무선환경에서 분산 임베디드 시스템을 위한 시간 동기화 기법

이윤준°, 홍영식
동국대학교 컴퓨터공학과
iamjuni@hanmail.net, hongys@dgu.ac.kr

A Clock Synchronization protocol for Distributed Embedded Systems in wireless environments

Youn-Jun Lee° , Young-Sik Hong
Dept. of Computer Engineering, Dongguk University

요 약

최근 무선 임베디드 시스템의 사용이 증가하면서 기존의 분산 환경에 무선 임베디드 시스템이 포함되기 시작하였고, 이를 고려한 분산 어플리케이션들이 개발되고 있다. Global clock과 동기화할 수 있는 GPS가 모든 무선 임베디드 시스템에 장착되지 않은 상황에서 분산된 임베디드 시스템간 혹은 고성능 컴퓨터와의 내부 동기화를 수행할 동기화 기법이 필요하다. 현재 무선환경에서의 동기화에 대한 연구들이 이루어지고 있지만 제한된 리소스의 임베디드 시스템에 그대로 적용하기 어렵다.

이에 본 논문에서는 무선 임베디드 시스템만이 가지는 제한사항을 고려하여 메시지 지연값의 변화량을 측정하여 적용할 수 있는 시간 동기화 기법을 제시하고 실험을 통해 그 성능을 평가한다.

1. 서 론

분산 시스템은 네트워크망으로 연결된 여러 컴퓨터들이 메시지 교환을 통해 상호 통신하고 협력하는 집합체를 말한다. 분산 시스템 구성요소간에 동기화, 순서화, 그리고 일관성 유지를 위해서 시간 동기화는 반드시 필요하다. 이를 위해 시간 동기화에 대한 많은 연구가 이루어져왔다. 최근 들어 무선 임베디드 시스템의 성능이 향상되고 보급률이 높아짐에 따라 기존의 분산 환경에도 무선 임베디드 시스템들이 포함되기 시작하였다. 따라서 이를 고려한 많은 분산 어플리케이션들이 개발되고 있다.

최근 무선 환경에서의 시간 동기화에 대한 많은 연구가 이루어졌지만, 리소스의 한계를 지닌 무선 임베디드 시스템에 그대로 적용시키는 것은 부적합하다. 외부 표준 시간과 동기화를 수행할 수 있는 GPS(Global Positioning System)는 아직까지 모든 임베디드 시스템에 장착되어 있지 않아 이에 대한 해결책이 될 수 없다. 이에 본 논문에서는 무선 임베디드 시스템을 고려하여 분산 환경에서 보다 정확한 정밀도를 가진 시간 동기화 기법을 제안한다.

본 논문의 2장에서는 무선환경에서의 불규칙한 메시지 지연을 처리하기 위한 기존 연구에 대해 알아보고, 3장에서 두 시스템 시간의 변화량으로부터 메시지 지연의 변화량을 계산하고 적용시킬 수 있는 동기화 구조를, 4장에서는 이에 따른 실험 결과를 보여준다. 마지막 5장에서는 본 논문에 대한 결론 및 향후 연구 과제에 대해 논한다.

2. 관련 연구

무선 임베디드 시스템은 메시지의 손실과 지연이 많은 무선환경의 특징과 제한된 리소스를 가지는 임베디드 시스템의 특징을 모두 가지고 있다. 무선환경만을 고려한 기존 연구의 경우 무선통신매체의 Broadcast 특성을 적용하여 두개의 Receiver가 동일한 메시지를 받으면 거의 같은 시점에 그 메시지를 받는다는 가정 하에서 이 문제를 해결하였다.[2] 그러나 Receiver가 임베디드 시스템인 경우 적은 리소스로 인해 메시지 수신시 시스템의 상태에 따라 시간 정보의 처리속도에서 많은 차이를 발생시킨다. 또한 유효범위를 두어 그 안에 도착된 메시지만을 유효한 값으로 처리하는 방식도 동일한 문제점을 안고 있다.[4]

rate-based Algorithm은 무선환경에서 메시지 손실과 지연을 감내하기 위해 제안되었다.[2] rate-based Algorithm은 두 시스템의 물리적 시간들이 시간의 흐름에 비례하여 일정한 비율(drift rate)로 drift된다는 전제하에 Slave가 연속된 측정을 통해 얻어지는 Master와의 drift rate를 사용하여 동기화 메시지가 도착하지 않는 경우에도 시간 동기화를 수행해 나갈 수 있다.

3. 제안한 동기화 기법

본 논문에서 제안한 동기화 기법은 Master-Slave구조를 기반으로 한다. 고성능 시스템인 Master는 시스템 시간만을 사용하고, 임베디드 시스템, 즉 Slave는 자신의 시스템 시간뿐만 아니라 주기적으로 송신되는 Master의

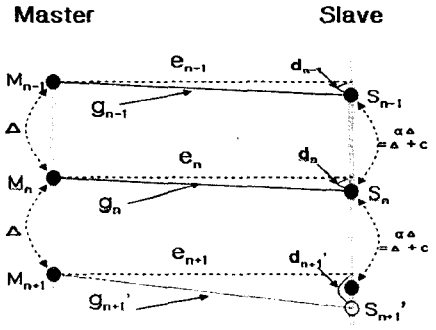
시스템 시간으로부터 동기화된 가상 시간을 함께 사용한다.

3.1 초기 동기화

두 시스템 간에 처음 메시지 교환이 발생하면 쿨스타트(Cool Start)효과로 인해 메시지 지연값의 변화가 크다. 따라서 제안하고 있는 시간 동기화 기법에서 초기 메시지 지연의 정확한 측정은 매우 중요하다. 쿨스타트 효과 이후 [그림 2]와 같이 약 90%의 메시지 지연값이 일정하다는 실험 결과로부터 일정수만큼의 반복된 메시지 지연값 측정이후 가장 빈도수가 높은 메시지 지연값을 초기 메시지 지연값(d_{init})으로 설정한다. 이때 메시지 지연값의 측정은 메시지를 송수신하여 측정하는 일반적인 방식을 그대로 사용한다.

3.2 재동기화

두 시스템의 물리적 시간은 시간의 변화에 비례하여 일정한 편차를 보이므로 재동기화는 반드시 필요하다. 제안하고 있는 동기화 기법에서도 일정주기(△)마다 재동기화 메시지를 보냄으로써 재동기화를 수행한다.



[그림 1] Master와 Slave의 재동기화

[그림 1]은 Master가 Slave에게 주기적인 재동기화 메시지를 보낼 때 Master와 Slave의 형태를 나타낸 그림이다.

M_n과 S_n은 n번째 동기화 메시지 송수신시 Master와 Slave의 시스템 시간이고, d_n은 이때 발생된 메시지 지연값, g_n은 메시지 송수신시 두 시스템의 시간차(S_n - M_n)이다. 이때 초기 메시지 지연값을 구하는 방식과 달리 재동기화시에는 오직 Master에서 Slave로 송신만 하므로 측정할 수 있는 값은 M_n 과 S_n 둘뿐이다. 따라서 d_n은 다음의 공식을 이용하여 구한다.

e_n을 Master가 메시지를 보내는 순간에 Slave와의 시간 차이값으로 정의하자.

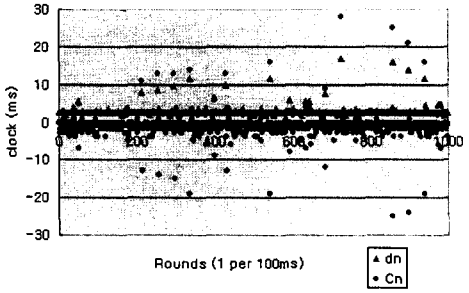
$$g_n = e_n + d_n \quad ...(1)$$

n번째 두 시스템 시간의 차이(g_n)와 n-1번째 동기화시 두 시스템 시간의 차이(g_{n-1})의 차를 C_n이라 하면 다음과 같다.

$$\begin{aligned} C_n &= g_n - g_{n-1} \\ &= (e_n + d_n) - (e_{n-1} + d_{n-1}) \\ &= (e_n - e_{n-1}) + (d_n - d_{n-1}) \\ &= (\alpha\Delta - \Delta) + (d_n - d_{n-1}) \\ &= C + (d_n - d_{n-1}) \quad ...(2) \end{aligned}$$

※ $C = \alpha\Delta - \Delta$, α는 drift rate

메시지 지연값 d_k 가 일정하다면 C_k의 값은 C가 된다.



[그림 2] 메시지 지연값 d_n 과 편차값 C_n의 관계

하지만 일반적으로 [그림 2]와 같이 대략 10%의 크고 작은 메시지 지연의 변화가 발생한다.

따라서 메시지 지연 값의 변화에 의해 발생하는 C_n의 오차를 줄여 C에 근접하도록 두 가지의 형태의 평균값을 사용하였다.

(a) $C_{avg(n)} = \Sigma C_n / n$

(b) $C_{avg(n)} = 2*\Sigma(g_n - g_{init}) / (n*(n+1))$

※g_{init} 메시지 지연 값 초기화시 설정값

식(a)는 일반적인 형태의 평균값을 구하는 식이다.[7] [그림 2]와 같이 두 시스템 간 시간차의 변화량이 한번 증가하게 되면 그 다음의 변화량은 그와 비슷한 값으로 감소하게 되어 식 (a)를 이용할 경우 증가 또는 감소했던 메시지 지연의 변화가 다음번 주기 때 거의 상쇄되어 결국 C에 근접한다. 그러나 식 (a)의 경우 마지막 값의 변화가 전체 평균값에 큰 영향을 주어 $C_{avg(n)}$ 값의 변화가 심하다.

마지막값에 의한 변화를 줄이기 위해 이전값들의 누적을 이용한 식 (b)를 유도하였다.

$$\begin{aligned} g_1 - g_{init} &= C_1 \approx C \\ g_2 - g_{init} &= C_1 + C_2 \approx 2*C \\ &\vdots \quad\quad \vdots \\ g_n - g_{init} &= C_1 + C_2 + \cdots + C_n \approx n*C \\ \Sigma(g_n - g_{init}) &= n*C_1 + (n-1)*C_2 + \cdots + 1*C_n \\ &\approx (n*(n+1)/2)*C \\ \therefore C &\approx 2*\Sigma(g_n - g_{init}) / (n*(n+1)) \end{aligned}$$

이전 값들에 가중치가 주어지므로 마지막 값의 변화가 전체 평균값에는 큰 영향을 주지 못한다. 즉 매 주기마다 다시 계산된 $C_{avg(n)}$의 변화가 작다.

현재의 두 시스템 간 시간차의 변화량과 두 식을 통해 구해진 C_{avg}의 값을 이용하여 메시지 지연의 변화량을 계산할 수 있다. 즉 식 (2)는 다음과 같이 쓸 수 있다.

$$C_n - C_{avg(n)} \approx (d_n - d_{n-1}) \quad ...(3)$$

이러한 과정을 통해 얻어진 메시지 지연의 변화량으로

Slave는 가상시간을 유지해 나간다.

3.3 가상 시간의 유지

Slave의 물리적 시간은 시스템 시간을 측정할 때 사용되며 실제 동기화시 가상시간(V)을 유지해나간다. n번째 메시지 수신 시 자신의 가상 시간(V_n)을 다음과 같이 설정한다.

$$V_n = M_n + d_n \quad ...(4)$$

※ $d_n = d_{init} + C_n - C_{avg(n)}$

이렇게 설정된 가상시간은 다음 재동기화 메시지가 도착하기 전까지 또다시 drift rate (α)가 고려된 형태로 유지된다.

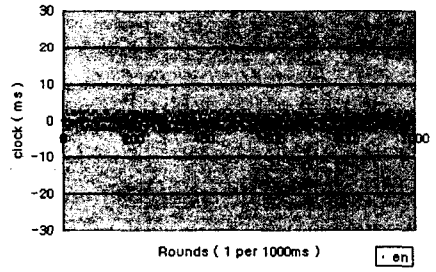
$V = V_n + (t * \alpha)$

(t: V_n로부터 경과된 Slave의 물리적 시간)

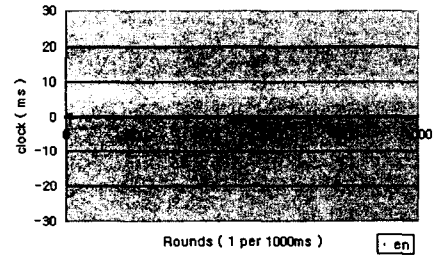
위의 형태로 가상시간을 유지함으로써 메시지 손실이 발생하더라도 일정 오차범위 내에서 동기화된 시간을 유지해 나갈 수 있다.

4. 실험 및 결과

Window 2000을 탑재한 펜티엄4 1.7GHz의 PC와 WinCE 3.0를 탑재한 ARM CPU의 PDA를 사용하여 실험하였다.



(a)식을 이용한 결과



(b)식을 이용한 결과

[그림 4] Master와 Slave간의 시간차

[그림 3]의 (a),(b)는 1000ms의 주기로 두 가지 형태의 동기화를 수행하였을 때 얻어진 Master와 Slave간의 시간차를 나타낸다. 실험시 [그림 2]와 같이 메시지 지연값의 변화가 너무 큰 경우(약 1%)는 필터링(filtering)하였다. 이와 같은 실험환경 하에서 일반적인 형태의 (a)식을 이용한 경우는 차의 범위가 ±3ms, (b)식을 이용한 경우 ±2ms의 시간차를 보였다. 즉 (b)와 같이 안정된 C의 근사값을 사용한 경우가 보다 나은 동기화 성능을 보였다.

5. 결론 및 향후 과제

본 논문에서는 무선 환경을 위한 IEEE 802.11 표준에서 권고하고 있는 Master/Slave 시간 동기화 구조를 기반으로 두 시스템 시간의 차이로부터 변화된 메시지 지연을 계산하여 적용하는 방식의 동기화 기법을 제안하여 무선 임베디드 시스템에서 발생하는 손실과 적은 리소스로 인해 메시지 수신 시 시스템의 상태에 따라 시간정보의 처리속도에서 많은 오차가 발생되는 문제의 해법을 제시하였다.

또한 연속된 두 시스템 시간의 차들의 근접한 값을 구하기 위해 일반적인 형태의 평균식과 누적치를 이용한 평균식을 적용하여 비교 실험해 보았다.

향후 과제로는 제안하고 있는 동기화 프로토콜의 성능에 큰 영향을 미치게 되는 정확한 초기 메시지 지연측정에 대한 연구와 더욱더 정밀한 실험환경에서의 성능평가이다.

6. 참고 문헌

[1]Emmanuelle Anceaume and Isabelle Puaut *"Performance Evaluation of Clock Synchronization Algorithms"*, 1998

[2]Horst F. Wedde, Wolfgang Freund, *"Harmonious Internal Clock Synchronization"* 12th Euromicro Conference on Real-Time Systems, pp.175-182, 2000

[3]K. H. (Kane) Kim, Chansik Im and Prasad Athreya, *"Realization of a Distributed OS Component for Internal Clock Synchronization in a LAN Environment"*, Proceedings of the fifth IEEE Symposium on Object-Oriented Real-Time Distributed Computing, pp.263-279, 2002.

[4]Mock, M., Frings, R., Nett, E. and Trikaliotios, S., *"Continuous clock synchronization in wireless real-time applications"*, Proceedings of The 19th IEEE Symposium on Reliable Distributed Systems, pp.125-132, 2000.

[5]Mock, M., Frings, R., Nett, E. and Trikaliotios, S., *"Clock synchronization for wireless local area networks"* The 12th Euromicro Conference on Real-Time Systems, pp.61-67, 2000.

[6]Wedde, H.F., J.A. Lind and G. Segbert, *"Achieving Internal Synchronization Accuracy of 30μs Under Message Delays Varying More Than 3 msec"*, Proceedings of the 24th IFAC/IFIP Workshop on REAL TIME PROGRAMMING, 1999.

[7]Y.S. Hong and J.H. No *"Clock Synchronization in Wireless Distributed Embedded Applications"*, IEEE Workshop on Software Technologies for Future Embedded Systems, pp.101-104, 2003