용했을 때 나타날 가능성이 있기 때문이다. 또한 데이터를 포함한 패킷에 대한 실험을 진행하는 등 다양한 환경에서의 결과를 확인해야 할 것이다.

참 고 문 헌

[1] J.V. Greunen, J. Rabaey, “Lightweight time synchronization for sensor networks,” Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications, pp.11~19, 2004.

[2] S. Ganeriwal, R. Kumar, M. Srivastava, “Timing-sync Protocol for sensor networks”, ACM SenSys 2003. 03.

[3] 김열상, 김현수, 전중남, "무선 센서 네트워크에서 에너지 효율성을 고려한 라우팅 트리 구축 알고리즘," 정보처리학회논문지C, 제16-C권, 제6호, pp.731~736, 2009. 12.

[4] 신재혁, 김영신, 전중남, "단일 플러딩 라우팅 알고리즘을 활용한 센서 네트워크의 시간 동기화 기법", 정보처리학회논문지C 제18-C권, 제1호, 2011. 2.

[5] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Time Synchronization using Reference Broadcasts." Proc. 5th Symp. Op. Sys. Design and Implementation, Boston, MA, Dec., 2002.

[6] Yoon, S., Veerarittiphan, C., and Sichitiu, M. L. 2007. Tiny-Sync: Tight time Synchronization for wireless sensor networks. ACM. Trans. Sens. Netw. 3, 2, Article 8 June, 2007.

[7] J. N. Al-KARAKI and A. E. KAMAL, "Routing Techniques in Wireless Sensor Networks: a Survey," IEEE Commun. Mag., Dec., 2004.

[8] W. Heinzelman, A. Chandrakasan and H. Balakrishnan "Energy-efficient communication protocols for wireless microsensor networks" Proceedings of the Hawaii International Conference on Systems Sciences, Jan., 2000.

신 재 혁

e-mail : naezang@lycos.co.kr

2009년 충북대학교 컴퓨터공학과(학사)

2009년~현 재 충북대학교 컴퓨터과학과 석사과정

관심분야 : 임베디드 시스템, 영상처리, SNS등

오 현 수

e-mail : fodorl@naver.com

2009년 청주대학교 전자공학과(학사)

2009년~현 재 충북대학교 컴퓨터과학과 석사과정

관심분야 : 임베디드 시스템, SoC 설계 등

전 중 남

e-mail : joongnam@cbu.ac.kr
1990년 연세대학교 전자공학과(공학박사)
1996년~1998년 미국 Texas A&M 연구교수
현 재 충북대학교 전자정보대학 교수
관심분야 : 컴퓨터구조, 임베디드시스템