비 선점 영역을 갖는 실시간 태스크에서 소비 전력을 고려한 태스크 스케줄링

·이정환, 김명준
충북대학교 실시간 시스템 연구실
junghwanz@lge.com, mjkim@cbucc.chungbuk.ac.kr

Energy-Aware Task Scheduling for Real-Time Tasks with Non-Preemption Sections

Jung-Hwan Lee, Myung-Jun Kim
Real-Time System Lab, Dep. Computer Sci., Chungbuk National University

요 약

현재 이동용 장치(Mobile Device)들에서 전력 소모는 사용자들의 요구에 따라 성능 다음으로 중요한 비중을 차지하고 있다. 특히 배터리 셀의 기술 증가에 비해 프로세서들의 성능 및 요구하는 소비전력이 크게 증가함에 따라 프로세서의 전력 소모를 최소화 하는 연구들이 많이 진행되고 있다. 특히 프로세서의 전력 소모가 많은 비중을 차지함에 따라 프로세서의 전력 소모를 낮추기 위한 방법으로 많은 프로세서들은 DVS(Dynamic Voltage Scaling)와DFS(Dynamic Frequency Scaling)를 지원한다. 실제 프로세서의 전력 소모는 공급전압에 의 제곱에 비례하고 동작 클럭(Clock) 주파수에 비례한다. 그러나 공급전압은 다시 동작 클럭 주파수에 비례함으로써 DVS와DFS를 지원하는 대부분의 프로세서는 동작 클럭 주파수를 낮춤으로서 많은 전력 소모를 줄일 수 있게 된다. 그러나 동작 클럭 주파수를 낮추게 되면 태스크들의 실행 시간이 길어지게 되어 실시간 시스템에서 실시간성을 보장하지 못하게 된다. 본 논문에서는 상호간에 공유자원을 갖는 태스크들의 실시간성을 보장하며 동작 클럭 주파수를 낮추는 알고리즘을 제안한다.

1. 서 론

배터리를 사용하는 노트북, PDA(Personal Digital Assistant), MP3 재생기, 그리고 개인 휴대 전화등과 같은 이동용 장치(Mobile Device)들에서 한번 충전 후 사용하는 시간의 사용자에 대한 요구가 증가하고 있다. 그러나 이동용 장치들의 성능이 계속 증가하고 전력 소모가 커지는 반면 배터리 셀 기술의 발전은 요구를 만족시키지 못하고 있어 이동용 장치들의 전력 소모를 낮추기 위한 연구들이 많이 진행되어 왔고 실제로 적용되고 있다[1, 2]. 시스템에서 프로세서가 차지하는 전력 소모의 양이 대부분 이므로 프로세서의 전력 소모를 낮추는 것은 필수적이라 할 수 있다. 이동용 장치들의 프로세서에서 전력 소모를 고려하지 않은 경우와 고려한 경우에 전력 소모 차이는 수배에서 수십 배에 차이가 나고 있어 현재 대부분의 이동용 장치들에서는 프로세서의 전력 소모를 고려한 전력 관리를 적용하고 있다. 프로세서의 전력 소모를 개선하기 위한 방법으로 현재 사용되는 있는 기술은 DVS 와 DFS 가 있다. DVS 는 프로세서의 성능에 따라 입력 되는 공급전압을 가변 함으로서 프로세서의 전력 소모를 줄이는 방법이고 DFS 는 프로세서의 성능을 낮추기 위해서 동작 클럭 주파수를 낮춤으로서 전력 소모를 줄이는 방법이다. 그러나 입력 전압은 동작 클럭 주파수에 거의 비례한다[3]. 다음 식은 전력 소비, 입력 전압(Supply Voltage), 클럭 주파수(Clock Frequency), 부하 커패시턴스(Load Capacitance)의 관계를 나타낸다.

P: 전력 소비량
C: 부하 커패시턴스
F: 클럭 주파수
V_S: 입력 전압

$$P = C \cdot F \cdot V_S^2 \quad (1)$$

그러나 입력 전압은 동작 클럭 주파수에 거의 비례함으로 위 식은 이상적으로 다시 다음과 같이 표현 가능하다.

$$P = C \cdot F^3 \quad (2)$$

위 식에 따라서 프로세서의 동작 클럭 주파수의 세제곱이 전력 소비량과 비례함을 알 수 있다. 그러므로 최대한 주파수를 낮추는 것이 프로세서의 소비전력을 낮추는 것이다. 프로세서의 동작 주파수는 프로세서의 속도를 나타내므로 그림 2와 같이 임의의 태스크가 10의 주파수로 시간 5에 끝낼 수 있다면 그림 1과 같이 5의 주파수로 10의 시간에 끝낼 수 있다. 이때 그림 1의 전력 소비 P는 C·1250이고 그림 2의 전력 소비 P는 C·10000이 된다. 그러므로 그림 1과 그림 2의 소비전력 차이는 8배의 차이를 갖게 된다. 만약 태스크가 실시간성을 요하는 태스크이고 10의 데드라인을 갖는다면 그림 1과 같이 프로세서의 클럭 주파수를 낮추는 것이 전력 소비에 효율적이면서 실시간성을 보장하는 방법이라는 것을 알 수 있다.

본 논문에서 나머지 구성은 다음과 같다. 2장에서는 실시간 시스템에서 전력 소비를 최소화 하기 위한 기존의 연구들과

본 논문에 연구가 적용될 시스템 모델을 정의한다. 3장에서는 본 논문과 관련된 이전연구인 DS알고리즘과 주파수 상속 알고리즘에 대해서 설명한다. 4장에서는 본 논문에서 제안한 비 선점 영역 상속 알고리즘과 EDF기반에 이 알고리즘을 사용시 모든 태스크들이 데드라인을 만족함을 증명한다.

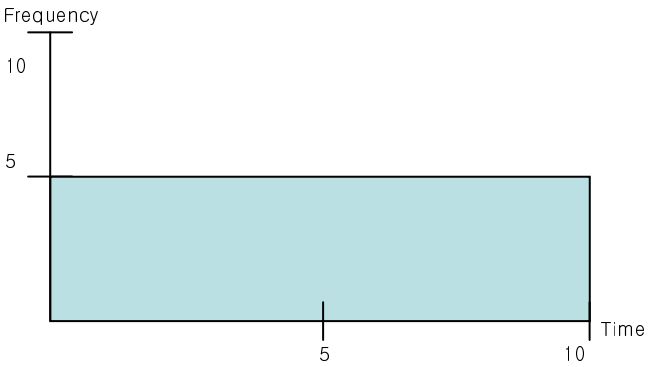


그림 1 5의 동작 클럭 주파수로 10의 시간에 일을 끝내는 경우.

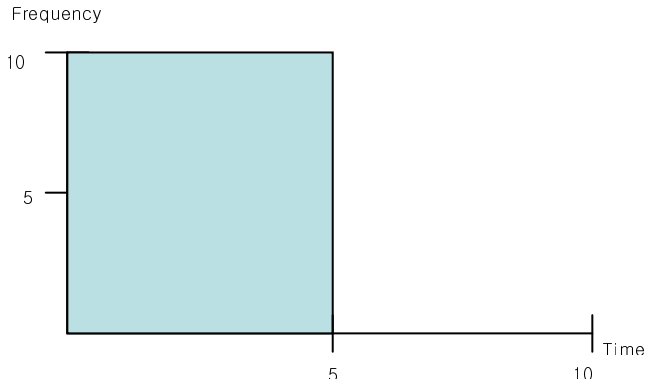


그림 2 10의 동작 클럭 주파수로 5의 시간에 일을 끝내는 경우.

2. 실시간 시스템에서 전력 소비 최적화

본 논문에서는 프로세서의 전력 소비를 최적화 하기 위하여 프로세서의 동작 클럭 주파수를 최대한 낮추면서 실시간성을 보장하기 위한 프로세서의 동작 클럭 주파수 변환 식을 제안한다. 기본적으로 실시간성을 보장하기 위한 스케줄링 방식은 EDF(Earliest Deadline First)를 사용한다.

이미 전력 소비를 고려한 독립적인 태스크 간의 EDF 스케줄러 기반의 알고리즘에 대한 분석과 정확성이 연구 되어졌고다[4]. 이와 같은 문제로 고정 우선순위 스케줄링은 Quan과 Hu[5] 의해서 연구되었고 NP-hard(Non-polynomial-hard)가 증명 되어졌다[6]. 그러나 실 세계에서는 독립적인 태스크뿐만 아니라 공유 자원에 의한 태스크간에 종속 성이 발생하는 경우가 빈번하다. 이러한 독립적이지 않은 태스크들은 태스크간에 동기화가 중요하다. 태스크 동기화가 필요한 스케줄링 알고리즘은 NP-hard 이나[7]-[8], 충분한 스케줄링 가능성(feasibility) 테스트에 대해서 연구 되어졌다[9][10]. 이런 태스크 스케줄링 가능성 테스트를 기반으로 프로세서의 클럭 주파수를 일정하게 내리는 연구가 되어졌다[11]. 이와 유사하게 Zhang과 Chanson은 비 선점 영역(non-Preemption Section)을 갖는 태스크에서 프로세서의 클럭 주파수를 일정하게 내리기 위한 연구를 하였고 DS(Dual-Speed)알고리즘을 제안하였다[12]. DS알고리즘은 태스크 내의 비 선점 영역에 대해서 만을 고려하고 태스크 동기화에 대해서는 고려하지 않았다. 이러한 문제로 같은 조건에서 태스크의 동기화를 고려한 클럭 주파수를 내리기 위한 알고리즘이 다시 연구되었다[13]. 그러나 이 알고리즘 또한 태스크가 여러 개의 다른 비 선점 영역을 가질 경우 데드라인을 만족하지 못하게 되는 문제가 있다. 본 논문에서는 이러한 문제점을 해결하기 위하여 기존에 계산식을 수정하고 이 알고리즘을 사용하여 프로세서의 클럭 주파수를 내려도 실시간성이 만족됨을 증명한다.

2.1 시스템 모델

본 논문은 주기적인 태스크를 고려한다. 주기적인 태스크는 연속된 일의 간격이 일정하고 이것을 주기라고 말한다.

시스템과 태스크의 특성은 다음과 같이 정의 한다.

(1) 시스템은 하나의 프로세서를 갖는다.
(2) 프로세서는 주파수 클럭과 전압 가변을 지원한다.
(3) 문맥 교환 시간은 고려하지 않는다.
(4) 태스크는 데드라인과 주기를 갖는다.
(5) 태스크의 주기와 데드라인은 같다.
(6) 태스크는 경성 실시간 태스크이다.
(7) 태스크 안에 여러 개의 비 선점 공유 영역이 존재한다.

태스크의 집합은 T 로 표기하고 각 태스크 $T_i \in T$ 이다. 다시 $T_i = \{P_i, D_i, C_i, B_i\}$이다.

P_i : 태스크의 주기

D_i: 태스크의 데드라인

C_i: 프로세서의 최대 속도에서 태스크의 최악의 경우의 실행시간

B_i: 프로세서의 최대 속도로 비 선점 영역에서 자신보다 우선순위가 낮은 태스크에 의한 최대 지연 시간.

태스크에 모든 작업이 데드라인을 만족하면 태스크는 경성 실시간을 만족한 것이라 말하고 태스크를 위한 프로세서의 이용률인 $U = \sum_{i=1}^{n} C_i / T_i \le 1$ 이어야 한다[14].

2.2 비 선점 영역을 가진 태스크에서 프로세서 클럭 속도를 내리는 경우 문제

태스크들간에 독립성이 보장되는 태스크들 간에는 우선순위가 높은 경우 낮은 태스크에 의해서 실행 시간이 지연되는 경우가 존재하지 않는다. 그러나 비 선점 영역을 가진 태스크들 사이에서는 우선순위가 높은 태스크가 낮은 태스크에 의해서 실행 시간이 지연되는 경우가 발생하게 된다. 그러므로 비 선점 영역을 가진 독립적이지 않는 태스크는 비

선점 영역에 대해서 하위 태스크에 의한 실행 시간의 지연을 고려하여야만 한다. 이러한 경우에 대한 예제는 R. Jejurikar 와 R. Gupta 의 연구에서 잘 보여준다[13].

그림 3(a)에서 T_1 = {5,5,2,3,}, T_2 = {40, 40, 4, 0}이다. 이때 프로세서의 이용률 U = (2/5) + (4/40) = 0.5 가 된다. 그러므로 비 선점 영역을 고려하지 않은 경우에 프로세서의 클럭 속도는 최대속도의 1/2 로 줄일 수 있게 된다. 태스크들의 프로세서 클럭 속도 변환 값에 집합을 N 이라 하고 각 태스크의 클럭 속도 변환 값 $N_i \in N$ 이라 한다. 이때 N_1 과 N_2 는 모두 0.5 가 된다. 그러므로 T_1 과 T_2 는 프로세서 최대 속도에서의 실행시간에 비해 2 배의 실행 시간을 갖게 된다.

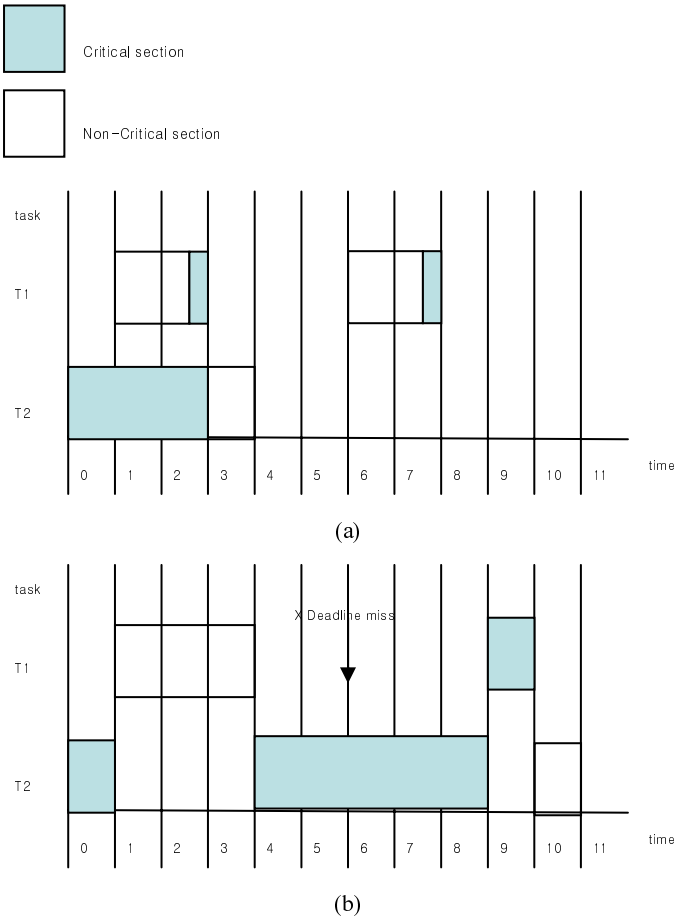


그림 3 비 선점 영역을 고려하지 않은 경우 프로세서 클럭 속도를 내림 (a) 태스크 도착 시간과 데드라인 (b) 비 선점 영역을 고려하지 않은 경우에 데드라인 맞추지 못하는 문제

그림 3 에 (b)는 비 선점 영역을 고려하지 않아 높은 우선순위의 T_1 이 T_2 를 선점하지 못하는 경우에 N_1 과 N_2 값을 적용하여 실행시간이 최대 프로세서 속도에 비해 2 배로 늘어나 T_1 이 데드라인을 만족하지 못함을 보여주고 있다.

3. DS 알고리즘과 주파수 상속 알고리즘

DS 알고리즘은 태스크가 비 선점 영역을 갖는 경우에 프로세서의 클럭 속도를 내리면서 각 태스크가 데드라인을 만족하기 위한 알고리즘이다[12]. DS 알고리즘은 독립적인 태스크들을 위해 프로세서의 낮은 속도와 독립적이지 않는 태스크들을 위해 프로세서의 높은 속도를 계산하여 태스크가 비 선점 영역에 의해서 실행시간이 지연될 경우에 높은 속도로 프로세서가 전환되게 하고 SRP(Stack Resource Policy)에 의해서 스케줄링을 한다. 그러나 DS 알고리즘은 태스크간의 동기화를 위한 PCP(Priority Ceiling Protocol)을 고려하지 않아 높은 우선순위의 지연시간이 증가하는 문제가 있다. 또한 DS 알고리즘에서 태스크간의 동기화를 위해 PCP 를 적용할 경우에 태스크의 데드라인을 만족하지 못하게 되어 이를 개선하기 위해 R. Jejurikar 와 R. Gupta 는 주파수 상속(Frequency Inheritance)과 프로세서의 클럭 속도를 내리기 위한 알고리즘 및 계산 식을 제안 하였다[11]. 태스크 동기화와 DS 알고리즘에 문제 및 이를 개선한 주파수 상속 알고리즘은 R. Jejurikar 와 R. Gupta 의 연구에서 보여진 그림 4 에서 알 수 있다[11].

그림 4 에 (b)에서 속도 L 과 H 의 계산은 DS 알고리즘 계산에 의해서 L = (2/5) + (4/40) = 0.5 이고 H = (2/5) + (3/5) = 1.0 이 된다. 프로세서의 클럭 속도가 처음에는 L 의 속도인 0.5 의 속도이다가 T_1 이 비 선점 영역을 가진 T_2 에 의해서 실행시간이 지연되면 프로세서의 클럭 속도를 1 로 전환하면서 태스크들의 데드라인은 만족하게 된다. 그러나 여기서는 PCP 를 고려하지 않아서 T_2 에 의해서 T_1 의 비 선점 영역까지 태스크의 실행이 지연되는 문제가 있다. 그림 4 에 (c)는 태스크 동기화를 위한 PCP 를 적용할 경우에 태스크 T_1 이 0.5 의 클럭 속도로 실행되어서 결국 데드라인을 만족하지 못하게 됨을 보여주고 있다. 그림 4 에 (d)는 이러한 태스크 동기화 문제 해결을 위해서 개선된 알고리즘과 주파수 상속에 의한 프로세서 클럭의 속도 감속을 보여주고 데드라인을 만족함을 보여주고 있다. 이를 위한 식 (3)과 알고리즘 1 은 R. Jejurikar 와 R. Gupta 가 프로세서의 클럭 속도 감속을 위해 제안하였다.

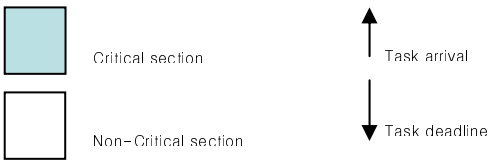
$$\left(\sum_{1\le r\le q}\frac{1}{n_r}\frac{C_r}{T_r}\right)+\frac{1}{n_i}\left(\frac{B_i}{D_i}+\sum_{q\le p\le i}\frac{C_p}{D_p}\right)=1 \qquad (3)$$

그림 4 에 (d)에서 N1 과 N2 는 식(3)과 알고리즘 1 에 의해서 다음과 같이 계산된다.

(1/N1)(3/5 + 2/5) = 1, N1 = 1.0

(1/N2)(0/40 + 2/5 + 4/40), N2 = 0.5

T1 의 프로세서 클럭 속도 값을 고려하여 N2 를 다시 계산하면 (1/1.0)(2/5) + (1/N2)(0/8 + 4/40) = 1, N2 = 0.167 이다.



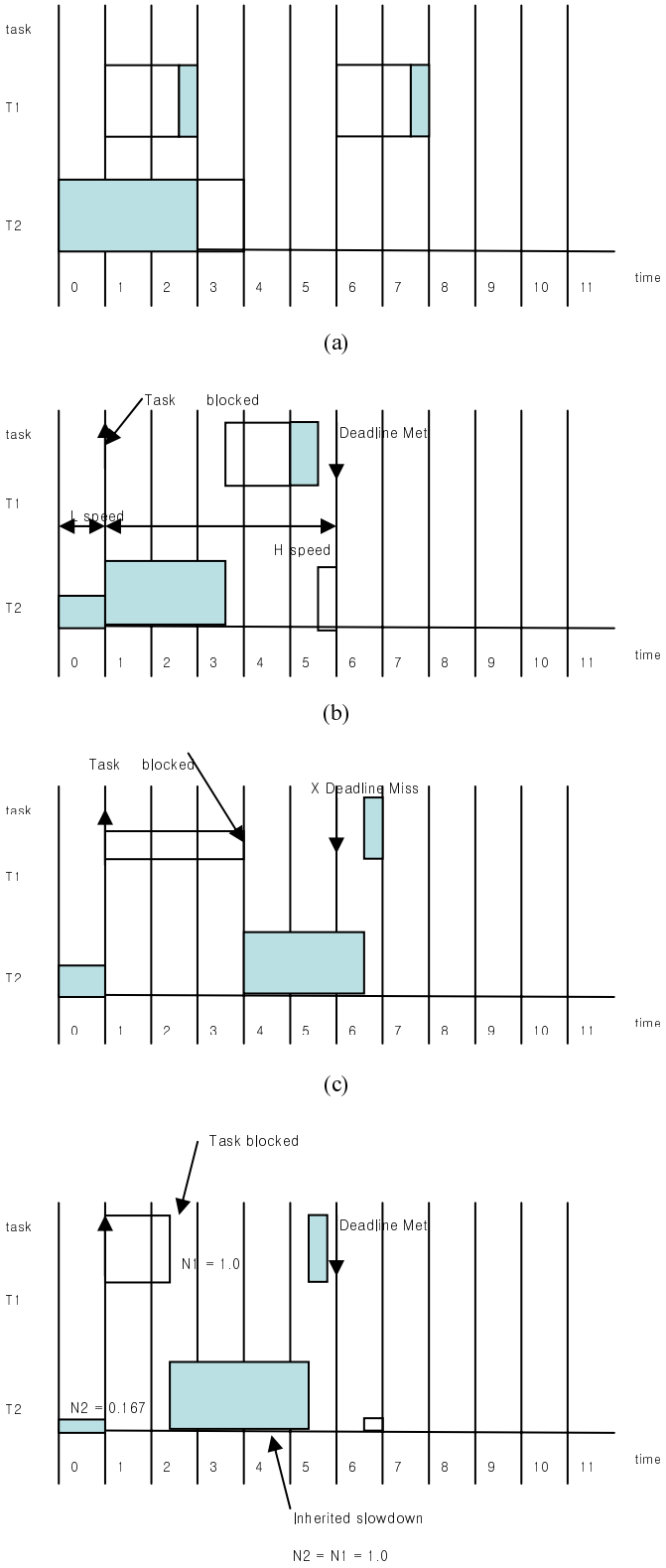


그림 4 비 선점 영역을 고려한 프로세서 클럭 내림 (a)태스크의 도착시간과 데드라인 (b) 태스크 동기화를 적용하지 않은 경우 DS 에 의한 방법 (c) 태스크 동기화를 적용한 경우 DS 에 의한 문제 (d) 태스크 동기화를 적용하고 주파수 상속에 의해 태스크 데드라인을 만족시킴

알고리즘 1. 프로세서 클럭 속도 감속

1: {데드라인은 순서가 낮을수록 짧다.}
2: q = 1; {초기화}
3: WHILE(q<=n) DO
4: FOR(i=q; i<=n; i++) DO
5: 식(3)에 의한 프로세서 클럭 속도 감속 값 계산
6: END FOR
7: Nm = $MAX_{i=q}^{n}$ (Ni)
8: FOR(i = q; i<=m ; i++) DO
10: N1 = Nm;
11: q = m + 1;
12: END WHILE

4. 비 선점 영역 상속 알고리즘

R. Jejurikar 와 R. Gupta 가 프로세서의 클럭 속도 감속을 위해 제안한 우선 순위 상속 알고리즘은[11] 하위의 태스크에 의해서 상위 태스크의 비 선점 영역에 대한 실행 시간의 지연은 발생하지 않지만 자신보다 상위의 태스크가 비 선점 영역에 대해서 자신보다 하위의 태스크에 의해 실행시간이 지연될 경우 자신의 태스크까지 실행시간이 지연되는 것에 대해 프로세서의 클럭 속도 감속을 고려하지 않아 이러한 경우가 발생될 때에는 태스크의 데드라인을 만족하지 못하게 된다. 예를 들어 T_1 = {5,5,2,3}, T_2 = {7,7,1,0}, 그리고 T_3 = {400, 400, 4, 0}인 경우에 식(3)에 의한 각 태스크의 프로세서 클럭 속도 감속 값은 다음과 같다.

$(1/N_1)(3/5 + 2/5) = 1$, $N_1 = 1.0$

$(1/N_2)(2/5 + 1/7 + 4/400) = 1$, $N_2 = 0.255$

$(1/1.0)(2/5) + (1/0.255)(1/7) + (1/N_3)(4/400) = 1$, $N_3 = 0.255$

그러므로 그림 5 의 (a)와 같이 T_2 는 자신의 데드라인을 맞추지 못하게 되는 문제가 발생하게 된다. 이러한 문제 해결을 위해 본 논문에서는 T_1 태스크의 비 선점 영역에 의한 T_2 태스크에서의 지연 시간을 고려하기 위하여 T_2 태스크 또한 T_1 태스크가 T_3 태스크에 의해 실행 시간에 지연이 발생하는 경우에 T_2 태스크는 T_1 태스크의 비 선점 영역을 상속 받아 T_2 태스크 또한 T_3 태스크에 비 선점 영역을 고려하게 하는 식을 제안한다.

$$\left(\sum_{1\le r\le q}\frac{1}{n_r}\frac{C_r}{T_r}\right)+\frac{1}{n_i}\left(\frac{B_i}{D_i}+\sum_{1\le r\le q}\frac{B_r}{D_i}+\sum_{q\le p\le i}\frac{C_p}{D_p}\right)=1 \quad (4)$$

식(4)를 알고리즘 1 에 5 번 줄에 사용하게 되면 식(4)에 의한 각 태스크의 프로세서 클럭 속도 감속 값은 다음과 같다.

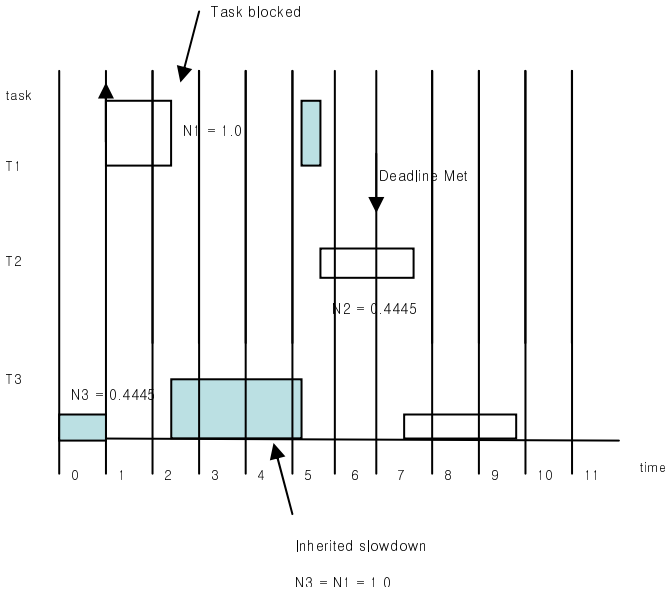
$(1/N_1)(3/5 + 2/5) = 1$, $N_1 = 1.0$

$(1/N_2)(2/5 + 4/7 + 4/400) = 1$, $N_2 = 0.571$

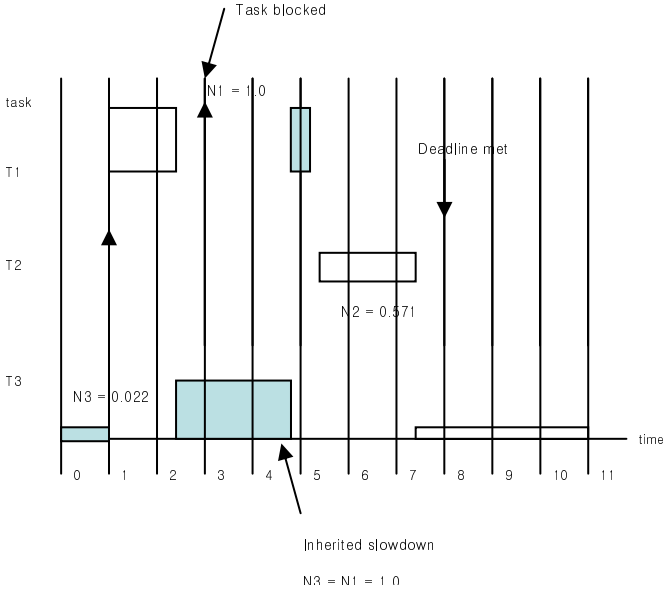
$(1/1.0)(2/5) + (1/0.571)(1/7) + (1/N_3)(4/400) = 1$, $N_3 = 0.022$

그러므로 그림 5 의 (b)와 같이 T_2 태스크 또한 데드라인을 만족하게 된다. EDF 스케줄링 기반에서 비 선점 영역 상속 알고리즘 식으로 계산된 프로세서의 클럭 속도 감속 값을 적용 시 모든 태스크는 데드라인을 만족한다. 식(5)는 EDF

스케줄링 기반에서 식(4)의 계산식으로 프로세서의 클럭 속도 감속 값을 적용했을 경우에 태스크들의 실행 시간을 나타낸다.



(a)



(b)

그림 5 상위 태스크에 비 선점 영역 고려 문제 (a)하위 태스크에 의한 상위 태스크에 지연을 고려하지 않은 중간 태스크에 스케줄링 문제 (b)상위 태스크에 비 선점 영역에 지연을 다른 태스크들에서 고려한 프로세서 클럭 속도 내림

$\frac{1}{D_i n_i}\sum_{k=1}^{i} B_k + \sum_{k=1}^{i} \frac{1}{n_k}\frac{C_k}{D_k}$ 은 프로세서 클럭 속도 감소를 적용한 모든 작업의 실행 시간이므로 EDF 기반의 스케줄링에서 이 시간이 1 보다 작거나 같은 경우에 모든 태스크는 데드라인을 만족할 수 있다.

$$\frac{1}{D_i n_i}\sum_{k=1}^{i} B_k + \sum_{k=1}^{i} \frac{1}{n_k}\frac{C_k}{D_k} \le 1 \qquad (5)$$

증명: 증명은 Baker[30]과 R. Jejurikar 와 R. Gupta[11]의 증명과 유사하다. 식(5)가 거짓 이라고 가정한다. 임의의 t 를 데드라인을 만족하지 못한 최초의 시간이라 하고 t'을 t 전에 마지막 시간이라 한다. 이때 t'전에 도착시간을 갖는 일의 실행 요청이 지연되는 경우가 없고 데드라인은 t 전이거나 t 와 같다. A 는 [t',t]안에 도착하는 일들의 집합이며 [t',t]내에서 데드라인을 갖는다. t'의 선택에 의해서 A 안에 일들의 요청은 [t', t]의 시간 동안에 항상 지연되는 일들이 존재한다. 그리고 이 시간 간격에는 절대 시스템의 휴지시간이 발생할 수 없다. EDF 우선순위 할당 정책에 의해서 오직 A 안에 일들만이 [t',t]시간에 시작된다. 그러나 t 보다 큰 데드라인을 갖는 일들도 A 안에 일이 요구하는 자원을 갖고 있는 경우에는 [t',t]에서 실행할 수 있다. 그와 같은 일들에 집합은 B 로 표기하고 각 일 $j_b \in B$ 는 t 보다 큰 데드라인을 갖고 실행한다. 모든 A 안에 일 t_i 대해서 $D_i \le X$ 이고 만약 j_b 가 t'시간에 실행 중이면 $X < D_b$ 이다. D_i 에서 데드라인이 큼에 따라 순서 값이 D_i 의 순서 값이 높아지게 되고 $A \subseteq \{t_1,\ldots,t_k\}$, $D_k \le X$, 그리고 $k < b$ 이다. [t't]내에서 자원에 의해 지연되는 일의 최고 속도는 B_i 에 의해서 결정된다[9]. 모든 A 안에 일 t_i 는 [t', t]안에서 프로세서 시간에 대한 요구 시간이 kC_i 보다 크지는 않으므로

$k = \left\lfloor \frac{X - D_i}{T_i} \right\rfloor + 1$ 이다.

주파수 상속 규칙에 의해서 공유자원을 가진 일이 다른 일의 실행을 지연한 경우에 최대 프로세서 속도 감속 값을 상속 받게 된다[11]. 그리고 본 논문에서 제안한 비 선점 영역 상속 알고리즘에서 자신보다 높은 우선순위의 일이 자신보다 하위의 일에 의한 실행시간 지연을 모두 고려함으로 전체 지연시간은 $\frac{1}{n_i}\sum_{k=1}^{i} B_k$ 으로 나타낼 수 있고 일 A 와 B 의 전체 시간은 다음과 같다.

$$\frac{1}{n_i}\sum_{k=1}^{i} B_k + \sum_{k=1}^{i} \frac{1}{n_k}\left(\left\lfloor \frac{X - D_k}{T_k} \right\rfloor + 1\right) C_k$$

태스크는 t 시간에 데드라인을 만족하지 못하므로 모든 일들에 대한 전체 실행시간은 X 보다 커야 한다. 그러므로 다음과 같이 다시 나타낼 수 있다.

$$\frac{1}{n_i}\sum_{k=1}^{i} B_k + \sum_{k=1}^{i} \frac{1}{n_k}\left(\left\lfloor \frac{X - D_k}{T_k} \right\rfloor + 1\right) C_k > X$$

이때 $\frac{X}{T_k} \geq \left\lfloor \frac{X}{T_k} \right\rfloor$ 이므로,

$$\frac{1}{Xn_i}\sum_{k=1}^{i} B_k + \sum_{k=1}^{i} \frac{1}{n_k}\left(\frac{X - D_k + T_k}{T_k X}\right) C_k > 1$$

$$= \frac{1}{Xn_i}\sum_{k=1}^{i} B_k + \sum_{k=1}^{i} \frac{1}{n_k}\left(1 + \frac{T_k - D_k}{X}\right)\frac{C_k}{T_k} > 1$$

$D_k \leq X \ \forall k, k=1,..,i$ 이고

$$\frac{1}{D_i n_i}\sum_{k=1}^{i} B_k + \sum_{k=1}^{i} \frac{1}{n_k}\left(1 + \frac{T_k - D_k}{X}\right)\frac{C_k}{T_k}$$

$$= \frac{1}{D_i n_i}\sum_{k=1}^{i} B_k + \sum_{k=1}^{i} \frac{1}{n_k}\frac{C_k}{D_k} > 1$$

그러므로 식 (5)를 부정하므로 모든 태스크들은 데드라인을 만족한다.

4. 결론 및 향후 연구 과제

본 논문은 R. Jejurikar and R. Gupta 가 제안한 주파수 상속[13]과 T.P. Baker 가 제안한 SRP 를 확장한 논문으로서 기존 논문에서는 태스크가 비 선점 공유 영역을 가지고 있지 않거나 다른 속성의 비 선점 영역을 가지고 있어 자신보다 하위 태스크에 의해 자신보다 상위 태스크가 지연됨으로써 자신까지 실행 시간이 지연되는 경우 데드라인을 만족하지 못하는 문제를 가지고 있다. 이러한 문제를 해결하기 위해 자신보다 상위의 태스크가 자신보다 하위의 태스크에 의해서 실행 시간의 지연이 발생하게 되는 비 선점 영역을 고려함으로써 프로세서 클럭의 속도를 감속시키고도 모든 데드라인을 만족한다. 그러나 본 논문에서는 자신 보다 상위 태스크들에 비 선점 영역에 대한 중복을 고려하지 않았다. 이러한 문제로 인하여 프로세서 클럭 속도를 더 낮게 내리지 못하게 된다. 향후 이러한 문제를 해결하고 검증하기 위한 연구가 필요하다. 또한 본 논문에서 제시한 프로세서 클럭을 내리기 위한 식이 새로운 알고리즘에 적용되지 않고 R. Jejurikar and R. Gupta 가 제안한 알고리즘에 추가함으로써 알고리즘의 최적화가 이루어 지지 않아 수행시간에 상당한 영향을 미칠 것으로 예상된다. 향후에는 이러한 알고리즘 최적화 및 실제 응용에서 적용 후 실험을 통한 성능 분석이 필요하다.

[1] J.R. Lorch and A.J. Smith. "Software strategies for portable computer energy management", IEEE Personal Communications, 5(3):60-73, 1998

[2] K.I. Farkas, J. Flinn, G. Back, D. Grunwald, and J.M. Anderson. "Quantifying the energy consumption of a pocket computer and a java virtual machine". In Proceedings of SIGMETRICS, page 252-263, 2000

[3] I. HONG, D. Kirovski, G. QU, M. Potkonjak, and M.B. Srivastava. "Power optimization of variable-voltage core-based systems". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 8(12):1702-1714, 1999

[4] W.kwon and T. Kim, "Optimal voltage allocation techniques for dynamically variable voltage processors", in Proc. Design Automation Conf., Anaheim, CA, 2003, pp. 125-130.

[5] G. Quan and X. Hu, "Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors," in Proc. Design Automation Conf., Las Vegas, NV, Jun. 2001, pp. 828-833.

[6] H. Yun and J. Kim, "On energy-optimal voltage scheduling for variable fixed-priority hard real-time systems," Trans. Embed. Comput. Syst., vol. 2, no. 3, pp. 393-430, Aug. 2003

[7] A.K.Mok, "Fundamental design problems of distributed systems for hard real-time environmonet," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, 1983

[8] J.A. Stankovic, M. Spuri, M. D. Natale, and G. Buttazzo, "Implications of classical scheduling results for real-time systems," IEEE Trans. Comput., vol. 28, no. 6, pp. 16-26, Jun. 1990.

[9] T.P. Baker, "Stack-based scheduling of real-time processes," J. Real-Time Syst., Vol.3, no. 1, pp. 67-99, Mar. 1991.

[10] L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," IEEE Trans. Comput., vol. 39, no. 9, pp. 1175-1185, Sep. 1990

[11] R. Jejurikar and R. Gupta, "Energy aware task scheduling with task syncrhronization for embedded real time systems," in Proc. Int. Conf. Compilers, Architecture and Synthesis Embedded Systems, Grenoble, France Oct. 2002, pp. 164-169.

[12] F. Zhang and S. T. Chanson, "Processor voltage scheduling for real-time tasks with non-preemptible sections,"in Proc. IEEE Real-Time Systems Symp., Austin, TX, Dec. 2002, pp. 235-245

[13] R. Jejurikar and R. Gupta, "IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems, Vol. 25, No. 6, Jun. 2006.

[14] J. W. S. Liu, Real-Time Systems. Upper Saddle River, NJ: Prentice Hall, 2000.