DOI: 10.3745/KIPSTC.2011.18C.1.015

단일 플러딩 라우팅 알고리즘을 활용한 센서 네트워크의 시간 동기화 기법

신 재 혁[†] · 김 영 신[††] · 전 중 남[†††]

요 약

일반적으로 센서 네트워크는 라우팅 트리를 구축한 후에 시간 동기화를 수행한다. 이로 인하여 시간 동기화가 늦어지고 교환하는 패킷이 증가하여 에너지를 많이 소모하는 문제를 유발한다. 본 논문에서는 한 번의 플러딩 과정으로 라우팅 트리를 구축하고 이와 동시에 시간 동기화를 수행하는 TSRA (Time Synchronization Routing Algorithm) 알고리즘을 제안한다. 라우팅 패킷에 패킷 수신 시간과 패킷 전송시간을 추가하여 두 노드간 시간 차이를 구하고, 시간 차이를 전송함으로써 노드들 간의 시간 동기화를 구현한다. 시뮬레이션에 의하여 제안하는 알고리즘은 기존의 동기화 알고리즘인 TPSN과 동등한 수준의 정확도를 보이면서 동기화 속도 및 에너지 소모 면에서 우수하다는 것을 입증하였다.

키워드 : 센서 네트워크, 시간동기화, 라우팅 트리

A Time Synchronization Method of Sensor Network using Single Flooding Algorithm

Jaehyuck Shin[†] · Youngsin Kim[††] · Joongnam Jeon[†††]

ABSTRACT

Usually time synchronization is performed after routing tree is constructed. This thesis proposes a time synchronization algorithm combined with single-flooding routing tree construction algorithm in a single path.

TSRA (Time Synchronization Routing Algorithm) uses routing packets to construct a routing tree. Two types of time information are added to the routing packet: one is the packet receiving time, and the other is the packet sending time. Time offset and transmission time-delay between parent node and child node could be retrieved from the added time information using LTS (Lightweight Time Synchronization) algorithm. Then parent node sends the time offset and transmission time to children nodes and children nodes can synchronize their time to the parent node time along the routing tree.

The performance of proposed algorithm is compared to the TPSN (Timing-sync Protocol for Sensor Networks) which is known to have high accuracy using NS2 simulation tool. The simulation result shows that the accuracy of time synchronization is comparable to TPSN, the synchronization time of all sensor nodes is faster than TPSN, and the energy consumption is less than TPSN.

Keywords : Sensor Network, Time Synchronization, Routing Tree

1. 서 론

무선 센서 네트워크(Wireless Sensor Network)란 다량의 센서들이 무선 방식을 통해 네트워크에 연결되어 있는 것으로써, 컴퓨팅 능력과 무선통신 능력을 갖춘 센서 노드를 자연 환경에 배치하여 자율적인 네트워크를 형성하고, 획득한 정보를 송수신하고, 네트워크를 통해 원격지에서 감시 및 제어 용도로 활용할 수 있는 기술이다.

무선 센서 노드들은 소형 및 경량화에 따라 제한된 컴퓨팅 능력과 전력 자원을 갖으며, 이로 인하여 전력 소비가

※ 이 논문(도서, 작품)은 2009년도 충북대학교 학술연구지원사업의 연구비 지원에 의하여 연구되었음.
† 준 회 원 : 충북대학교 컴퓨터과학과 석사과정
†† 정 회 원 : 네오엠텍 주식회사 과장
††† 종신회원 : 충북대학교 컴퓨터공학부 교수(교신저자)
논문접수 : 2010년 3월 23일
수 정 일 : 1차 2010년 8월 25일
심사완료 : 2010년 10월 17일

적은 라우팅 알고리즘이 필요하며, 또한 각 노드들은 여러 가지 이유로 자신의 로컬 시간을 갖고 있기 때문에 이를 일치시키기 위하여 시간 동기 과정이 필요하다.

기존의 시간 동기화 알고리즘인 TPSN(Time-sync Protocol for Sensor Network)[1]은 시간 동기를 위하여 라우팅 트리를 구축한 후에 시간 동기 프로토콜을 수행하였다. 그리고 한 번의 플러딩 과정만으로 노드들 간의 부모-자식 관계를 형성하여 라우팅 트리를 구축하는 RTAF (Routing-Tree construction Algorithm by single Flooding) 알고리즘[3]도 제안된 바 있다. 본 논문에서는 RTAF의 라우팅 트리 구축 패킷에 라운드 패킷 수신시간과 패킷 전송 시간을 추가하여 한 번에 라우팅 트리 구축과 시간 동기화를 수행하는 알고리즘을 제안한다. 송수신 횟수가 감소하여 노드의 에너지 소비를 줄이는 효과를 기대할 수 있다.

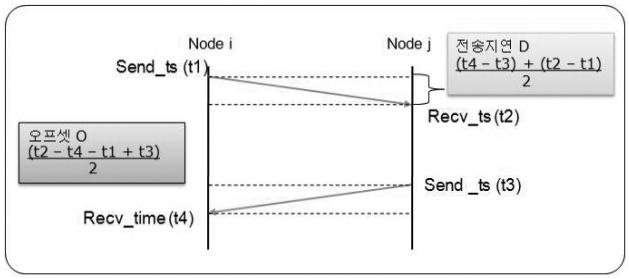
2장에서 관련 연구로 센서 네트워크의 시간 동기화 알고리즘들을 소개하고, 3장에서 본 논문에서 제안하는 시간 동기 라우팅 알고리즘인 TSRA(Time Synchronization Routing Algorithm)을 소개한다. 4장에서는 본 알고리즘을 이용한 라우팅 트리 구축 및 시간 동기화까지 소요되는 시간과 에너지 소모량, 각 노드의 시간차를 TPSN과 비교하고, 마지막으로 5장에서 결론을 제시한다.

2. 관련 연구

일반적으로 센서 네트워크의 라우팅 프로토콜은 평면 기반 라우팅, 위치 기반 라우팅, 그리고 계층 기반 라우팅 프로토콜로 구분할 수 있다[5]. RTAF는 계층 기반 프로토콜에 해당하며, 기본적으로 플러딩 프로토콜에 의존한다. 플러딩 프로토콜[2]을 사용하는 노드가 비콘 패킷을 수신하면 이것을 브로트캐스트하여 모든 이웃한 노드에게 전달하며, 이 과정은 최종 노드에 도달할 때까지 반복된다. RTAF 라우팅 프로토콜은 플러딩 방식으로 라우팅 패킷을 보낼 때, 새로운 신호를 받은 노드들이 주변의 노드에게 신호를 전달하는 정방향 패킷과, 되돌아오는 패킷인 역방향 패킷을 이용하여 부모-자식 간의 관계를 만들어 싱크노드를 중심으로 트리구조를 완성시키는 알고리즘이다. 이 알고리즘은 전방향 패킷을 사용하여 부모노드를 등록하고 역방향 패킷을 자식노드로 등록하여, 한 번의 플러딩 만으로 라우팅 트리를 구축한다[3].

시간 동기화 알고리즘으로는 두 노드간 시간차이를 구하는 LTS[4]알고리즘, 수신 노드들 간의 시간을 동기화 하는 RBS(Reference Broadcast Synchronization) 알고리즘[7], 송신자-수신자간의 시간동기화 방법을 사용하는 TPSN, Tiny-Sync[8] 등이 있다.

LTS 기법은 두 노드 간에 서로의 송수신 타임스탬프를 주고받음으로써 노드간의 오프셋을 측정하여 시간을 동기화하는 방법이다. (그림 1)과 같이 노드 i와 j간에 패킷을 주고



(그림 1) LTS 방식에서의 두 노드간 오프셋과 전송 지연시간

받을 때, 전달 시점인 네 가지의 시점을 이용하여 오프셋과 전송 지연 시간을 구할 수 있다.

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 시간동기화는 다루지 않는다.

TPSN은 LTS기법을 이용하여 무선 센서 네트워크를 동기화하는 기법이다. TPSN은 두 단계로 나누어서 동기화를 진행한다. 첫 번째 단계에서는 네트워크의 각 노드에 레벨을 할당하여 계층적 토폴로지를 형성한다. 이 단계는 단일 플러딩에 의한 라우팅 트리 프로토콜로 대치할 수 있다. 두 번째 단계는 하위 레벨의 노드가 상위 레벨의 노드에 시간 동기를 수행한다. 동기화 단계의 마지막 시점에 모든 노드는 루트 노드의 시간에 동기 되어 네트워크 전체의 동기가 이루어진다. TPSN 기법은 기존 무선 센서 네트워크 시간 동기 기법들에 비해 정확도가 높은 장점을 가진다. 그러나 다수의 노드들이 몰려 있는 경우에 시간 동기화에 참여하는 노드와 패킷수가 많아져서 네트워크에 오버헤드가 발생한다. 이로 인해 패킷 전송을 위한 에너지 소모가 증가하고, 시간동기화가 늦춰질 수 있는 단점이 있다.

Tiny-Sync는 두 노드들 간의 오프셋과 클럭 드리프트를 고려하여 시간동기화를 하는 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 표류율이 정확해질 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.

3. TSRA 시간동기화 라우팅 트리 구축 알고리즘

제안하는 “단일 플러딩 라우팅 알고리즘을 활용한 센서 네트워크의 시간 동기화 기법”은 기존 센서네트워크의 라우팅과, 시간동기화가 별개로 구현되어 발생하는 시간 동기화

과정의 지연 문제와 다수의 패킷 교환으로 인한 에너지 소모 문제를 해결하고자 한다. 본 알고리즘은 기본적으로 단일 플러딩 트리구축 알고리즘에 받은 시간과 보내는 시간 정보를 추가한다. 부모 노드로 선정된 노드는 브로드캐스트를 통하여 자식노드에게 패킷을 보낸다. 그리고, 되돌아오는 역방향 패킷의 시간을 측정하고, 그 시간 정보를 LTS 기법으로 산정한 오프셋 값과 전송 지연시간 등 시간 동기화를 위한 데이터를 추출한다. 이 과정을 되풀이 하여 모든 자식의 오프셋 값과 전송 지연시간을 구할 수 있다. 또한 싱크 노드의 홉부터 끝단에 있는 노드의 홉까지 오프셋을 누적하여 전달함으로써 모든 홉의 노드들이 싱크 노드의 시간에 맞출 수 있다.

3.1 이론적 배경

이 알고리즘은 라운드 패킷을 사용하여 라우팅 트리를 구축함과 동시에 부모-자식 노드 간 시간 차이를 구한다. 라운드 패킷이 하위 노드로 전달되는 것을 정방향 패킷이라 하고, 상위 노드로 전달되는 것을 역방향 패킷이라고 한다. 라운드 패킷을 수신한 하위 노드는 새로운 라운드가 시작된 것을 인지하고, 신호를 보낸 노드를 부모로 등록한다. 하위 노드는 자신의 아이디와 부모의 아이디를 넣어 라운드 신호를 재전송하며, 이 패킷을 수신한 상위 노드는 부모 아이디와 자신의 아이디를 비교하여 같은 경우 보낸 노드를 자식으로 등록하여 부모-자식 간의 관계를 맺는다. 이때 보내는 라우팅 패킷 구조는 (그림 2)와 같고 패킷은 다음과 같은 정보를 갖는다.

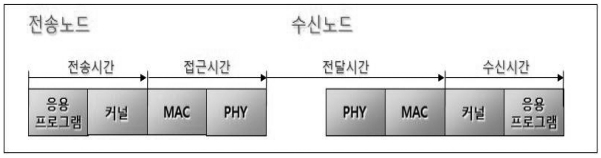
ID (2)	Hop (2)	Round (2)	PID (2)	Recv_ts (8)	Send_ts (8)

(그림 2) Round 패킷의 구조

- ID : 노드의 식별번호
- Hop : 트리에서의 노드 차수
- Round : 라운드 순서번호
- PID : 부모 노드의 식별번호
- Recv_ts : 새로운 라운드 신호를 받은 시간
- Send_ts : 현재 라운드 패킷을 보내는 시간

라운드 패킷의 Recv_ts와 Send_ts를 이용하여 시간차인 오프셋과 전송 지연시간을 구할 수 있다. 오프셋은 각각 두 노드가 가지고 있는 로컬 시간 차이이다. 전송 지연시간은 신호를 보내고 받을 때까지 걸리는 시간으로 전송시간과 접근 시간, 전달시간, 수신시간으로 (그림 3)과 같이 나눌 수 있다.

- 전송시간(Send Time): 응용 프로그램에서 데이터를 전송하기 위해 전송 명령을 수행했을 때, 실제로 전송을 시도하는 과정에서 지연되는 시간으로 커널 프로세싱, 스케줄링, 시스템 콜 등과 같은 현재 시스템 부하에 따라 달라진다.
- 접근시간(Access Time): 생성된 메시지가 RF 전송을 하기 위해 전송 채널로 접근하는 과정에서 기다리는 시간으로, 사용되는 MAC 기법의 종류에 따라 달라진다.
- 전달시간(Propagation Time): RF 신호가 송신 측에서 수신 측에 도착하는데 걸리는 시간으로 두 노드간의 거리에 따라 달라지며, 센서 네트워크에서 빛의 속도로 전송하기 때문에 전송 시간과 접근 시간에 비해 무시해도 될 정도로 작다.
- 수신시간(Receive Time): 수신한 데이터가 어플리케이션에 도달하는데 걸리는 지연 시간이다.



(그림 3) 전송 지연시간

이상의 방법으로 오프셋과 전송 지연 시간 값을 구할 수 있는데, 구한 오프셋과 지연시간 값을 비동기 노드에 더하면 두 노드 간 시간동기화가 이루어진다. 이때 동기화 정보를 담은 패킷은 더블형의 단일 패킷으로 유니캐스트를 통하여 목적지 노드에 전달된다.

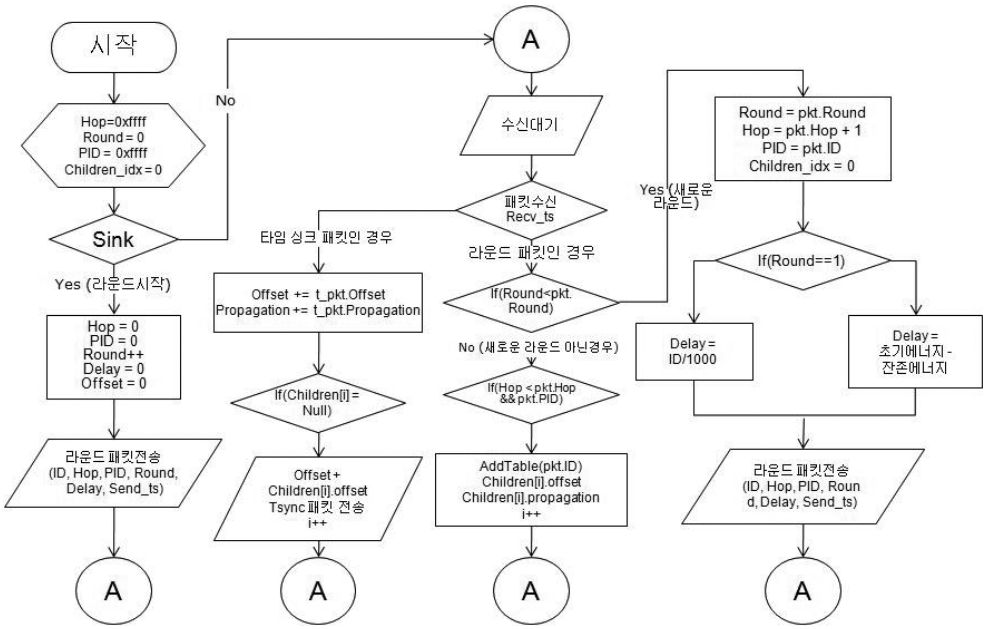
3.2 TSRA 기법

3.2.1 TSRA 구축 알고리즘

본 논문에서 제안하는 TSRA(Time Synchronization Routing Algorithm)의 순서도는 (그림 4)와 같다. 먼저 Hop과 Round, PID, Children_idx를 초기화 한다. Children_idx는 해당 노드의 자식이 증가할 때마다 1씩 증가한다. 싱크 노드는 가장 높은 홉이기 때문에 Hop을 0으로 하고, 부모가 없는 최상위 노드이기 때문에 PID도 0으로 만든다. 현재 새로운 라운드를 시작하므로 Round를 1 증가시키고 기준 노드이기 때문에 Offset 값도 0으로 만든다. 그리고 지연 시간 없이 바로 패킷을 전송한다.

새로운 라운드 패킷을 받은 노드들은 받은 시간을 기록하여 Recv_ts 필드에 넣고, 라운드는 새로운 라운드로 갱신한다. 그리고 라운드 패킷의 ID필드에 있는 아이디를 부모 아이디로 등록하고, 패킷의 홉에 1을 증가 시킨다. 그 후 delay 시간을 첫 라운드에는 자신의 ID를 1000으로 나눈 값을 주고, 첫 라운드가 아닌 경우 초기에너지에서 잔존 에너지를 뺀 수로 지연 시간을 준다. 이렇게 계산한 값을 현재 시간과 더해 보내는 시간, 즉 Send_ts에 넣어 라운드 패킷을 브로드캐스트 한다.

첫 번째 라운드에는 모든 노드의 잔존 에너지가 같기 때문에 서로 다른 시간에 라운드 신호를 보낼 수 있도록 하는

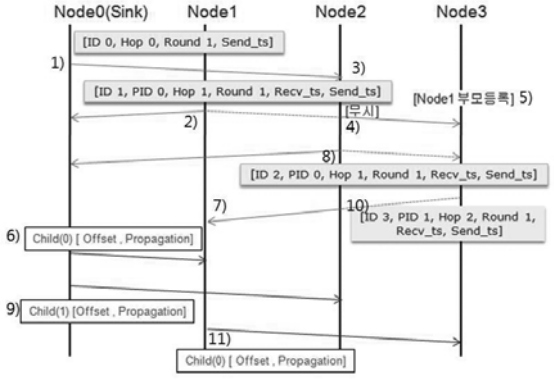


(그림 4) 라우팅 시간동기화 순서도

것이고, 두 번째 라운드부터는 같은 홉에서 많은 잔존 에너지를 가진 노드가 신호를 먼저 보내도록 하여 주위의 노드들이 에너지 잔존량이 많은 노드를 부모 노드로 선택하도록 유도하기 위한 과정이다. 이를 통하여 잔존 에너지 량이 많은 노드를 에너지 소모가 많은 클러스터 헤더로 유도해 노드들의 에너지 소모를 균등하게 하여 전체적인 네트워크의 수명을 향상시킬 수 있다.

3.2.2 TSRA의 패킷교환

노드들의 패킷 교환 과정은 (그림 5)와 같다. 라운드는 첫 번째 라운드라 가정한다. Node 0은 싱크 노드이고 Node 1과 Node 2는 싱크노드의 이웃 노드, Node 3은 Node 1과 Node 2의 이웃 노드이다. 각 노드간 패킷을 교환하는 과정은 다음과 같다.



(그림 5) 라운드 패킷과 시간 동기화 패킷 전송

1) Node 0이 이웃 노드에게 새로운 라운드 신호를 보내게 되면, Node 1과 Node 2는 Node 0에게 받은 라운드 패킷에 받은 시간을 기록하고, 패킷의 ID를 부모 ID로 등록한다.
2) 받은 라운드 패킷에 홉을 1 증가 시키고, 자신의 라운드를 갱신한 후, 보내는 시간을 기록하여 라운드 패킷을 보낸다.
3) Node 2는 Node 1과 비슷한 시간에 Node 0에게 라운드 신호를 받아 자신의 라운드와 홉을 갱신하고 라운드 신호를 만들어 대기하고 있게 되는데, 이때 첫 라운드이기 때문에 새로운 라운드를 받은 후 전송 할 때의 delay 시간은 자신의 ID/1000 이 되므로 Node 1의 라운드 패킷 전송순서가 Node 2보다 빠를 것이다.
4) Node 1의 라운드 패킷을 받은 Node 2는 자신의 라운드와 비교하는데, 자신의 라운드와 같은 라운드이고, 홉도 같은 홉이므로 이 신호를 무시한다.
5) Node 3은 Node 1의 패킷을 받으면 받은 시간을 기록하고, 새로운 라운드로 인식하여, 자신의 라운드를 패킷의 라운드로 갱신하고, 패킷의 홉에 1홉을 더해 자신의 홉으로 만들고, 보낸 Node1을 부모 노드로 등록한다.
6) Node 1의 신호를 받은 Node 0은 받은 시간을 기록하고, 새로운 라운드 신호가 아니기 때문에 Hop과 PID 필드를 검사하게 되는데, 홉이 자신의 것보다 높고, PID도 자신의 ID와 같으므로, 이 패킷을 보낸 ID를 자식테이블에 등록하게 된다. 이때, 이전에 자신이 보냈던 시간을 t1 이라 놓고 Node 1이 보낸 Recv_ts 를 t2, Send_ts 를 t3, 그리

고 Node 1이 보낸 패킷을 Node 0이 받은 시간을 t4로 하여 (그림 6)의 공식으로 Node 1의 오프셋과 전송 지연시간을 구한다.

$$Offset = \frac{(t2-t4)-(t1+3)}{2}$$

$$\mathrm{P}ropagation = \frac{(t4-t1)-(t3-t2)}{2}$$

(그림 6) 오프셋과 전송 지연시간

7) Node 0은 (그림 8)의 공식으로 구한 값을 Child(0) [Offset, Propagation] 에 넣고 자식 테이블의 인덱스를 1 증가 시켜서 다음에 들어오는 자식의 값을 채울 수 있도록 준비한다.
8) 다음으로는 Node 2가 라운드 패킷을 방송하게 되는데 Node 2와 Node 3은 라운드가 같고 패킷의 홉이 자신의 것보다 낮으므로 신호를 무시한다.
9) Node 0은 Node 2가 보낸 신호를 받아 받은 시간을 기록하고 자신의 ID와 패킷의 PID를 비교해 같음을 확인하고, 자식 테이블에 추가한다. 이 때에도 Node 1의 오프셋과 전송 지연 시간을 구할 때와 마찬가지로 값을 구하고 Child(1)[Offset, Propagation] 에 추가하고 자식 인덱스를 증가시킨다.
10) Node 3이 라운드 패킷을 전송하게 되는데 Node 2는 Node 3이 보낸 라운드 패킷의 홉이 자신보다 높은 것을 확인하고, 자신의 ID와 패킷의 PID 필드를 비교한다. 하지만 같지 않기 때문에 패킷을 버린다.
11) Node 1은 Node 3이 보낸 신호를 받아 자신의 ID와 부모의 PID가 같음을 확인하고 Node 3을 자식으로 등록하고 시간차와 전송 지연시간을 구해 자식 테이블을 만든다.
12) 일정 시간이 지나면 Node 0부터 자식 테이블에 등록된 노드에게 시간차와 전송 지연 시간을 유니캐스트로 전송한다. 이때 자식에게 전달되는 시간차와 전송 지연 시간은, 자신의 시간차와 전송지연 값에 자식의 시간차와 전송지연 값을 더해 전달하게 된다.

위의 방식으로 자식이 있는 노드는 자신의 오프셋 값을 자식의 오프셋에 더해 전달하고 이를 받은 노드는 자신의 시간에 오프셋과 전송 지연시간 값으로 자신의 시간을 수정한다. 이상의 과정을 마치게 되면 노드들이 트리 구조를 만듦과 동시에 구성된 모든 노드들은 싱크 노드의 시간에 맞출 수 있게 된다.

4. 시뮬레이션 및 성능 분석

4.1 시뮬레이션 환경

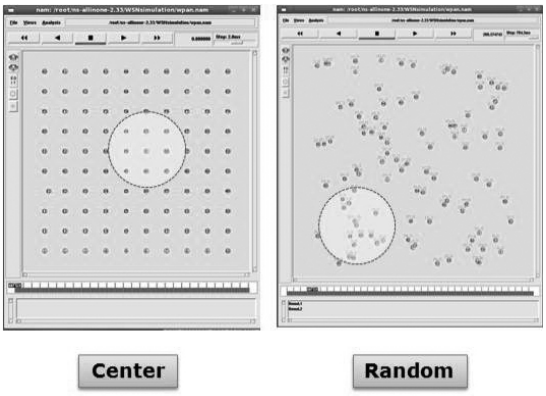
시뮬레이션 환경은 <표 1>과 같다. NS2는 OTCL 스크립트와 C++ 언어를 사용한다. OTCL은 NS2를 위해 만들어진 스크립트 언어로 TCL의 오브젝트 기반 프로그래밍까지 지원할 수 있도록 기능을 확장한 스크립트 언어이다. TCL (Tool Command Language)은 버클리 대학에서 개발한 스크립트 언어로 기능이 유사한 어플리케이션들이 반복되는 문제점을 해결하기 위하여 처음 고안되었다. 이 스크립트 언어는 변수나 제어 구조 등과 같은 프로그래밍 툴인 동시에 C 라이브러리 인터페이스를 통한 언어의 확장성 측면에서 매우 편리하다. 실험 결과는 NS2 시뮬레이션 결과인 트레이스(trace) 파일을 분석하여 정리하였으며, NAM (Network Animator)을 통해서 시뮬레이션 결과를 눈으로 확인할 수 있다.

〈표 1〉 시뮬레이션 환경

시뮬레이션 환경	사용 도구
사용 OS	Fedora core 8
시뮬레이션 툴	Network Simulater 2.33
사용 언어	C++, OTCL

시뮬레이션 환경 설정은 (그림 7)에 보이는 것처럼 격자형(grid)에 10m 간격으로 100개를 배치하였다. 싱크 노드의 위치는 중앙에 배치하였고 다른 노드들을 랜덤하게 배치하여서 실험하였다. (그림 7)에서 점선으로 된 원의 중앙에 싱크노드가 위치한다.

한 라운드는 200초 간격으로 노드의 구조가 새로 구성되며 노드마다 상이한 시간차를 갖게 하기 위해 싱크노드를 제외한 모든 노드들의 초기 시간차를 -10초에서 +10초까지 랜덤으로 주었다. 라운드마다 노드들의 시간이 틀어지게 하기 위해 클럭 드리프트를 1초당 -0.001부터 0.001초까지 랜덤하게 설정하였다. 라운드는 총 10 라운드로 2,000초까지의 시간 동안의 라운드별 시간 오류와 시간 동기화 구축까지의 소모시간을 측정하였다.



(그림 7) 노드의 위치

실험환경 설정은 <표 2>와 같다. Zigbee(802.15.4) 환경으로 설정하였으며, 전력 소비량 기준은 버클리 모트 전력 소모량 측정표[6]를 참고하였다. 제시된 에너지 소비에 대한 기준 모트는 Mica2이고 무선 통신 유형은 CC1000을 기준으로 환경 변수를 설정 하였다.

실험은 크게 TPSN과 TSRA 두 종류로 구분되며 각 알고리즘마다 격자형과 랜덤형으로 100개의 노드를 분포하여 진행된다. 또한 각 경우마다 클럭 드리프트가 없다고 가정했을 때와 -0.001초부터 0.001초 사이의 랜덤 값을 주었을 때, 총 8가지의 경우로 진행하였다. 실험 결과는 시뮬레이션을 하는 동안 내장된 분석기를 통하여 라운드, 노드의 ID, 부모ID, 홉, 자식노드 수, 초기 오류시간, 클럭 드리프트, 부모 노드와의 시간 차이인 오프셋과 전송 지연시간, 동기화가 완료된 시점의 시간과 잔존 에너지양을 산출하여 텍스트 파일 형태로 출력하였다.

〈표 2〉 환경 설정

NS2의 실험 환경 변수	
노드의 배치 형태 / 노드수	격자형 / 100, 랜덤형 / 100
Mote	MICA2 - ATmega128
Channel Type	Channel/Wireless Channel
Network Interface	Phy/WirelessPhy/802.15.4
Mac Protocol	Mac/802.15.4
Energy Consumption rate	rxPower 0.3 W
	txPower 0.4 W
Offset	-10 ~ 10 sec (Random)
Node clock drift	드리프트가 없는 경우 & -1 ~ 1 ms/sec (Random)

실험을 통해서 얻고자 하는 성능 지표들은 다음과 같다.

4.1.1 라운드별 초기 오류의 분포

라운드가 진행될 때마다 TSRA 알고리즘을 통해 노드의 구조를 결정하고, 시간을 동기화 하는 과정을 거친다. 시간 동기화는 초기 오류시간을 수정하는 것으로 진행되는데, 시간 동기화가 이루어지고 있는지를 확인하기 위해 한 라운드가 끝나는 시점에서의 오류 시간으로 시간 동기화가 잘 이루어지고 있는지 검증한다.

4.1.2 홉 수에 따른 시간동기화 소요시간

홉 수가 증가함에 따라 시간 동기화 구축이 완료된 시간을 나타내는 값이다. 싱크노드와의 거리에 따른 시간 동기화 완료 시간을 측정하면, 네트워크의 전체적인 시간 동기화 완료 시간을 예측할 수 있다.

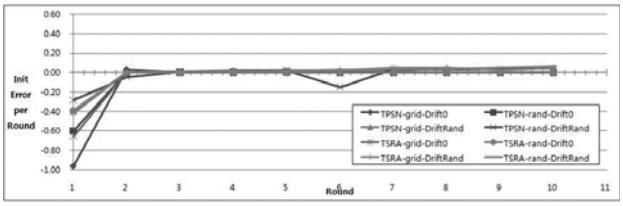
4.1.3 라운드별 에너지 소모량

시간 동기화가 완료되는 시점의 잔존 에너지를 측정하여 본 알고리즘의 에너지 효율을 검증한다.

4.2 실험 결과 분석

4.2.1 라운드별 초기 오류의 분포

(그림 8)은 비교 대상인 두 알고리즘의 라운드가 시작할 때의 초기 오류 분포이다. 이 오차 값은 100개의 노드에 대한 평균값이며 라운드를 진행하면서 시간 동기화 알고리즘에 의해 수정된다.



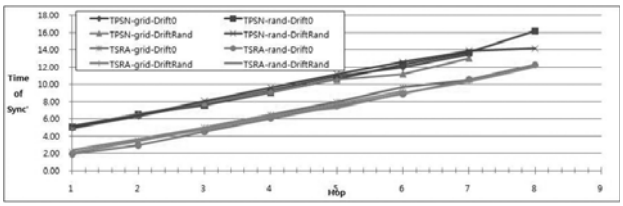
(그림 8) 라운드별 초기 오류 분포

TPSN과 TSRA 알고리즘들은 시간이 동기화된 후 라운드가 시작할 때의 초기 오류는 0에 가까워지며, 이것은 모든 노드가 싱크노드의 시간에 동기화 되고 있다는 것을 의미한다. 클럭 드리프트 값이 랜덤하게 생성될 때는 TPSN과 TSRA 모두 초기 오류 시간의 정확도에 약간의 오차가 발생한다. 결과적으로 두 알고리즘 모두 LTS를 기반으로 시간 동기화를 수행하므로 시간 오차의 관점에서는 큰 차이가 없다.

6 라운드의 경우 TPSN 방식의 랜덤 노드 분포와 랜덤 드리프트 값의 경우 오차율이 커진 이유는 패킷 충돌이 많이 발생하여 동기화 동작을 수행하지 못하고 이전 라운드부터 클럭 드리프트에 의한 오류가 누적되었기 때문이다. 하지만 다음 라운드에서 시간 동기화가 수행된 후 오차가 수정되는 것을 볼 수 있다. 실제로 시간 동기화 완료 후 오차는 클럭 드리프트가 0인 경우 0.000X 수준이고, 클럭 드리프트를 준 경우는 0.00X 정도의 오차를 보였다.

4.2.2 홉 수에 따른 시간동기화 소요시간

노드가 시간 동기화를 완료하는 시점을 기록하여 TPSN과 TSRA의 시간 동기화 소요시간을 측정할 수 있으며, 그 결과는 (그림 9)이다.



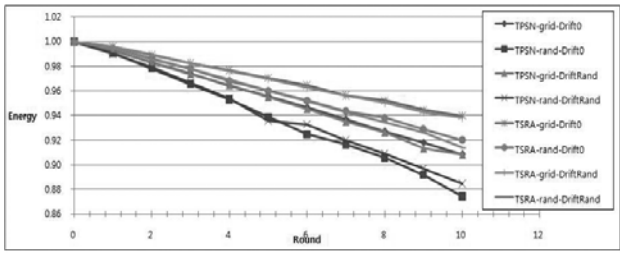
(그림 9) 홉 수에 따른 시간 동기화 소요시간

두 알고리즘 모두 홉에 비례하는 동기 시간을 보여주는데, 이것은 부모 노드가 동기화 된 후 자식 노드들을 동기시키기 때문이다. 실험 결과 TSRA 알고리즘의 경우 TPSN

알고리즘 보다 약 1.5초 정도 빠르게 동기화 되는 것을 알 수 있었다. 이는 TPSN의 경우 라우팅 단계에서 1개, 시간 동기화 단계에서 3개 총 4개의 패킷을 사용하고, TSRA는 라우팅과 시간 동기화를 통합 2개의 패킷으로 진행이 되기 때문이다.

4.2.3 라운드별 에너지 소모

라운드별 에너지 소모량에 대한 결과는 (그림 10)에 나타내었다. TSPN 알고리즘을 사용한 시간 동기화 후의 에너지 소모량 중 가장 적게 소모하는, '클럭 드리프트가 없는 격자형'에서의 결과가 TSRA 알고리즘을 '랜덤한 노드분포와 클럭 드리프트'를 가진 환경에서의 에너지 소모량보다 많은 것을 확인할 수 있었다. 두 알고리즘의 수행에 소요되는 패킷 수에 차이에 의하여 이러한 결과가 나타난다. TPSN의 에너지 소모가 TSRA 기법 보다 상대적으로 크다는 것을 보여주며 임의의 노드 분포에서 TSRA 기법이 에너지 효율이 좋다는 것을 보여준다.



(그림 10) 라운드별 에너지 소모량

5. 결 론

무선 센서네트워크의 라우팅과, 시간동기화는 데이터의 신뢰성을 부여하기 위한 가장 기초적인 사전 작업이라 할 수 있다. 이전의 동기화 알고리즘들은 네트워크의 구조를 만들기 위한 과정과 시간동기화가 따로 구현이 되어 있어서 실제 데이터를 측정하고 보낼 수 있는 상태가 되기까지 다수의 패킷 신호와 절차가 필요하였다.

본 연구는 "단일 플러딩에 의한 센서 노드들의 라우팅 트리 구축 알고리즘"과 LTS 기법을 응용하여 단일 플러딩 방법으로 시간 동기화까지 이룰 수 있게 하였다. 단일 플러딩 라우팅 패킷에 시간정보를 기록하게 하여 단일 라운드 패킷만으로 센서 네트워크의 구조를 설정하고 시간동기화를 위한 정보를 추출해 낼 수 있도록 하였고, LTS기법을 사용하여 시간동기화를 함으로써 센서네트워크 노드들의 시간 오차를 상당부분 줄일 수 있었다. 단일 라우팅을 통해 패킷교환을 줄이고 네트워크 구축을 짧은 시간 내에 마칠 수 있었으며, 특정 노드에게 에너지 소모가 이루어지는 경우를 방지하여 네트워크의 전체적인 수명을 늘릴 수 있게 하였다. 또한 TPSN과의 비교 시뮬레이션으로 단일 플러딩을 이용한 시간 동기화 방법의 유리함을 확인하였다.

향후 연구과제로 네트워크 필드의 크기와 노드의 수가 더욱 많아 질 때, 라운드 라우팅 패킷의 충돌문제가 얼마나 빈번해 지는지, 또 구축 시간이 얼마나 늘어나는지에 대한 연구가 필요하다. 그리고 현재는 시뮬레이션을 통해 누적시간 오류와 구축 시간을 측정하였으나, 실제 센서 노드에 적용하여 측정하지 못하였으므로 실제 센서노드 환경을 구성하여 평가해야 할 것이다.

참 고 문 헌

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

[2] Jamal N. Al-Karaki, Ahmed E. Kamal, "Routing techniques in wireless sensor networks: a survey," IEEE Wireless Communications, pp.6-28, December 2004.

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

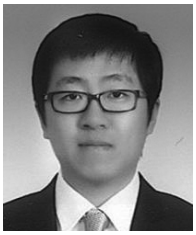
[4] J.V. Greunen, J. Rabaey, "Lightweight time synchronization for sensor networks," Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications, pp.11-19, 2004.

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

[6] 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.

[7] 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.

[8] 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.



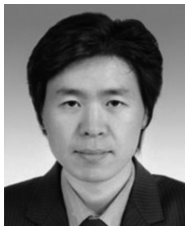
신 재 혁

e-mail : naezang@lycos.co.kr

2009년 충북대학교 컴퓨터공학과(학사)

2009년~현 재 충북대학교 컴퓨터과학과 석사과정

관심분야 : 임베디드 시스템, 영상처리, SNS등



김 영 신

e-mail : kys516@empal.com
2003년 충북대학교 정보통신공학과(학사)
2009년 충북대학교 전자계산학과(공학석사)
2009년~현 재 네오엠텍 주식회사 과장
관심분야 : 임베디드 시스템, 센서네트워크 등

전 중 남

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