실시간 제약 조건의 동적/정적 변환을 통한 클록 동기화 문제 해결

유 민수, 홍 성수
서울대학교 전기공학부 실시간 운영체제 연구실

Revisiting Clock Synchronization Problems: Static and Dynamic Constraint Transformation for Correct Timing Enforcement

Minsoo Ryu and Seongsoo Hong
Seoul National University

요 약

본 논문에서는 클록들을 주기적으로 동기화하는 분산 실시간 시스템에서 주어진 태스크의 시간 제약(timing constraint)을 변환시는 두가지 기법을 제안한다. 전형적인 이산 클록 동기화(discrete clock synchronization) 알고리즘은 클록의 값을 순간적으로 보정(correct)하여 클록의 시간이 불연속적으로 진행하게 한다. 이러한 시간상의 불연속성은 태스크의 시작제한시간(release time)이나 종료시한(deadline)과 같은 이벤트를 잃어버리거나 다시 발생시키는 오류를 범하게한다.

클록 시간의 불연속성을 피하기 위해 일반적으로 연속 클록 동기화(continuous clock synchronization) 기법이 제안되었지만 소프트웨어적으로 구현되기에는 많은 오버헤드를 유발시키는 문제점이 있다. 이에 따라 연속 클록 동기화는 PLL (Phase-Locked Loop)을 이용한 별도의 하드웨어를 사용하는 것이 보통이다. 본 논문에서는 연속 클록 동기화 기법을 사용하는 대신, 태스크의 시간 제약을 동적으로 변환시키는 DCT (Dynamic Constraint Transformation) 기법을 제안하였다. DCT는 소프트웨어적으로 구현이 가능하여 새로운 하드웨어를 필요로 하지 않으며, 이를 통해 기존의 이산적으로 동기화된 시스템에서 클록 시간의 불연속성에 의한 문제점들을 해결할 수 있다.

또 다른 문제점으로서, 클록의 물리적인 특성으로 인해 동기화된 클록들이 상한된(bounded from the above) 오차(skew)를 갖는다는 것이다. 이러한 오차는 지역 클록(local clock)에 대해 만족될 수 있는 임의의 실시간 제약 조건이 전역 클록(global clock)에 대해서는 만족되지 않을 수 있음을 의미한다. 본 논문에서는 이를 위해 먼저 두가지의 스케쥴링 가능성, 지역적 스케쥴링 가능성(local schedulability)과 전역적 스케쥴링 가능성(global schedulability)을 정의하고, 실시간 제약을 정적으로 변환시키는 SCT (Static Constraint Transformation) 기법을 제안하였다. SCT를 통해 지역적으로 스케쥴링 가능한 태스크는 전역적으로 스케쥴링이 가능하므로, 단지 지역적 스케쥴링 가능성만을 검사하면 스케쥴링 문제를 해결할 수 있도록 하였다.

1. 서 론

분산 실시간 시스템에서 전역 클록(global clock)은 분산된 컴퓨터들이 외부 사건(event)을 관측하거나 실시간 스케쥴링을 행함에 있어 기준이 되는 시간값을 제공한다. 하지만 컴퓨터들이 공간적으로 분산되어있으므로 이러한 전역 클록은 본질적으로 존재할 수 없다. 또한 물리적인 특성으로 클록은 온도나 압력과 같은 외부 환경에 대해 약간씩 오차를 보이는 드리프트(drift)를 가진다. 따라서 분산된 클록들을 초기에 정확히 동기화시킨다 하더라도 점차 서로 발산하게 된다. 이에 따라 분산된 클록들을 주기적으로 동기화시키기 위한 알고리즘들이 제안되어 왔는데, 대표적인 것들로서 Interactive Convergence Algorithm (CNV) [2], Interactive Consistency Algorithm (COM, CSM) [2], Fault Tolerant Average Algorithm (FTA) [7]등이 있다 [2, 3, 4, 5,].

하지만 위와 같은 방법들은 주로 고장감내(fault-tolerancy)에 초점을 맞춘 것으로서 클록 동기화 알고리즘이 실제로 적용될 때 발생하는 문제에 대해서는 아직 주목할 만한 연구가 수행된 바 없다. 일반적으로 클록 동기화 알고리즘들은 (1) 이산 클록 동기화(discrete clock synchronization)와 (2) 연속 클록 동기화(continuous clock synchronizstion)의 두가지 방법으로 구현될 수 있다. 이산 클록 동기화는 분산된 클록들을 일시에 보정(correct)시키는 반면, 연속 클록 동기화는 보정(correction)값을 동기화 주기(resynchronization period) 내에 고르게 분포시킨다. 연속 클록 동기화는 클록의 매 틱(tick)마다 보정시켜야 하므로 소프트웨어적으로 구현하기에는 어려움이 따른다. 일반적으로 연속 클록 동기화는 PLL (Phase-Locked Loop)과 같은 하드웨어 기법으로 구현될 수 있다. 따라서 경제적이고 구현이 용이한 이산 클록 동기화 기법이 널리 사용되어 왔는데, 이는 클록 시간을 불연속적으로 진행시켜 실시간 시스템과 같은 응용 분야에서는 심각한 오류를 일으킨다. 클록의 현재 값에 의해 계산된 보정값에 따라 클록의 시간은 과거의 시각, 또는 미래의 시각으로 보정되는데,

이로 인해 실시간 태스크(real-time task)의 종료시한(deadline)이나 시작제한시간(release time)과 같은 사건(event)을 잃어버리거나 다시 만나게 되는 문제점이 야기된다.

본 논문에서는 이러한 문제점을 해결하기 위해 태스크의 시간 제약을 동적으로 변환시키는 DCT (Dynamic Constraint Transformation) 기법을 제안한다. 다음 절에서 볼 수 있듯이 클록들이 이산적으로 동기화 되더라도 연속적으로 동기화 되었을 때의 클록 값은 쉽게 유도될 수 있다. 이 때 얻어진 연속 함수, 즉 연속 클록 동기화에 얻어진 클록의 함수를 이용해 태스크들이 갖는 시간 제약을 변환시키는데 이러한 기법을 DCT라 한다. DCT에 의해 실시간 태스크들은 마치 연속적으로 동기화된 클록에 따라 스케쥴되는 것처럼 수행되므로 클록 시간의 불연속성에 의한 문제를 해결할 수 있다.

클록 동기화가 적용될 때 발생하는 또 다른 문제점은 실시간 스케쥴링과 연관되어 있다. 클록들이 주기적으로 동기화된다 하더라도 클록 값들의 차이는 존재하기 마련인데 이를 클록 오차(skew)라 일컫는다. 임의의 컴퓨터에 할당되어 수행되는 실시간 태스크들은 그 컴퓨터에 내장된 지역 클록(local clock)에 따라 스케쥴되는데, 이 때 태스크의 시간 제약은 전역 클록(global clock)에 대해 만족되어야 한다. 하지만 클록 오차로 인해 지역 클록(local clock)과 전역 클록(global clock)이 가리키는 시간이 일치하지 않으므로 태스크의 시간 제약들은 클록 오차를 고려해 변환되어야 한다. 본 논문에서는 이를 위해 SCT (Static Constraint Transformation)를 제안하였으며, SCT에 의해 지역 클록에 의해 스케쥴링이 가능한 태스크는 전역 클록에 의해서도 스케쥴링이 가능하게 된다.

2. 클록 동기화 모델

2.1 정상 클록

임의의 지역 클록(local clock) C는 외부 환경에 따라 이상적인 기준 클록(reference clock) C_r으로부터 발산하게 되는데, 기준 클록 시간 t_r에서 C의 클록 값 t은 $C(t_r)=t$과 같은 함수로 표현될 수 있다. 이 때, 지역 클록이 다음과 같은 성질을 갖게 될 때 클록 동기화가 가능할 뿐만 아니라 시스템의 올바른 동작을 보장할 수 있다.

정의 1. 클록 $C(t_r)$ 이 연속, 단조 증가, 미분가능하고 $|\frac{dC(t_r)}{dt_r}-1|$이 상한되어 있으면, C_r에 대해 **정상** 클록이라 한다

위의 정의에 따르면 정상적인 클록은 연속성과 단조 증가성을 가질 뿐만 아니라 $|\frac{dC(t_r)}{dt_r}-1|$로 표현되는 드리프트 비율(drift rate) ρ가 상한값을 가진다. 드리프트 비율이 상한될 때 클록 동기화에 의해 클록 오차(skew)가 상한될 수 있다.

2.2 이산 클록 동기화

드리프트 비율로 인해 지역 클록은 클록 오차가 존재하게 되는데 이는 다음과 같이 정의될 수 있다.

$$\Delta(t_r)=C_r(t_r)-C(t_r)=t_r-C(t_r) \quad (1)$$

T_{clk}을 클록 동기화 주기라 하자. 임의의 시간 t^k_r에서 계산된 보정(correction)값을 Φ_k라 하면 이산적으로 동기화되는 클록의 함수는 다음과 같이 주어진다.

$$C^d(t_r)=C(t_r)+\sum_{k=0}^{\infty}\Phi_k u(t_r-t^k_r) \quad (2)$$

이 때 $u(t)$는 다음과 같이 주어진다.

$$u(t)=\begin{cases}0 & \text{if } t<0\\ 1 & \text{if } t\geq 0\end{cases}$$

2.3 연속 클록 동기화

연속적으로 동기화되는 클록의 함수는 보정값을 동기화 주기 구간에 고르게 분포함으로써 얻을 수 있다. 연속적으로 동기화된 클록의 함수를 C^c라 하면, 이는 다음과 같이 구해진다.

$$\begin{aligned}t^c &= C^c(t_r)\\ &= C^d(t_r)-\Phi_k+\frac{C^d(t_r)-C^d(t^k_r)}{T_{clk}}\Phi_k \quad (3)\\ &= (1+\frac{\Phi_k}{T_{clk}})C^d(t_r)-(1+\frac{C^d(t^k_r)}{T_{clk}})\Phi_k\end{aligned}$$

식 (3)에 따르면 $C^c(t_r)$은 연속 함수이다. 또한 대부분의 경우 $T_{clk}\geq|\Phi_k|$이 성립하므로 $(1+\frac{\Phi_k}{T_{clk}})>0$에 따라 $C^c(t_r)$은 단조 증가한다고 할 수 있다. 정리 1은 $C^c(t_r)$이 정상 클록임을 보여주고 있다.

정리 1. *C가 정상 클록이고 $k\geq 0$에 대해 $T_{clk}\geq|\Phi_k|$이면 C^c는 정상 클록이다. 즉 C^c는 다음을 만족한다.*

(P1) C^c의 드리프트 비율이 상한되어 있다.

(P2) C^c는 연속이다.

(P3) C^c는 단조 증가한다.

증명. 생략

3. 연속 클록 동기화를 위한 정적 변환 SCT

3.1 태스크 모델

일반적으로 실시간 태스크는 세가지 유형의 시간 제약을 갖는데, 이는 주기(period) 조건, 시작제한시간(release time) 조건, 종료시한(deadline) 조건으로 분류될 수 있다. 주기적인 태스크 τ_i의 j번째 수행을 $\tau_{i,j}$로 나타내자. 주기, 시작제한시간, 종료시한을 각각 $\langle T_i, r_i, d_i\rangle$로 나타내면, $\tau_{i,j}$의 절대 시작제한시간(absolute release time), 절대 종료시한(absolute deadline)은 다음과 같이 얻어진다.

$$R_{i,j}=jT_i+r_i,\ D_{i,j}=jT_i+d_i. \quad (4)$$

이외에 $\tau_{i,j}$가 실제로 수행되면서 정의될 수 있는 인자들로서는 수행시간(execution time), 시작 시간(start time), 종료 시간(finish time)이 있다. 이들을 각각 $e_{i,j}$, $S_{i,j}$, $F_{i,j}$로 나타내자. 만약 위에서 정의된 파라미터들이 C^c에 의해 측정된다면 이는 각각, $e^c_{i,j}$, $F^c_{i,j}$, $S^c_{i,j}$로 표현한다.

동일한 컴퓨터에서 수행되는 태스크들은 지역 스케쥴러 $\Pi(C^c, y)$에 의해 스케쥴되며, $\Pi(C^c, y)$는 지역 클록 C^c 와 스케쥴링 알고리듬 y에 의해 태스크들을 스케쥴한다..

3.2 실시간 스케쥴링

앞서 밝힌 바와 같이 클록 오차(skew)로 인해 지역 클록에 의해 스케쥴되는 임의의 태스크는 전역 클록에 대해 시간 제약을 만족시키지 못할 수 있다. 이 문제를 명확히 하기 위해 지역적 스케쥴링 가능성(local schedulability)과 전역적 스케쥴링 가능성(global

schedulability)을 다음과 같이 정의한다.

정의 2. 태스크 $\tau_{i,j}$가 $\Pi(C^x, y)$에 의해 스케줄되며 다음과 같은 조건을 만족한다면, $\tau_{i,j}$는 $\Pi(C^x, y)$에 의해 **지역적으로 스케줄링 가능하다고** 한다.

$$R_{i,j} \le S^x_{i,j} \le F^x_{i,j} \le D_{i,j} \quad (5)$$

정의 3. 태스크 $\tau_{i,j}$가 $\Pi(C^x, y)$에 의해 스케줄되며 다음과 같은 조건을 만족한다면, $\tau_{i,j}$는 $\Pi(C^x, y)$에 의해 **전역적으로 스케줄링 가능하다고** 한다.

$$R_{i,j} \le c^x(S_{i,j}^x) \le c^x(F_{i,j}^x) \le D_{i,j} \quad (6)$$

3.3 정적 변환 기법

정의 2와 3에 따르면 클록 오차로 인해 지역적으로 스케줄링 가능한 태스크는 전역적으로 스케줄링이 가능하다고 할 수 없다. 하지만 최대 클록 오차를 고려한 다음의 SCT에 의한 변환을 통해 전역적 스케줄링 가능성을 보장할 수 있다.

* SCT

1. $T_i \xrightarrow{SCT} \widehat{T}_i = T_i$.
2. $r_i \xrightarrow{SCT} \widehat{r}_i = r_i + \Delta$
3. $d_i \xrightarrow{SCT} \widehat{d}_i = d_i - \Delta$

정리 2. *주기적 태스크 τ_i가 $\Pi(C^c, y)$에 의해 지역적으로 스케줄링 가능하면, $\Pi(C^c, y)$전역적으로 스케줄링 가능하다.*
증명. 생략.

4. 이산 클록 동기화를 위한 동적 변환 DCT

DCT의 기본적인 개념은 종료시한이나 시작제한시간과 같은 태스크의 시간 제약을 이산 클록 함수상에서 연속 클록 함수상으로 변환시키는 것이다. 이를 위한 사상(mapping)은 무한히 많이 존재하지만 다음과 같은 특성을 지녀야한다.

C1. DCT로 변환된 시간 제약의 시간적 순서가 뒤바뀌지 않아야 한다.

C2. DCT로 변환된 태스크의 지역적 스케줄링 가능성의 분석이 용이해야 한다.

C2. DCT에 의해 전역적 스케줄링 가능성이 보장되어야 한다.

위와 같은 특성을 만족시키기 위해 다음과 같은 사상을 DCT에 이용한다.

$$\Theta: t^c \xrightarrow{c^c} t^r \xrightarrow{C^d} t^d \quad (7)$$

$$\Theta(t_c) = C^d(c^c(t^c)) = \left(\frac{T_{clk}}{T_{clk}+\Phi_k}\right)\left(t^c + \left(1 + \frac{C^d(c(kT_{clk}))}{T_{clk}}\right)\Phi_k\right). \quad (8)$$

위의 사상을 이용하여 DCT는 다음과 같이 정의된다.

* DCT

1. $\widehat{R} \xrightarrow{DCT} \widehat{R} = \Theta(\widehat{R})$
2. $\widehat{D} \xrightarrow{DCT} \widehat{D} \triangleq \Theta(\widehat{D})$

다음의 보조 정리 1과 정리 3, 4는 DCT에 의해 위의 조건 C1, C2, C3가 만족됨을 보여준다.

보조 정리 1. *$\widehat{T}$이 $\Pi(C^d, y)$에 의해 스케줄되고 $\widehat{T}$이 $\Pi(C^c, y)$에 의해 스케줄되면, 다음이 성립한다.*

$$c^c(\widehat{S^c_{i,j}}) = c^d(\widehat{\widehat{S^d_{i,j}}}),$$
$$c^c(\widehat{F^c_{i,j}}) = c^d(\widehat{\widehat{F^d_{i,j}}}).$$

증명. 생략.

정리 3. *태스크 $\widehat{\tau_{i,j}}$가 $\Pi(C^d, y)$에 의해 지역적으로 스케줄링 가능할 필요충분 조건은 태스크 $\widehat{\tau_{i,j}}$가 $\Pi(C^c, y)$에 의해 지역적으로 스케줄링 가능함이다.*
증명. 생략.

정리 4. *태스크 $\tau_{i,j}$가 $\Pi(C^d, y)$와 DCT에 의해 전역적으로 스케줄링 가능할 필요충분 조건은 태스크 $\tau_{i,j}$가 $\Pi(C^c, y)$와 SCT에 의해 전역적으로 스케줄링 가능함이다.*
증명. 생략.

5. 결론

본 논문에서는 클록 동기화를 실시간 시스템에 적용할 때 발생하는 문제점을 살펴보고 이에 대한 해결책으로서 시간적 제약의 정적/동적 변환 기법을 제안하였다. 정적 변환 기법인 SCT를 통해 지역적으로 스케줄링 가능한 태스크가 전역적으로 스케줄링 가능하도록 하였다. 동적 변환 기법인 DCT는 PLL을 이용한 별도의 하드웨어를 사용하지 않으면서 기존의 이산 클록 동기화상에서 발생하는 시간의 불연속성 문제를 소프트웨어적으로 해결하였다.

향후에는 제안된 SCT와 DCT를 본 연구실에서 개발한 실시간 운영체제인 ARX [1]에 구현하여 이의 오버헤드 및 성능을 측정하고자 한다. 나아가서는 고장 감내 시스템을 위해 시스템에 오류가 발생했을 때에도 주어진 시간 제약을 만족시킬 수 있도록 본 연구를 확장하고자 한다.

6. 참고 문헌

[1] Y. Seo. J. Park, and S. Hong. Supporting preemptive user-level threads for embedded real-time systems. Technical Report SNU-EE-TR-1998-1, School of Electrical Engineering, Seoul National University, August 1998.

[2] L. Lamport and P. M. Melliar-Smith. Synchronizing clocks in the presence of faults. *Journal of the ACM*, 32(1):52-78, January 1985.

[3] P. Pamanathan, D. D. Kandlur, and K. G. Shin. Hardware-assisted software clock synchronization for homogeneous distributed systems. *IEEE Transactions on Computers*, C-39(4):514-524, April 1990.

[4] K. G. Shin and P. Ramanathan. Synchronization of a large clock network in the presence of malicious faults. *IEEE Transactions on Computers*, C-36(1):2-12, January 1987.

[5] H. Kopetz. and W. Ochsenreiter. Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers*, C-36(8):933-940, August 1987.