Journal of Korea Multimedia Society Vol. 14, No. 7, July 2011(pp. 940-948)

결함허용을 고려한 실시간 임베디드 태스크 스케줄러

전태건†, 김창수††

요 약

본 논문에서는 단일 처리기를 가지는 임베디드 시스템에서 실시간성과 결함 허용을 고려한 태스크 스케줄러를 설계하고 구현한다. RMS(Rate Monotonic Scheduling) 기법을 이용하여 주기적 태스크를 실행하고 태스크의 실행 마감 시간을 보장하며 잉여 시간을 관리함으로써 비주기적 태스크의 실행 및 완료 방법을 제시한다. 또한 백업 시간을 관리함으로써 일시적인 태스크의 단일 결함을 허용하기 위한 결함 허용 기법을 제공한다. 주기적 태스크와 비주기적 태스크의 응답시간을 조절할 수 있는 주기적 태스크의 중요도를 제시한다. 마지막으로 시뮬레이션을 통해 제시한 방법의 결과를 분석하고 평가한다.

A Real-Time Embedded Task Scheduler considering Fault-Tolerant

Taegun Jeon†, Changsoo Kim††

ABSTRACT

In this paper, we design and implement a task scheduler that considers real-time and fault tolerance in embedded system with a single processor. We propose a method how it can meet the deadlines of periodic tasks using RMS and complete the execution of aperiodic tasks by calculating surplus times from a periodic task set. And we describe a method how to recover of a transient fault task by managing backup time. We propose an important level of periodic tasks that can control the response time of periodic and aperiodic tasks. Finally, we analyse and evaluate the proposed methods by simulation.

Key words: Fault-Tolerant(결함허용), Real-Time(실시간), Embedded System(임베디드 시스템), Task Scheduler(태스크 스케줄러)

1. 서 론

최근 임베디드 시스템은 하드웨어 기술의 발달로 시스템의 수행 능력이 증가하고 있으며 이를 이용하는 사용자의 요구 또한 다양하게 증가하였다. 또한 산업 제어 시스템, 항공기, 우주 왕복선, 스마트 자동차 등과 같은 작업을 수행함에 있어서 완료 시간에 대한 제약을 만족해야하는 경우가 있다[1-4]. 이러한 작업 완료 시간에 대한 제약을 만족시키기 위해서 시스템은 외부 반응에 대해서 즉각적으로 응답하는 실시간성을 제공할 수 있어야 하며 이를 위해서는 실행되는 태스크의 속성(태스크의 주기성, 실행시간, 마감시간 등)에 따른 스케줄링 정책을 제공할 필요가 있다.

안전에 민감한 임베디드 실시간 시스템인 의료장비나 자동차 엔진 제어, 고속전철 제어 등의 시스템에 적용되는 임베디드 운영체제는 다양한 이유로 실행되는 작업의 결함이 발생할 수 있다. 이러한 시스템에서 발생하는 작업의 결함으로 인한 시스템의 결함은 경제적 손실 뿐만아니라 심지어 인간의 생명에

※ 교신저자(Corresponding Author) : 김창수, 주소 : 부산광역시 남구 대연3동 599-1 부경대학교 대연캠퍼스 1호관 302호실, 전화 : 051)629-6245, FAX : 051)629-6230, E-mail : cskim@pknu.ac.kr
접수일 : 2010년 12월 16일, 수정일 : 2011년 3월 14일
완료일 : 2011년 6월 21일
† 정회원, (주)인사이트정보
(E-mail: nicejtg@naver.com)
†† 종신회원, 부경대학교 IT융합응용공학과

영향을 미칠 수 있다. 또한, 임베디드 시스템의 특성상 사람이 직접적으로 접근하기 어려운 곳에 설치될 수가 있으며 이러한 시스템에서 시스템의 결함이 발생한 경우 사람이 즉각적으로 직접 제어하기 어려운 경우가 있다. 따라서 임베디드 시스템은 태스크 결함으로 인해 발생할 수 있는 문제점들을 해결할 수 있어야 하며, 이를 위해서는 기존의 결함 허용 기법들에 대한 응용 및 임베디드 실시간 시스템에 적용할 수 있는 방법에 대한 연구가 필요하다.

본 논문은 임베디드 시스템에서 실시간성과 결함허용을 보장하는 태스크 스케줄러를 설계하고 구현한다. 구현된 스케줄러는 실시간성을 보장하기 위해서 태스크의 특성(태스크의 주기성, 비주기성, 태스크의 실행시간, 태스크의 마감시간 등)을 고려하여 우선순위 기반인 RMS(Rate Monotonic Scheduling) 기반에서 태스크들을 스케줄링한다. 또한 태스크 수행 중 결함이 발생할 경우 결함이 발생한 태스크를 재실행함으로써 결함을 복구하는 기법을 제공한다. 그리고 주기적 태스크의 실행에 있어서 중요도라는 개념을 도입함으로써 주기적 태스크 중에서 빠른 응답시간이 필요한 경우 이 태스크에 대해서 선택적으로 빠른 응답 시간을 제공할 수 있도록 한다. 이를 위해 본 논문에서는 태스크 스케줄러와 관련된 기존 연구들을 분석하여 임베디드 태스크 스케줄러에 적용할 수 있는 방법을 찾은 후, 스케줄러를 설계하고 구현한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 태스크 실시간 스케줄링 알고리즘과 태스크 결함 허용 알고리즘에 대해 기술한다. 3장에서는 본 논문에서 제안하는 태스크 스케줄러를 설계하고 스케줄러 구성 요소에 대해 기술한다. 4장에서는 3장에서 제안한 스케줄러를 구현하고 그 결과를 분석한 후, 마지막으로 5장에서는 결론을 맺는다.

2. 관련 연구

Liu와 Layland[5]가 제안한 RMS(Rate Monotonic Scheduling)기법은 고정 우선순위 기반 선점형 스케줄링 방식 하에서는 최적의 스케줄링 방법임을 증명하였으며 태스크 집합의 실행 가능 여부를 검사하여 태스크의 실행 가능성을 판단하였다. 그러나 [5]의 전체 처리기 이용률을 만족하지 않는 태스크 집합이라 할지라도 RMS에 의해 실행이 가능함을 Lehoczky [6] 등이 제시하였다. Joseph 등[7]과 Audsley 등[8]은 태스크의 최악 반응시간(worst case response time)을 계산함으로써 태스크 스케줄 가능성을 분석하는 방법을 제안하였으며 Tindell 등[9]은 임의의 마감시한을 가지는 태스크들에 대해 최악 반응 시간 계산을 적용하여 스케줄 가능성 분석을 위한 필요충분조건을 제시하였다.

Lehoczky와 Ramos-Theul[10]은 고정 우선순위 시스템에서 비주기적 태스크 스케줄링을 위해서 슬랙 스틸링(slack stealing) 알고리즘을 제안하였으며, 정경훈 등[11]과 김병훈 등[12]은 실시간 센서 노드 플랫폼에서 실시간성을 제공하기 위하여 주기적 태스크의 마감시한을 보장하고 비주기적 태스크의 응답시간을 최소화하는 방법을 제공하였다. 김희헌 등[13]은 마감시간이 있는 주기적 태스크의 마감시간을 보장하고 마감시간이 없는 비주기 태스크의 빠른 응답시간을 보장하는 방법을 제안하였다. 정경훈 등[14]은 임베디드 실시간 운영체제 수준에서 주기적 및 비주기적 태스크 스케줄링뿐만 아니라 태스크들의 일시적인 결함을 복구할 수 있는 임베디드 실시간 태스크 관리 메커니즘을 제안하였다.

Gosh 등[15]은 두 개의 태스크 요청들 사이에 결함이 발생한 태스크의 재실행에 이용될 수 있는 슬랙 타임(slack time)을 확보함으로써 같은 처리기 상에서 결함 태스크의 재실행을 통해 일시적인 결함을 허용하는 RMS 알고리즘을 제안하였다. Yu Chen 등[16]은 단일 프로세서 상에서 태스크들 간의 의존성이 없다는 가정 하에 주기적 실시간 태스크들의 실행시간을 높일 수 있는 결함허용 실시간 스케줄링 기법(ICFTRM: Imprecise Computation Fault-Tolerant Rate-Monotonic)을 제안하였다. Y. S. Hong[17]은 분산 실시간 시스템에서 태스크의 처리기 이용률에 따라 태스크를 분류하여 primary/backup 기반으로 결함이 발생한 primary 태스크의 재실행을 backup 태스크에서 재실행함으로써 결함을 복구하는 주기적 태스크들을 위한 결함 허용 스케줄링 방법을 제안하였다.

3. 태스크 스케줄러 설계

3.1 시스템 구성

본 논문에서 제안하는 결함허용을 고려한 실시간

임베디드 태스크 스케줄러(FTRTS: Fault-Tolerant Real-Time embedded task Scheduler)의 전체 구조와 구성 요소는 그림 1과 같다.

제안한 스케줄러는 실행 가능성 검사 관리자(ECM: Executability Check Manager), 타임 관리자(BSTM: Backup Surplus Time Manager), 결함 허용 관리자(FTM: Fault-Tolerant Manager), 비주기적 태스크 스케줄러(APTS: Aperiodic Task Scheduler)와 주기적 태스크 스케줄러(PTS: Periodic Task Scheduler)로 구성되어 있다. 각 구성요소에 대해서는 3.2절~3.6절에서 상세히 기술하며 본 논문에 적용한 태스크 집합에 대해서 다음과 같은 가정을 한다.

- 주기적 태스크 집합은 모든 태스크들의 초기 시작 시간이 동일한 태스크 집합이다.
- 시스템은 하나의 처리기를 가진다.
- 태스크 선점의 비용과 스케줄링 오버헤드는 고려하지 않는다.
- 비주기적 태스크의 실행은 마감시간을 고려하지 않은 즉, 연성 비주기적 태스크를 고려한다.
- 태스크의 결함 발생을 위한 비용은 고려하지 않는다.

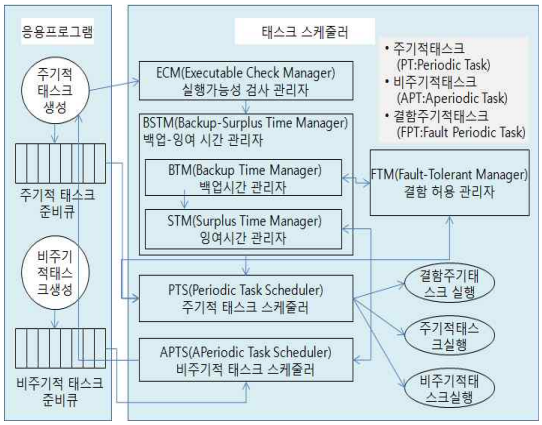


그림 1. FTRTS 구성도

3.2 ECM(Executability Check Manager)

ECM은 주기적 태스크의 속성값(태스크의 실행시간, 마감시간) 그리고 결함 허용을 위한 태스크 백업 시간을 고려하여 주기적 태스크들의 실행 가능성을 검사한다. 이를 위해 본 논문에서는 결함을 복구하기 위한 백업 시간([15])과 스케줄 가능성 검사 방법([6])을 이용한다.

그림 2는 ECM의 수행 흐름도를 나타낸다.

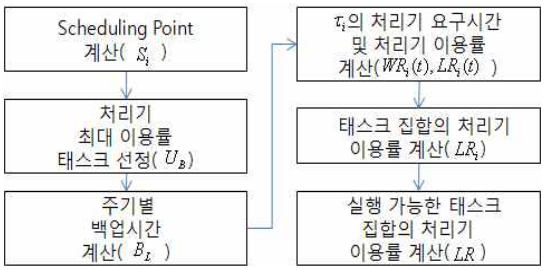


그림 2. ECM 수행 흐름도

주기적 태스크 집합(τ)에 포함된 주기적 태스크(τ_i) 의 속성값을 이용하여 스케줄링 포인트($S_i = t$)와 하이퍼 주기를 계산한다. t는 τ_i의 각 주기인 마감시간이며 처리기 이용률을 계산하는 기준 시간이 된다. 그리고 수식 (1)을 이용하여 태스크 집합의 처리기 이용률 중에서 최대 처리기 이용률을 찾은 후, 하이퍼 주기에 대한 태스크들의 주기별 백업 시간을 수식 (2)에 의해 계산한다.

$$U_B = \max(U_i),\ U_i = C_i / T_i \tag{1}$$

$$B_L = U_B * (lT_j - kT_i) \tag{2}$$

수식 (3)을 이용하여 백업 시간을 적용한 τ_i의 처리기 요구시간을 계산하고 수식 (4)를 이용하여 처리기 이용률을 계산한다. 수식 (5)와 수식 (6)을 이용하여 태스크 집합의 처리기 이용률과 실행 가능한 태스크 집합의 처리기 이용률을 각각 계산한 후, $LR \leq 1$인 경우 태스크 집합은 실행 가능하고 그렇지 않은 경우 실행 불가능으로 판단한다.

$$WR_i(t) = \sum_{j=1}^{i} C_j \cdot \lceil t / T_j \rceil + B_L \tag{3}$$

$$LR_i(t) = WR_i(t)/t \tag{4}$$

$$LR_i = \min_{\{0 < t \leq T_i\}} LR_i(t) \tag{5}$$

$$LR = \max_{\{1 \leq i \leq n\}} LR_i \tag{6}$$

표 1은 주기적 태스크 집합 τ={(2,10), (2,15), (6,30)}가 주어진다고 가정할 경우의 처리기 이용률을 나타내며, LR_1=0.40, LR_2=min(0.60, 0.60)=0.60, 그리고 LR_3=min(1.20, 1.00, 0.90, 0.72)=0.72가 된다. 따

표 1. ECM 처리기 이용률 예

i	τ_i	t	$WRi(t)$	$LRi(t)$	LRi	LR
1	τ_1	10	$B1+C1 = 4$	0.40	0.40	
2	$\tau_1 \sim \tau_2$	10	$B1+C1+C2 = 6$	0.60		
		15	$B1+B2+2C1+C2 = 9$	0.60	0.60	
3	$\tau_1 \sim \tau_3$	10	$B1+C1+C2+C3 = 12$	1.20		
		15	$B1+B2+2C1+C2+C3 = 15$	1.00		
		20	$B1+B2+B3+2C1+2C2+C3 = 18$	0.90		
		30	$B1+B2+B3+B4+3C1+2C2+C3 = 22$	0.72	0.72	**0.72**

라서 태스크 집합 τ의 전체 처리기 이용률은 LR =max(0.40, 0.60, 0.72)=0.72가 되며 LR<1이므로 주기 태스크 집합 τ는 실행이 가능하다.

3.3 BSTM(Backup-Surplus Time Manager)

BTM(Backup Time Manager)은 결함 허용을 위한 백업으로 할당된 타임 슬롯(time slot)을 관리하며 결함 태스크의 요청에 백업 시간을 할당한다. 또한 모든 태스크의 결함 허용을 위해서 Overloading 기법[18]을 이용한다. 그리고 백업 시간은 3.2절의 수식 (2)에 의해서 설정된다. 복구 태스크의 마감시간을 보장하면서 재실행에 충분한 실행 시간을 확보하기 위해서 교체(Swapping) 방법을 이용한다. 즉, 결함이 발생하지 않는 상태에서 태크스들이 실행할 경우 이미 할당된 백업 타임은 실행되는 태스크의 시간 뒤로 이동함으로써 결함이 발생한 태스크의 재실행이 마감시간 내에서 보장되도록 한다.

STM(Surplus Time Manager)은 주기적 태스크 집합의 마감시간을 만족하고 비주기적 태스크의 실행을 보장하기 위해서 처리기 잉여시간을 관리한다. 이를 위해 본 논문에서는 주기적 태스크의 실행 시간을 고려한 [19]에서 제안한 잉여시간(slack time) 계산 알고리즘에 BTM에 의한 백업 시간을 적용하여 최대 잉여시간을 계산한다.

3.4 PTS(Periodic Task Scheduler)

주기 태스크 스케줄러인 PTS는 주기적 태스크에 대해 우선순위 기반의 스케줄링을 수행하며 실행 과정은 다음과 같다. 비주기적 태스크와 결함 태스크가 존재하지 않을 경우, RMS에 의해 주기적 태스크 집합을 스케줄링하며, 비주기적 태스크가 실행을 위해 준비큐에 도착하고 비주기적 태스크 실행을 위한 잉여시간이 존재하는 경우, 비주기적 태스크 스케줄러인 APTS를 호출하여 비주기적 태스크를 실행한다. 또한 주기적 태스크가 실행 중에 결함이 발생할 경우 이를 결함 허용 매니저인 FTM에 통보하여 결함 태스크를 재실행함으로써 복구를 수행할 수 있도록 한다.

3.5 FTM(Fault-Tolerant Manager)

FTM은 결함이 발생하였을 경우 BSTM의 구성요소 중 하나인 BTM에서 제공하는 백업 시간을 이용하여 결함이 발생한 태스크를 재실행한다.

본 논문에서의 결함은 일시적으로 발생하며 단일 태스크에 의한 결함은 다른 태스크에 영향을 미치지 않는다고 가정한다. 그러므로 이러한 결함은 태스크를 재실행함으로써 복구될 수 있다. 또한 단일 태스크의 결함에 대한 복구만을 고려함으로써 복수의 결함에 대한 복구로 인한 처리기 이용률 감소를 해소할 수 있다.

태스크 τ_r이 복구 모드에서 새로운 태스크 τ_i가 도착했을 경우 태스크의 스케줄링은 RMS 기법인 우선순위에 의한 선점 방식에 의해 이루어진다. 그러나 새로운 태스크 τ_i의 우선순위가 복구 태스크 τ_r의 우선순위보다 높고 마감시간이 큰 경우는 우선순위에 의해 스케줄링 되지 않으며 복구 태스크의 수행이 완료될 때까지 기다렸다가 실행한다. 즉, 비록 τ_i의 우선순위가 τ_r보다 높더라도 τ_r의 마감시간을 보장하기 위해서 τ_i를 블록 태스크 큐로 전환한 후 기존의 τ_r을 실행함으로써 복구 태스크의 마감시간을 보장한다.

3.6 APTS(Aperiodic Task Scheduler)

비주기 태스크 스케줄러인 APTS는 비주기적 태스크에 대한 요청이 발생할 경우 이 태스크를 실행하기 위해서 잉여시간 관리자인 STM에 필요한 처리기 시간을 요청하며, STM으로부터 할당받은 처리기 잉여시간에 따라 비주기적 태스크의 생성 및 스케줄 여부를 결정한다. 잉여시간을 계산하고 관리하기 위해서 본 논문에서는 주기적 태스크에 중요도를 적용한다. 중요도란 주기적 태스크의 응답시간이며 응답시간이 짧을수록 중요도가 높고 그렇지 않은 경우 중요도가 낮다고 가정한다. 또한 중요도는 태스크 집합을 스케줄링하기 전에 고정적으로 제공된다고 가정한다. 빠른 응답시간을 요구하지 않는 주기적 태스크의 경우 최악의 마감시간을 보장하면서 최대한 마감시간 가까이에서 실행하도록 하였다. 그 결과 미리 확보되는 처리기의 잉여시간을 비주기적 태스크에 할당할 수 있다. 즉, 주기적 태스크 집합 중에서 빠른 응답시간을 필요로 하는 태스크에 대한 최선 실행시간을 보장하면서 비주기적 태스크의 응답시간 또한 향상시킬 수 있다.

주기적 태스크에 중요도를 적용하지 않을 경우 비주기적 태스크의 실행 시작 시간을 최소화할 수 있으며 이는 비주기적 태스크의 응답시간을 최소화할 수 있다.

주기적 태스크 집합 τ=(C,T)={(2,10), (2,15), (6, 0)}를 고려하자. 그림 3은 주기적 태스크 집합 중에서 태스크 τ_1에 대해서 빠른 응답시간을 요구한다라고 가정할 경우의 주기적 태스크의 중요도를 고려한 태스크 할당 테이블이다. 즉, 태스크 τ_1은 가장 빠른 응답시간을 필요로 하며 그렇지 않은 주기적 태스크들은 최악 마감시간 내에 실행을 완료하게 된다. 그러므로 비주기적 태스크를 위한 잉여시간은 시간 4~10과 시간 16~17이다.

4. 구현 및 결과 분석

본 장에서는 FTRTS를 구현하고 테스트하기 위해서 uC/OS-II ver. 2.76 운영체제의 내부 태스크 스케줄러를 수정 및 추가하였으며 주기적 태스크의 마감시간내 실행 완료, 비주기적 태스크의 실행 및 완료, 주기적 태스크의 결함 발생시 마감시간내 실행 완료 보장과 중요도에 따른 응답시간을 비교하기 위해서 시뮬레이션을 수행한다.

시뮬레이션에 적용된 주기적 태스크 집합은 표 2와 같으며 타임 틱(tick) 발생 시간을 10ms단위로 적용하여 테스트하였다.

그림 4는 표 2에서 제시한 주기적 태스크 집합을 이용하여 ECM를 실행한 결과로써 태스크 집합의 처리기 이용률은 0.9125이며 이는 이 태스크 집합이 실행 가능함을 나타내고 있다.

BTM은 백업 시간 할당 테이블을 생성하기 위해서 먼저 주기적 태스크 집합에 포함된 모든 태스크들의 주기에 대한 최소 공배수(Hyper Period)를 계산한다. 이는 주기적 태스크 집합에 포함된 모든 태스크들의 최소 공통 주기가 된다. 표 2에서 제시한 주기적 태스크 집합을 이용한 하이퍼 주기는 1200이 된

표 2. 시뮬레이션 태스크 집합 (단위 : ms)

Task Set	Exec. Time	Period
Priority 4	5	100
Priority 5	7	100
Priority 6	10	200
Priority 7	25	200
Priority 8	30	200
Priority 9	20	300
Priority 10	25	300
Priority 11	30	400
Priority 12	40	400

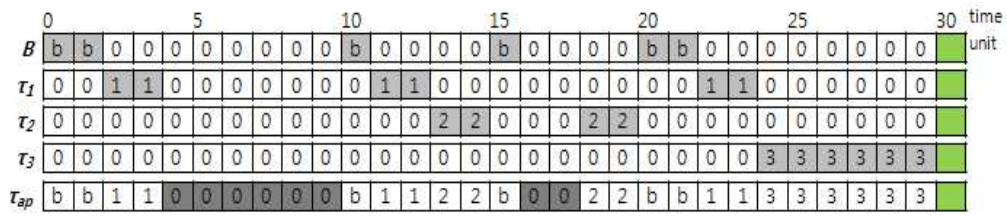


그림 3. 태스크 할당 테이블(중요도 : 주기적 태스크 τ_1)

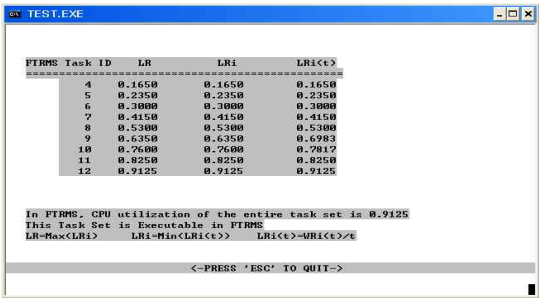
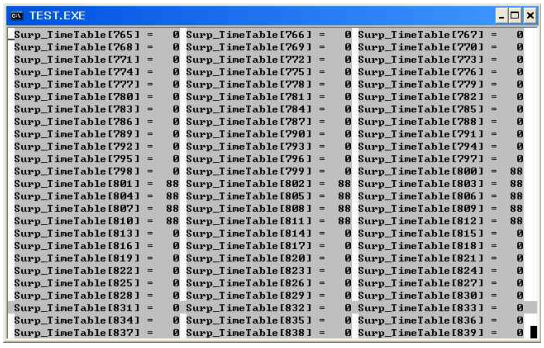

TEST.EXE
PIRMS Task ID LR LRi LRi(t)
4 0.1650 0.1650 0.1650
5 0.2350 0.2350 0.2350
6 0.3000 0.3000 0.3000
7 0.4150 0.4150 0.4150
8 0.5300 0.5300 0.5300
9 0.6350 0.6350 0.6983
10 0.7600 0.7600 0.7817
11 0.8250 0.8250 0.8250
12 0.9125 0.9125 0.9125
In PIRMS, CPU utilization of the entire task set is 0.9125
This Task Set is Executable in PIRMS
LR=Max(LRi) LRi=Min(LRi(t)) LRi(t)=WRi(t)/t
<-PRESS 'ESC' TO QUIT->


그림 4. 실행 가능성 판단 결과(실행 가능)

다. 하이퍼 주기가 생성되면, 이 주기값에 대하여 각 주기적 태스크들의 주기를 반영한 전체 태스크 집합의 부분 주기값을 계산한다. 마지막으로 수식 (2)에서 계산된 태스크 중에서 최대 처리기 이용률을 각 부분 주기 범위에 할당함으로써 백업 시간 할당 테이블을 생성한다.

그림 5는 하이퍼 주기에 대한 백업 시간 할당 테이블을 시뮬레이션 한 결과의 일부를 나타낸다. 그림에서 나타나는 값 중에서 "0"으로 표기된 곳은 주기적 태스크와 비주기적 태스크의 처리에 할당될 타임 슬롯이며, "88"로 표기된 곳은 결함이 발생한 태스크를 재실행하기 위해 확보되는 백업 시간을 나타내는 타임 슬롯이다.

비주기적 태스크의 실행과 완료를 보장하기 위해서 처리기 잉여시간을 계산하며 이를 위해 주기적 태스크의 중요도에 대해서 3.6절에서 기술하였다. RMS에서 제안한 우선순위 기법을 기본 알고리즘으로 적용하며 또한 주기적 태스크 집합 중에서 빠른 응답시간을 필요로 하는 태스크에 대해 비주기적 태스크보다 먼저 실행을 보장함으로써 빠른 응답 시간을 제공한다.

그림 5. 백업 시간 할당 테이블

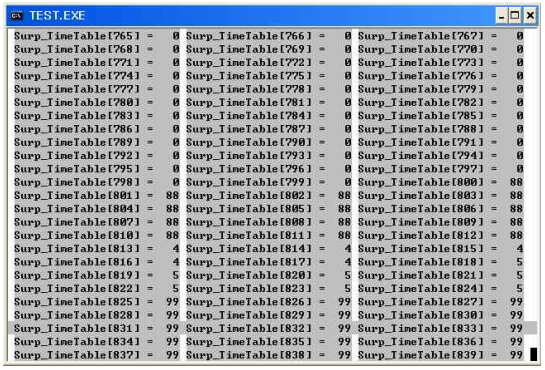


그림 6. 태스크 할당 테이블(중요도 2)

그림 6은 주기적 태스크의 중요도를 2로 설정하였을 경우의 결과이다. 타임 슬롯 813에서 824까지 나타난 값은 우선순위 1과 2에 해당하는 주기적 태스크의 실행 시간을 의미하며, "99"로 나타난 값인 타임 슬롯 825에서 839까지는 비주기적 태스크의 실행을 위해 확보되는 타임 슬롯이다. 타임 슬롯 765에서 799까지는 비주기적 태스크를 먼저 실행한 후, 우선순위 3이상의 태스크들이 실행하게 될 타임 슬롯이 된다.

그림 7은 표 2에서 제시된 9개의 주기적 태스크들을 수행시킨 결과이다. 태스크 집합이 수행되고 있을 때, 각 태스크의 주기, 남은 주기, 실행시간, 실행완료를 위해 남은 실행시간을 확인 할 수 있으며, 주기적 태스크들이 마감시간내에 완료됨을 알 수 있다.

그림 8은 결함 복구와 비주기적 태스크 실행을 동시에 처리하는 결과 화면이다. 결함 허용의 경우 결함이 발생한 태스크의 우선순위는 6이며 결함이 발생한 시간은 시간 5동안 지연을 시킨 후에 발생하도록 하였으므로 217이 된다. 시간 217에서 결함 태스

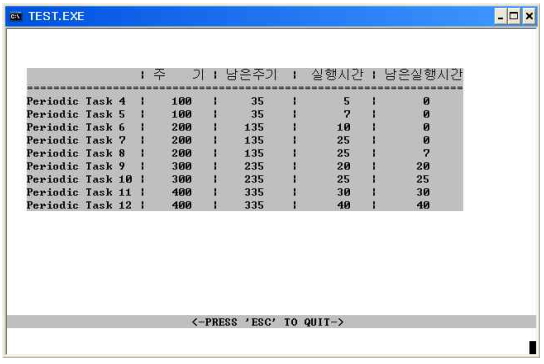

TEST.EXE
주 기 남은주기 실행시간 남은실행시간
Periodic Task 4 100 35 5 0
Periodic Task 5 100 35 7 0
Periodic Task 6 200 135 10 0
Periodic Task 7 200 135 25 0
Periodic Task 8 200 135 25 7
Periodic Task 9 300 235 20 20
Periodic Task 10 300 235 25 25
Periodic Task 11 400 335 30 30
Periodic Task 12 400 335 40 40
<-PRESS 'ESC' TO QUIT->

그림 7. 주기적 태스크 실행 결과

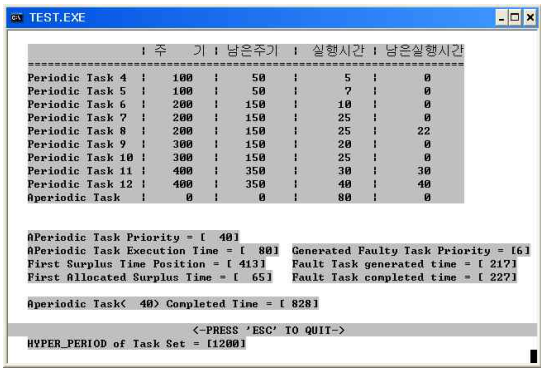


그림 8. 결함 복구 및 비주기적 태스크 실행

크의 복구 작업이 실행되며 우선순위 6인 태스크의 실행 시간이 10이므로 복구가 완료되는 시간은 227이 된다. 비주기적 태스크의 실행 시간은 80이며 최초로 할당 받은 잉여시간은 시간 380에서 20이 된다. 그리고 비주기적 태스크의 실행이 완료된 시간은 795가 된다.

그림 9는 주기적 태스크 집합의 중요도에 따른 응답 시간을 나타낸다. 주기적 태스크 집합의 중요도에 따른 응답 시간의 변화율을 분석하기 위해서 최대 잉여시간을 비주기적 태스크에 할당할 필요가 있으므로 비주기적 태스크의 실행 시간을 400으로 설정하였으며 중요도를 1~9로 증가시켜 가면서 응답시간을 측정하였다. 그 결과 주기적 태스크의 중요도가 증가할수록 주기적 태스크의 응답시간이 감소함을 보였다.

그림 10은 주기적 태스크 집합의 중요도에 따른 비주기적 태스크의 응답시간을 시뮬레이션한 결과이다. 비주기적 태스크의 실행 시간은 80으로 설정하였다. 주기적 태스크 집합의 중요도가 증가할수록 주

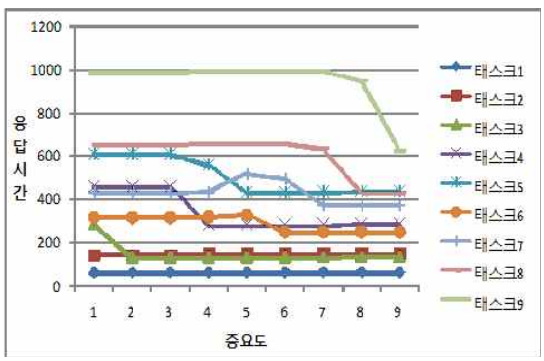


그림 9. 태스크의 중요도에 따른 응답 시간

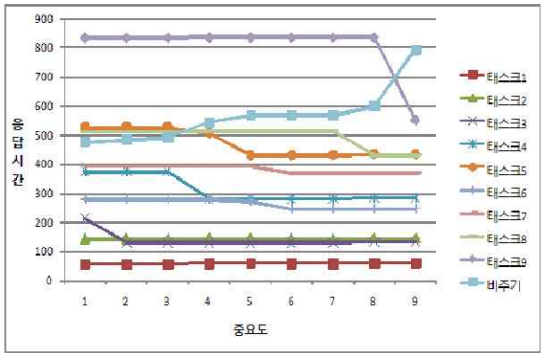


그림 10. 중요도에 따른 주기적/비주기적 태스크의 응답시간

기적 태스크의 응답시간이 감소하며 이로 인해 비주기적 태스크의 응답시간이 증가함을 보였다. 이는 태스크를 스케줄링할 경우, 주기적 태스크의 중요도를 고려함으로써 비주기적 태스크의 응답시간을 조절할 수 있음을 나타낸다. 즉, 주기적 태스크의 중요도를 증가시킴으로써 주기적 태스크의 빠른 응답시간을 제공할 수 있으며 주기적 태스크의 중요도를 감소시킴으로써 비주기적 태스크의 응답시간을 감소시킬 수 있다. 이는 주기적 태스크에 중요도를 적용함으로써 주기적 태스크와 비주기적 태스크의 응답시간을 조절할 수 있음을 확인하였다. 또한 주기적 태스크의 중요도를 적용하지 않을 경우 비주기적 태스크의 실행 시작 시간을 최소화할 수 있으며 이는 비주기적 태스크의 응답 시간 또한 최소화할 수 있다.

5. 결 론

본 논문에서는 임베디드 시스템에서 실시간성과 결함 허용을 제공하는 태스크 스케줄러를 설계하고 구현하였다. 실시간성을 제공하기 위해서 스케줄러에 주기적 태스크들을 실행하기 위한 RMS 알고리즘을 적용하였으며 비주기적 태스크들을 실행하기 위한 잉여시간을 제공하는 방법을 제시하였다. 또한 일시적인 결함이 발생한 태스크의 재실행을 통해 결함을 허용하는 결함 복구 기법도 제시하였다.

제안한 스케줄러는 주기적 태스크 집합을 스케줄링하기 전에 주기적 태스크들의 실행 가능성을 검사하는 ECM, 실행 중인 태스크에서 결함이 발생할 경우 태스크를 재실행하여 결함을 복구하기 위한 백업 시간을 계산하고 관리하는 BTM(Backup Time Manager), 비주기적 태스크가 사용할 처리기 여유

시간을 관리하는 잉여시간 관리자(STM : Surplus Time Manager), 주기적 태스크들의 스케줄링을 담당과 비주기적 태스크의 요청이 발생할 경우 비주기적 태스크 스케줄러를 호출하여 비주기적 태스크를 실행하도록 하며 태스크의 결함이 발생할 경우 결함허용 관리자의 결함 복구 작업을 지원하는 PTS (Periodic Task Scheduler), 태스크에서 일시적인 결함이 발생할 경우 백업시간 관리자에게 백업시간을 요청하고 할당받은 백업 시간을 이용하여 결함이 발생한 태스크를 복구하는 FTM(Fault Tolerant Manager), 비주기적 태스크의 요청이 발생할 경우 잉여시간 관리자로부터 비주기적 태스크가 사용할 수 있는 처리기의 잉여시간을 할당받은 후 비주기적 태스크를 실행하는 APTS(APeriodic Task Manager)로 구성되어 있다.

구현된 스케줄러를 시뮬레이션한 결과 주기적 태스크들의 실행 마감시간 보장과 비주기적 태스크의 실행 및 완료를 보장하였다. 또한 주기적 태스크에 중요도를 적용하지 않았을 경우 비주기적 태스크의 빠른 응답시간을 제공할 수 있으며 중요도를 적용하여 빠른 응답시간이 필요한 주기적 태스크의 응답시간을 줄일 수 있다. 즉, 주기적 태스크에 중요도를 적용함으로써 주기적 태스크와 비주기적 태스크의 응답시간을 조절할 수 있음을 확인하였다. 그러므로 실시간 임베디드 시스템에서 태스크집합의 스케줄링 정책을 수립할 때 예측하여야 할 태스크들의 시간 응답성을 주기적 태스크의 중요도를 이용하여 결정할 수 있다. 또한, 제안한 결함 복구 기법은 태스크의 일시적인 결함 발생에도 정상적인 태스크들의 마감시간을 보장하며 또한 결함이 발생한 태스크의 마감시간도 만족하는 것을 확인할 수 있었다.

참 고 문 헌

[1] J.T. Baldwin, *Predicting and Estimating Real-Time Performance*. Embedded Systems Programming, 8(2), 1995.

[2] L. Doyle and J. Elzey, "Successful Use of Rate Monotonic Theory on a Formidable Real Time System," In 11th IEEE Workshop on Real-Time Operating Systems and Software, pages 74-78. IEEE, 1994.

[3] H Kopetz, "Automotive Electronics-Present State and Future Prospects," In FTCS 25, 1995.

[4] K. W. Tindell, "Fixed Priority Scheduling of Hard Real-Time Systems," PhD thesis, Univ of York, UK, 1994.

[5] C.L Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *JACM*, 20(1), pp. 46-61, 1973.

[6] John Lehoczky, Lui Sha, and Ye Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization And Average Case Behavior," *RTSS* pp. 166-171, 1989.

[7] M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," *The BCS Computer Journal*, Vol.29, No.5, pp. 390-395, 1986.

[8] N. C. Audsley, A. Burns, M. Rihardson, and A. Wellings, "Hard Real-Time Scheduling: The Deadline-Monotonic Approach," In Proc. of the 8th IEEE Workshop on Real- Time Operating Systems and Software, pp. 133-137, 1991.

[9] K. W. Tindell, A. Burns, and A. J. Wellings, "An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks," *Real-Time Systems,* Vol.6, No.2, pp. 133-151, 1994.

[10] J.P. Lehoczky and S. Ramos-Thuel, "An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed-Priority Preemptive Systems," Proc. 13th Real-Time Systems Symp., pp. 110-123, 1992.

[11] 정경훈, 김병훈, 이동건, 김창수, 탁성우, "확장성 및 실시간성을 고려한 실시간 센서 노드 플랫폼의 설계 및 구현," 한국통신학회논문지 제32권, 제8호, pp. 509-520, 2007.

[12] 김병훈, 정경훈, 탁성우, "주기 및 비주기 태스크의 효율적인 관리를 위한 실시간 센서 노드 플랫폼의 설계," 정보처리학회지 제14-C권 제4호 통권, 제114호, pp. 371-382, 2007.

[13] 김희헌, 박학봉, 박문주, 박민규, 조유근, 조성

재, “잉여 여유시간을 이용한 연성 비주기 태스크들의 효율적인 스케줄링,” 정보과학회논문지, 시스템 및 이론 제36권, 제1호, 2009.

[14] 정경훈, 탁성우, 김창수, “결함허용이 가능한 임베디드 실시간 태스크 관리 메커니즘,” 한국멀티미디어학회논문지, v.10, no.7, pp.882-892, 2007년 7월.

[15] S. Ghosh, R. Melhem, D. Moss, and J. Sensarma., “Fault-tolerant rate-monotonic scheduling,” *Real-Time Systems*, Vol.15, No. 2, pp. 149-181, 1998.

[16] Yu Chen and Guangze Xiong, “Imprecise Computation Fault-Tolerant Rate-Monotonic Scheduling,” ICA3PP'02, Algorithms and Architectures for Parallel Processing, 2002.

[17] Y. S. Hong and H. W. Goo, “A Fault-Tolerant Technique for Scheduling Periodic Tasks in Real-Time Systems,” Proc. of the Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, 2004.

[18] S. Ghosh, D. Mosse, and R. Melhem, “Implementation and Analysis of a Fault-Tolerant Scheduling Algorithm,” *IEEE Transactions on Parallel and Distributed Systems*, 1997.

[19] J.P. Lehoczky and S. Ramos-Thuel, “An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed-Priority Preemptive Systems,” Proc. 13th Real-Time Systems Symp., pp. 110-123, 1992.

전 태 건

1997년 부경대학교 전자계산학과 (학사)
1999년 부경대학교 전자계산학과 (석사)
2011년 부경대학교 정보공학과 (박사)
2011년~현재 (주)인사이트정보
관심분야 : 임베디드 운영체제, 센서네트워크, 실시간 스케줄링, 분산병렬처리, GIS/UIS

김 창 수

1991년 중앙대학교 컴퓨터공학과 박사
2006년~현재 유비쿼터스 부산 도시협회 방재분과위원장
2006년~현재 (사)그레고리장학회 이사
2009년~현재 한국멀티미디어학회 정책자문위원
1992년~현재 부경대학교 IT융합응용공학과 교수
관심분야 : 방재 IT, UIS/GIS, 운영체제, 시멘틱 웹, 재난관리, 공간검색, 도시방재 등