주기성과 산발성 태스크가 혼합된 시스템을 위한 전력절감 스케쥴링 기법

Power-Aware Scheduling for Mixed Real-Time Tasks

공민식, 정근재, 송예진, 정명조, 조문행, 이철훈
충남대학교 공과대학원 컴퓨터공학부

Min-Sik Gong(msgong@cnu.ac.kr), Gun-Jae Jeong(gjjeong@cnu.ac.kr),
Ye-Jin Song(yjsong@cnu.ac.kr), Myoung-Jo Jung(mjjung@cnu.ac.kr),
Moon-Haeng Cho(root4567@cnu.ac.kr), Cheol-Hoon Lee(clee@cnu.ac.kr)

요약

본 논문에서는 주기적으로 생성(release)되는 주기성 태스크(Periodic Task)들과 산발적으로 생성되는 산발성 태스크(Sporadic Task)들이 혼합된 실시간 시스템을 위한 전력관리 스케쥴링 기법을 제안한다. 각각의 태스크는 최소주기, 최악수행요구시간과 마감시간 등으로 정의된다. 본 논문에서 제안한 동적 전압조정(Dynamic Voltage Scaling : DVS) 알고리즘인 DVSMT(DVS for mixed tasks)는 태스크의 실시간 마감시간을 보장하면서 작업이 종료됐을 때, 수행하는 동안 사용한 사이클 중 다른 태스크들이 할당한 수행 사이클을 자신의 마감시간까지 온라인 상태에서 균등 분배함으로써 공급전압(또한 동작 주파수)을 동적으로 조정한다. 이러한 기법으로 더 많은 에너지를 절감할 수 있다. 제안한 알고리즘은 실시간 운영체제에 쉽게 통합될 수 있기 때문에 제한된 배터리 전력을 이용하는 휴대용 기기 및 센서망 노드 등에 적용할 수 있다. 시뮬레이션 결과들은 DVSMT가 주기성 태스크들로만 구성된 시스템과 주기성 태스크들 및 산발성 태스크들이 혼합된 시스템에서 기존의 알고리즘보다 대략 60% 까지 에너지가 절감됨을 보였다.

■ 중심어 : | 주기성 태스크 | 산발성 태스크 | 동적 전압조정 | 전력관리 스케쥴링 |

Abstract

In this paper, we address a power-aware scheduling algorithm for a mixed real-time system which consists of periodic and sporadic tasks, each of which is characterized by its minimum period, worst-case execution requirement and deadline. We propose a dynamic voltage scaling algorithm called DVSMT(DVS for mixed tasks), which dynamically scales down the supplying voltage(and thus the frequency) using on-line distribution of the borrowed resources when jobs complete while still meeting their deadlines. With this scheme, we could reduce more energy consumption. As the proposed algorithm can be easily incorporated with RTOS(Real-Time Operating System), it is applicable for handheld devices and sensor network nodes that use a limited battery power. Simulation results show that DVSMT saves up 60% more than the existing algorithms both in the periodic-task and mixed-task systems.

■ keyword : | Periodic Tasks | Sporadic Tasks | Dynamic Voltage Scaling | Power-Aware Scheduling |

* 본 연구는 정보통신부의 선도기반기술개발사업의 지원으로 수행되었습니다.

접수번호 : #061106-001
접수일자 : 2006년 11월 06일
심사완료일 : 2006년 12월 21일
교신 저자 : 이철훈, e-mail : clee@cnu.ac.kr

I. 서 론

최근 컴퓨터/통신분야에서는 휴대폰(cellular phone), 유비쿼터스 센서 네트워크(ubiquitous sensor network), DMB(Digial Multimedia Broadcasting) 및 무인로봇과 같은 배터리로 동작하는 모바일 및 이동형 플랫폼/장치가 발전하고 있다. 이와 같은 응용분야에서는 장치를 오랫동안 사용할 수 있도록 배터리 수명을 연장하는 것이 매우 중요하다. 점점 소형화되고 컴퓨팅 성능의 증가와 더불어 디지털 캠코더, 휴대폰 및 휴대용 의료장비와 같은 다양한 임베디드 시스템에서는 더욱 정교하고, 지능적으로 제어하는 소프트웨어를 수행하기 위한 강력한 마이크로 프로세서의 사용이 증가하고 있다. 마이크로 프로세서의 동작 주파수가 수 백MHz에서 수 GHz 까지 빨라짐에 따라 전력소모와 발열 문제가 중요한 쟁점이 되고 있다.

최근 10년 전부터 실시간 마감시간(Real time deadline)을 보장함과 동시에 전압과 주파수를 조정함으로써 실시간 시스템의 에너지를 절감하기 위한 동적전압조정(Dynamic voltage Scaling : DVS) 기법에 대한 많은 연구와 개발 노력이 있어 왔다[1-8]. 이것은 오늘날 대다수 마이크로 프로세서에서 사용하고 있는 CMOS 회로가 구동전압에 의존하므로 주파수를 줄였을 때 프로세서는 더 낮은 전압에서 동작할 수 있고, CMOS 회로에서 에너지 소모는 공급전압의 제곱에 비례하기 때문에 가능한 것이다($E \propto V^2$) [9]. 실시간 태스크가 최악계산요구(Worst-case Computation Requirements) 로 정의되더라도, 대부분의 작업은 최악계산시간보다 훨씬 적게 소요된다[10]. 대부분 DVS 알고리즘들은 태스크가 사용하지 않고 남은 슬랙시간(slack time)을 이용하여 마감시간을 보장하면서 동시에 공급전압(또한 동작 주파수)을 낮게 조정한다.

*Aydin et al*은 주기성 태스크로 구성된 시스템에서 가장 높은 우선순위 태스크의 남아있는 실행시간을 이용하여 우선순위기반 슬랙 분배 기법을 적용한 DRA (Dynamic Reclaiming Aggressive)을 제안하였다[6]. 최근 제안된 OLDVS(On-line DVS)는 일반적인 태스크 모델을 위한 온라인 슬랙 관리 알고리즘이다[5]. DRA와 OLDVS 알고리즘은 태스크의 문맥교환시 다음에 수행할 가장 높은 우선순위를 갖는 태스크에게만 전체 슬랙을 할당한다.

Phillai and Shin[2]은 주기성 태스크들로 구성된 시스템을 위한 Cycle-conserving EDF(ccEDF)라는 새로운 DVS 알고리즘을 제안했다[2]. 이 알고리즘은 시스템의 모든 태스크들에게 슬랙을 균등 분배함으로써 각 태스크의 작업이 생성(release) 또는 완료됐을 때 동작 주파수에 대한 조정계수를 계산한다. 산발성 태스크로 구성된 시스템(sporadic task system)에 대해서는 Qadi, *et al.*[4]가 슬랙시간을 사용하지 않고 작업들이 생성될 때에만 프로세서 주파수를 올리거나 내리는 DVSST 알고리즘을 제안하였다.

본 논문에서는 주기성과 산발성 태스크들이 혼합된 시스템을 위한 DVSMT라는 DVS 알고리즘을 제안하였다. DVSMT 알고리즘은 어떤 태스크가 완료되는 시점에서 태스크가 수행하고 남은 시간을 다른 태스크들에게 할당된 사이클과 함께 자신의 마감시간까지 균등 분배함으로써 주파수 조정계수를 계산한다.

제안한 알고리즘은 각 태스크의 작업이 생성되는 시점과 종료시점에 오직 한번의 계산을 필요로 한다($O(1)$). 그렇기 때문에 실시간 운영체제에 쉽게 통합될 수 있다. 시뮬레이션을 통해 DVSMT가 주기성 태스크만으로 구성된 시스템과 주기성 및 산발성 태스크들이 혼합된 시스템에서 모두 기존의 다른 알고리즘보다 60%까지 에너지를 절감하는 결과를 확인하였다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 본 논문에서 고려한 시스템 모델(system model)을 설명하고, 제안한 알고리즘의 상세 설명과 연구 내용을 기술한다. 3장에서는 시뮬레이션 결과를, 마지막으로 4장에서는 결론 및 향후 연구과제에 대하여 기술한다.

II. 주기성과 산발성 태스크가 혼합된 시스템을 위한 동적전압조정 알고리즘

1. 시스템 모델

본 논문은 실시간 태스크들이 EDF(Earliest Deadline First) 스케쥴링 정책 하에서 스케쥴링되는 선점형 경성

실시간 시스템(hard real-time system)을 제안한다. 본 논문에서 고려한 응용 분야는 태스크 집합 $T=\{T_1, T_2, ..., T_n\}$에서 각 태스크 $T_{i,j}=(p_i, e_i)$는 주기(period) p_i와 최악수행시간(worst case execution time : WCET) e_i를 가진 주기성 태스크(periodic task) 또는 산발성 태스크(sporadic task)이다. 또한 각 태스크는 상대적인 마감시간(relative deadline) D_i를 가지며, $D_i=p_i$로 가정한다. 모든 태스크들은 독립적이고 선점할 수 있다. 각 태스크 T_i가 $j \leq 1$인 생성 시간(release time) $r_{i,j}$와 마감시간(Deadline) $d_{i,j}$를 가진 많은 작업(job) $J_{i,j}=(r_{i,j}, d_{i,j})$을 생성하게 된다. 만약 T_i가 주기성 태스크이면 $r_{i,j+1}=r_{i,j}+p_i$이고, 산발성 태스크이면 $r_{i,j+1} \geq r_{i,j}+p_i$ 이다. 즉, 산발성 태스크 작업들의 최소 도착시간(minimum inter-arrival time)은 자신의 주기인 것이다. 두 가지 형태의 태스크에서 $d_{i,j}=r_{i,j}+p_i=r_{i,j}+D_i$ 를 가정한다. 태스크 T_i의 이용률(utilization) u_i는 자신의 주기에 대한 수행 이용률이다($u_{i,j}=e_i/p_i$). 총 이용률 u_{tot}는 모든 태스크들의 이용률의 합으로 아래 식과 같다.

$$u_{tot}=\sum_{i=1}^{n} u_i$$

구동전압을 가변할 수 있는 프로세서는 구동 전압과 동작 주파수가 각각 동작범위인 $[v_{\min}, v_{\max}]$과 $[f_{\min}, f_{\max}]$ 사이에서 조정할 수 있다고 가정한다. 각 동작 주파수는 하나의 최소 공급전압으로 연결될 수 있다(f_i, v_i). EDF 스케쥴링 정책하에서 프로세서의 동작 주파수를 조정하기 위한 조정계수는 생성된 모든 태스크들의 현재 총 이용률 u_{tot}에 의해 결정한다. 주파수 조정계수 α는 프로세서의 최고 주파수 $f_{\max}$에 대한 현재 프로세서의 동작 주파수 f_i의 비율로써 $\alpha=f_i/f_{\max}$와 같이 나타낸다.

본 논문은 EDF 스케쥴링 정책하에서 주기성과 산발성 태스크로 구성된 시스템를 위한 전력절감 알고리즘을 고려하기 때문에, 식 (1)을 만족하는 필요충분조건이 태스크 집합에 대해 실행가능하다.

$$\sum_{i=1}^{n} \frac{e_i}{p_i} \leq \alpha \quad (1)$$

2. 제안한 기법

태스크의 작업이 생성될 때, 작업의 실제 수행요구시간을 알 수 없다. 만약 현재 작업이 일찍 끝난다면, 작업을 완료한 태스크의 슬랙은 주파수를 다시 조정하기 위해 다음에 수행할 태스크들에 의해 사용될 수 있다.

본 논문에서 제안한 기법은 태스크의 완료시점에서 자신의 작업을 수행하는 동안 소모하였던 다른 태스크의 자원 또는 사이클을 자신의 마감시간까지 대기 중인 모든 작업들에게 균등하게 분배하는 개념으로 이를 통해 동적으로 전압을 조정한다.

제안한 알고리즘 개념을 설명하면 다음과 같다. 최악의 경우 총이용률이 1보다 작거나 같은 실시간 태스크 집합 T를 생각하자. 예를 들면 [그림 1(a)]에서 $u_i+u_j+u_k \leq 1$ 인 경우이다. 태스크들이 생성되었을 때 태스크들의 최악수행요구 e_i 는 [그림 1(b)]에서 보는 바와 같이 작업들의 도착시간에서 마감시간까지 분배될 수 있다.

c_i는 $t_1 \geq r_i$ 과 $t_2 \geq d_i$ 조건에서 구간 $[t_1, t_2)$에 할당된 태스크 T_i 의 자원 또는 사이클을 의미하며, T_i의 총자원의 일부분이다. 다시 말해서, c_i는 구간 $[t_1, t_2)$에 할당(또는 분배)된 CPU 사이클(cycle) 수를 의미한다. 즉, $c_{i,[t_1,t_2)}=u_i(t_2-t_1)$이다. 그러므로 c_i는 최악수행요구시간 e_i 보다 작거나 같다($c_i \leq e_i$).

정의 : 실시간 태스크 집합과 수행구간 $[t_1, t_2)$이 주어졌을 때, 작업 $J_{i,j}$의 빌린자원 $B_{i,j}$ 는 식 (2)와 같다.

$$B_{i,j}=\sum_{k=1, k\neq i} c_k \quad (2)$$

빌린자원 $B_{i,j}$는 작업 $J_{i,j}$를 수행하기 위해 태스크 T_i에 의해서 소모된 CPU 사이클의 일부분으로써 다른 태

스크들이 구간 $[t_1, t_2)$에 할당한 사이클의 총합이다. $J_{i,j}$이 수행하는 동안 경과한 시간 $E_{i,j}$는 $t_2 - t_1$ 이다. T_i에 의해 소모된 자원들이 대기 중인 모든 태스크들을 위해 보존되지 않았다면, 최악 시나리오에서 마감시간을 놓치게 될 것이다. 그러므로 수행동안 다른 태스크들로부터 빌린자원들은 반드시 되돌려 주어야 한다.

이론 : 연속적인 주파수와 전압 레벨을 가진 실시간 시스템에서 실시간 태스크 집합과 수행구간 $[t_1, t_2)$이 주어졌을 때, 종료시간까지 주어진 작업 $J_{i,j}$의 빌린자원 $B_{i,j}$는 최악수행 시나리오에서 $u_i\{D_i - (t_2 - t_1)\}$와 같다.

증명 : 최악수행요구시간 e_i 을 가진 태스크 T_i 는 [그림 1(b)]에서와 같이 생성하자마자 작업 $J_{i,1}$을 수행한다고 하자. 태스크 T_i 는 구간 $[t_1, t_2)$에서 가장 높은 우선순위 태스크이고, 구간 $[t_1, t_2)$에 할당된 T_j 와 T_k의 자원(각각 c_j와 c_k)을 수행하는 동안 소모한다. 최악의 수행 시나리오에서, 작업의 사이클 보존 때문에 $c_{i,1} + c_{i,2} = c_{i,1} + c_j + c_k$이라는 것을 확실히 알 수 있다. 수식을 정리하면, $c_{i,2} = c_j + c_k$이다. 정의에 따라, 빌린자원 $B_{i,j}$는 $c_j + c_k$와 같다. 따라서 $B_{i,1} = c_{i,2}$이다. 여기서 $c_{i,2}$는 $u_i(d_i - t_2) = u_i\{D_i - (t_2 - t_1)\} = u_i(D_i - E_{i,j})$와 같다. 결과적으로 빌린자원 $B_{i,j}$는 식 (3)과 같다.

$$u_i\{D_i - (t_2 - t_1)\} = u_i(D_i - E_{i,j}) \qquad (3)$$

식 (3)으로부터 작업 $J_{i,j}$의 완료시점에 빌린자원 $B_{i,j}$는 구간 $[r_{i,j} + E_{i,j}, d_{i,j})$에서 분배될 수 있다. 그래서 T_i 의 현재 이용률 $u_{i,j}$는 $d_{i,j} - (r_{i,j} + E_{i,j})$에 대한 $B_{i,j}$의 비율로써 식 (4)와 같다.

$$u_{i,j} = \frac{B_{i,j}}{d_{i,j} - r_{i,j} - E_{i,j}} = \frac{B_{i,j}}{D_{i,j} - E_{i,j}} \qquad (4)$$

태스크 T_j가 [그림 1(c)]에서 보는 바와 같이 구간 $[t_1, t_2)$에서 T_i에 의해 선점되었고, t_2에 다시 수행되는 경우를 생각하자. 최악수행 시나리오에서 $c_{j,1} + c_{j,2} + c_{j,3} = c_k + c_{j,2} + c_i$ 임을 알 수 있다. 그래서 $B_{i,1}$은 $c_j + c_k$과 같다. 즉, $B_{i,1} = c_{j,1} + c_{j,3} = c_j + c_k$이다. 식 (4)는 T_j가 더 높은 우선순위의 태스크들에 의해 선점되는 조건에서도 역시 유효함을 알 수 있다.

식 (3)과 식 (4)로 부터 현재 이용률 $u_{i,j}$ 는 최악수행 시나리오에서 항상 최악 이용률 $u_i = e_i/p_i$와 같다.

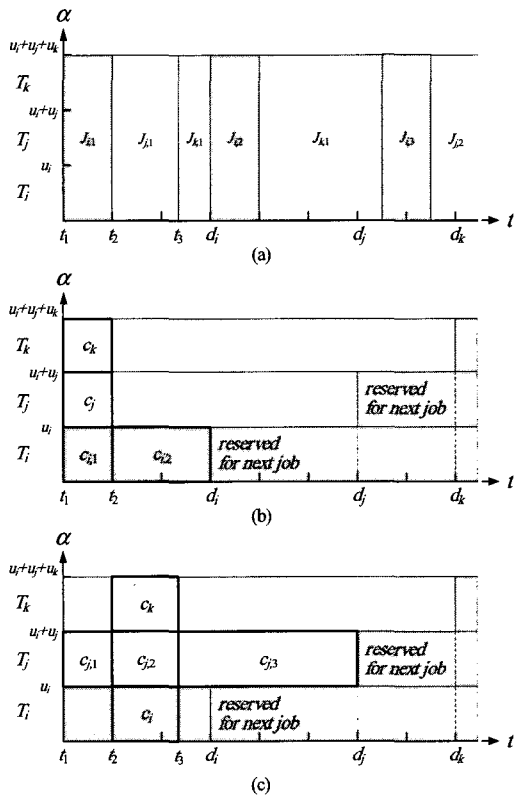


그림 1. DVSMT 알고리즘 개념을 설명하기 위한 예제

태스크가 자신의 최악수행시간보다 일찍 완료된다면, 빌린자원 $B_{i,j}$와 경과시간 $E_{i,j}$ 는 감소될 것이다. 그러므로 현재 이용률 $u_{i,j}$는 최악 이용률 u_i 보다 더 작다. 즉, $u_{i,j} < u_i$ 이다. 태스크에 의해 소모된 빌린자원들을 사용하여 이용률을 다시 계산함으로써 CPU의 속도를

줄일 수 있다는 것이다. 감소한 이용률은 태스크 자신의 다음에 도착하는 작업을 위해 다시 생성될 때까지 사용된다.

3. DVSMT 알고리즘

최악 시나리오에서 생성되는 모든 태스크들이 수행 가능함을 보장하기 위해서는 수행 사이클을 보존하는 알고리즘을 만들어야 한다. 이를 위해 DVSMT(DVS for Mixed Tasks)는 완료된 태스크가 빌린자원들을 대기 중인 모든 태스크들에게 자신의 마감시간까지 되돌려 준다.

작업 $J_{i,j}$가 생성되는 시점에서는 실제로 수행되는 시간을 알 수 없으므로 수행 사이클이 보존된다는 가정하에 태스크의 최악수행요구가 필요하다. 모든 태스크가 생성되는 경우를 가정하여 계산한 최악 총이용률 $u_{tot} = \sum_{i=1}^{n} u_i$을 이용하는 초기 주파수 레벨을 시작하는 정적인 전압조정 알고리즘[3]과는 달리 DVSMT는 생성되는 작업들의 최악 이용률 u_i을 이용하여 가능한 최저 주파수 레벨에서 시작한다. 작업 $J_{i,j}$가 완료되는 시점에는 수행하는 동안에 다른 태스크들에게 빌린자원(또는 수행 사이클)을 알 수 있다. 빌린자원 $B_{i,j}$는 대기 중인 모든 태스크에 되돌려 주기 위해 구간 $[r_{i,j} + E_{i,j}, d_{i,j})$에서 균등하게 분배된다.

본 논문에서는 생성된 태스크들에 대하여 각 작업의 마감시간을 넘기지 않도록 태스크들을 유지하기 위해 큐(queue) dQ를 도입하였다. dQ는 EDF 스케쥴 정책에 따라 마감시간에 의해 정렬되며, 작업의 도착시점, 마감시간 및 완료시점에 갱신된다. 갱신방법은 다음과 같다. 작업 $J_{i,j}$가 생성 될 때, 이전에 수행한 작업 $J_{i,j-1}$가 dQ에 없다면 현재 도착한 $J_{i,j}$를 큐에 삽입 후 정렬한다. 그렇지 않으면 $J_{i,j-1}$를 새롭게 도착한 $J_{i,j}$로 대체한다. 만약에 태스크 T_i가 자신의 마감시간에 다음 수행될 작업 $J_{i,j+1}$이 도착하지 않으면, dQ에서 $J_{i,j}$의 정보를 제거한다.

DVSMT는 [그림 2]에서 보는 바와 같다. 시스템이 *IDLE* 상태, 즉 수행하는 태스크들이 없을 경우에 조정계수는 0이고 dQ는 비운다.

t 시점에 $J_{i,j}$가 생성될 때, 이용률 $u_{i,j}$는 e_i/p_i로 초기화되고 경과시간 $E_{i,j}$와 빌린자원 $B_{i,j}$ 는 0으로 재 초기화된다. DVSMT는 dQ에 있는 모든 작업들에 대하여 현재의 총이용률을 계산한다. 그리고 프로세서 속도를 선택한다. 프로세서 속도는 조정계수 보다 큰 주파수들 중에 가장 낮은 주파수로 선택된다.

현재 시간 t에서 $J_{i,j}$가 현재 수행중인 작업 $J_{k,j}$ 보다 우선순위가 높으면 $J_{k,j}$는 $J_{i,j}$에 의해 선점된다. 선점되는 $J_{k,j}$의 경과시간 $E_{k,j}$와 빌린자원 $B_{k,j}$이 다음과 같이 계산된다. $B_{k,j}$는 이전에 수행하기 위해 빌린자원과 선점되기 직전까지 수행하기 위해 빌린자원의 합으로 계산된다.

이를 수식으로 표현하면,$B_{k,j} = B_{k,j} + (t-l)(\alpha - u_k)$이다. 여기서 l은 가장 최근에 문맥을 전환한 시간이며, α는 프로세서의 현재 조정계수 또는 속도를 의미한다. 또한 $E_{k,j}$는 이전에 수행하는데 경과된 시간과 현재의 경과시간의 합으로써 $E_{k,j} = E_{k,j} + (t-l)$이다. 경과시간은 현재의 조정계수 또는 속도와는 관계가 없다.

$J_{i,j}$가 현재 수행중인 작업 $J_{k,j}$ 보다 우선순위가 낮으면 $J_{i,j}$의 이용률 $u_{i,j}$을 반영하여 프로세서의 속도를 결정하고 $J_{k,j}$는 수행을 계속한다.

$J_{i,j}$가 시간 t에서 완료될 때, $B_{i,j}$과 $E_{i,j}$는 앞에서 언급한 것과 같이 계산된다. DVSMT는 $J_{i,j}$의 새로운 이용률을 $u_{i,j} = B_{i,j}/(d_{i,j} - r_{i,j} - E_{i,j}) = B_{i,j}/(D_i - E_{i,j})$로 계산하고 프로세서의 속도를 결정한다. 그 다음에 대기 큐에서 우선순위가 가장 높은 작업이 수행된다.

DVSMT 알고리즘은 실시간 운영체제에 쉽게 일체화되고 동적전압조정을 위한 처리시간이 크게 필요하지 않다. 또한 DVSMT는 $O(1)$ 계산이 필요하다. 동적전압조정에서 가장 심각한 오버헤드는 하드웨어적인 전압 전환시간이다. 그러나 각 태스크마다 2번 이상의 전환은 필요하지 않기 때문에 쉽게 설명될 수 있고 태스크의 최악수행시간에 포함될 수 있다.

위에서 언급했던 것처럼 DVSMT는 각각의 문맥전환 시점에서 프로세서의 공급전압과 동작 주파수를 조정한다.

```
on initial or IDLE State
    α = 0.0;
    Empty dQ;

on arrival time of J_{i,j} :
    if J_{k,j} is preempted by J_{i,j}
        B_{k,j} = B_{k,j} + (t−l)(α−u_{k,j});
        E_{k,j} = E_{k,j} + (t−l);
    endif;

    B_{i,j} = 0;  // reset the borrowed resources
    E_{i,j} = 0;  // reset the elapsed time

    if J_{i,j−1} exists in dQ then
        replace J_{i,j−1} with J_{i,j};
    else
        insert J_{i,j} into dQ;
    endif;
    sort dQ by the deadline under the EDF
    set u_{i,j} to e_i/p_i;
    scale_voltage_and_frequency();

on completion time of J_{i,j} :
    B_{i,j} = B_{i,j} + (t−l)(α−u_{i,j});
    E_{i,j} = E_{i,j} + (t−l);
    update u_{i,j} = B_{i,j}/(D_i − E_{i,j});
    scale_voltage_and_frequency();

on deadline d_{i,j} of J_{i,j} :
    extract J_{i,j} from dQ ;
    scale_voltage_and_frequency();

scale_voltage_and_frequency():
    select the lowest frequency
        f_i ∈ { f_min, ..., f_max | f_min < ... < f_max }
    such that Σ_{i=1}^{m} u_i ≤ f_i/f_max, for {J_{i,j}} ∈ dQ
    select the supply voltage associated with f_i;
```

그림 2. DVSMT 알고리즘의 Pseudo-Code

4. 예제 해설

$u_{tot}=1$인 주기성과 산발성 태스크가 혼합된 시스템을 가정한다. 시스템은 $T_1(4,1)$, $T_2(8,2)$ 및 $T_3(10,5)$로 구성되며, 각각의 태스크는 $T_i(p_i,e_i)$로 정의한다. 여기서 T_1과 T_2 는 주기성 태스크이고 T_3는 산발성 태스크이다.

실시간 태스크들이 최악의 환경으로 규정됨에도 불구하고 일반적으로 최악의 경우보다 빨리 수행을 완료한다. 본 예제에서는 각 태스크의 실제 수행시간은 $\hat{e_1}=0.5$, $\hat{e_2}=1$ 및 $\hat{e_3}=5$ 이고, T_3의 작업들은 $t=0$, 12에서 각각 생성된다고 가정한다. 동적전압조정 기법이 적용되지 않은 EDF 스케쥴 결과는 [그림 3]과 같으며, 제안한 DVSMT 알고리즘에 의한 동적전압조정과 스케쥴링 과정은 [그림 4]와 같다.

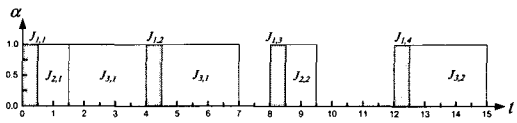


그림 3. EDF 스케쥴링 결과

$t=0$에서 T_1, T_2, T_3의 작업 $J_{1,1}$, $J_{2,1}$, $J_{3,1}$은 동시에 생성된다. 알고리즘에 의해 생성된 작업들은 dQ에 삽입되고 마감시간에 따라 정렬되며, dQ에 저장된 작업들의 각 이용률 u_i를 이용하여 현재의 총이용률 u_{tot}가 계산된다. $t=0$에서 $J_{1,1}$은 가장 높은 우선순위의 작업이므로 조정계수 α=1(u_{tot}=1)로 수행하기 시작한다. $J_{1,1}$은 $t=0.5$에 자신의 WCET보다 일찍 완료되며 [그림 4(a)], 시구간 $[0,0.5)$에서 T_2과 T_3이 할당한 수행 사이클을 소모했다.

식 (2)에 의해 $J_{1,1}$이 빌린자원 B_1은 $E_{1,1}(u_2+u_3)$로 계산되고 $E_{1,1}=0.5$, $u_2=0.25$, $u_3=0.5$이므로 B_1 =0.375이다. 빌린자원 B_1은 식 (4)에 의해 현재 시간 t =0.5과 자신의 마감시간 $t=4$ 사이에 대기 중인 태스크들에게 균등 분배된다. 식 (4)에 의한 이용률 $u_{1,1}$은 $B_1/(D_1-E_{1,1})$=0.375/(4−0.5)=0.107로 계산된다. dQ에 저장되어 있는 T_1의 작업 $J_{1,1}$은 아직 마감시간을 지나지 않았으므로 dQ에 계속 저장되며, $J_{1,1}$의 최악시간요구에 의한 이용률 u_1은 새롭게 계산된 이용률 $u_{1,1}$으로 대체된다. dQ에 저장된 작업들의 각 이용률 u_i를 이용하여 현재의 총이용률 u_{tot}를 다시 계산한다. 계산된 현재 총이용률 u_{tot}은 0.857이다. $J_{2,1}$은 새로운 조정계수 α=0.857으로 작업을 시작한다.

t=1.67에 $J_{2,1}$이 완료된다. $J_{2,1}$의 경과시간 $E_{2,1}$은 1.167이고, $B_{2,1}$은 0.708로 $u_{2,1}$은 0.104로 계산된다. 따라서 [그림 4(b)]와 같이 알고리즘에 의해 주파수 조정계수가 0.711로 계산된다.

t=4에서 $J_{1,2}$이 생성되고 우선순위에 의해 $J_{1,2}$은 $J_{3,1}$을 선점한다. dQ에 있는 $J_{3,1}$ 정보는 경과시간 $E_{3,1}$과 빌린자원 $B_{3,1}$은 각각 2.33과 0.492로 갱신된다. 또한 dQ에 있는 $J_{1,1}$ 정보는 $J_{1,2}$ 정보로 갱신되므로 u_1=0.25이다. 따라서, $J_{1,2}$는 주파수 조정계수 α=0.854로 작업을 시작한다[그림 4(c)].

t=4.59에서 $J_{1,2}$는 완료되면, 조정계수 α가 0.707로 재계산되고 $J_{3,1}$가 재시작된다.

t=8에서 $J_{1,3}$과 $J_{2,2}$이 생성되며, 조정계수 α는 1로 다시 계산된다. 그러나 $J_{3,1}$는 t=8에서 우선 순위가 가장 높기 때문에 수행을 계속하고 t=8.93에서 작업이 완료된다[그림 4(d)]. $J_{3,1}$의 총 경과시간은 구간 [1.67, 4), [4.59, 8)과 [8, 8.93)의 경과시간의 합으로 $E_{3,1}$=6.67이다. 또한 총 빌린자원은 각각의 수행구간에서의 빌린자원들의 총합으로 $B_{3,1}$=1.663이다. 따라서 $u_{3,1}$=0.5 이다.

t=10.59에서 대기 중인 태스크와 수행 중인 태스크가 더 이상 없으므로 조정계수 α는 0으로 초기화된다.

DVSMT 알고리즘에 의해 스케줄링한 최종 결과는 [그림 4(d)]와 같다. 알고리즘은 태스크가 생성하는 시점과 완료시점에 동적으로 프로세서의 주파수와 전압을 조정한다. 이는 최악의 시나리오에서도 생성되는 모든 태스크들의 마감시간을 보장함과 동시에 실행 가능함을 보장한다.

III. 시뮬레이션 결과

실시간 시스템에서 전압조정에 의한 에너지 절감 정도를 평가하기 위하여 RTSIM[14]를 이용하여 시뮬레이션을 수행하였다. RTSIM은 실시간 시스템을 위한 시뮬레이터이며, 전통적인 스케줄링 알고리즘뿐만 아니라 새로운 DVS 알고리즘을 시뮬레이션 할 수 있다. 또한 실시간 DVS 시스템에서 동적으로 전압과 주파수를 조정할 때, 하드웨어의 특성을 고려하여 시뮬레이션 할 수 있다. 본 논문에서는 몇 가지 시뮬레이션 결과를 도출했고 가장 중요한 시스템 파라미터 작용을 이해할 수 있었다.

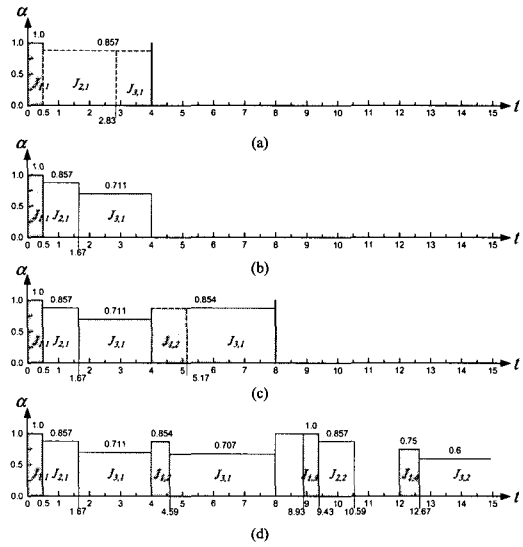


그림 4. DVSMT 알고리즘 예제

본 논문에서 몇 가지 시뮬레이션 조건을 다음과 같이 가정하였다. 시뮬레이션은 주어진 전압레벨에서 매 동작 사이클 동안 일정한 양의 에너지를 소모한다고 가정한다. 프로세서에 의해 소모된 에너지만 계산되고, 다양한 명령어에 의한 영향은 고려하지 않는다. 또한, 제어용 중지 기능이 프로세서에 제공되어 휴지시간(idle time) 동안은 에너지 소모가 없다고 가정한다. 특히 문맥전환 시간과 동작 주파수와 전압의 전환시간에 의한 오버헤드는 고려하지 않는다. 실시간 태스크 집합은 20개의 태스크로 구성되고, 주기와 최악요구시간으로 구성된 태스크 정보는 다음과 같이 무작위로 생성하였다. 각 태스크는 3가지 주기별 즉, 짧은 주기(1-10ms), 중간주기(10-100ms), 그리고 긴주기(100-1000ms) 등으로 동일한 확률로 균등하게 분포되도록 한다. 주기성과 산발성 태스크가 혼합된 태스크 집합에 대해서는 산발성 태스크를 5개로 제한하고 태스크 집합의 총 이용률 u_{tot}이 1이 되도록 하였다. 실시간 태스크의 실제 수행시간 변화는 가우시안(gaussian) 확률분포를 따르며, 각각의 조건은 시뮬레이션 조건에 따라 다르게 입력하였다.

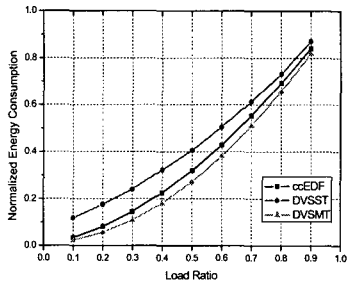
제안한 알고리즘을 평가하기 위해 ccEDF, DVSST, 그리고 non-DVS 알고리즘과 비교하였다. 또한 DVS 기능이 제공되는 완전한 프로세서와 PXA250[13]과

TM5800[14]과 같은 실제 프로세서에 대하여 시뮬레이션을 수행하여 그 영향을 비교하였다. 완전한 프로세서는 동작범위 $[v_{\min}, v_{\max}]$과 $[f_{\min}, f_{\max}]$ 내에서 연속적으로 공급전압과 동작 주파수를 동적으로 조정할 수 있다. 그러나 실제 프로세서들은 이산적인 전압과 주파수 레벨을 갖는다. 다음은 시뮬레이션에 적용한 프로세서들의 공급전압과 동작 주파수를 요약한 것이다.

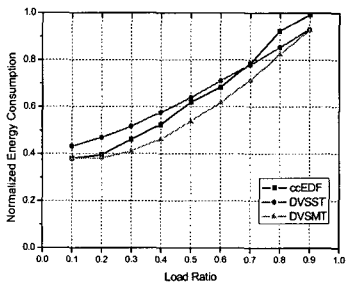
Ideal CPU : $[v_{\min}, v_{\max}]$=[0, 1],
$[f_{\min}, f_{\max}]$=[0, 1]
PXA250 : {(100, 0.85), (200, 1.0), (300, 1.1), (400, 1.3)}
TM5800 : {(300, 0.8), (433, 0.875), (533, 0.95),
(667, 1.05), (800, 1.15), (900, 1.25),
(1000, 1.3)}.

먼저 주기성 태스크의 부하율(load ratio) 영향을 평가하기 위한 시뮬레이션 결과는 [그림 5]와 같으며, non-DVS 알고리즘에 의해 계산된 에너지 소모로 표준화(normalize)하여 도시하였다. 여기서, 부하율은 최악요구시간 e_i에 대한 실제 수행시간 $\hat{e_i}$의 비($\hat{e_i}/e_i$)이다. 태스크의 실제 수행시간의 평균(mean)은 부하율로 결정하고, 인접 부화율과 간섭을 최소화하기 위해 표준편차는 0.1 · WCET/3으로 결정하였다. 따라서 실제 수행시간의 99.7%는 시구간 [부하율 - 0.1 · WCET, 부하율 + 0.1 · WCET] 내에 있다. 시뮬레이션 결과는 제안한 DVSMT 알고리즘이 이상적인 프로세서에서 ccEDF에 비해 25%까지, DVSST에 비해 55%까지 에너지를 더 절감할 수 있음을 보였다. 또한 실제 프로세서에 대해서도 다른 알고리즘에 비해 15%까지 더 절감함을 보였다. 에너지 절감은 부하율이 감소할수록 증가하고, DVSMT와 다른 알고리즘 사이의 에너지 소모차는 부하율이 증가할수록 커진다.

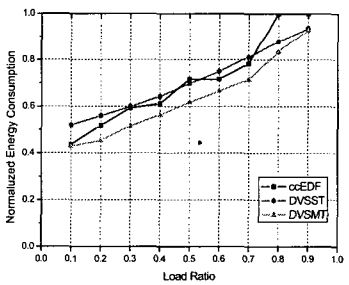
[그림 6]은 DVSMT에서 부하율과 CPU의 조정레벨 특성에 따른 시뮬레이션 결과를 한꺼번에 보기 위해 [그림 5]를 정리한 것이다. 결과로부터 동적전압레벨이 많을수록 이상적인 CPU에 가까워짐을 알 수 있고 PXA250과 TM5800 프로세서의 경우, 부하율 70%이상은 이득이 없음을 알 수 있다.



(a) IDEAL CPU



(b) TM5800



(c) PXA250

그림 5. 주기성 태스크의 부하율 변화에 따른 표준화된 에너지 소모

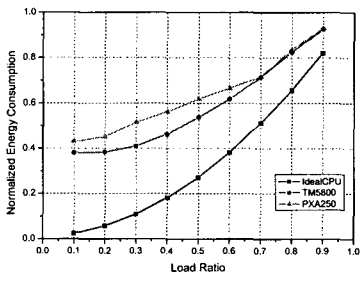
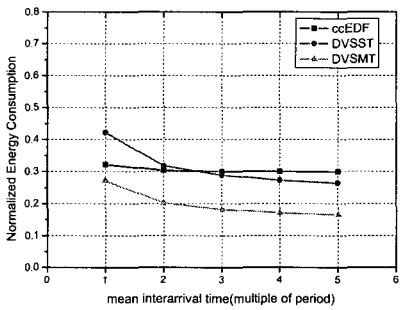


그림 6. DVSMT에 의한 주기성 태스크의 부하율 변화에 따른 표준화된 에너지 소모

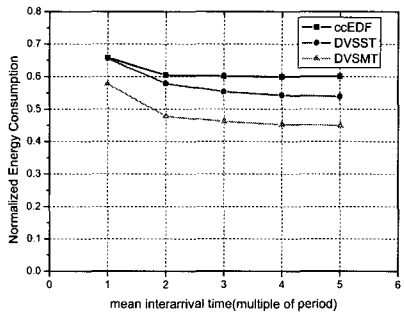
주기성과 산발성 태스크가 혼합된 태스크 집합에 대하여 부하율 0.5인 경우에서의 산발성 태스크의 도착시간 사이의 시간(inter-arrival time)에 대한 영향을 분석하기 위해 시뮬레이션 하였다. 혼합 태스크 집합은 주기성 태스크 15개, 산발성 태스크 5개로 결정하였다.

특히 ccEDF가 주기성 태스크만을 고려한 알고리즘이지만, RTSIM이 이벤트 구동(event-driven) 시뮬레이터이므로 알고리즘은 변경하지 않았다. 시뮬레이션 결과는 [그림 7]과 같다. 도착시간 사이의 시간은 평균이 태스크의 상대 마감시간의 배수인 지수분포(exponential distribution)를 갖는다. 시뮬레이션 결과로부터 산발성 태스크에 도착시간 사이의 시간에 거의 영향을 받지 않음을 알 수 있었다.

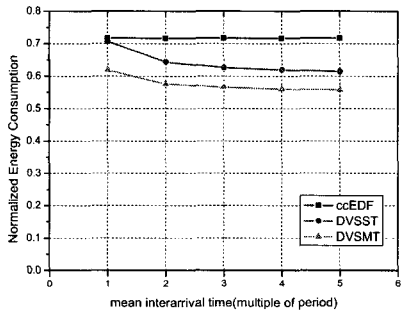
DVSMT는 이상적인 프로세서인 경우 ccEDF에 비해 45%, DVSST에 비해 60% 까지 절감함을 확인하였다. 또한, PXA250과 TM5800인 경우에도 ccEDF에 비해 25%, DVSST에 비해 20% 까지 절감하였다.



(a) IDEAL CPU



(b) TM5800



(c) PXA 250

그림 7. 혼합 태스크 집합에서 산발성 태스크의 inter-arrival 변화에 따른 표준화된 에너지 소모

IV. 결론과 향후과제

본 논문에서 주기성과 산발성 태스크들이 혼합된 경성 실시간 시스템에서의 동적전압조정 기법을 사용하는 전력관리기법인 DVSMT(DVS for a mixed task set) 알고리즘을 제안하였다. 제안한 알고리즘은 태스크들의 마감시간을 만족하면서 에너지 절감을 향상시키기 위해, 온라인으로 빌린자원을 대기 중인 태스크들에게 자신의 마감시간까지 분배하는 것이다.

DVSMT는 태스크가 도착할 때와 완료되는 시점에서만 프로세서의 동작전압과 동작 주파수를 온라인으로 조정하므로 실시간 운영체제에 쉽게 통합시킬 수 있다. 또한, 혼합 태스크 집합에서도 효율적으로 적용할 수 있음을 보였다.

시뮬레이션 결과에서 알 수 있듯이 제안한 알고리즘은 이상적인 프로세서에서는 다른 알고리즘에 비해 60%, 실제 프로세서인 PXA250과 TM5800에 비해 25% 까지 에너지 절감을 확인하였다.

향후 연구과제로는 유비쿼터스 환경하에서의 시간적 지역성(temporal locality)을 갖는 이동장치의 동작을 반영한 실시간 전력관리기법을 연구할 것이다.

참 고 문 헌

[1] F. Gruian, "Hard real-time scheduling for low energy using stochastic data and DVS processors,"

Proc. Int'l Symposium on Low-Power Electronics and Design (ISLPED'01), pp.46-51, 2001.

[2] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," Proc. 18th ACM Symposium on Operating System Principles (SOSP'01), pp. 89-102, 2001.

[3] A. Dudani, F. Mueller, and Y. Zhu, "Energy-Conserving Feedback EDF Scheduling for Embedded Systems with Real-Time Constraints," Proc. of the joint conf. on Languages, compilers and tools for embedded systems: software and compilers for embedded systems(LCTES-SCOPES 2002), pp.213-222, 2002.

[4] A. Qadi, S. Goddard, and S. Farritor, "A dynamic voltage scaling algorithm for sporadic tasks," Proc. of the 24th IEEE Int'l Real-Time Systems Symposium (RTSS'03), pp.52-62, 2003.

[5] C. H. Lee and K. G. Shin, "On-line dynamic voltage scaling for hard real-time systems using the EDF algorithm," Proc. of the 25th IEEE Int'l Real-Time System Symposium (RTSS'04), pp. 319-327, 2004.

[6] H. Aydin, R. Melhem, D. Mosse, and P. M. Alvarez, "Power-aware scheduling for periodic real-time tasks," IEEE Trans. on Computers, Vol.53, pp.584-600, 2004.

[7] S. S. Lee, "Low-Power Video Decoding on Variable Voltage Processor for Mobile Multimedia Applications," ETRI Journal, Vol.27, No.5, pp.504-510, Oct. 2005.

[8] X. Zhong and C. Z. Xu, "Energy-Aware Modeling and Scheduling of Real-Time Tasks for Dynamic Voltage Scaling," Proc. of the 26th IEEE Int'l Real-Time Systems Symposium (RTSS'05), pp.366-375, 2005.

[9] T. D. Burd and R. W. Brodersen, "Energy efficient CMOS microprocessor design," Proc. 28th Hawaii Int'l Conf. on System Sciences, pp. 288-297, 1995.

[10] R. Ernst and W. Ye, "Embedded Program Timing Analysis Based on Path Clustering and Architecture Classification," Proc. Int'l Conf. Computer-Aided Design (ICCAD'97), pp.598-604, 1997.

[11] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," J. ACM, Vol.20, No.1, pp.46-61, 1973.

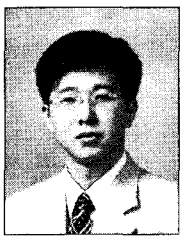
[12] http://rtsim.sssup.it

[13] http://developer.intel.com/design/intelxscale

[14] http://www.transmeta.com

저 자 소 개

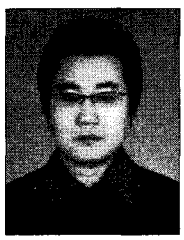
공 민 식(Min-Sik Gong) 정회원



- 1988년 : 충남대학교 전자공학과 (공학사)
- 1990년 : 충남대학교 전자공학과 (공학석사)
- 1990년 ~ 현재 : 국방과학연구소 선임연구원
- 2003년 ~ 현재 : 충남대학교 컴퓨터공학과 박사과정

<관심분야> : 내장형 실시간 시스템, 실시간 운영체제 및 저전력 실시간 스케쥴링

정 근 재(Gun-Jae Jeong) 준회원



- 2006년 : 충남대학교 전자공학과 (공학사)
- 2006년 ~ 현재 : 충남대학교 컴퓨터공학과 컴퓨터시스템 및 멀티 미디어전공 석사과정

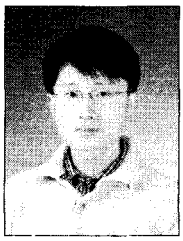
<관심분야> : 내장형 실시간 시스템, 파일 시스템

송 예 진(Ye-Jin Song) 준회원

- 2006년 : 충남대학교 전자공학과 (공학사)
- 2006년~ 현재 : 충남대학교 컴퓨터공학과 컴퓨터시스템 및 멀티 미디어전공 석사과정

<관심분야> : 내장형 실시간 시스템, 자바가상머신

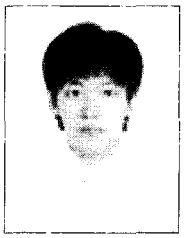
정 명 조(Myoung-Jo Jung) 정회원



- 2003년 : 충남대학교 컴퓨터공학과 (공학석사)
- 2003년 ~ 현재 : 충남대학교 컴퓨터공학과 박사과정

<관심분야> : 자바 가상머신, 실시간 스케줄링 및 실시간 운영체제

조 문 행(Moon-Haeng Cho) 정회원



- 2004년 : 충남대학교 컴퓨터공학과 (공학사)
- 2006년 : 충남대학교 컴퓨터공학과 (공학석사)
- 2006년 ~ 현재 : 충남대학교 컴퓨터공학과 박사과정

<관심분야> : 내장형 실시간 시스템, 실시간 컴퓨팅, 실시간 유비쿼터스 컴퓨팅 및 초소형 저전력 실시간 운영체제

이 철 훈(Cheol-Hoon Lee) 정회원

- 1983년 : 서울대학교 전자공학과 (공학사)
- 1983년 ~ 1986년 : 삼성전자 컴퓨터개발실 연구원
- 1988년 : 한국과학기술원 전기 및 전자공학과 (공학석사)
- 1992년 : 한국과학기술원 전기 및 전자공학과(공학박사)
- 1992년 3월 ~ 1994년 2월 : 삼성전자 컴퓨터사업부 책임연구원
- 1994년 2월 ~ 1995년 2월 : Univ. of Michigan 객원연구원
- 2004년 2월 ~ 2005년 2월 : Univ. of Michigan 교환교수
- 1995년 ~ 현재 : 충남대학교 컴퓨터공학과 교수

<관심분야> : 병렬처리, 운영체제, 실시간 커널 및 결함허용 컴퓨팅