무선 센서 네트워크에서의 시각동기를 위한 재귀적 클럭 스큐 추정 방법

김동진, 맹세영, 방종대, 이연우, 정민아, 이성로

목포대학교

srlee@mokpo.ac.kr

Recursive Clock Skew Estimators for Time Synchronization in Wireless Sensor Networks

Dongjin Kim, Seyeong Maeng, Jongdae Bang, Yeonwoo Lee, Min-a Jung, Seong Ro Le

Mokpo National Univ.

요 약

무선 센서네트워크에서의 시각동기는 MAC 계층에서부터 APP 계층에 이르기까지 거의 모든 계층에서 다양한 목적을 위해 매우 중요한 기술이다. 본 논문에서는 무선 센서네트워크에서의 에너지 효율적인 시각동기를 위한 실시간 클럭 스큐 추정 방법을 제시한다. 재귀적 최소제곱법을 통해 오프셋 보정 정보들을 얻을 때마다 클럭 스큐가 실시간적으로 추정 및 갱신되며, 아울러 스큐 추정을 위해 각 센서노드에 저장해야할 정보를 최소화한다. 제안한 클럭 스큐 추정 방법은 기존의 클럭 오프셋 보정 방법과 쉽게 통합될 수 있으며, 이 경우 보다 정확하고 효율적인 시각동기화가 가능해진다. 시뮬레이션 및 실험 결과를 통해 제안한 클럭 스큐 추정 방법을 통한 시각동기 정확도의 향상을 보인다.

Ⅰ. 서론

일반적으로 센서네트워크를 구성하는 노드들은 자체적으로 타이머(클럭)를 보유하고 있으며 MAC (Medium AccessControl) 계층, NWK (Network) 계층, APP (Application)

계층에서의 요구로 인해 일부 또는 전체 노드의 타이머가 동기화되는 것이 필요하다.

무선 센서네트워크는 인터넷 또는 LAN 환경과는 다르기 때문에 무선 센서네트워크를 위한 시각동기 프로토콜은 그만의 특수한 요구사항을 필요로 한다. 첫째, 배터리로 동작하는 센서노드의 가용 에너지는 한정되어 있기 때문에 시각동기는 반드시 에너지 효율적인 방법으로 동작할 수 있어야 한다. 둘째, 센서노드는 보통 크기가 작으므로 컴퓨팅 능력과 저장 공간에도 제약이 따른다. 따라서 NTP 또는 GPS와 같은 전통적인 시각동기 방법들은 복잡도와 에너지 관련 이슈, 경제성, 제한된 노드 크기 등으로 인해 무선 센서네트워크에 적용하기에는 적합하지 않다[1].

앞서 언급한 요구사항들을 고려하여, 본 논문에서는 재귀적 클럭 스큐 추정을 통한 무선 센서네트워크에서의 시각동기 기법을 제시한다. 클럭 스큐는 재귀적 최소제곱법 (recursive least squares)을 통해 추정 및 갱신되며, 오프셋 보정과 스큐의 보정은 동시에 수행되도록 한다. 제안한 추정 방법을 통해 스큐 보정을 위해 각 센서노드에 저장되는 데이터의 양을 최소화할 수 있다. 아울러 시각동기의 정확도를 향상시켜 빈번한 재동기에 따른 통신 부담을 줄일 수 있도록 한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 관련 연구를 소개하고, Ⅲ장에서는 재귀적 클럭 스큐 추정 방법을 제안한다. Ⅳ장에서는 실험을 통해 성능 평가를 수행하고 결론을 맺는다.

Ⅱ. 관련 연구

센서네트워크에서의 시각동기를 위한 많은 프로토콜들이 연구되어 왔으며, 이 프로토콜들은 대체로 공통적인 특징을 갖고 있다. 센서노드들 간 클럭(타임스탬프) 정보의 교환을 통해 시각동기를 수행하고, 메시지 송수신 과정에서의 비결정적인(nondeterministic) 요소들의 영향을 최소화한다는 것이 그것이다. 이들은 receiver-receiver (R-R) 방식, sender-receiver (S-R) 방식, 단방향 메시지 확산 (one-way dissemination) 방식의 세 가지로 분류할 수 있다.

R-R 동기화 방식에서는 한 노드가 비컨 메시지를 이웃 노드들에게 주기적으로 브로드캐스트 전송하며, 이를 수신한 노드들은 수신 시각의 차이를 이용하여 클럭 오프셋을 구할 수 있으며 시각을 동기화시킬 수 있다. RBS(Reference Broadcast Synchronization)[2]는 R-R 동기화 방식의 대표적인 프로토콜이다. S-R 동기화 방식에서는 한 쌍의 노드 간에 핸드셰이크(handshake) 프로토콜을 통해 동기화가 이뤄지며, 이 방식을 따르는 대표적인 시각동기 프로토콜에는

TPSN(Timing-Sync Protocol for Sensor Networks)[3]이 있다. 단방향 메시지 확산 방식에서는 기준 노드가 이웃 노드들에게 시각 정보를 브로드캐스트 전송하고, 이를 수신한 각 노드는 브로드캐스트 메시지의 수신 시각들을 기록하여 선형회귀(linear regression) 테이블을 통해 로컬 클럭과 기준 노드의 클럭 간 변환을 하게 된다. FTSP(Flooding Time Synchronization Protocol)[4]는 단방향 메시지 확산 방식의 대표적인 프로토콜이다.

Ⅲ. 제안하는 재귀적 클럭 스큐 추정 방법

본 논문에서는 재귀적 최소제곱법을 이용한 실시간 클럭 스큐 추정 방법을 제공한다. 오프셋 보정과 관련해서는 기존의 방식들(TPSN, RBS, FTSP 등)이 수정 없이 그대로 적용될 수 있으며, 클럭 오프셋 정보를 얻을 때마다 클럭 스큐는 재귀적으로 추정 및 갱신되어 저장해야 할 데이터 정보를 최소화할 수 있다.

노드 B가 노드 A와 동기화하려는 상황을 가정하자. 노드 A와 B가 하나 혹은 그 이상의 동기 메시지를 서로 교환하게 되면, 그림 1에서와 같이 노드 B는 오프셋 보정을 통해 노드 A와 잠시 동기화할 수 있다. 그러나 실제 무선 네트워크 환경에서는 다양한 지연 요소들이 메시지 전송 과정에 영향을 미치게 되어 시각동기의 문제를 더욱 어렵게 만들고, 따라서 노드 간 클럭 오프셋과 스큐를 추정하기 위해서는 연속적인 동기 메시지의 교환이 필요하다.

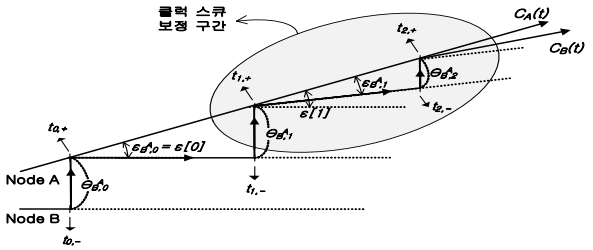

그림 1. 오프셋 보정 및 재귀적 클럭 스큐 추정

i-번째 동기 메시지의 교환이 끝난 후 노드 B는 상대 클럭 오프셋 $\hat{\theta}^A_{B,i}$을 계산하게 되고, 곧바로 현재 자신의 클럭값에 이 오프셋을 더함으로써 노드 A의 클럭과 동기화할 수 있다. 이후 (i+1)-번째 오프셋 보정이 수행되면, 노드 B 입장에서는 i-번째 단계와 (i+1)-번째 단계 사이의 상대 클럭 스큐를 계산할 수 있게 된다. 0-번째 오프셋 보정 시점에서는 아직 스큐 추정을 할 수 있는 충분한 정보가 없기 때문에 오프셋 보정만 수행된다. 1-번째 오프셋 보정이 수행되고 난 후에야, 상대 클럭 스큐 $\epsilon^A_{B,0}$의 계산이 식 (1)을 통해 가능해진다.

$$\epsilon^A_{B,i} = \frac{\hat{\theta}^A_{B,i+1}}{t_{i+1,-} - t_{i,+}} \qquad (1)$$

1-번째 오프셋 보정 이후 상대 클럭 스큐값을 얻을 수 있으나, 현 시점에서 이 값은 노이즈를 포함하고 있기 때문에 추정 신뢰도가 높지 않다. 따라서 실제 클럭 스큐와 추정치와의 오차를 줄이기 위해 선형회귀 분석에 사용되는 최소제곱법을 사용하여 접근하도록 한다. (i+1)-번째(i≥2) 오프셋 보정 이후 추정되는 상대 클럭 스큐 $\hat{E}[i]$에 대한 재귀적 형태의 식을 정리하면 다음과 같다.

$$\hat{E}[i] = \frac{i}{i+1}\hat{E}[i-1] + \frac{1}{i+1}\epsilon[i]$$

$$= \hat{E}[i-1] + \frac{1}{i+1}\epsilon^A_{B,i}\left(1 + \hat{E}[i-1]\right) \qquad (2)$$

여기에서 $\epsilon[i]$는 상대 클럭 스큐를 나타내는 새로운 변수로서 다음과 같이 정의된다.

$$\epsilon[i] = \hat{E}[i-1] + \epsilon^A_{B,i} + \hat{E}[i-1]\epsilon^A_{B,1} \qquad (3)$$

$\epsilon[i]$는 i-번째 오프셋 보정 시점부터 (i+1)-번째 오프셋 보정 시점까지의 상대 클럭 스큐를 나타낸다는 점에서 $\epsilon^A_{B,i}$와 비슷하다. 그러나 $\epsilon[i]$는 노드 B 클럭의 본래의 주파수로부터의 노드 A 클럭의 주파수 오프셋을 의미하며, $\epsilon^A_{B,i}$는 노드 B 클럭의 보정된 주파수로부터의 주파수 오프셋을 의미한다. 여기서 $\hat{E}[i]$과 $\epsilon[i]$의 초기값은 다음과 같다.

$$\hat{E}[0] = \epsilon[0] = \epsilon^A_{B,0}$$

식 (2)을 통해, 상대 스큐 추정치를 갱신시키기 위해 필요한 정보는 세 가지임을 확인할 수 있다. 첫째는 이전 단계에서의 스큐 추정치 $\hat{E}[i-1]$, 둘째는 (i+1)-단계에서의 오프셋 보정을 통해 얻은 상대 클럭 스큐 $\epsilon^A_{B,i}$, 셋째는 관측 스큐 정보의 수 i이다. 여기서 $\epsilon[i]$에 대한 정보는 필요치 않다는 점에 주목하자.

Ⅳ. 성능 분석 및 결론

제안한 클럭 스큐 추정 방법을 적용한 시각동기의 실제 성능을 확인하기 위해 동기 오차를 측정하기 위한 실내 실험을 수행하였다. 이를 위해 자체 제작 센서노드 플랫폼에 제안한 알고리즘을 구현하였다. 센서노드 플랫폼은 MSP430 MCU와 CC1100 RF 트랜시버의 하드웨어로 구성되며, 초소형 OS와 네트워크 프로토콜 스택이 탑재되어 있다. 또한 ±20ppm의 주파수 오차를 갖는 클럭 기반의 32.768㎑ 타이머를 사용하였으며, MAC 계층에서 타임스탬핑이 수행되도록 하였다. 비컨이 브로드캐스트 전송되는 주기를 변경하며, 매 4초마다 시각동기 오차를 측정하여 표 1에 결과를 나타내었다. 예견했던 바와 같이 비컨 전송의 주기가 길어짐에 따라 시각동기의 정확도는 감소하였지만, 전체적으로 평균 100㎲ 이하의 동기 수준을 보였다.

표 1. 비컨 주기에 따른 동기 오차의 평균 및 최대편차 (1 Tick ≒ 30.5㎲)

비컨 주기	시각동기 오차			
	평균편차		최대편차	
	㎲	Tick	㎲	Tick
1분	35.7	1.17	113.2	3.71
2분	42.1	1.38	130.6	4.28
4분	53.1	1.74	193.8	6.35
8분	70.5	2.31	272.8	8.94
16분	94.3	3.09	369.6	12.11

제안한 방법을 통해 오프셋 보정과 스큐 보정이 동시에 수행되도록 하며, 아울러 이 과정에서 각 노드가 저장해야 할 데이터의 양을 최소화할 수 있도록 하였다. 오프셋 보정과 관련해서는 기존의 방법들이 수정 없이 제안한 클럭 스큐 추정 방법에 적용될 수 있다. 실험 결과를 통해 제안한 방법이 클럭 스큐의 영향에 따른 클럭 오프셋 오차를 상당히 감소시키고 장기적인 시각동기의 안정성을 향상시킴을 확인하였다. 이는 적정 수준의 동기 수준을 유지하기 위한 재동기 주기를 길게 가져갈 수 있도록 하여, 이에 따른 에너지 소모를 줄일 수 있는 효과가 있다.

ACKNOWLEDGMENT

이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 대학중점연구소 지원사업으로 수행된 연구임 (2011-0022980)

본 연구는 지식경제부 및 정보통신산업진흥원의 IT융합 고급인력과정 지원사업의 연구결과로 수행되었음 (NIPA-2012-H0402-12-1001)

참 고 문 헌

[1] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, “Clock Synchronization for Wireless Sensor Networks: a Survey,” Ad-Hoc Networks, Vol. 3, No. 3, pp. 281-323, 2005.

[2] J. Elson, L. Girod, and D. Estrin, “Fine-Grained Network Time Synchronization Using Reference Broadcasts,” Proceedings 5th Symposium on Operating System Design and Implementation, pp. 147-163, 2002.

[3] S. Ganeriwal, R. Kumar, and M. B. Srivastava, “Timing-Sync Protocol for Sensor Networks,” Proceedings First International Conference on Embedded Network Sensor Systems (SenSys), pp. 138-149, 2003.

[4] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, “The Flooding Time Synchronization Protocol,” Proceedings Second International Conference on Embedded Networked Sensor Systems (SenSys), pp. 39-49, 2004.