센서 네트워크에서의 누적 시간 정보를 이용한 시간 동기화

노진홍° 홍영식
동국대학교 컴퓨터공학과
{jhno°, hongys}@dongguk.edu

Time Synchronization using the Accumulative Time Information in Wireless Sensor Networks

Jinhong No° Youngsik Hong
Department of Computer Engineering, Dongguk Universigy

요 약

최근 무선 통신의 발달과 임베디드 시스템의 확산으로 주변 상황을 감지하고 통신할 수 있는 저전력 장치인 무선 센서 네트워크에 대한 연구가 많이 진행되고 있다. 무선 센서 네트워크를 구성하는 노드들 간의 동기화, 순서화, 그리고 일관성 유지를 위하여 시간 동기화는 반드시 필요하다. 하지만 지난 20여 년간 연구되어진 분산 시스템에서의 시간 동기화 방법들은 풍부하지 않은 자원과 에너지 소모 등을 고려해야 하는 무선 센서 네트워크의 특성상 그대로 적용하기에는 어려운 점이 많다. 이에 본 논문에서는 무선 센서 네트워크에서의 누적 시간 정보를 이용한 시간 동기화 방법을 제안한다. 무선 브로드캐스트의 특성을 활용하여 시간 동기화의 제약 조건을 완화하여 높은 정확성을 제공하면서 한 번의 브로드캐스트 메시지만으로 시간을 동기화함으로써 에너지 손실을 최소화하였다. 이를 위해 송신자-수신자 방식의 구조에서 송신자는 시간 동기화를 위한 시간 정보를 브로드캐스트하고, 수신자는 누적된 시간 정보를 통해 편차와 편차율을 계산하여 송신자의 시간을 추정하고 동기화된 시간인 가상 시간을 계산하였다. 전체 센서노드들의 시간이 동기화할 수 있음을 분석하고 실험을 통해 효율성을 확인하였다.

1. 서 론

최근 무선 센서 네트워크(Wireless Sensor Networks, WSN)에 대한 연구가 활발히 이루어지고 있다. WSN은 주변 상황을 감지하고 공동으로 감지된 데이터를 처리하여 전달하는 대량의 센서 노드(sensor node)들로 구성되고, 이를 활용할 수 있는 응용 분야는 접근할 수 없는 동물을 관찰하는 분야에서 침입 감지나 장치를 모니터링하는 분야까지 아주 다양하다. 대부분의 이러한 응용 프로그램들은 이벤트가 발생한 시간을 결정하기 위해 센서 노드들이 자체 시간을 유지하며, 정보를 감지하고 의미 있는 정보를 만들기 위해 다양한 신호 처리 기법을 사용한다. 이러한 신호 처리 기법들은 이벤트들 사이의 올바른 순서를 결정해야 하므로 센서 노드들 간의 시간은 동기화가 필요하다.

하지만 WSN에서의 시간 동기화는 다음과 같은 다양한 제약 조건을 가지고 있다[1]. 첫 번째로 WSN의 노드들은 제한된 배터리와 대역폭을 가지는 등 풍부하지 못한 자원을 가지고 있으므로 시간 동기화 알고리즘들은 패킷 전송을 적게 하는 등의 경량화된 방법(lightweighted method)을 사용해야 한다. 다음으로 무선 브로드캐스트는 패킷 손실뿐만 아니라 패킷 충돌을 발생시켜 패킷들의 전달 시간이 지연될 수 있다. 이로 인해 시간 동기화의 정확성을 떨어뜨릴 수 있으므로 이를 해결해야한다. 세 번째로 센서 노드들은 기존의 분산 환경에서 사용하는 컴퓨터들에 비해 값싼 크리스탈(crystal)을 사용하므로 시간을 측정하는 것이 부정확하다. 그러므로 시간 편차가 부정확한 점을 감안하여 시간을 동기화해야 한다. 마지막으로 다양한 응용 프로그램들이 서로 다른 시간 동기화 정밀도를 요구하므로 이러한 요구들을 조절하기 위해 유연성 있는 시간 동기화를 제공할 필요가 있다.

이에 본 논문에서는 무선 센서 네트워크에서 사용할 수 있는 오버헤드가 적으면서도 높은 정밀도를 제공하는 시간 동기화 알고리즘을 제안한다. 제안된 알고리즘은 메시지의 효율적인 전송을 위해 계층구조를 생성하는 레벨 탐색 단계(level-discovery phase)와 누적 시간 정보와 편차율(drift rate)을 측정하는 선 동기화 단계(pre-synchronization phase), 그리고 가상 시간(virtual time)을 측정하여 시간을 동기화하는 시간 동기화 단계(time synchronization phase)의 세 단계로 구성된다. 레벨 탐색 단계와 선 동기화 단계는 초기에 한번만 실행되며 이후에는 동기화 주기마다 시간 동기화 단계만이 실행된

다. 제안된 알고리즘은 에러가 감쇠된 누적 시간 정보와 MAC 계층(media access control layer)에서의 타임스탬핑(timestamping)을 사용하여 시간 정확성을 높였다. 또한 한 번의 브로드캐스트만으로 이웃 노드들에게 시간 동기화에 필요한 정보를 제공하고 계층구조를 사용하여 메시지 전송을 최소화함으로써 에너지 효율적이다. 실험을 통해 1μs 이하의 평균 정확성을 가지고 있음을 알 수 있었다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 센서 네트워크에서의 시간 동기화와 관련된 연구들을 설명하고, 3장에서는 본 논문에서 제안하는 시간 동기화 알고리즘에 대하여 설명한다. 4장에서는 제안된 방법을 실제 구현하여 실험한 결과를 통하여 성능을 평가하고 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

최근에 무선 센서 네트워크에서 사용할 수 있는 많은 시간 동기화 알고리즘들[2, 3, 4, 5]이 개발되었다. 모든 시간 동기화 알고리즘들은 동기화를 위해 노드들 간에 시간 정보를 교환한다. 하지만 네트워크를 통한 메시지 송수신시 발생하는 여러 가지 비결정적(non-deterministic) 요소들은 시간 정보를 부정확하게 하므로 시간 동기화의 정확성을 떨어뜨린다. 이를 해결하기 위해 기존 알고리즘들은 비결정적 요소들로 발생하는 동기화 에러를 추정하고 보정하여 시간 정보를 계산하는 방법에서 차이가 있다고 볼 수 있다. [6]에서는 시간 동기화 방법에 에러를 발생 시키는 네 가지 기본 구성요소를 다음과 같이 정리하였다.

·송신 시간(send time) : 송신자가 메시지를 생성하는데 소요되는 시간으로서 운영체제에서 발생하는 지연과 네트워크 인터페이스(network interface)에 메시지를 전달하는 시간이 포함된다.

·접근 시간(access time) : 생성된 메시지가 전송을 위해 전송 채널로 접근하기 위해 기다리는 시간으로서 어떤 MAC 기법을 사용하느냐에 따라 다르다.

·전파 시간(propagation time) : 패킷이 송신자를 떠나 수신자의 네트워크 인터페이스까지 전송되는데 필요한 시간으로 빛의 속도로 전파되는 무선 네트워크에서는 아주 적은 시간이다.

·수신 시간(receive time) : 수신자의 네트워크 인터페이스가 메시지를 수신한 후 호스트에게 메시지 도착을 통지하는데 소요되는 시간이다. 이는 일반적으로 네트워크 인터페이스가 메시지 수신 신호를 생성하는데 필요한 시간이다.

RBS(Reference Broadcast Synchronization) 알고리즘[2]은 수신자-수신자 방법을 사용하여 송신시간과 접근시간 에러를 없애는 방법이다. 그림 1과 같이 일반적으로 무선 환경에서의 수신자-수신자 방법은 동기화 에러를 발생시키는 요인으로 수신 시간만을 고려하면 되므로 높은 정확성을 얻을 수 있다. RBS에서는 한 노드가 주기적으로 무선 비콘(wireless beacon) 메시지를 이웃노드에게 브로드캐스트하고, 이 메시지를 받은 이웃 노드들은 받은 시간을 참고 시간(reference point)으로 사용한다. 무선 브로드캐스트의 특성상 이웃 노드들은 거의 동시에 메시지를 수신한다는 점을 활용하여 이웃 노드들은 참고 시간을 서로 교환하여 비교하며 선형 회귀 함수를 통해 편차율을 계산하고 시간을 동기화한다. 하지만 이 방법은 이웃 노드들 간의 교환되는 브로드캐스트와 선형 회귀 함수와 같은 복잡한 계산이 오버헤드를 발생시킨다는 단점이 있다.

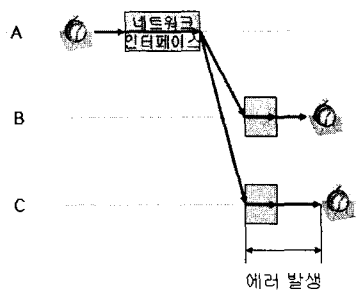

그림 1 무선 브로드캐스트의 특성

TPSN(Timing-sync Protocol for Sensor Networks)[3]은 전형적인 송신자-수신자 방법을 사용하여 시간을 동기화하는 알고리즘이다. 이는 계층 구조를 생성하기 위해 센서 노드들의 레벨을 결정하는 레벨 탐색 단계와 SNTP(Simple Network Time Protocol) [7]과 유사한 방법으로 시간을 동기화하는 시간 동기화 단계로 나누어진다. 시간 동기화 단계에서는 그림 2와 같이 기본적인 두 방향 메시지 교환(two-way message exchange) 방법을 통해 전파 지연시간과 시간 편차를 계산하여 시간을 보정하는 방법을 사용한다. MAC 계층에서 시간을 측정하므로 시간 정보는 RBS 방법보다 정확하지만 편차율을 계산하지 않기 때문에 오랜 동기화 주기를 가지는 경우에는 정확성이 떨어질 수 있다는 단점을 가지고 있다.

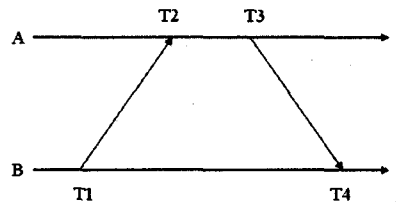

그림 2. 두 방향 메시지 교환

3. 누적 시간 정보를 이용한 시간 동기화 기법

본 논문에서 제안하는 시간 동기화 기법은 레벨 탐색 단계, 선 동기화 단계 그리고 시간 동기화 단계의 세 단계로 동작한다.

레벨 탐색 단계는 초기에 한 번만 실행되며 TPSN과 유사하게 동작하지만 차이점은 부모 노드뿐만 아니라 자식 노드들도 모두 결정하게 된다는 점이다. 초기에 루트 노드는 레벨을 0, 부모 노드 ID를 0으로 설정하여 레벨 탐색 패킷을 전송한다. 레벨 탐색 패킷은 송신 노드의 ID, 부모 노드의 ID, 자신의 레벨 그리고 보내는 시간으로 구성된다. 이웃 노드들은 이 패킷을 수신하여 레벨을 1 증가시켜 자신의 레벨로 설정하고, 송신 노드를 부모 노드로 설정한 후 레벨 탐색 패킷을 다시 브로드캐스트한다. 부모노드의 ID가 자신의 ID와 같은 패킷을 수신하였다면 패킷을 송신한 노드를 자식 노드로 설정한다. 이 작업은 네트워크의 전체 노드가 레벨을 설정할 때까지 계속되며 그림 3은 레벨 탐색 단계를 나타낸다.

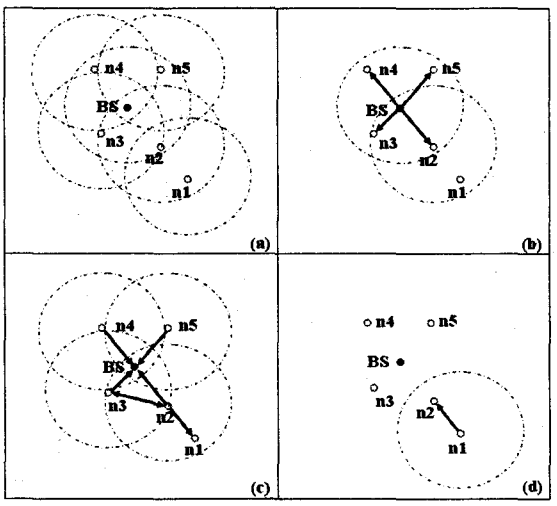

그림 3. 레벨 탐색 단계: (a) 초기 상태 (b) 루트 노드 브로드캐스트 (c) 이웃노드들 브로드캐스트 (d) 반복 작업

레벨 탐색 단계를 통해 계층적 구조가 생성되었다면 편차율을 계산하기 위해 선 동기화 단계가 실행된다. 루트 노드부터 송신 노드의 ID, 자식 노드의 ID 그리고 보내는 시간으로 구성된 선 동기화 패킷을 브로드캐스트한다. 이웃 노드들은 이 패킷을 수신하여 편차율을 측정하고 자식 노드가 있는 노드들은 새로운 선 동기화 패킷을 브로드캐스트한다. 자신의 ID와 자식 노드의 ID가 같았다면 부모 노드에게 자신의 ID, 측정된 편차율, 패킷에 포함된 시간과 전송 시간을 포함한 응답 패킷을 전송한다. 응답 패킷을 받은 부모 노드는 오프셋과 가상 시간을 측정한다. 편차율과 오프셋, 가상 시간을 측정하는 방법은 이후에 설명하기로 한다.

시간 동기화 단계는 시간 동기화 주기마다 이전에 계산된 가상 시간을 자식 노드들에게 알려주는 단계이다. 선 동기화단계와 유사하게 동작하며 루트 노드부터 선 동기화 패킷의 내용에 가상시간이 추가되어 브로드캐스트한다. 이 패킷을 수신한 이웃 노드들은 가상 시간을 통해 오프셋과 이전 단계에서 계산된 편차율을 사용하여 시간을 동기화하고 다음에 사용될 편차율을 계산한다. 자신이 부모 노드라면 새로운 시간 동기화 패킷을 전송하고, 자신의 ID와 자식 노드의 ID가 같았다면 선 동기화 응답 패킷의 내용과 같은 패킷을 부모 노드에게 전송한다. 응답 패킷을 받은 부모 노드는 오프셋과 가상 시간을 측정한다. 그림 4는 선 동기화 단계와 시간 동기화 단계가 동작하는 방법을 나타낸다.

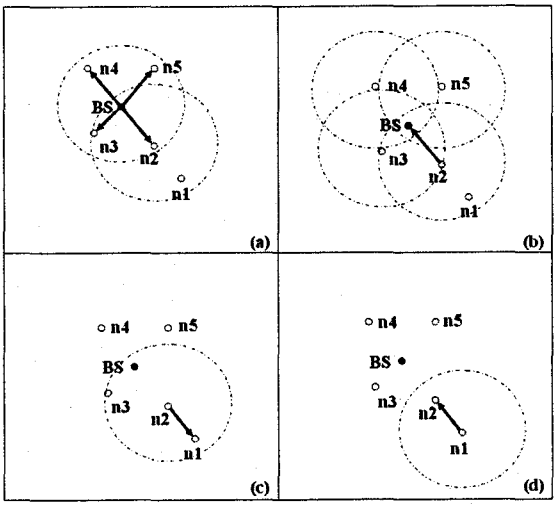

그림 4. 동기화 단계: (a) 루트 노드 브로드캐스트 (b) 이웃 노드 응답 (c) 부모 노드 브로드캐스트 (d) 반복 작업

편차율과 오프셋을 계산하는 방법을 설명하기에 앞서 누적 시간 정보를 계산하는 방법을 설명한다. 그림 5에서 A 노드는 T_n 시간에 브로드캐스트를 하고, 이웃 노드인 B가 t_n 시간에 메시지를 받은 후 t_{n+1} 시간에 응답 메시지를 전송하고, A 노드가 T_{n+1} 시간에 메시지를 받는 작업을 반복하는 상황을 고려해보자. 여기서 D_n은 (T_{n+1} - T_n) 으로 브로드캐스트 사이의 시간이고, d_n은 (t_{n+1} - t_n) 으로 브로드캐스트 수신 사이의 시간이다. 즉, D_n은 A의 시간으로 본 동기화 주기이며 d_n은 그에 해당하는 B의 시간이다.

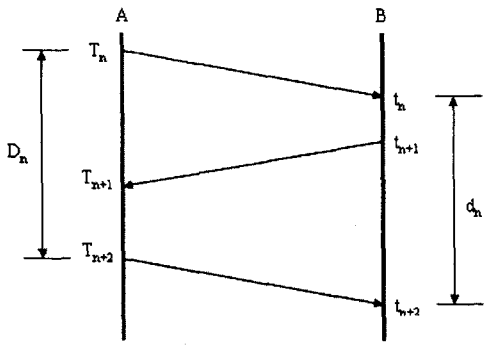

그림 5. 메시지 교환 방법

A_n을 D_n과 d_n의 시간 차이라고 하면 다음과 같은 등식이 성립된다.

$$(T_{n+2}-T_n)-(t_{n+2}-t_n)=D_n-d_n=A_n \quad (1)$$

A_n은 수식 1에서처럼 메시지 교환시마다 D_n과 d_n만을 계산하여 얻을 수도 있지만 다음과 같이 누적된 값을 통해서도 계산할 수도 있다.

$$A_n=(D_n+d_n)\times\frac{\sum_{i=1}^{n}(D_i-d_i)}{\sum_{i=1}^{n}(D_i+d_i)} \quad (2)$$

식 (2)를 이용하여 계산된 A_n은 시간 동기화 주기가 정확히 지켜지지 않아 D_n과 d_n이 불규칙한 경우에도 (Dn+d1)이 큰 값이므로 계산된 값에 크게 영향을 미치지 못하여 A_n은 일정한 값을 유지할 수 있다는 장점이 있다. 이는 4장에서 실험을 통하여 알아보도록 한다.

노드 A의 시간이 T일 때 노드 B의 시간을 t라고 한다면 편차율 a와 오프셋 b를 통하여 다음과 같이 노드 B의 시간을 노드 A의 시간으로 변환할 수 있다.

$$T=at+b \quad (3)$$

식 (1)과 식 (3)을 사용하여 편차율 a를 구하면 다음과 같다.

$$a=\frac{T_{n+2}-T_n}{T_{n+2}-T_n-A_n}=\frac{D_n}{D_n-A_n} \quad (4)$$

편차율을 구했다면 식 (3)을 통해 오프셋을 구할 수 있다.

$$b=\frac{(T_n+T_{n+1})-a(t_n+t_{n+1})}{2} \quad (5)$$

누적 시간정보인 A_n과 편차율 a는 매 브로드캐스트 메시지를 수신할 때마다 계산되며 오프셋 b는 응답 메시지를 수신할 때마다 계산되는 정보이다. 송신자는 수신자의 편차율과 수신 시간을 응답 메시지로 수신하여 오프셋을 측정한다. 수신자의 메시지 수신 시간, 편차율 그리고 오프셋을 사용하면 수신자의 시간을 송신자의 시간으로 변환하고 이를 가상 시간이라고 한다. 무선 브로드캐스트의 특성상 브로드캐스트 메시지는 모두 거의 동시에 받은 시간이므로 가상 시간은 주변 노드들에게 동기화 시점을 제공할 수 있다.

4. 구현 및 성능 평가

제안된 시간 동기화 방법을 TPSN과 비교 실험하기 위해 TinyOS 운영체제[8]를 사용하는 MICAz[9] 상에서 프로토타입을 구현하였다. NesC[10] 언어로 시간 동기화 알고리즘을 구현하고 커널을 일부 수정하였다. 관련 연구에서 언급했던 송신 시간과 접근 시간 에러를 감소시키기 위해 패킷 전송시에는 MAC 계층에서 패킷이 네트워크 카드에서 전송되기 바로 전에 시간을 측정하고, 수신시에는 네트워크 카드에 패킷이 수신되기 시작할 때 시간을 측정하여 수신 시간 에러를 감소하도록 커널을 수정하였다. 다음은 실험 환경과 파라미터이다.

표 1. 실험 환경 및 파라미터

파라미터	값
모트	MPR2400 MICAz
인터페이스 보드	MBH 500CA
운영체제	TinyOS 1.1.14
사용 언어	NesC 1.1.2
노드 수	3개
동기화 주기	10초
총 실험 시간	10분

제안된 시간 동기화 방법과 TPSN의 성능을 비교하기 전에 앞 장에서 설명한 A_n을 메시지 수신시의 정보만을 이용해 계산하는 식 (1)의 방법과 누적 시간 정보를 사용한 식 (2)의 방법을 비교하였고 그림 6은 이를 나타내고 있다.

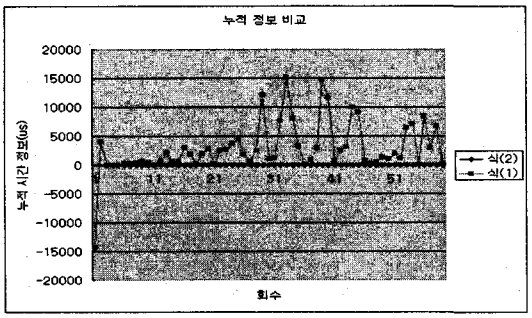

그림 6. 누적 시간 정보 비교

그림에서 보듯이 누적 시간 정보를 사용한 식 (2)를 통해 계산된 A_n이 더 안정적인 것을 알 수 있다. 이는 부정확한 타이머로 인해 동기화 주기인 D_n이 변하는 경우에도 일정하게 A_n을 계산할 수 있음을 의미한다. 일정한 A_n을 사용하면 긴급한 데이터를 전송하거나 패킷 충돌 등으로 인해 시간 동기화가 늦어지는 경우에도 영향을 크게 받지 않는다.

그림 7은 TPSN과 본 논문에서 제안된 시간 동기화 알고리즘의 성능을 보여주고 있다.

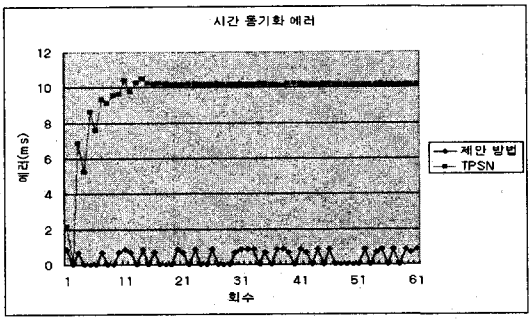

그림 7. 성능 평가

그림에서 보듯이 TPSN은 동기화 회수가 15회 정도가 되어야 성능이 일정한 반면 제안된 논문은 처음부터 일정한 성능을 보여주고 있다. 일정한 A_n을 사용함으로써 편차율과 오프셋 역시 일정하게 유추되므로 동기화시 발생하는 에러가 크지 않다는 것을 알 수 있다.

다음은 제안된 논문과 TPSN의 시간 동기화 성능을 분석한 표이다.

표 2. 시간 동기화 에러의 평균과 분산

기 준	제안된 방법	TPSN
평 균	0.40	9.63
분 산	0.33	6.43
최 대	0.86	10.50
최 소	0	0.18

표 2에서 보듯이 본 논문에서 제안된 알고리즘이 TPSN에 비해 약 20배의 성능이 좋아진 것을 알 수 있다. TPSN은 오랜 동기화 주기를 가지는 경우에는 정확성이 떨어질 수 있으므로 동기화 주기가 실험보다 큰 경우는 성능이 더욱 좋아질 것으로 예측된다.

5. 결론

본 논문에서는 무선 센서 네트워크에서 사용할 수 있는 오버헤드가 적으면서도 높은 정밀도를 제공하는 시간 동기화 알고리즘을 제안하였다. 제안된 알고리즘은 레벨 탐색 단계, 선 동기화 단계 그리고 시간 동기화 단계의 세 단계로 구성되어 동작한다. 그리고 한 번의 브로드캐스트만으로 이웃 노드들에게 시간 동기화에 필요한 정보를 제공하며 계층구조를 사용하여 메시지 전송을 최소화함으로써 기존 방법보다 에너지 효율적이다. 또한 동기화될 시간을 계산하기 위해 누적된 시간정보를 사용하며 MAC 계층에서의 타임스탬핑을 사용함으로써 시간 정확

성을 높였다. MICAz 모트에 구현하여 성능을 평가함으로써 1μs 이하의 평균 정확성과 1μs 이하의 분산을 가지고 있음을 알 수 있었다.

향후 연구로는 다양한 동기화 주기를 사용한 실험과 멀티 홉(multi-hop) 네트워크에서의 실험을 통해 성능을 평가할 것이다. 또한 센서 노드들의 고장을 감내할 수 있는 기능을 추가할 계획이다.

6. 참고 문헌

[1] H. Dai and R. Han, "TSync : A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks," Mobile Computing and Comm. Review, 2004.

[2] 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.

[3] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing Sync Protocol for Sensor Networks," ACM SenSys, Los Angeles, CA, Nov. 2003.

[4] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. "The flooding time synchronization protocol," Proc. 2nd international conference on Embedded networked sensor systems, 2004.

[5] J. V. Greunen and J. Rabaey, "Lightweight Time Synchronization for Sensor Networks," Proc. 2nd ACM Int'l. Conf. Wireless Sensor Networks and Apps., San Diego, CA, Sept. 2003.

[6] H. Kopetz and W. Schwabl. "Global time in distributed real-time systems". Technical Report 15/89, Technische Universit¨at Wien, 1989.

[7] Simple Network Time Protocol, (SNTP) version 4. IETF RFC 2030.

[8] TinyOS, http://webs.cs.berkeley.edu/tos

[9] Crossbow Technology Inc., "MICAz wireless measurement system," http://www.xbow.com, June 2004.

[10] D. Gay, P. Levis, R. Behren, M. Welsh, E. Brewer, and D, Culler, "The nesC Language: A Holistic Approach to Networked Embedded Systems", Proceedings of Programming Language Design and Implementation (PLDI) 2003, June 2003.