분산 실시간 태스크의 스케쥴가능성 개선을 위한 지터 분석

(A Jitter Analysis for Improved Schedulability of Distributed Real-Time Tasks)

김 태 웅 † 신 현 식 †† 장 래 혁 ††
(Taewoong Kim) (Heonshik Shin) (Naehyuck Chang)

요 약 분산 실시간 시스템에서 선행 태스크의 완료에 의해 활성화되는 태스크는 활성화 지터를 갖는 주기적 태스크로 모델링될 수 있다. 태스크의 활성화 지터는 선행 태스크의 최악과 최선 응답시간의 차로 정의된다. 기존의 방법은 최선 응답시간을 실제보다 훨씬 작은 값으로 가정하기 때문에, 활성화 지터와 낮은 우선순위를 갖는 태스크들의 최악 응답시간이 과대평가된다. 본 논문은 최선 응답시간을 보다 정확하게 산정하고 활성화 지터의 한계를 줄이기 위해 새로운 최선 응답시간 분석기법을 제안한다. 제안된 기법은 태스크들간의 상대적 위상을 고려하여 최선 응답시간을 구한다. 활성화 지터의 정확한 분석은 다른 태스크들의 최악 응답시간을 감소시키고 스케쥴가능성을 증가시킨다. 모의실험의 결과는 제안된 분석 기법이 최선 응답시간과 최악 응답시간의 정확도를 각각 최대 40%와 6%로 개선함을 보여 준다.

Abstract In distributed real-time system, a task activated by the completion of its preceding task can be modeled as a periodic task with activation jitter. An activation jitter of a task is defined as the difference between the worst case and the best case response time of its preceding task. Because the existing approaches assume that the best case response time is much smaller than the actual one, the activation jitter and the worst case response time of lower priority tasks are overestimated. This paper proposes a new analysis technique to calculate the best case response time more precisely and to reduce the activation jitter bounds. The proposed technique obtains the best case response time by considering the relative phase between tasks. The precise analysis of the activation jitters can reduce the worst case response time of other tasks and increase the schedulability. The simulation results show that the proposed analysis technique improves the accuracy of the best case and the worst case response time up to 40% and 6%, respectively.

1. 서 론

분산 실시간 시스템은 경성 실시간 시스템을 구축하는 하나의 방법으로서 이에 대한 많은 연구가 수행되어 왔다[2, 3, 4, 5]. 전형적인 분산 실시간 시스템은 다수의 프로세서 노드와 이들을 연결하는 네트워크로 구성된다. 이러한 시스템의 예로는 공정 제어 시스템, 항공기 제어 시스템, 레이다 추적 시스템, 자동 항법 장치등이 있다. 분산 실시간 시스템에서 각 프로세서 노드에서 수행되는 태스크들은 지역 태스크(local task)와 종단 태스크(end-to-end task)로 분류된다. 지역 태스크는 주기적으로 활성화되어 해당 노드와 관련된 제어 루프를 수행하며 다른 노드에 있는 태스크와 통신을 하지 않는다. 여기에서 태스크의 활성화는 해당 태스크가 준비(ready) 상태가 되어 OS 스케쥴러의 실행 큐(run queue)에 삽입되는 것을 의미한다. 종단 태스크는 동일한 주기를 가지고 서로 다른 노드에서 수행되는 일련의 서브태스크들로 구성되며, 메시지 전송을 통해 서브태스

† 비 회 원 : 서울대학교 컴퓨터공학과
twkim@comp.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학과 교수
shinhs@comp.snu.ac.kr
naehyuk@comp.snu.ac.kr
논문접수 : 1999년 8월 25일
심사완료 : 2000년 3월 27일

크들간의 선행 관계를 유지한다. 즉, 첫번째 서브태스크는 타이머에 의해 주기적으로 활성화되지만, 두번째 서브태스크부터는 선행 태스크가 완료시에 보내는 메시지의 도착에 의해 활성화된다. 지역 태스크의 응답시간은 해당 태스크의 활성화 시점부터 완료시점까지 걸리는 시간을 의미하는 반면, 종단 서브태스크의 응답시간은 첫번째 서브태스크의 활성화 시점부터 해당 서브태스크의 완료 시점까지 걸리는 시간으로 정의된다. 그리고 종단 태스크의 종단 응답시간(end-to-end response time)은 첫번째 서브태스크의 활성화 시점부터 마지막 서브태스크의 완료 시점까지 걸리는 시간으로 정의된다. 본 논문에서 지역 태스크와 종단 태스크들은 경성 종료시한을 가진다고 가정한다. 따라서, 태스크 스케줄링 알고리즘은 태스크의 최악 응답시간이 항상 종료시한보다 작거나 같도록 보장해야 한다. 실시간 시스템의 태스크 스케줄링 알고리즘으로 고정 우선순위에 기반한 선점형 스케줄링이 널리 사용된다[8, 9]. 그 이유는 대부분의 실시간 운영체제에서 고정 우선순위 스케줄링을 지원하고 작은 실행시간 오버헤드를 가지기 때문이다.

종단 태스크의 서브태스크들은 선행 태스크의 완료와 메시지 도착에 의해 활성화되므로, 메시지 전달시간을 고려하지 않으면 서브태스크의 활성화간 시간은 선행 태스크의 최악 응답시간과 최선 응답시간의 차만큼 주기와 차이를 가질 수 있다. 평균적으로 보면 주기적으로 활성화되는 태스크의 활성화간 시간과 주기와의 편차를 태스크의 활성화 지터(activation jitter)라고 부르고, 선행 태스크의 최악 응답시간과 최선 응답시간의 차로 정의된다. 일반적으로 지역 태스크들과 종단 태스크의 첫번째 서브태스크는 선행 태스크를 가지지 않고 타이머에 의해 주기적으로 활성화되므로 활성화 지터는 0으로 가정한다.

실시간 태스크의 활성화 지터를 줄이는 스케줄링 방법으로는 DCTS(distance constrained task scheduling) [10, 11]와 순환 실행체제(cyclic executive)[12]가 있다. DCTS는 실시간 태스크의 주기를 조화(*harmonic*) 주기로 변환하여 스케줄하는 방식이다. 그러나, 이 방법은 높은 프로세서 이용률을 갖는 태스크 집합에 대해서 적용하기 어렵고[11], 변환된 태스크 집합이 프로세서 이용률을 증가시키는 단점을 가지고 있다. 순환 실행체제는 오프라인 분석을 통하여 생성된 스케줄 테이블에 따라 태스크를 수행하는 방식으로 활성화 지터를 줄일 수 있다. 그러나, 태스크 집합이 변경되면 스케줄 테이블을 새로 생성해야 하는 단점을 가지고 있다.

고정 우선순위에 기반한 실시간 태스크 스케줄링 알고리즘에서 활성화 지터를 가지는 태스크를 고려한 최악 응답시간 분석 기법은 Tindell에 의해 제안되었다[6, 7]. Tindell의 분석 방법에서는 선행 태스크의 최선 응답시간을 0에 가까운 아주 작은 값으로 가정하기 때문에 태스크의 활성화 지터는 선행 태스크의 최악 응답시간과 같은 값을 가지게 된다. 그러나, 일반적인 경우 태스크의 최선 응답시간은 0이라고 볼 수 없기 때문에 Tindell의 태스크 모델에서의 활성화 지터는 실제값보다 훨씬 크게 된다. 태스크의 활성화 지터를 정확하게 계산하기 위해서는 선행 태스크의 최선 응답시간을 구해야 한다. Gutierrez 외 저자들은 태스크의 활성화 지터를 계산하기 위해 태스크의 최선 응답시간 분석기법을 제안하였고 이를 본 논문에서는 GGH 방법이라고 부른다[1]. 그러나, GGH 방법은 실제로 발생할 수 없는 상황을 최선 경우로 가정하여 태스크의 최선 응답시간을 구하기 때문에 실제 최선 응답시간보다 작은 값을 산출한다. 따라서, 태스크의 활성화 지터를 실제보다 큰 값으로 추정한다. 본 논문은 GGH 방법을 개선한 최선 응답시간 분석기법을 제안한다. GGH 방법에서는 고려되지 않은 활성화 지터를 갖지 않는 태스크를 포함하여 최선 응답시간을 분석하고, 구해진 활성화 지터에 의해 분산 실시간 태스크의 스케줄가능성을 분석한다.

본 논문의 구성은 다음과 같다. 2절에서 분산 실시간 시스템의 모델과 가정에 대해 기술한다. 3절에서 실시간 태스크의 최악·최선 응답시간 분석기법을 제안하고, 종단 최악 응답시간의 계산에 종단 서브태스크들간의 네트워크를 통한 메시지 전송 지연시간을 고려한 분석기법을 제시한다. 4절에서 제안된 최선 응답시간 분석기법을 기존의 방법과 모의실험을 통해 비교하고 활성화 지터가 태스크들의 최악 응답시간에 미치는 영향을 평가한다. 5절에서 결론 및 앞으로의 연구방향에 대해 기술한다.

2. 시스템 모델

분산 실시간 시스템에서 n개의 태스크 $\tau_1, \tau_2, \ldots, \tau_n$은 네트워크로 연결된 다수의 프로세서 노드에 정적으로 할당되어 수행된다[2, 3, 4, 5]. 각 프로세서 노드에서 수행되는 태스크들은 지역 태스크와 종단 태스크로 분류된다. 지역 태스크는 주기적으로 해당 노드와 관련된 제어 루프를 수행하며 다른 태스크와 통신을 하지 않는다. 종단 태스크는 동일한 주기를 가지고 수행되는 일련의 서브태스크들로 구성되며, 메시지 전달을 통해 서브태스크들간의 선행 관계를 유지한다. 서브태스크 τ_i가

서브태스크 τ_j보다 먼저 수행되면 $\tau_i \Rightarrow \tau_j$로 나타낸다. 네트워크를 통한 메시지 전송 지연시간을 0으로 가정한다면 τ_j는 τ_i가 수행을 완료함과 동시에 활성화된다고 볼 수 있다. 최선 응답시간과 최악 응답시간의 분석을 용이하게 하기 위해 일단 네트워크에 의한 메시지 전송 지연시간을 0으로 가정하고, 3.3절에서 메시지 전송 지연시간을 고려하기로 한다.

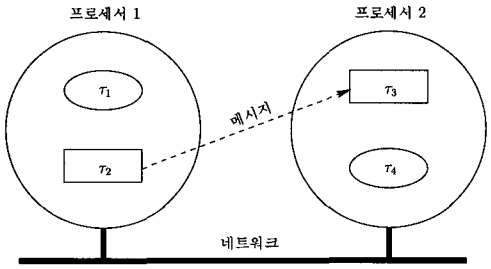


그림 1 분산 실시간 시스템의 예

표 1 그림 1에 제시된 태스크들의 매개변수

태스크		수행시간	주기	선행 관계
프로세서 1	τ_1	2	5	$\tau_2 \Rightarrow \tau_3$
	τ_2	3	7	
프로세서 2	τ_3	2	7	
	τ_4	6	10	

분산 실시간 시스템의 예로서 그림 1은 2개의 프로세서 노드와 4개의 태스크로 구성된 시스템을 나타낸다. 그림 1에 제시되어 있는 태스크들은 표 1과 같은 수행시간, 주기, 선행 관계를 갖는다. 그림 1에서 τ_1과 τ_4는 지역 태스크를 나타내고 τ_2와 τ_3은 종단 태스크의 서브태스크를 나타낸다. τ_1, τ_2, τ_4는 자신의 주기마다 활성화되어 수행되는 반면, τ_3은 선행 관계를 유지하기 위해 τ_2가 보낸 메시지가 도착하면 활성화되어 수행을 하게 된다. τ_2가 주기적으로 수행되어 메시지를 전송하므로 τ_3는 평균적으로는 선행 태스크 τ_2와 같은 주기로 활성화된다. 그러나, 실제로 τ_3의 활성화는 그림 2에 제시된 것과 같이 τ_2의 주기와 일치하지 않을 수 있다. τ_3의 활성화는 선행 태스크 τ_2의 완료시점에 의존적이기 때문에 τ_3의 활성화간 걸리는 시간은 5와 9로 τ_2의 주기(=7)보다 작거나 클 수 있다. 그림 2에서 태스크의 스케쥴링은 고정 우선순위에 기반한 RMS 알고리즘[8]이 사용되고 네트워크를 통한 메시지 전송 지연시간은 0으로 가정하였다. RMS를 사용하기 때문에 프로세서 1에서 τ_1은 τ_2보다 높은 우선순위를 가지고, 프로세서 2에서 τ_3는 τ_4보다 높은 우선순위를 가진다. τ_3의 첫번째 활성화(시점 5)는 τ_2가 최악 응답시간을 나타내는 경우이고, τ_3의 두번째 활성화(시점 10)는 τ_2가 최선 응답시간을 나타내는 경우이고, τ_3의 세번째 활성화(시점 19)는 τ_2가 다시 최악 응답시간을 나타내는 경우이다. τ_3의 활성화간 시간은 최악 응답시간과 최선 응답시간의 차만큼 선행 태스크의 주기와 차이를 가질 수 있다. 즉, 그림 2에서 τ_2의 최악 응답시간은 5이고 최선 응답시간은 3이므로 τ_3의 활성화 지터는 2임을 알 수 있다.

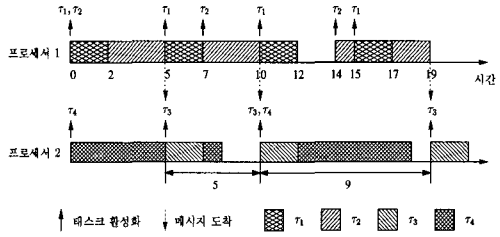


그림 2 지역 태스크와 종단 서브태스크들의 수행

본 논문에서 사용되는 주요 기호들을 표 2에 나타낸다. 표 2에서 τ_i는 지역 태스크 또는 종단 태스크의 서브태스크를 의미한다. 태스크 τ_i의 실제 수행 시간은 $c_i^{\min}$과 $c_i^{\max}$사이의 값을 가진다. 지역 태스크 τ_i의 경우 태스크의 상대적 경성 종료시한은 주기 p_i와 같다. 따라서, τ_i의 최악 응답시간 R_i^w는 p_i보다 작거나 같아야 한다. 종단 태스크가 τ_j, τ_k, τ_l의 서브태스크들로 구성되고 $\tau_j \Rightarrow \tau_k \Rightarrow \tau_l$의 선행 관계를 가진다고 하자. 종단 태스크의 서브태스크들은 동일한 주기를 가진다($p_j = p_k = p_l$). 이 종단 태스크의 종단 최악 응답시간은 마지막 서브태스크의 최악 응답시간 R_l^w으로 정의되고 종단 종료시한인 주기보다 작거나 같아야 한다. 종단 태스크의 서브태스크들은 활성화 지터를 가질 수 있다. 일반적으로 지역 태스크들과 종단 태스크의 첫번째 서브태스크는 타이머에 의해 주기적으로 활성화되므로 활성화 지터는 0으로 가정한다.

표 2 주요 기호

기호	의미
$c_i^{\min}$	τ_i 의 최소 수행시간
$c_i^{\max}$	τ_i 의 최대 수행시간
p_i	τ_i 의 주기
prec(j)	prec(j) = i if $\tau_i \Rightarrow \tau_j$
$hp(i)$	τ_i 가 수행되는 프로세서에서 τ_i 보다 높은 우선순위를 갖는 태스크들의 집합
R_i^b	τ_i 의 최선 응답시간
R_i^w	τ_i 의 최악 응답시간
J_i	종단 서브태스크 τ_i 의 활성화 지터: $R^w_{\text{prec}(i)} - R^b_{\text{prec}(i)}$

3. 실시간 태스크의 스케줄가능성 분석

3.1 최악 응답시간 분석

실시간 태스크의 스케줄가능성은 태스크의 최악 응답시간을 구한 후 해당 태스크의 종료시한보다 작거나 같은가로 판별한다[6, 15]. 실시간 태스크가 활성화 지터를 가지게 되면 활성화간 시간이 자신의 주기보다 작을 수 있다. 이러한 상황이 발생하면 낮은 우선순위를 가지는 태스크의 최악 응답시간에 영향을 미치게 된다. 활성화 지터를 고려한 최악 응답시간의 분석은 모든 태스크가 동시에 활성화되는 임계 시간(critical instant)[8]이 높은 우선순위를 가지는 태스크가 활성화 지터만큼 늦게 활성화된 시간과 일치하는 상황에 기초를 두고 있다. 실시간 태스크의 최악 응답시간을 나타내면 수식 (1)과 같다.

$$R_i^w = \begin{cases} \Delta_i^w, & \tau_i \text{가 지역 태스크} \\ \Delta_i^w + R^w_{\text{prec}(i)}, & \tau_i \text{가 종단 서브태스크} \end{cases} \quad (1)$$

$$\begin{aligned} \Delta_i^w(0) &= c_i^{\max} \\ \Delta_i^w(k+1) &= c_i^{\max} + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{J_j + \Delta_i^w(k)}{p_j} \right\rceil c_j^{\max} \end{aligned} \quad (2)$$

수식 (1)에서 Δ_i^w는 수식 (2)에 제시된 $\Delta_i^w(k)$의 수렴값으로 정의되고 태스크 τ_i가 활성화되고 완료될 때까지 걸리는 최대 시간을 의미한다. B_i는 공유 자원에 대한 동기화 프로토콜인 PCP(priority ceiling protocol)를 사용할 때 발생하는 낮은 우선순위를 갖는 태스크의 수행에 의한 지연시간을 의미한다[13]. 태스크 τ_i가 종단 서브태스크이면 최악 응답시간은 수식 (1)과 같이 선행 태스크의 최악 응답시간과 자신이 활성화되어 완료할 때까지 걸리는 시간으로 정의된다. 그리고, 태스크 τ_i가 종단 태스크의 첫번째 서브태스크인 경우에는 선행 태스크가 없으므로 수식 (1)에서 $R^w_{\text{prec}(i)}=0$으로 정의된다. 수식 (2)에 제시된 점화식을 이용한 Δ_i^w의 계산은 Tindell의 분석기법에 기반을 두고 있다[6].

종단 태스크의 서브태스크 τ_i의 활성화 지터 J_i는 표 2와 같이 선행 태스크의 최악 응답시간과 최선 응답시간의 차로 계산된다. Tindell의 모델에서는 R_i^b를 0에 가까운 아주 작은 값으로 가정하기 때문에 서브태스크의 활성화 지터는 선행 태스크의 최악 응답시간과 같은 값을 가지게 된다($J_i = R^w_{\text{prec}(i)}$). 그러나, 일반적인 경우 태스크의 최선 응답시간은 0이라고 볼 수 없기 때문에, Tindell의 태스크 모델에서의 활성화 지터는 실제값보다 훨씬 크게 된다. 실제값보다 크게 산정된 활성화 지터는 해당 태스크보다 낮은 우선순위를 가지는 태스크의 최악 응답시간을 증가시키기 때문에 태스크의 최악 응답시간 분석의 정확도를 떨어뜨리게 된다. 또한 태스크의 최악 응답시간이 실제보다 크게 계산되고 해당 태스크가 후행 태스크를 가지면, 후행 태스크의 활성화 지터도 증가되어 연쇄적으로 부정확한 최악 응답시간을 산출하게 된다. 그림 3은 부정확한 활성화 지터 예측이 태스크의 최악 응답시간에 미치는 영향을 보여주고 있다.

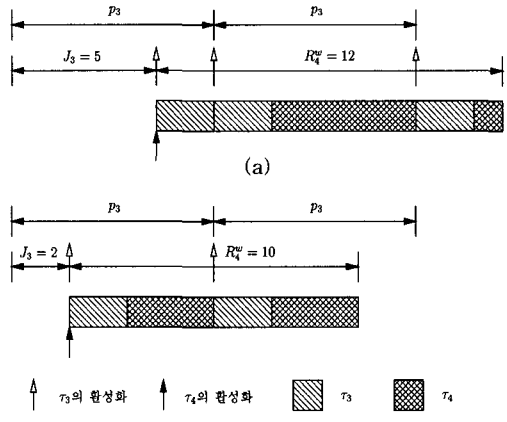


그림 3 τ_4의 최악 응답시간 분석

그림 3은 표 1에 제시된 τ_4의 최악 응답시간을 수식 (1)과 (2)를 이용하여 분석한 것이다. 프로세서 2에서 τ_4는 τ_3보다 우선순위가 낮기 때문에 τ_3의 활성화 지터

J_3에 의해 최악 응답시간이 영향을 받게 된다. 즉, 활성화 지터 J_3만큼 늦게 활성화된 τ_3는 주기마다 활성화되고, τ_4는 이러한 τ_3와 동시에 활성화될 때 최악 응답시간을 나타낸다. 그림 3의 (a)에 나타낸 바와 같이 Tindell의 모델을 적용하여 J_3을 선행 태스크 τ_2의 최악 응답시간인 5로 가정하여 τ_4의 최악 응답시간 R_4^w을 구하면, τ_4의 최악 응답시간은 12이므로 종료시한 10을 만족시키지 못한다. 그러나, 그림 3의 (b)는 그림 2에서 보인 바와 같이 선행 태스크 τ_2의 최악 응답시간과 최선 응답시간의 차인 2로 J_3을 정확하게 계산하여 τ_4의 최악 응답시간을 구하면 종료시한이 만족됨을 보여주고 있다.

3.2 최선 응답시간 분석

한 프로세서 노드에서 수행되는 실시간 태스크가 최선 응답시간을 나타내는 상황은 자신보다 우선순위가 높은 태스크에 의해 선점되는 시간이 가장 작고, 태스크 자신의 수행시간이 가장 작을 때 발생한다. 한 태스크가 우선순위가 높은 태스크들에 의해 선점되는 시간이 가장 작은 경우는 자신보다 우선순위가 높은 태스크들이 최선 응답시간으로 완료됨과 동시에 해당 태스크가 활성화되고 우선순위가 높은 태스크들의 다음번 활성화는 가능한한 늦게 발생해야 한다. 우선순위가 높은 태스크들의 활성화가 가능한한 늦게 발생해야 하는 이유는 선점의 영향을 최소화하기 위해서이다. GGH 방법은 태스크의 최선 응답시간을 계산할 때 자신보다 높은 우선순위를 갖는 태스크들이 모두 비활성화되어 있다고 가정한다[1]. 그러나, 이와같은 가정은 활성화 지터를 갖지 않는 태스크들이 존재하는 경우에는 적용되지 않기 때문에 실제 최선 응답시간보다 훨씬 작은 값을 얻게 된다.

예를 들면, 표 3과 같은 태스크 집합이 있을 때 τ_1은 지역 태스크를 나타내고, τ_2는 종단 태스크의 첫번째 서브태스크를 나타낸다. 2절에서 기술한 바와 같이 τ_1과 τ_2는 활성화 지터를 갖지 않는다. τ_2의 최선 응답시간을 GGH 방법으로 구하면, 그림 4의 (a)과 같이 τ_1이 완료된 직후 τ_2가 활성화되어(시점 8) 최소 수행시간인 3만큼 수행하는 것을 가정하여 구할 수 있다. τ_1의 주기가 10이므로 τ_2는 단위 시간 2만큼 수행 후(시점 10) τ_1에 의해 선점되고 시점 19에서 완료된다. 따라서, GGH 방법에서 구한 τ_2의 최선 응답시간은 11이다. 그러나, τ_1과 τ_2의 주기와 활성화 지터를 갖지 않는다는 점을 고려하면 그림 4의 (a)와 같은 상황은 실제로는 발생하지 않는다. 즉, τ_2의 주기는 τ_1의 주기의 배수이므로 그림 4의 (b)와 같이 τ_2는 항상 τ_1과 동시에 활성화된다. 따라서, τ_2의 최선 응답시간은 19가 된다. τ_2의 최악 응답시간은 20이므로 τ_3의 활성화 지터 J_3은 GGH 방법에 의하면 $20-11=9$로 계산되는 반면, 개선된 방법에 의하면 $20-19=1$로 계산된다. 이상에서 기존의 GGH 방법을 사용하면 실제 태스크들의 수행시 발생할 수 없는 상황을 가정하여 최선 응답시간을 분석하므로 실제 최선 응답시간보다 훨씬 작은 최선 응답시간을 얻게됨을 보였다. 이와같이 최선 응답시간이 부정확하게 추정되면 결국 후행 태스크의 활성화 지터를 증가시켜 정확한 태스크의 스케쥴가능성 분석을 방해하게 된다.

표 3 최선 응답시간 분석을 위한 태스크들의 매개변수

태스크		$(c_i^{\min}, c_i^{\max})$	p_i	선행 관계
프로세서 1	τ_1	(8, 8)	10	$\tau_2 \Rightarrow \tau_3$
	τ_2	(3, 4)	30	
프로세서 2	τ_3	(3, 5)	30	

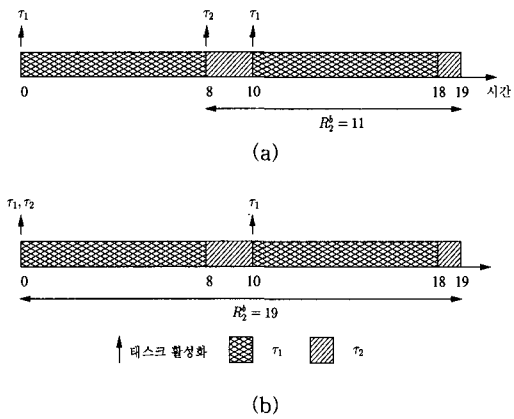


그림 4 표 3에 제시된 τ_2의 최선 응답시간 분석
(a) GGH 방법, (b) 개선된 방법

제안된 최선 응답시간 분석은 활성화 지터를 갖지 않는 경우와 활성화 지터를 갖는 경우에 따라 최선 응답시간을 나타내는 태스크들간의 상대적 위상(phase)이 달라짐에 근거를 두고 있다. 지역 태스크와 종단 태스크의 서브태스크들의 최선 응답시간은 수식 (3)과 같이 정의된다. 수식 (3)에서 Δ_i^b는 태스크 τ_i가 활성화되어 완료하기까지 걸리는 최소 시간을 의미한다.

$$R_i^b = \begin{cases} \Delta_i^b, & \tau_i \text{가 지역 태스크} \\ \Delta_i^b + R_{\text{prec}(i)}^b, & \tau_i \text{가 종단 서브태스크} \end{cases} \quad (3)$$

태스크 τ_i가 최선의 경우로 완료되기 위해서는 τ_i가 최소 수행시간 $c_i^{\min}$을 필요로 하고 자신보다 높은 우선순위를 갖는 태스크들에 의해 선점되는 시간이 가장 작아야 한다. Δ_i^b는 자기자신의 수행시간과 높은 우선순위 태스크들에 의해 선점되는 최소 시간의 합으로 구해지며, 수식 (4)와 같은 점화식으로 나타낼 수 있다. Δ_i^b는 $\Delta_i^b(k)$의 수렴값으로 정의된다.

$$\begin{aligned} \Delta_i^b(0) &= c_i^{\min} \\ \Delta_i^b(k+1) &= c_i^{\min} + \sum_{\forall j \in hp(i)} \left\lceil \frac{\max[0, \Delta_i^b(k) - x_j]}{p_j} \right\rceil c_j^{\min} \end{aligned} \quad (4)$$

where

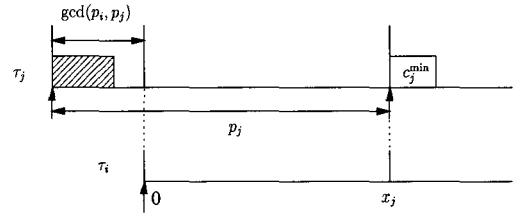
$$x_j = \begin{cases} p_j - \max[1, \lfloor \Delta_j^b / \gcd(p_i, p_j) \rfloor] \gcd(p_i, p_j) & \text{if } J_i = 0 \wedge J_j = 0 \\ p_j + J_j - \Delta_j^b & \text{otherwise} \end{cases}$$

수식 (4)에서 합항은 높은 우선순위 태스크들에 의해 선점되는 시간을 포함시키기 위해 필요하다. 그리고 x_j는 τ_i의 활성화 시점을 기준으로 하여 태스크 τ_i보다 높은 우선순위를 가지는 태스크 τ_j가 활성화되는 시점을 의미한다. 태스크 τ_i가 선점되는 시간과 태스크 τ_j의 활성화 시점 x_j는 태스크들간 상대적 위상에 따라 그 값이 달라지므로, 태스크들이 활성화 지터를 갖지 않는 경우($J_i=0 \wedge J_j=0$)와 활성화 지터를 갖는 경우($J_i \neq 0 \vee J_j \neq 0$)로 구분하여 구해야 한다.

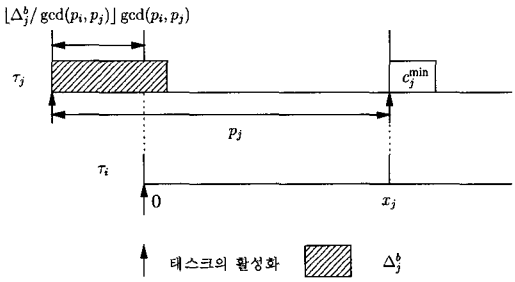
Case 1. 활성화 지터를 갖지 않는 경우

활성화 지터를 갖지 않는 태스크 τ_i와 τ_j가 있다고 가정하자. 태스크 τ_i는 주기의 배수($kp_i, k=0,1,2,\ldots$)가 되는 시점마다 활성화되어 수행되고, 태스크 τ_j는 태스크 τ_i보다 높은 우선순위를 가지며, 주기의 배수($lp_i, l=0,1,2,\ldots$)가 되는 시점마다 활성화되어 수행된다. RMS 알고리즘을 사용하는 경우 태스크 τ_i보다 높은 우선순위를 갖는 τ_j는 p_i보다 작거나 같은 주기 p_j를 갖는다. 1) p_i가 p_j의 배수인 경우에 τ_i는 τ_j와 항상 동시에 활성화된다. 즉, τ_i는 τ_j가 완료된 후 수행을 시작할 수 있게 되므로 τ_i의 최선 응답시간은 τ_j의 수행시간만큼 증가하게 된다. 따라서 τ_i가 수행을 완료할 때까지 τ_j에 의해 선점되는 최소 시간은 $\left\lceil \frac{\Delta_i^b}{p_j} \right\rceil c_j^{\min}$와 같다.

2) p_i가 p_j의 배수가 아닌 경우에 τ_i와 τ_j의 활성화 시점간의 차($kp_i - lp_j$)는 주기들의 최대 공약수 $\gcd(p_i, p_j)$의 배수와 같다. 그림 5의 (a)와 같이 $\Delta_j^b \leq \gcd(p_i, p_j)$이면, τ_j의 활성화시점부터 $\gcd(p_i, p_j)$시간 후에 τ_i가 활성화되는 경우가 τ_j에 의한 선점이 가장 작게 발생하는 경우이다. τ_i의 활성화 시점을 기준 시점(0)으로 하여 τ_j가 다시 활성화되는 시점 x_j는 $p_j - \gcd(p_i, p_j)$와 같다.



(a)



(b)

그림 5 활성화 지터가 없는 경우의 최선 응답시간: (a) $\Delta_j^b \leq \gcd(p_i, p_j)$, (b) $\Delta_j^b > \gcd(p_i, p_j)$

$\Delta_j^b > \gcd(p_i, p_j)$일 때, τ_i가 최선 응답시간을 나타내려면 τ_i와 τ_j의 활성화 시점간의 차가 $\gcd(p_i, p_j)$의 $\lfloor \Delta_j^b / \gcd(p_i, p_j) \rfloor$ 배이거나 $\lceil \Delta_j^b / \gcd(p_i, p_j) \rceil$ 배이어야 한다. 전자의 경우는 τ_j의 완료전에 τ_i가 활성화되는 경우이고 후자의 경우는 τ_j의 완료 후에 τ_i가 활성화되는 경우이다. 전자가 최선 응답시간을 나타내는 경우는 전자의 활성화 시점에서는 τ_j에 의한 선점이 발생하지 않지만 후자의 활성화 시점에서는 τ_j에 의한 선점이 발생하는 경우이다. 후자가 최선 응답시간을 나타내는 경우는 전자의 활성화 시점과 후자의 활성화 시점에서 모두 τ_j에 의한 선점이 발생하는 경우이다. 오프라인 분석에서 구하는 최선 응답시간은 실제 응답시간의 하한을 구하는 것이므로 전자의 경우와 후자의 경우보다 작거나 같은 값을 얻으면 된다. 따라서, 그림 5의 (b)와 같이 τ_i와 τ_j의 활성화 시점간의 차가 $\gcd(p_i, p_j)$의 $\lfloor \Delta_j^b / \gcd(p_i, p_j) \rfloor$ 배로 가정하고, τ_j의 수행에 의해 지연

되는 시간을 오프라인에 정확하게 계산할 수 없으므로 0으로 가정한다. 그림에서 보는 바와 같이 x_j는 $p_j - \lfloor \Delta_j^b / \gcd(p_i, p_j) \rfloor \gcd(p_i, p_j)$와 같다. 그림 5의 (a)와 (b)를 종합하여 τ_i가 수행을 완료할 때까지 τ_j에 의해 선점되는 최소 시간은 수식 (5)와 같이 구할 수 있다.

$$\left\lceil \frac{\max[0, \Delta_i^b - x_j]}{p_j} \right\rceil c_j^{\min} \quad (5)$$

where $x_j = p_j - \max[1, \lfloor \Delta_j^b / \gcd(p_i, p_j) \rfloor] \gcd(p_i, p_j)$

수식 (5)는 1) p_i가 p_j의 배수인 경우에도 적용할 수 있다. p_i가 p_j의 배수이면 $\gcd(p_i, p_j)$는 p_j와 같고, Δ_j^b는 p_j보다 작거나 같아야 하므로 수식 (5)에서 x_j는 0으로 계산된다. 따라서, 활성화 지터를 갖지 않는 경우에 τ_j에 의해 선점되는 최소 시간은 수식 (5)와 같이 구할 수 있다.

Case 2. 활성화 지터를 갖는 경우

태스크 τ_i와 τ_i보다 높은 우선순위를 갖는 태스크 τ_j가 있다고 하자. 먼저, τ_i의 활성화 지터 유무에 상관없이 τ_j는 활성화 지터를 갖는다고 가정하자. 태스크 τ_i는 τ_j가 수행을 완료하여 비활성화되어 있을 때, 활성화되면 최선 응답시간을 나타낼 수 있다. 활성화 지터를 갖는 경우는 활성화 시점이 일정하지 않기 때문에 GGH 방법과 마찬가지로, 그림 6과 같이 τ_j가 τ_i의 활성화 시점보다 Δ_j^b전에 활성화되었다고 가정한다[1]. 태스크 τ_j의 활성화간 최대 시간은 $p_j + J_j$이므로, 태스크 τ_i가 태스크 τ_j에 의해 선점되는 최소 시간은 수식 (6)과 같이 구할 수 있다.

$$\left\lceil \frac{\max[0, \Delta_i^b - x_j]}{p_j} \right\rceil c_j^{\min} \quad (6)$$

where $x_j = p_j + J_j - \Delta_j^b$

태스크 τ_i는 활성화 지터를 가지고, 태스크 τ_j가 활성화 지터를 갖지 않으면 수식 (6)에서 J_j = 0으로 설정하여 x_j를 계산한다. 따라서, 활성화 지터가 갖는 경우에 τ_j에 의해 선점되는 최소 시간은 수식 (6)과 같이 구할 수 있다.

수식 (5)와 (6)을 종합하면, 수식 (4)와 같이 Δ_i^b의 계산에 τ_j에 의해 선점되는 시간을 포함시킬 수 있다. 수식 (3)과 (4)를 이용하여 태스크 τ_i의 최선 응답시간을 구하기 위해서는 Δ_j^b의 값이 필요하므로, 각 프로세서 노드에서 우선순위 순서대로 태스크의 최선 응답시간을 구한다.

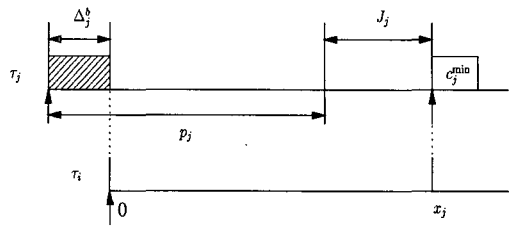


그림 6 활성화 지터를 갖는 경우의 최선 응답시간

지금까지 제안된 최선 응답시간 분석기법으로 태스크 응답시간의 하한을 구하는 방법을 살펴보았다. 기존의 연구인 GGH 방법에서 제시한 최선 응답시간 분석기법도 태스크 응답시간의 하한으로 사용될 수 있다. 그러나, GGH 방법은 태스크들간의 위상을 고려하지 않았기 때문에, 실제 최선 응답시간보다 훨씬 작은 값으로 최선 응답시간을 산출하여 활성화 지터를 실제보다 크게 계산하게 만든다. 제안된 최선 응답시간 분석기법은 GGH 방법의 단점을 개선하여 실제 최선 응답시간에 근접한 최선 응답시간을 산출한다.

정리 1. 수식 (3)과 (4)를 이용한 제안된 최선 응답시간 기법은 GGH 방법의 최선 응답시간 ≤ 본 논문의 최선 응답시간 ≤ 실제 최선 응답시간의 관계를 만족시킨다.

증명) 1. GGH 방법의 최선 응답시간 ≤ 본 논문의 최선 응답시간

GGH 방법과 본 논문의 방법은 높은 우선순위 태스크에 의한 선점시간의 계산 방식이 다르다. 수식 (4)에서 알 수 있듯이, 우선순위가 높은 태스크 τ_j의 활성화 시점인 x_j의 값이 작을수록 선점에 의한 영향이 커지므로 태스크 τ_i의 응답시간은 증가하게 된다. 태스크들이 활성화 지터를 갖지 않는 경우에서 GGH 방법에서는 x_j는 $p_j - \Delta_j^b$로 구할 수 있다[1]. 제안된 방법은 그림 5와 같이, $\Delta_j^b \le \gcd(p_i, p_j)$인 경우에 x_j는 $p_j - \gcd(p_i, p_j)$로 계산하여 GGH 방법보다 x_j를 작거나 같게 계산하여 GGH 방법보다 최선 응답시간이 커지게 된다. $\Delta_j^b > \gcd(p_i, p_j)$의 경우에 제안된 방법의 x_j값은 GGH 방법의 x_j값보다 크게 계산되어 GGH 방법보다 τ_j에 의한 선점 시간이 작게 계산될 수 있다. 그러나, $\Delta_j^b \le \gcd(p_i, p_j)$의 경우까지 종합하여 고려하면 최선 응답

시간은 GGH 방법보다 크게 계산된다. 4.1절의 실험 결과가 이를 뒷받침한다. 태스크들이 활성화 지터를 가지는 경우는 GGH 방법과 동일하게 x_j를 구하여 최선 응답시간을 구하므로 부등식을 역시 만족시킨다.

2. 본 논문의 응답시간 ≤ 실제 최선 응답시간

태스크 τ_i의 실제 실행시간은 $c_i^{\min}$보다 크거나 같고 $c_i^{\max}$보다 작거나 같다. 최선 응답시간은 태스크들의 응답시간이 가장 작아지는 상대적 위상에서 최소 실행시간을 갖는 경우로 계산하기 때문에 실제 최선 응답시간보다 항상 작거나 같게 된다. □

3.3 네트워크를 통한 메시지 전송 지연시간

3.2절까지의 최선 응답시간과 최악 응답시간 분석은 네트워크를 통한 메시지 전송 지연시간을 0으로 가정하여 수행되었다. 본 절에서 분산 실시간 응용에 널리 사용되는 CAN(controller area network) 버스[14]를 대상으로 하여 메시지 전송 지연시간을 고려하도록 하겠다. CAN 버스는 최대 1 Mbps의 속도를 지원하고 프레임 헤더의 식별자(identifier)를 사용하여 메시지의 종류와 우선순위를 표현한다. CAN 버스의 매체 접근 제어는 CSMA/CD에 우선순위에 의한 중재를 추가한 형태이다. 즉, 매체가 사용되지 않는 것을 감지하면 각 노드는 프레임의 첫부분인 메시지의 식별자를 전송하여 전송을 시도한다. 여러 노드가 동시에 전송을 시도하면 가장 높은 우선순위를 갖는 식별자를 보낸 노드를 제외한 노드들은 충돌을 감지하게 되고 전송을 중단한다. 만약 매체가 사용중이면 현재 전송중인 메시지의 전송이 완료되어 매체가 사용되지 않을 때까지 각 노드는 전송 시도를 연기한다. 따라서, CAN 버스에서는 대기중인 메시지 중 가장 우선순위가 높은 메시지가 선점없이 전송된다.

서브태스크들간의 메시지 전송 지연시간을 고려하기 위해 그림 7과 같이 CAN 버스를 프로세서로, 메시지를 종단 태스크의 서브태스크로 모델링한다. 그림 7은 태스크간의 선행 관계 $\tau_{send} \Rightarrow \tau_{recv}$를 메시지 전송을 고려하기 위해 $\tau_{send} \Rightarrow \tau_m \Rightarrow \tau_{recv}$으로 변환한 것이다. 기존의 연구 [1, 5]에서도 그림 7과 유사한 모델은 제시하였으나, 실제 네트워크에 대한 분석은 수행되지 않았다. τ_m의 수행시간 $c_m^{\min}$은 메시지 m이 최소 길이를 가질 때의 전송 시간을 나타내고, $c_m^{\max}$는 메시지 m이 최대 길이를 가질 때의 전송 시간을 나타낸다. 주기 p_m은 τ_{send}의 주기 p_{send}와 같고 활성화 지터 J_m은 $R^w_{\text{prec}(m)} - R^b_{\text{prec}(m)}$로 정의된다. $R^w_{\text{prec}(m)}$와 $R^b_{\text{prec}(m)}$는 각각 τ_{send}의 최악 응답시간과 최선 응답시간을 의미한다.

CAN 버스에서의 메시지 전송은 프로세서에서 고정

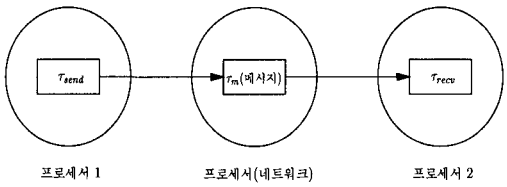


그림 7 네트워크를 통한 메시지 전송의 모델

우선순위에 기반한 비선점 스케줄링 알고리즘을 사용하여 태스크를 수행시키는 것과 동일하다. 메시지가 전송이 시작되면 다른 메시지에 의해 선점되지 않기 때문에, τ_m의 최선 응답시간은 메시지가 활성화됨과 동시에 전송이 시작되는 경우에 발생한다. 따라서, 수식 (7)과 같이 나타낼 수 있다.

$$R^b_m = c_m^{\min} + R^b_{\text{prec}(m)} \tag{7}$$

τ_m의 최악 응답시간은 수식 (8)과 같이 활성화 후 전송을 시작할 때까지의 최대 대기시간 (Q^w_m), 최대 전송시간 ($c_m^{\max}$), 그리고 선행 태스크의 최악 응답시간 ($R^w_{\text{prec}(m)}$)의 합으로 구할 수 있다[14].

$$R^w_m = Q^w_m + c_m^{\max} + R^w_{\text{prec}(m)} \tag{8}$$

$$\begin{aligned} Q^w_m(0) &= 0 \\ Q^w_m(k+1) &= B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{Q^w_m(k) + J_j + t_{bit}}{p_j} \right\rceil c_j^{\max} \end{aligned} \tag{9}$$

Q^w_m은 수식 (9)에 제시된 $Q^w_m(k)$의 수렴값으로 정의된다. 수식 (9)에서 t_{bit}는 CAN 버스에서 한 비트를 전송하는 시간을 의미하고, 매체 접근 직전에 활성화된 높은 우선순위를 갖는 메시지에 의한 대기시간 증가를 고려하기 위해 필요하다. B_m은 수식 (10)과 같이 정의되고 메시지가 활성화되었을 때 이미 전송중인 낮은 우선순위를 갖는 메시지에 의해 지연되는 시간을 의미한다. $hp(m)$은 m보다 높은 우선순위를 갖는 메시지들의 집합을 의미한다.

$$B_m = \max_{\forall j \in lp(m)} \{c_j^{\max}\} \tag{10}$$

수식 (10)에서 $lp(m)$은 m보다 낮은 우선순위를 갖는 메시지들의 집합을 의미한다.

이상에서 메시지 전송 지연시간의 최선과 최악 응답시간을 유도하였다. 메시지를 수신하는 태스크 τ_{recv}의 활성화 지터는 정의에 의하여 $J_{recv} = R^w_m - R^b_m$로 계산하여 3.1절과 3.2에 제시된 분석기법을 이용하여 종단 응답시

간을 구한다.

4. 성능 평가

4.1 최선 응답시간의 비교

제안된 최선 응답시간 분석기법의 정확도를 측정하기 위해 모의실험을 수행하였다. 우선적으로 [16, 17, 18]의 연구에서 인용된 태스크 집합들을 대상으로 하여 실제 최선 응답시간을 측정하였다. 표 4는 모의 실험에 사용된 태스크 집합 중 일부를 나타낸다. 태스크의 실제 최선 응답시간은 모의실험 시간동안 측정한 해당 태스크의 응답시간 중 최소값으로 결정된다. 최선 응답시간 분석기법들을 비교하기 위해 모의실험에서 얻은 각 태스크의 실제 최선 응답시간으로 GGH 방법의 최선 응답시간과 제안된 방법의 최선 응답시간을 각각 나눈 값을 구하였다. 그리고, 이 값을 최선 응답시간의 정확도로 부르겠다. 분석기법에서 얻은 최선 응답시간은 실제 최선 응답시간보다 작거나 같아야 하므로 최선 응답시간의 정확도는 1보다 작거나 같고, 1에 가까울수록 정확도가 높다고 볼 수 있다. 그림 8은 단일 노드에서 구한 태스크들의 최선 응답시간의 정확도를 나타낸다.

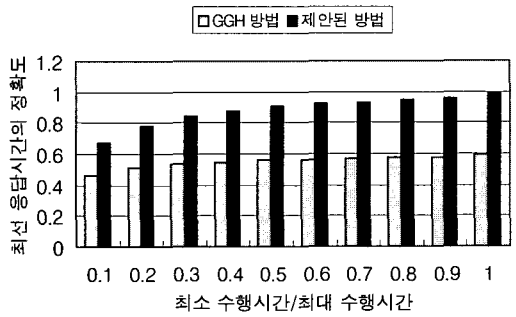


그림 8 단일 노드에서 최선 응답시간의 정확도

모의실험에서 태스크의 최소 수행시간/최대 수행시간의 비율을 0.1~1로 변화시켰고, 실제 태스크 수행시간은 최소 수행시간과 최대 수행시간 사이의 구간에서 균일 분포(uniform distribution)를 갖는다고 가정하였다. 모의실험은 태스크 주기들의 최소공배수*10에 해당하는 시간동안 수행하였고 RMS 스케쥴링 알고리즘을 사용하였다. 그림 8에 나타낸 값들은 각 태스크들에 대해 구한 최선 응답시간의 정확도들의 평균값들이다. 그림 8에 나타낸 바와 같이 제안된 방법을 이용하면 기존의 GGH 방법보다 최대 40%까지 개선되어 실제 최선 응답시간에 근접한 값을 얻을 수 있게 된다. 따라서, 후행 태스크의 활성화 지터를 기존의 방법보다 정확하게 추정할 수 있게 된다. 그림 8에서 최소 수행시간/최대 수행시간의 비율이 1에 가까울수록 최선 응답시간의 정확도가 높아지는 이유는 모의실험에서 사용된 태스크의 실제 수행시간의 범위가 작아지기 때문이다. 즉, 분석 기법에서 태스크의 최소 수행시간을 기준으로 하여 최선 응답시간을 구하기 때문에 실제 태스크 수행시간과 최소 수행시간의 편차가 작을수록 최선 응답시간의 오차가 작아지게 된다.

4.2 활성화 지터의 영향

활성화 지터가 분산 실시간 태스크의 최악 응답시간에 미치는 영향을 측정하기 위해 모의실험을 수행하였다. 모의실험에 사용된 태스크 집합을 얻기 위해 지역 태스크와 종단 태스크들을 임의로 생성한 다음, 2개의 프로세서 노드에 각각 4개의 지역 태스크들과 3개의 종단 태스크의 서브태스크들을 갖도록 할당하였다. 지역 태스크들은 100에서 2,000사이의 주기를 갖도록 생성하였고, 종단 태스크들의 서브태스크들은 500에서 1,000사이의 주기를 갖고 한 종단 태스크의 서브태스크들은 모두 동일한 주기를 갖도록 생성하였다. 프로세서 이용률이 각각 50%, 60%, 70%, 80%, 90%로 변화되도록 태스크 수행시간을 변경하여 태스크 집합을 프로세서 이용률당 100개씩 생성하였다. 한 프로세서 노드에서 태스크의 수행은 RMS 스케쥴링을 사용하였다.

프로세서 이용률의 변화에 따른 최선 응답시간의 정확도를 측정한 결과는 그림 9와 같다. 그림 9에 나타낸 값은 최소 수행시간/최대 수행시간의 비율을 1로 고정시켜 얻은 값이다. 즉, 태스크의 실제 수행시간은 항상 최대 수행시간과 동일한 값을 가진다. 그 이유는 태스크의 실제 수행시간이 변화하면 프로세서 이용률이 변화하기 때문에 프로세서 이용률을 일정하게 유지하기 위함이다. 그림 9에서 제안된 방법은 실제 최선 응답시간에 대한 정확도가 약 0.9를 나타내는 반면, GGH 방법은 최선 응답시간의 정확도가 0.77 정도의 값을 갖는다. 최선 응답시간 정확도가 그림 8에 나타낸 최선 응답시간 정확도보다 낮은 값을 갖는 이유는 다음과 같다. 그림 8의 실험에 사용된 태스크 집합들은 실제 실시간 시스템의 태스크들로서 주기들사이의 배수관계를 많이 갖는 특성을 가지지만, 임의로 생성된 태스크 집합들은 주기들사이의 배수관계를 상대적으로 적게 갖기 때문이다.

즉, 태스크 주기들이 서로 배수관계를 갖는 경우에 제안된 방법이 GGH 방법보다 월등한 최선 응답시간 정

표 4 모의실험에 사용된 태스크 집합들

(a) GAP 태스크 집합 [16]

태스크	최대 수행시간	주기
1	7	250
2	14	250
3	10	400
4	30	500
5	50	500
6	80	590
7	90	800
8	20	800
9	50	1000
10	30	2000
11	10	2000
12	30	2000
13	10	2000
14	10	2000
15	30	2000
16	10	10000
17	10]	10000

(b) 신호처리 태스크집합 [17]

태스크	최대 수행시간	주기
1	135	1200
2	69	1600
3	119	1600
4	736	4000
5	736	4000
6	336	8000
7	536	8000
8	536	8000
9	536	8000
10	536	60000
11	536	60000
12	839	120000
13	321	400000
14	546	1200000
15	2845	1200000

(c) INS 태스크 집합 [17]

태스크	최대 수행시간	주기
1	12	25
2	43	400
3	103	625
4	203	10000
5	1003	10000
6	250	12500

(d) Submarine 태스크 [17]

태스크	최대 수행시간	주기
1	50	100
2	9	500
3	41	1000
4	5	2500
5	33	3000
6	2	4000

(e) 44% 이용률 집합 [18]

태스크	최대 수행시간	주기
1	2	54
2	6	108
3	16	216
4	30	270
5	4	360
6	12	432
7	10	540
8	15	675
9	10	1080
10	40	1200

(f) 69% 이용률 집합 [18]

태스크	최대 수행시간	주기
1	6	54
2	6	114
3	5	240
4	64	432
5	30	540
6	45	675
7	72	720
8	36	900
9	20	1080
10	105	1200

(g) 88% 이용률 집합 [18]

태스크	최대 수행시간	주기
1	3	54
2	10	108
3	28	216
4	14	300
5	72	432
6	110	540
7	36	600
8	66	900
9	20	1080
10	40	1200

확도를 나타냄을 알 수 있다. 프로세서 이용률이 높은 경우(그림 9의 90%) 최선 응답시간의 정확도가 떨어지는 이유는 3.2절에서 가정한 태스크가 자신보다 우선순위가 높은 태스크들이 완료된 후 활성화되는 경우가 실제로는 거의 발생하지 않기 때문이다.

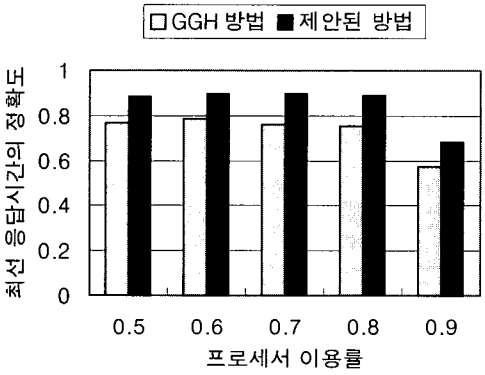


그림 9 프로세서 이용률 변화에 따른 최선 응답시간의 정확도

활성화 지터가 태스크의 최악 응답시간에 미치는 영향을 측정하기 위해 종단 태스크의 서브태스크들보다 우선순위가 낮은 태스크들의 실제 최악 응답시간을 모의실험을 통해 측정하였다. 왜냐하면 활성화 지터가 최악 응답시간에 영향을 미치는 태스크들은 해당 종단 태스크의 서브태스크보다 우선순위가 낮은 태스크들이기 때문이다. 모의실험을 통해 구한 실제 최악 응답시간으로 3.1절의 분석기법으로 얻은 최악 응답시간을 나눈 값으로 최악 응답시간의 정확도를 계산하였다. 분석기법에서 얻은 최악 응답시간은 실제 최악 응답시간보다 크거나 같아야 하므로 최악 응답시간의 정확도는 1보다 크거나 같고, 작을수록 정확도가 높다고 볼 수 있다. 그림 10에 나타낸 값들은 종단 태스크의 서브태스크들보다 우선순위가 낮은 태스크들의 최악 응답시간 정확도들의 평균을 의미한다. 그림 10에서 종단 서브태스크들의 활성화 지터를 계산할 때 사용되는 선행 태스크의 최선 응답시간을 각각 GGH 방법과 제안된 방법을 사용하여 구하였다. 그림 10에 나타낸 바와 같이 제안된 방법이 활성화 지터를 정확하게 계산함으로써 최대 6%까지 최악 응답시간의 분석의 정확도를 높임을 알 수 있다. 따라서, 부정확한 활성화 지터 추정으로 인하여 최악 응답시간이 불필요하게 크게 분석되는 점을 개선할 수 있다. 최선 응답시간 계산의 정확성을 현저히 높였음에도 불구하고, 최악 응답시간의 정확도 개선이 상대적으로 작은 이유는 최악 응답시간 분석에서 가정한 최악 경우가 모의실험에서 발생하지 않았기 때문이다.

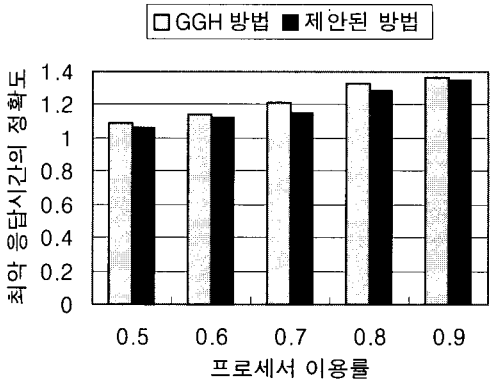


그림 10 최악 응답시간의 정확도

5. 결 론

분산 실시간 시스템에서 종단 서브태스크들은 고정 우선순위 스케쥴링 알고리즘에 의해 수행되면 활성화 지터를 가지게 된다. 활성화 지터를 가지는 태스크들이 존재할 때 스케쥴가능성을 분석하기 위해서는 선행 태스크들의 최악과 최선 응답시간을 오프라인에 알 수 있어야 한다. 기존의 분석 방법은 최선 응답시간을 실제 최선 응답시간보다 훨씬 작은 값으로 추정하기 때문에 태스크들의 최악 응답시간을 증가시키는 결과를 초래하였다. 본 논문은 이러한 단점을 개선하기 위해 실제 최선 응답시간에 가까운 최선 응답시간 분석기법을 제시하였고 모의실험을 통하여 제안된 방법이 기존의 방법보다 정확한 최선 응답시간을 산출함을 보였다. 그리고, 제안된 최선 응답시간 분석을 통해 얻은 태스크의 활성화 지터가 낮은 우선순위를 갖는 실시간 태스크들의 최악 응답시간에 미치는 영향을 모의실험을 통하여 측정하였다. 모의실험의 결과에 의하면 활성화 지터를 정확하게 추정함으로써 우선순위가 낮은 태스크들의 최악 응답시간이 실제 최악 응답시간에 근접하게 되었다.

기존의 활성화 지터를 줄이는 실시간 태스크 스케쥴링에 대한 연구[10, 11]는 고정된 태스크 수행시간을 가정하고 있다. 그러나, 실제 시스템에서는 수행 환경의 영향에 의해 태스크들이 가변적인 수행시간을 갖게 된다. 본 연구의 향후 과제로는 태스크의 최소 수행시간과 최대 수행시간이 알려져 있을 때 태스크의 완료간 시간

(inter-completion time)이 작은 편차를 갖는 적응성 스케줄정책에 대한 연구를 들 수 있다.

참 고 문 헌

[1] J. C. P. Gutierrez, J. J. G. Garcia, and M. G. Harbour, "Best-Case Analysis for Improving the Worst-Case Schedulability Test for Distributed Hard Real-Time Systems," *Proc. of Euromicro Workshop on Real-Time Systems*, pp. 35-44, 1998.

[2] J. J. G. Garcia and M. G. Harbour, "Optimized Priority Assignment for Tasks and Messages in Distributed Hard Real-Time Systems," *Proc. of Workshop on Parallel and Distributed Real-Time Systems*, pp. 124-132, 1995.

[3] D-T. Peng, K. G. Shin, and T. F. Abdelzaher, "Assignment and Scheduling Communicating Periodic Tasks in Distributed Real-Time Systems," *IEEE Transactions on Software Engineering*, Vol. 23, No. 12, pp. 745-758, 1997.

[4] K. W. Tindell, A. Burns, and A. J. Wellings, "Allocating Real-Time Tasks. An NP-Hard Problem Made Easy," *Real-Time Systems*, Vol. 4, No. 2, pp. 145-166, 1992.

[5] J. Sun and J. Liu, "Synchronization Protocols in Distributed Real-Time Systems," *Proc. of International Conference on Distributed Computing Systems*, pp. 38-45, 1996.

[6] K. W. Tindell, "An Extendible Approach for Analysing Fixed Priority Hard Real-Time Systems," *Real-Time Systems*, Vol. 6, No. 2, pp. 133-151, 1994.

[7] K. W. Tindell, A. Burns, and A. J. Wellings, "Analysis of Hard Real-Time Communications," *Real-Time Systems*, Vol. 9, No. 2, pp. 147-171, 1995.

[8] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM*, Vol. 20, No. 1, pp. 46-61, 1973.

[9] N. Audsley, A. Burns, M. Richardson, and A. J. Wellings, "Hard Real-Time Scheduling: The Deadline Monotonic Approach," *Proc. of IEEE Workshop on Real-Time Operating Systems and Software*, pp. 133-137, 1991.

[10] C-C. Han and K-J. Lin, "Scheduling Distance-Constrained Real-Time Tasks," *Proc. of Real-Time Systems Symposium*, pp. 300-308, 1992.

[11] K-J. Lin and A. Herkert, "Jitter Control in Time-Triggered Systems," *Proc. of Hawaii International Conference on System Sciences*, Jan., 1996.

[12] C. Locke, "Software Architecture for Hard Real-Time Applications: Cyclic Executives vs. Fixed Priority Executives," *Real-Time Systems*, Vol. 4, No. 1, pp. 37-53, 1992.

[13] L. Sha, R. Rajkumar, and J. Lehoczky, "Priority Inheritance Protocols: An Approach to Real-Time Synchronization," *IEEE Transactions on Software Engineering*, Vol. 39, No. 9, pp. 1175-1185, 1990.

[14] K. W. Tindell, H. Hansson, and A. J. Wellings, "Analysing Real-Time Communications: Controller Area Network(CAN)," *Proc. of Real-Time Systems Symposium*, pp. 259-263, 1994.

[15] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, "Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling," *Software Engineering Journal*, Vol. 8, No. 5, pp. 285-292, 1993.

[16] C. D. Locke, D. R. Vogel, and T. J. Mesler, "Building a Predictable Avionics Platform in Ada: A Case Study," *Proc. of Real-Time Systems Symposium*, pp. 181-189, 1991.

[17] T-S. Tia, J. W.-S. Liu, and M. Shankar, "Aperiodic Request Scheduling in Fixed-Priority Preemptive Scheduling," *Technical Report UIUCDCS-R-94-1859*, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1994.

[18] T. M. Ghazalie and T. P. Baker, "Aperiodic Servers in a Deadline Scheduling Environment," *Real-Time Systems*, Vol. 9, No. 1, pp. 31-67, 1995.

김 태 웅

1993년 서울대학교 컴퓨터 공학과 공학사. 1995년 서울대학교 컴퓨터 공학과 석사. 1995년 ~ 현재 서울대학교 컴퓨터 공학과 박사과정 재학중. 관심분야는 실시간 시스템, 실시간 운영체제, 분산 시스템임

신 현 식

1973년 서울대학교 응용물리학과 공학사. 1980년 미국 텍사스대학교 의공학 석사. 1985년 미국 텍사스대학교 컴퓨터공학과 박사. 1986년 ~ 현재 서울대학교 컴퓨터공학과 교수. 관심분야는 분산시스템, 실시간시스템임

장 래 혁

1989년 2월 서울대 제어계측공학과 졸업. 1992년 2월 동대학원 석사. 1996년 2월 동대학원 박사. 1997년 1월 미시간대학교 Research Fellow. 1997년 10월 ~ 현재 서울대학교 컴퓨터공학과 조교수. 관심분야는 내장형 시스템, 저전력 시스템, 실시간 시스템, 디지털 시스템 설계 및 구현