http://dx.doi.org/10.3745/KTCCS.2013.2.8.343

Power-Aware Real-Time Scheduling based on Multi-Granularity Resource Reservation

Joohyung Sun† · Hyeonjoong Cho††

ABSTRACT

We proposes a power-aware fixed-priority real-time scheduling algorithm for multimedia service, called static voltage scaling algorithm with multi-granularity resource reservation (STATIC-MULTIRSV). The multi-granularity resource reservation was introduced to deliver higher system utilization and better temporal isolation than the traditional approaches in [2]. Based on this, our STATIC-MULTIRSV is designed to reduce the power consumptions while guaranteeing that all I-frames of each video stream meet their deadlines. We implemented the proposed algorithm on top of ChronOS Real-time Linux [6]. We experimentally compared STATIC-MULTIRSV with other existing methods which showed that STATIC-MULTIRSV reduce power consumption by maximum 15% compared to its experimental counterparts.

Keywords : Real-time Scheduliing, Dynamic Voltage Scaling, Multimedia, Resource Reservation, Multi-granularity

다중 세분화 자원 예약 기반의 저전력 실시간 스케쥴링 기법

선 주 형† · 조 현 중††

요 약

본 논문에서는 멀티미디어 서비스를 위한 파워-효율적인 고정 우선순위의 실시간 스케쥴링 알고리즘으로 다중 세분화 자원 예약 기반의 정적 전압 조절 알고리즘 (STATIC-MULTIRSV)을 제안한다. 다중 세분화 자원 예약은 전통적인 데드라인 기반의 자원 예약 보다 높은 연산 자원 사용률과 더 나은 태스크간 독립성을 제공하는 모델로서 [2]에서 소개된바 있다. 다중 세분화 모델을 기반으로 제안된 STATIC-MULTIRSV 알고리즘은 비디오 스트림들의 I-프레임들을 각각의 데드라인을 모두 만족하는 것을 보장하면서 파워 소비를 줄이도록 고안되었다. 제안된 알고리즘은 실시간 리눅스[6] 상에서 구현을 통해 실험적으로 기존의 방법에 비해 파워 소모를 최대 15%까지 줄였음을 보였다.

키워드 : 실시간 스케쥴링, 동적 전압 조절, 멀티미디어, 자원 예약, 다중 세분화

1. 서 론

최근 스마트폰 등을 통한 DMB, IPTV, 화상회의 등 다양한 멀티미디어 서비스들의 증가로 인해 임베디드 시스템은 사용자가 만족할만한 서비스의 질(QoS, Quality of Service)을 제공하기 위한 많은 양의 연산을 필요로 한다. 이러한 많은 연산은 필연적으로 높은 전력 소모를 야기하므로 제한된 배터리 수명을 갖고 있는 휴대 및 이동이 가능한 기기들은 전력 소모를 고려한 설계가 필수적이다.

임베디드 시스템에서 운영되는 실시간 소프트웨어는 시스템의 목적을 달성하기 위해 주어진 시간적인 요구 사항(Deadline, 이하 데드라인)을 만족시킬 수 있도록 설계되어야 한다. 대표적인 실시간 서비스인 멀티미디어 서비스는 데드라인의 만족 정도가 서비스의 QoS로 표현될 수 있는 실시간 서비스로 사용자가 원하는 QoS를 제공하기 위해서는 QoS를 정량적으로 표현하고 이를 달성하기 위해 시스템을 설계하는 것이 필요하다.

전통적인 멀티미디어 서비스의 QoS 표현에는 확률적인 방법이 있는데, 이는 태스크의 데드라인 만족(Deadline-Meet, 이하 데드라인 만족)의 시간적 분포를 표현할 수 없다는 단점이 있다. 즉, 100번의 태스크 실행 중 80번의 데드라인 만족이 일어나는 경우와 10번 중 8번의 데드라인 만족

※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2011-0011534).
† 비 회 원 : 고려대학교 컴퓨터정보학과 박사과정
†† 종신회원 : 고려대학교 컴퓨터정보학과 부교수
논문접수 : 2013년 3월 22일
수 정 일 : 1차 2013년 5월 31일
심사완료 : 2013년 5월 31일
* Corresponding Author : Hyeonjoong Cho(raycho@korea.ac.kr)

이 일어나는 경우가 존재할 때, 서비스의 질 면에서는 두 경우가 상이하지만, 확률적으로는 같은 80%로 이들을 구분 할 수 없다. 이러한 단점을 해결하기 위해 매 일정 시간 동안 데드라인 만족(그리고 불만족)의 분포에 제한을 두어 멀티미디어 서비스의 QoS 및 데드라인 불만족에 대한 시스템의 내성(Tolerance)을 확률적이 아닌 결정적으로(Deterministic) 정확하게 표현할 수 있는 약경성 (Weakly-hard) 실시간 제약이 제안되었다[1].

한편 멀티미디어 서비스의 QoS를 필요로 하는 실시간 시스템의 특성을 나타내기 위한 또 다른 방법으로 자원 예약 기반의 다중 세분화 (Resource Reservation-based Multi-Granularity, 이하 MG) 모델이 제안되었다[2]. 이 예약 기반 모델은 기존의 최악 상황을 고려한 예약 모델과는 다르게 MPEG 비디오 스트림 같은 많은 멀티미디어 어플리케이션의 (1) 연산 자원 요구량이 급격히 변하고, (2) 최악상황의 연산 자원 요구량이 평균 연산 자원 요구량에 비해 너무 크고, (3) 연산량이 많은 요청에 의한 태스크의 긴 버스트 시간(Burst Time) 및 (4) 가끔씩 생기는 데드라인 불만족을 허용할 수 있는 특징들을 모두 고려했다. 본 연구에서도 멀티미디어 서비스를 처리하기 위해 이 MG 모델을 실시간 태스크에 적용시킨다.

한편 회로의 전력 소모를 줄이기 위해 많은 집적 회로 제조사들은 클락 주파수와 연동되어 있는 공급 전압을 상황에 따라 동적으로 조절함으로써 전력소모를 줄일 수 있는 동적 전압 조절(Dynamic Voltage Scaling, DVS)기능을 포함한 저전력 프로세서들을 개발해왔다. 인텔의 XScale, AMD의 모바일 애슬론, 삼성의 Cortex가 좋은 예다. CMOS 회로의 에너지 소비량은 공급 전압의 2차식()[3]을 따르기 때문에, 공급 전압을 낮추는 방법은 에너지 소비를 줄이는 효과적인 기술 중에 하나이다. Pillai[4]는 시스템 시작 전에 결정되는 정적 우선순위를 갖는 실시간 태스크의 온라인 DVS 스케쥴링 기법을 제안했다. Pillai는 시스템 시작 전에 최악상황 반응시간 (Worst-case Response Time, WCRT 또는 WR) 분석을 기반으로 모든 태스크들의 데드라인을 만족하는데 필요한 클락 속도 중 가장 낮은 수치를 찾아 놓고, 시스템 시작 후 태스크가 일찍 끝남으로써 생기는 시스템의 남는 시간 (Slack Time) 을 자신 보다 높은 우선순위를 갖는 태스크에게 추가로 할당해 ccRM (cycle-conserving Rate Monotonic) 알고리즘을 제안한바 있다.

본 연구에서는 앞서 설명한 MG 모델에 맞는 WCRT 분석 방법을 새롭게 설계하고, 이를 이용해 시스템 시작 전에 태스크들의 데드라인을 만족하는데 필요한 클락 속도 중 가장 낮은 값을 찾는 방법을 제안한다. 또 (1) 최소한의 멀티미디어 QoS를 제공하면서도, (2) 시스템의 파워 소모를 줄이기 위해 단일 프로세서 상의 멀티미디어 서비스를 위한 저전력 실시간 스케쥴링 알고리즘을 제안한다.

제안 알고리즘의 성능을 기존 방법과 비교 및 분석하기 위해 ChronOS Real-time Linux[6]가 설치된 DELL i7-2630QM에서 이들을 구현하고, 구현 시스템에서 발생하는 실제 파워를 전기 계측/측정 장비인 WT210[5]으로 측정하며 실험하였다.

2. 본 론

2.1 시스템 모델

1) 태스크 모델

n개의 주기적 태스크들의 집합은 $\{\tau_1, \tau_2, ..., \tau_i, ..., \tau_n\}$으로 표기한다. 각각의 태스크들은 서로 독립적이며, 우선순위 순서로 나열되어 있다. 태스크들의 우선순위는 주기가 짧을수록 높은 우선순위를 갖는 정적 우선순위 기반의 RM (Rate-Monotonic) 스케쥴링으로 결정 한다. 즉, 주기(T)가 가장 짧은 τ_1이 가장 높은 우선순위를 갖고, 주기가 가장 긴 τ_n이 가장 낮은 우선순위를 갖는다. 각각의 태스크는 $\tau_i = \{prio_i, T_i, C_i, D_i\}$으로 표기되며, $prio_i$는 태스크의 우선순위, T_i는 주기, C_i는 최악상황의 실행시간, D_i는 데드라인이다. 여기에서 T_i와 D_i는 같다고 가정한다($T_i = D_i$).

2) MG 모델

전통적으로 사용했던 예약 모델 $\{C, T, D\}$ 대신에 Saewong이 제안한 MG 모델을 사용한다[2]. MG 모델은 $\{C, T, D\}$, $\{C^x, \varepsilon^x T\}$, ..., $\{C^y, \varepsilon^y T\}$로 표기하고, 여기서 $\varepsilon^x < ... < \varepsilon^y$, 그리고 $\forall_i, \varepsilon^i \in Z+$로 x, y는 각 Reserve 단계를 나타내며, ε은 각 Reserve 주기를 순서대로 구분해준다. 여기서 $\{C, T, D\}$는 가장 높은 단계의 Reserve로 매 T시간 마다 C만큼 제공되며, D는 데드라인이다. $\{C^x, \varepsilon^x T\}$는 한 단계 낮은 Reserve로 시간 $\varepsilon^x T$마다 평균적으로 C^x만큼 제공된다는 뜻이다. 각각의 세분화 시간 구간($\varepsilon^x T$)마다 그 단계에 해당하는 자원(C^x)이 계속 제공된다. 만약에 태스크가 예약된 양보다 더 많이 사용하려고 한다면 자원을 제한하고 태스크가 더 이상 실행되지 않도록 한다. 이는 태스크의 우선순위를 다른 일반 태스크(비실시간 태스크)와 같거나 또는 더 낮은 우선순위까지 강등시킴으로써 구현된다.

2.2 고정 클락 속도 결정 알고리즘

1) 스케쥴링 동작 및 클락 속도 결정 방법

제안한 알고리즘 STATIC-MULTIRSV는 MULTIRSV[2]의 스케쥴링과 같이 RM (Rate-Monotonic) 스케쥴링을 채용한다. STATIC-MULTIRSV는 MG 모델 기반의 실시간 태스크가 주어졌을 때 스케쥴 가능성(Schedulability)을 유지하면서 파워 소모를 줄이기 위해 시스템의 시작 전에 주파수를 정적으로 설정하는데, 이 주파수는 RM 스케쥴러가 주어진 실시간 태스크들의 데드라인을 모두 만족시키는 가능한 가장 낮은 주파수로 결정된다. 해당 주파수는 MG 모델 적용을 위해 확장된 Pillai[4]의 최악상황 반응시간 분석을 바탕으로 계산된다.

RM 스케쥴링에서 태스크의 최악상황 반응시간은 태스크의 최악상황 실행시간과 자신보다 더 높은 우선순위를 갖는 태스크들이 선점하는 시간의 총 합으로 구해진다. 시간 구간 $(t_0,\ t_0+t)$에서 클락 속도 요소를 고려하는 최대 선점시간은 다음 식으로 계산되며 $P_i^{(t_0,t_0+t)}$로 표기한다. 이는 [2]에서의 최대 선점시간 계산을 따르면서 DVS를 고려하여 시스템에서 주어진 클락 속도 $S_l \in \{S_1,\ ...,\ S_{n_f} \mid S_1<...<S_{n_f}\}$ (n_f는 클락 속도 요소의 총 개수) 중 선택된 속도 S_l를 기반으로 최악상황 실행시간(C_i)을 $\frac{C_i}{S_l}$로 새로 계산한 식이다.

$$P_i^{(t_0,t_0+t)}=\mathrm{MIN}(\lfloor \frac{t}{\varepsilon_i^{g_i}T_i}\rfloor \frac{C_i^{g_i}}{S_l}+\lfloor \frac{t-m^{g_i}}{\varepsilon_i^{(g_i-1)}T_i}\rfloor \frac{C_i^{(g_i-1)}}{S_l}$$
$$+...+\lfloor \frac{t-(\sum_{j=2}^{g_i}m^j)}{\varepsilon_i^1 T_i}\rfloor \frac{C_i^1}{S_l}+(t-(\sum_{j=1}^{g_i}m^j)),\lfloor \frac{t}{\varepsilon_i^{g_i}T_i}\rfloor \frac{C_i^{g_i}}{S_l}$$
$$+\lfloor \frac{t-m^{g_i}}{\varepsilon_i^{(g_i-1)}T_i}\rfloor \frac{C_i^{(g_i-1)}}{S_l}+...+\lceil \frac{t-(\sum_{j=2}^{g_i}m^j)}{\varepsilon_i^1 T_i}\rceil \frac{C_i^1}{S_l},\ \dots\ ,$$
$$\lfloor \frac{t}{\varepsilon_i^{g_i}T_i}\rfloor \frac{C_i^{g_i}}{S_l}+\lceil \frac{t-m^{g_i}}{\varepsilon_i^{(g_i-1)}T_i}\rceil \frac{C_i^{(g_i-1)}}{S_l},$$
$$\lceil \frac{t}{\varepsilon_i^{g_i}T_i}\rceil \frac{C_i^{g_i}}{S_l}) \qquad (1)$$

여기에서 $m^x=\lfloor \frac{t-m^{(x+1)}}{\varepsilon_i^x T_i}\rfloor \varepsilon_i^x T_i$, $m^{(g_i+1)}=0$, $C_i^1=C_i$, $\varepsilon_i^1=1$, g는 세분화 단계(g_2는 2단계).

다중 세분화 리소스 예약 시스템에서 τ_i의 최악상황 반응시간 w은 다음 식을 따른다[2].

$$w^{k+1}=\frac{C_i}{S_l}+\sum_{j<i}P_j^{(0,w^k)} \qquad (2)$$

여기서 $P_j^{(0,w^k)}$는 구간 $(0,w^k)$에서 τ_i보다 높은 우선순위를 갖는 태스크들의 선점시간이다.

식 (2)는 시스템에서 주어진 클락 속도 중에서 다음의 조건을 만족하는 가장 작은 클락 속도 요소 S_l을 찾기 위해 이용된다. k는 0부터, 즉, $w^0=\frac{C_i}{S_l}$부터 재귀적으로 시작하고, $w^{k+1}=w^k$일 때 "성공"으로 끝나서 가장 최근에 선택된 S_l을 고정 클락 속도로 설정하거나, w^{k+1}가 D_i을 넘어가게 되면 ($w^{k+1}>D_i$) "실패"로 끝나 고정 클락 속도를 S_{n_f}로 설정한다. 식 (2)를 통해 선택된 S_l에 대해서 "성공"일 경우, 각각의 태스크에 대해서 최악상황 반응시간이 $w^{k+1}=w^k \le D_i$이므로 RM 스케쥴링 기반의 최악상황 반응시간 분석[4]에 의해서 스케쥴 가능하다.

식 (2)의 시간 복잡도는 주어진 태스크들의 총 개수(n)와 가능한 클락 속도의 총 개수(n_f) 뿐만 아니라 세분화 단계(g_i)에 의해서도 영향을 받는다. 여기서 n_f와 g_i는 상수로, 시스템 시작 전에 연산되는 식 (2)의 복잡도는 $O(n^2)$이다. 그리고 시스템 동작 시에는 복잡도가 $O(n)$인 RM 스케쥴링 기반으로 동적으로 동작한다.

2) 예제

다음 Table 1과 Fig. 1은 제안한 알고리즘이 MG 모델 기반의 일반적인 실시간 태스크 집합이 주어졌을 때 고정 클락 속도를 찾는 예제이다. Table 1은 MG 모델 기반의 태스크 집합이다. 시스템 시작 전에 주어진 태스크 집합과 식 (2)로 고정 클락 속도를 계산한다. 먼저, 시스템이 제공하는 고정 클락 속도 S_l에 대응되는 주파수(frequency)가 $freq_i$ ={0.75, 0.8, 0.85, 0.9, 0.95, 1.0}으로 주어진다고 가정한다.

먼저 $freq_2$=0.8일 때, 최악상황 반응시간을 계산해 스케쥴 가능성을 계산하면, τ_1일 때 w^0=1.25, w^1=1.25. 즉, $w^0=w^1$ =1.25<3으로 "성공", τ_2일 때 w^0=1.25, w^1=2.5, w^2=2.5. 즉, $w^1=w^2$=2.5<4으로 "성공", τ_3일 때 w^0=1.25, w^1=3.75, w^2=4.5, w^3=5.5, w^4=6.25. 즉, w^4=6.25<6으로 "실패"이기 때문에 다음 $freq_3$=0.85로 앞에서 했던 방법으로 다시 계산한다. 그럼 주어진 $freq_3$에서 모든 태스크 τ_1, τ_2, τ_3의 최악상황 반응시간이 모두 데드라인을 넘지 않기 때문에 $freq_3$에 대응되는 고정 클락 속도가 정해진다.

Fig. 1은 고정 클락 속도가 정해졌을 때 실제로 일어나는 스케쥴링을 보여준다. Fig. 1A는 $freq_n$=1로 스케쥴링 되고, Fig. 1B는 $freq_2$=0.8로 스케쥴링 되는데 앞에서 스케쥴 가능성 테스트에서 "실패"였기 때문에 t=6에서 τ_3이 데드라인 불만족된다. Fig. 1C는 스케쥴 가능성을 처음 "성공"했던 $freq_3$=0.85로 스케쥴링 되는 것을 보여준다. 앞서 수식에서 예측된대로 스케쥴 가능성을 "성공"했기 때문에 모든 태스크가 데드라인을 만족함을 시각적으로 알 수 있다.

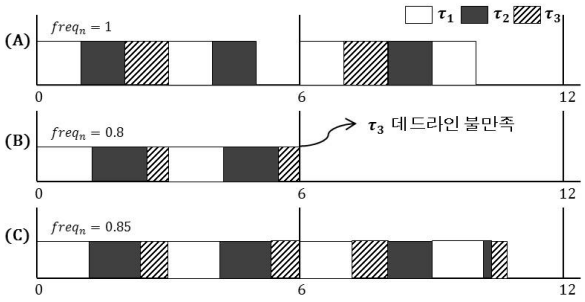


Fig. 1. Real-time Scheduling

Table 1. Real-time Task Sets

Tasks	MG reservation model, *Reserve*
τ_1	{ {1, 3, 3}, {4, 30} }
τ_2	{ {1, 4, 4}, {4, 40} }
τ_3	{ {1, 6, 6}, {5, 60} }

2.3 실험 방법

제안한 알고리즘 STATIC-MULTIRSV을 ChronOS Real-time Linux[6]가 설치된 DELL i7-2630QM 플랫폼 상에서 구현하고, 시스템의 파워 소모를 전기 계측/측정 장비인 WT210[5]으로 측정했다. ChronOS는 실시간 리눅스 패치로, 리눅스에서 실시간 처리를 가능(CONFIG_PREEMP_RT)하게 해줌으로써 실시간 스케쥴링을 위한 리눅스 커널 테스트베드를 제공해준다. STATIC-MULTIRSV과의 성능 비교를 위해 기존 MULTIRSV를 실험 비교 대상으로 삼았다.

2.4 실험 플랫폼 및 환경

실험 환경이나 검증 방법은 [2]의 실험과 동일하게 설정했다. 실험 플랫폼의 리눅스 상에는 기본 프로세스들이 디폴트로 동작하고, 동시에 비디오 프레임들과 실시간/비실시간 태스크들이 동작한다. Table 2는 [2]의 실험에 사용한 프레임들의 통계치로, 모든 비디오 스트림은 IBBPBBPBBPBB의 GOP(Group of Picture) 패턴을 사용한다. 그 외의 태스크는 5개의 실시간 태스크, 5개의 비실시간 태스크로 짧거나(1~10ms), 중간쯤이거나(10~100ms), 긴(100~1000ms) 주기로 균일 확률분포(Uniform Distribution) 함수에 의해 결정된다. 그리고 각각의 시나리오에 따라 주어진 전체 Utilization과 주기에 맞게 임의로 최악상황 실행시간이 설정되고, 시스템 시작 후의 실제 실행시간은 최악상황에 가까운 시간으로 설정된다. 먼저 프레임 태스크의 MPEG4 비디오 디코딩을 위한 간단한 다중 세분화 Reserve는

$$R = \{\{dt_{\max}, T, T\}, \{dt_{avg}*gop, T*gop\}\} \quad (3)$$

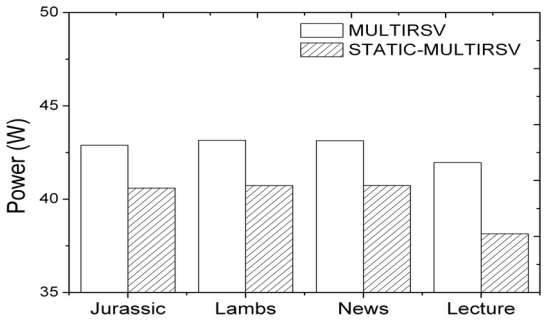
로 주어지는데, 여기에서 gop는 GOP의 크기, T는 프레임의 주기, $dt_{\max}$는 최대 디코딩 시간, dt_{avg}는 평균 디코딩 시간이다.

Table 2. Video Frame Statistics

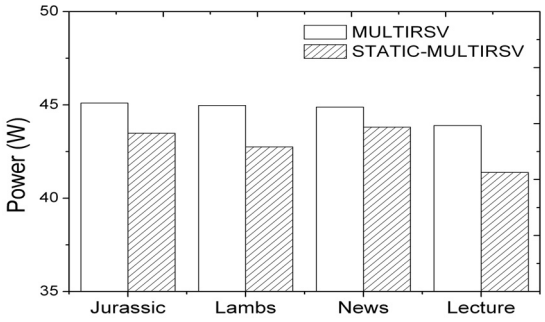
프레임 통계	Jurassic	Lambs	News	Lecture
압축률 (YUV:MP4)	9.92	13.22	10.52	36.27
프레임 비율	25	25	25	25
실행시간 (ms)	3.6e+06	3.6e+06	9e+05	3.6e+06
프레임 크기의 평균	3.8e+03	2.9e+03	3.6e+03	1e+03
프레임 크기의 분산	5.1e+06	5.2e+06	6.3e+06	8.3e+05
프레임 크기의 공분산	0.59	0.80	0.70	0.87
프레임 크기의 최소	72	158	123	344
프레임 크기의 최대	16745	22239	17055	7447
평균 시스템 사용률	0.103	0.0905	0.1025	0.064

프레임 이외의 실시간 태스크들의 다중 세분화 Reserve는 다음 식으로 주어진다.

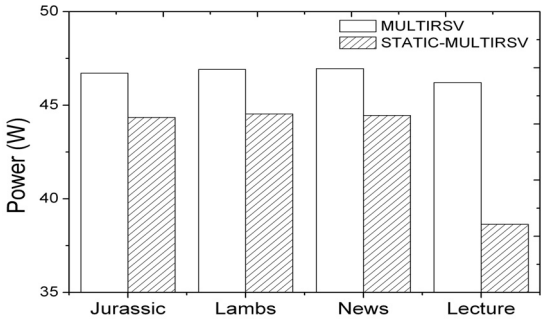
$$R = \{\{C_i, T_i, T_i\}, \{C_i, T_i\}\} \quad (4)$$



(A) RT-U=0.4 / U=0.5



(B) RT-U=0.65 / U=0.75

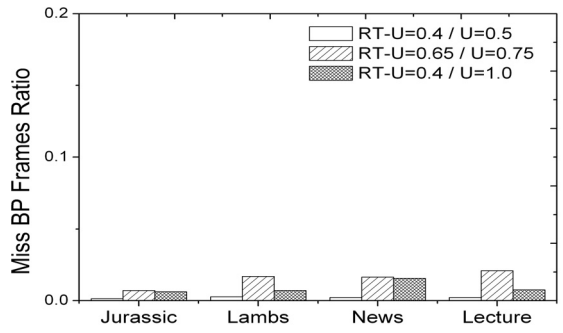


(C) RT-U=0.4 / U=1.0

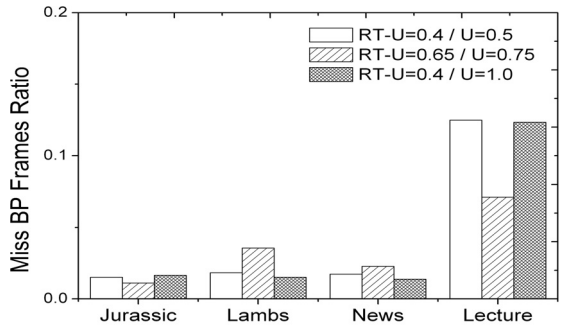
Fig. 2. Power-efficient experimental results

스트림을 위한 (m,k)-firm 데드라인 조건[7]은 $(m_{mpeg}, k_{mpeg}) = (\frac{dt_{avg}*gop}{dt_{\max}}, gop)$로 [2]와 같은 방법으로 실험한다. 각 프레임의 비디오 디코딩 시간은 해당하는 프레임 크기에 비례한다. 다음 측정치는 스케쥴링 알고리즘 성능을 평가하는데 사용된다.

- Miss frames: 모든 프레임들의 데드라인 불만족 개수
- Miss I-frames: 모든 프레임들의 총 개수 대비 I-프레임 데드라인 불만족 개수 비율
- Miss BP-frames: 모든 프레임들의 총 개수 대비 B 및 P-프레임 데드라인 불만족 개수 비율
- Dyn: dynamic error([7]에서 정의한 k개의 연속된 프레임에서 m개 미만으로 시간 조건이 만족됐을 때 생기는 Dynamic Failure와 같은 의미) 대비 가능한 k개의 연속된 프레임 집합들의 총 개수



(A) MULTIRSV



(B) STATIC-MULTIRSV

Fig. 3. Experimental results for performance (No Missed I-frames and No Dynamic Error)

실험은 앞에서 언급한 총 10개의 실시간 및 비실시간 태스크들과 Jurassic, Lambs, News, Lecture의 각 프레임에 대하여 총 세 개의 실험 시나리오로 진행한다. 먼저 각 시나리오에 대해서 실시간 태스크들의 시스템 사용률 (Real-time Utilization, 이하 RT-U), 비실시간 태스크들의 시스템 사용률 (Non- real-time Utilization, 이하 NRT-U), 전체 시스템 사용률(System Utilization, 이하 U)을 서로 다르게 정하는데, 이 사용률에는 비디오 프레임 사용률이 포함되어 있지 않다. 즉, 비디오 프레임 사용률을 제외한 시스템의(실시간 태스크들의) 작업량을 의미한다.

세 가지 시나리오는 각각 (1) 적은 작업량: RT-U=0.4/U=0.5, (2) 높은 작업량: RT-U=0.65 /U=0.75, (3) 적은 작업량, 높은 백그라운드 작업량(NRT-U=0.6): RT-U=0.4/U=1.0으로 앞서 설명한 알고리즘 성능에 대한 요소와 파워 소모에 중점을 맞춰 실험을 진행한다.

2.5 실험 결과

Fig. 2와 Fig. 3은 알고리즘, 시나리오, 비디오 프레임에 대한 실험 결과를 보여준다. 모든 실험에 대해서 Miss I-frames와 Dyn이 일어나지 않는다. 이는 본 논문에서 제안한 기법이 최소한의 멀티미디어 QoS를 보장함을 의미한다. 또 Fig. 2A와 Fig. 2C에서 RT-U가 적을 때, 기존의 알고리즘 보다 최대 15%의 파워 효율을 증가시켰음을 보여준다. 파워 효율이 증가된 이유는 RT-U가 적을수록 식 (2) 중 높은 우선순위를 갖는 태스크들의 선점시간(P)이 적으므로 더 낮은 고정 클락 속도를 사용할 수 있기 때문이다.

반면, Fig. 2B와 같이 RT-U가 RM의 스케쥴 가능성의 한계 0.69에 가까워질 경우 실시간성 보장을 위해 정적 알고리즘의 클락 속도를 낮추는 것이 어려우므로 파워 효율이 좋지 않다. 대신 낮은 클락 속도에서는 태스크의 실행 시간이 길어져 예약된 Reserve 양보다 더 많은 시간을 사용하게 되므로, Fig. 3에서 STATIC-MULTIRSV가 기존의 알고리즘보다 더 많은 B, P-프레임의 데드라인을 놓치고 있는 것을 볼 수 있다. 특히, Lecture 비디오 실험에서 상대적으로 많은 Miss BP-frames이 생기는 이유는 최대 프레임 크기가 작아서 식 (2)의 선점시간이 적어지고, 다른 프레임에 대한 실험에 비해 더 낮은 고정 클락 속도를 사용하게 되기 때문이다.

그럼에도 불구하고 Miss I-frames와 Dyn이 모든 실험에 대해 일어나지 않으므로 STATIC-MULTIRSV는 최소한의 멀티미디어 QoS를 제공하고 있음을 보여준다. I-프레임은 모든 비트 정보들을 갖고 있어서 디코딩 시 다른 프레임을 필요로 하지 않는 독립형 프레임이다. 만약에 GOP는 12, fps는 25인 멀티미디어 스트림이 존재할 때, 1초에 2~3개의 I-프레임만이라도 디코딩 한다면 영상이 0.5초 이상 멈춰있는 현상을 없앨 수 있다. 반면 P/B-프레임들은 I-프레임을 참조하여 디코딩되는 프레임이다. 즉 P/B-프레임 디코딩 태스크가 데드라인을 만족한다하더라도 I-프레임의 정보 없이는 디코딩될 수 없으므로 I-프레임 디코딩 태스크가 데드라인을 만족하는 것은 중요하다. 그러므로 STATIC-MULTIRSV는 최소한의 멀티미디어 QoS를 해치지 않도록 I-프레임 태스크의 데드라인을 항상 만족시킬 수 있는 최적 클락 속도를 찾아 동작한다.

3. 결 론

본 논문에서는 단일 프로세서 상의 멀티미디어 서비스를 처리하기 위해 실시간성을 포함한 최소한의 멀티미디어 QoS를 제공하면서 시스템의 파워 소모를 줄이는 임베디드 실시간 스케쥴링 기법 STATIC-MULTIRSV를 제안했다. STATIC-MULTIRSV는 멀티미디어 의 QoS를 보장하기 위해 MG 모델을 채택하고, 파워 소모를 줄이기 위해 DVS 기술을 사용했다. 실제 구현을 통해 실험적으로 제안 스케쥴링 알고리즘이 멀티미디어의 최소 성능을 보장하면서, 파워 효율을 최대 15%까지 증가시킴을 보였다.

참 고 문 헌

[1] Bernat, G., Burns, A., Liamosi, A. “Weakly hard real-time systems,” Computers, IEEE Trans. on, Vol.50, No.4, pp.308-321, Apr., 2001.

[2] Saowanee Saewong, Ragunathan (Raj) Rajkumar. “Multi-Granularity Resource Reservations”. IEEE RTSS, 2005.

[3] BURD, T. D., BRODERSEN, R. W. “Energy efficient CMOS microprocessor design”. Hawaii International Conference on System Sciences, 1995.
[4] Padmanabhan Pillai, Kang G. Shin. “Real-time dynamic voltage scaling for low-power embedded operating systems”. SIGOPS Oper. Syst. Rev. 35, 5 (October 2001), pp.89-102.
[5] YOKOGAWA. “White Paper of WT210/WT230 DIGITAL POWER METERS”. http://tmi.yokogawa.com/technical - library/ white-papers/wt210wt230-digital-power-meters/
[6] Systems Software Research Group at Virginia Tech. “ChronOS Real-time Linux”. http://chronoslinux.org/wiki /Main_Page
[7] Hamdaoui, M., Ramanathan, P. “A dynamic priority assignment technique for streams with (m, k)-firm deadlines,” Computers, IEEE Trans. on, Vol.44, No.12, pp.1443-1451, Dec., 1995.

선 주 형
e-mail : joohyung_sun@korea.ac.kr
2010년 고려대학교 컴퓨터정보학과(학사)
2012년 고려대학교 컴퓨터정보학과(석사)
2012년～현 재 고려대학교 컴퓨터정보학과 박사과정
관심분야 : Real-time scheduling, Energy-aware algorithm, Resource reservation

조 현 중
e-mail : raycho@korea.ac.kr
1996년 경북대학교 전자공학부(학사)
1998년 포항공과대학교 전자전기공학과 (석사)
2006년 9월 Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg VA(Ph.D)
2009년～현 재 고려대학교 컴퓨터정보학과 부교수
관심분야 : Real-time computing for embedded systems, Human-computer interactions for smart devices