JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 18, No. 7 : 1704~1712 July. 2014

실행 유휴 시간 분배 정책에 따른 실시간 전력 관리 스케줄링 기법의 성능 평가

탁성우*

Performance Evaluation of Real-Time Power-Aware Scheduling Techniques Incorporating Idle Time Distribution Policies

Sungwoo Tak*

Department of Computer Science and Engineering, Pusan National University, Pusan 609-735, Korea

요 약

실시간 태스크의 스케줄링 가능성 검사를 위해 미리 설정된 태스크의 최악 실행 시간보다 태스크의 실제 실행 시간이 짧은 경우, 최악 실행 시간에서 남은 실행 유휴 시간이 발생한다. 발생된 실행 유휴 시간은 실시간 전력 관리 스케줄링 기법을 통해 배터리 기반 센서 노드의 전력 소비 감소에 활용될 수 있다. 이에 본 논문에서는 발생된 남은 최악 실행 유휴 시간을 분배하여 실시간 전력 관리 스케줄링 기법에서 활용할 수 있도록 세 가지 분배 정책을 제안하였다. 제안한 분배 정책은 보수적, 중도적, 그리고 공격적 실행 유휴 시간 분배 정책으로 각각 구분하였다. 그리고 분배 정책 유형에 따른 실시간 전력 관리 스케줄링 기법의 성능 평가는 전력 소비 측면에서 비교 분석하였다.

ABSTRACT

The unused Worst-Case Execution Time (WCET) allocated to a real-time task occurs when the actual execution time of the task can be far less than the WCET preassigned to the task for a schedulability test. Any unused WCET allocated to the task can be exploited to reduce the power consumption of battery-powered sensor nodes through real-time power-aware scheduling techniques. From the distribution perspective of the unused WCET, the unused WCET distribution policy is classified into three types: Conservative Unused WCET (CU-WCET), Moderate Unused WCET (MU-WCET), and Aggressive Unused WCET (AU-WCET) distribution policies. We evaluated the performance of real-time power-aware scheduling techniques incorporating each of three unused WCET distribution policies in terms of low power consumption.

키워드 : 실시간 태스크, 최악 실행 시간, 실제 실행 시간, 실시간 전력 관리 스케줄링

Key word : Real-time task, Worst-case execution time, Actual execution time, Real-time power-aware scheduling

접수일자 : 2014. 03. 31 심사완료일자 : 2014. 04. 21 게재확정일자 : 2014. 05. 12

* **Corresponding Author** Sungwoo Tak (E-mail:swtak@pusan.ac.kr, Tel:+82-51-510-2387)
Department of Computer Science and Engineering, Pusan National University, Pusan 609-735, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2014.18.7.1704 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

본 논문에서는 제한된 에너지 자원을 사용하는 센서 노드에서 실시간 태스크 스케줄링과 전력 관리 기능을 동시에 고려하는 실시간 전력 관리 스케줄링 기법을 제안하고 비교 분석하였다.

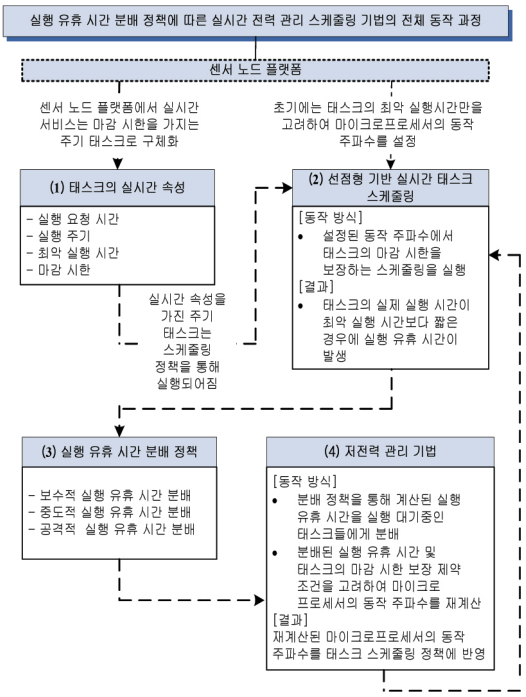

그림 1. 실시간 전력 관리 스케줄링 기법의 동작 과정
Fig. 1 Operation procedures for the proposed real-time power-aware scheduling technique

그림 1은 실시간 전력 관리 스케줄링 기법의 전체 동작 과정을 보여준다. 실시간 전력 관리 스케줄링 기법의 주요 4 개 구성 인자는 (1) 태스크의 실시간 속성, (2) 선점형 기반 실시간 태스크 스케줄링, (3) 실행 유휴 시간 분배 정책, 그리고 (4) 저전력 관리 기법이다. 센서 노드 플랫폼에서 실행되는 실시간 서비스는 마감시한을 가지는 실시간 태스크로 구체화되며, 태스크는 실행 요청 시각 (Release Time), 실행 주기 (Period), 최악 실행 시간 (WCET: Worst-Case Execution Time), 그리고 마감 시한 (Deadline) 속성을 가진다. 실시간 태스크는 선점형 기반 실시간 태스크 스케줄링 정책을 통해 실행 순서가 결정된다. 태스크의 실행 순위 제어와 이에 따른 프로세서의 선점 권한을 제어하여 효율적인 전력 소비 및 실시간 서비스를 제공할 수 있다. 따라서 비선점형 대신 선점형 기반 태스크 스케줄링 정책을 사용한다. 대표적인 센서 노드 플랫폼으로 사용되고 있는 TinyOS는 비선점형 스케줄링 정책을 사용하기 때문에 실시간 서비스를 요구하는 태스크의 마감 시한을 보장할 수 없다[1]. 태스크 스케줄링 및 실행을 담당하는 마이크로프로세서의 초기 동작 주파수 설정에서는 태스크의 최악 실행 시간만을 고려한다. 그러나 각 태스크의 실제 실행 시간 (ACET: Actual Execution Time)은 최악 실행 시간보다 작은 경우가 빈번하게 발생할 수 있다[2]. 본 논문에서는 최악 실행 시간과 실제 실행 시간간의 차이를 실행 유휴 시간이라는 용어로 기술하였다. 이러한 실행 유휴 시간은 배터리 기반 센서 노드의 전력 관리에 활용될 수 있는 중요한 성능 인자로 고려될 수 있다. 이에 그림 1의 단계 (3)에서 기술한 바와 같이, 보수적 및 중도적, 그리고 공격적 실행 유휴 시간 분배 정책으로 각각 구분한 후, 실시간 태스크 스케줄링 및 저전력 관리 기법에서 활용한다.

그림 1의 단계 (4)에서는 저전력 관리 기법을 기술하였다. 실행 유휴 시간을 활용할 수 있는 저전력 기법으로는 DPM (Dynamic Power Management) 및 DVS (Dynamic Voltage Scaling) 기법이 있다. DPM기법은 실행 유휴 시간 동안 마이크로프로세서를 유휴 상태 (Idle State)로 전환하여 센서 노드의 전력 소비를 줄인다. DPM 기법을 사용하고 있는 센서 노드 플랫폼으로는 TinyOS가 있다. DVS 기법은 마이크로프로세서에 입력되는 태스크 부하에 따라 마이크로프로세서에 공급되는 동작 주파수를 동적으로 조절하여 프로세서의 구동 전압을 낮춘다. 이를 통해 센서 노드 플랫폼의 전력 소비를 감소시켜 DPM기법보다 더 효율적인 전력 소비를 제공할 수 있다 [3-6]. 본 논문에서는 DVS 기법을 사용하며, 태스크의 마감시한을 만족하는 범위 내에서 동작 주파수를 조절하고자 한다.

$$V \propto f,\ 0 < f \leq 1 \tag{1}$$

$$WCET \propto \frac{1}{f},\ ACET \propto \frac{1}{f} \tag{2}$$

$$P(ACET) \propto V^2 \times C \times f \times ACET \tag{3}$$

CMOS 회로 기반 마이크로프로세서의 구동 전압 V는 주파수 조정 계수 f에 비례한다 (식 (1)). f가 1이면, 마이크로프로세서는 최대 동작 주파수로 동작한다. 주파수 조정 계수 f가 $1/N$으로 감소하면, 마이크로프로세서가 처리해야 할 태스크의 최악 실행시간 $WCET$ 및 실제 실행시간 $ACET$는 N배로 증가하게 되어 태스크의 응답 시간이 느려진다 (식 (2)). 식 (3)은 태스크의 실제 실행시간 $ACET$동안 소비되는 전력량 $P(ACET)$를 나타낸다 [6]. C는 커패시턴스 부하 상수를 나타낸다. 식 (3)에서 보는 바와 같이, 마이크로프로세서의 전력 소비는 구동 전압의 제곱 및 동작 주파수에 비례한다 [4-5]. 예를 들어, 주파수 조정 계수 f가 1이고 실제 실행시간 $ACET$가 1일 때, $P(ACET)$는 V^2C가 된다.

그리고 f를 $1/N$으로 낮추면, 식 (2)와 식(3)에 의해 $P(ACET)$는 $\left(\frac{V}{N}\right)^2 \times C \times \frac{1}{N} \times N$가 된다. 이는 $\left(\frac{V}{N}\right)^2 \times C$로 정리된다. 즉, 동작 주파수를 $1/N$로 낮추면, 구동 전압도 $1/N$으로 감소되어 센서 노드의 전력 소비는 $(1/N)^2$만큼 감소된다. 참고 문헌 [7]에서는 시간 t동안의 에너지 소비량을 동작 주파수의 제곱으로 정의하였다. 이는 식 (1)부터 식 (3)까지 기술한 내용이다. 주파수 조정 계수가 1일 때, 태스크의 실제 실행 시간이 1초이고 1초 동안 소모되는 전력이 1인 경우에, 주파수 조정 계수 f를 1/8 (= 0.125)로 조절하면, 식 (3)에서 기술한 바와 같이 소모되는 전력은 주파수 조정 계수의 제곱인 0.015625 (=0.125×0.125)만큼 감소된다. 그리고 식 (2)에서 기술한 바와 같이, 실행 완료 시간은 8 (= 1/0.125)로 증가한다.

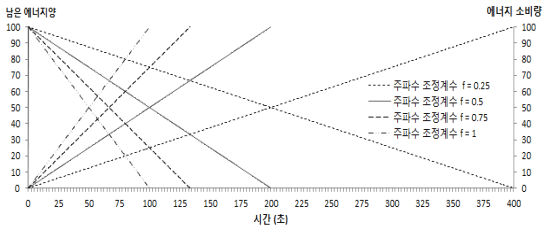

그림 2. 배터리 용량 변화
Fig. 2 Battery capacity traces

그림 1에서는 2개의 시뮬레이션 결과를 보여준다. 왼쪽 세로축에서는 남은 에너지양이 100인 상태에서 주파수 조정 계수 별 시간에 따른 에너지양의 감소를 보여준다. 오른쪽 세로축에서는 시간에 따른 누적 에너지 소비량을 보여준다. 그리고 주파수 조정 계수 f가 1일 때 1초 동안 소비되는 에너지는 1이며, 태스크의 실제 실행 시간은 100초로 설정하였다. 주파수 조정 계수가 0.25인 경우, 0.25초마다 0.0625 (= 0.25 × 0.25)만큼 에너지가 감소하여 1초당 0.25 (= 0.0625/0.25초 × 1초)만큼 에너지가 소비된다. 그리고 실행 시간은 4배로 증가하여 400초가 된다. 따라서 400초가 되면, 남은 에너지양은 0 (= 100 - (0.25/초 × 400초))이 된다. 즉, 주파수 조정 계수가 감소되면, 배터리 수명이 연장됨을 알 수 있다.

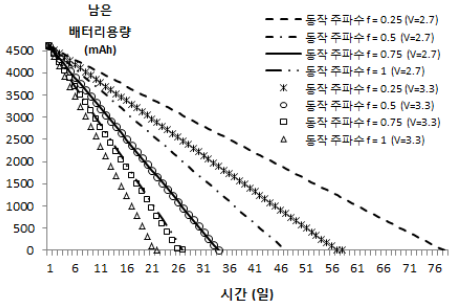

그림 3. 전류량에 따른 배터리 용량 변화
Fig. 3 Battery capacity traces over varying currents

표 1. ATMega128L의 동작 전류량
Table. 1 Active current consumption of ATMega128L

동작 주파수 조정 계수 (동작 주파수)	구동 전압: 2.7V	구동 전압: 3.3V
0.25 (2MHz)	2.5mA	3.4mA
0.5 (4MHz)	4.17mA	5.7mA
0.75 (6MHz)	5.83mA	7.5mA
1 (8MHz)	7.5mA	9.375mA

그림 3에서는 Atmel사의 ATMega128L 마이크로프로세서의 동작 전류량에 따른 배터리 수명을 시뮬레이션 하였다. 표 1에서 보는 바와 같이 측정된 동작 전류량은 동작 주파수 및 배터리 구동 전압에 따라 변한다. 측정 값은 ATMega128L 데이터시트 값과 유사하다 [8]. 그림 3의 시뮬레이션에서는 2300mAh 용량을 가지는 2개의 AA 건전지를 사용한다고 가정하였다. 시뮬레이션에 사용된 Atmel사의 ATMega128L 마이크로프로세서는 미국 버클리대학에서 제작한 Mica 센서

보드를 포함하여 많은 센서 보드에서 사용하고 있다 [9]. ATMega128L의 구동 전압으로는 2.7V와 3.3V를 고려하였다. 그림 3에서 구동 전압이 2.7V이고 주파수 조정 계수가 1인 경우, 25.09일 (= (2300mA × 2) × 3600초 / (7.5mA × 1초)) 만큼의 배터리 수명을 가진다. 그림 2의 왼쪽 세로축에서 보여준 배터리 소모량은 그림 3에서 보여주는 주파수 조정 계수의 제곱 형태와 유사하게 감소하는 것을 확인하였다.

Ⅱ. 실시간 전력 관리 스케줄링 기법

본 논문에서 고려하고 있는 주요한 2가지 사항은 다음과 같다. 첫째, 실시간 태스크는 주어진 마감시한 내에 실행을 완료해야 하기에 빠른 응답시간을 요구하지는 않는다. 따라서 태스크의 마감시한을 만족하는 범위 내에서 마이크로프로세서의 동작 주파수를 감소시켜 실행 완료 시간이 지연되더라도 전력 소비를 감소시킬 수 있다. 둘째, 센서 노드 플랫폼에서 실행되는 개별 태스크의 실제 실행시간이 최악 실행시간보다 작은 경우가 대부분이기 때문에 태스크 부하 변동에 따라 발생되는 실행 유휴 시간을 이용한다. 태스크의 이른 실행 완료에 의해 발생된 실행 유휴 시간은 우선순위가 높은 태스크에게 할당된다. 실행 유휴 시간을 할당받은 태스크는 자신의 마감 시한 내에서 마이크로프로세서의 동작 주파수를 최대한 낮추어 센서 노드의 전력 소비를 감소시킨다. 태스크 $Task_i$의 실시간 속성을 기술하는 변수들은 다음과 같다. i는 태스크 식별자이다. $Task(R)_i$는 실행 요청시각, $Task(P)_i$는 실행 주기, $Task(D)_i$는 상대적 마감시한, $Task(WCET)_i$는 최악 실행시간, 그리고 $Task(ACET)_i$는 실제 실행시간을 나타낸다.

표 2. 태스크 집합
Table. 2 A set of tasks

$Task_i$	$Task(R)_i$	$Task(P)_i$	$Task(D)_i$	$Task(WCET)_i$	$Task(ACET)_i$	
$Task_1$	0	8	8	3	2	1번째 실행의 실시간 속성
$Task_2$	0	10	10	3	1	
$Task_3$	0	14	14	1	1	
$Task_1$	8	8	8	3	1	2번째 실행의 실시간 속성
$Task_2$	10	10	10	3	1	
$Task_3$	14	14	14	1	1	

실행 유휴 시간 분배 정책에 사용되는 태스크 집합은 표 2와 같다. 시간의 단위는 초로 설정하였다. 그리고 표 2에 설정된 시간 정보들은 최대 동작 주파수 조정 계수가 1일 때이다.

2.1. 보수적 실행 유휴 시간 분배 정책

보수적 실행 유휴 시간 분배 정책에 대한 4개의 동작 규칙 (CU-WCET, Conservative Unused WCET Rule)은 다음과 같다.

- CU-WCET 규칙 1: 실행 초기 혹은 실행 대기 중인 태스크가 없는 경우에는 태스크의 최악 실행 시간만을 고려하여 동작 주파수 조정 계수를 설정한다.
- CU-WCET 규칙 2: 태스크의 이른 실행 완료에 의해 발생된 실행 유휴 시간은 다른 모든 태스크에게 균등하게 배분한다.
- CU-WCET 규칙 3: CU-WCET 규칙 2에서 균등하게 배분한 실행 유휴 시간과 태스크의 최악 실행 시간만을 고려하여 동작 주파수 조정 계수를 설정한다.

이러한 규칙 외에 모든 실행 유휴 시간 분배 정책에 공통적으로 EDF (Earliest Deadline First)기반 스케줄링 규칙이 적용된다. EDF 기반 스케줄링 규칙에서는 태스크의 실행 완료에 요구되는 마이크로프로세서 이용률이 1이하이고, 태스크의 주기는 마감시한 보다 크거나 같고, 실시간 속성은 마감시한만을 고려한다. 마이크로프로세의 이용률은 식 (4)와 같이 계산된다.

$$\sum_{i=1} \frac{Task(WCET)_i}{Task(D)_i} \tag{4}$$

이러한 보수적 실행 유휴 시간 분배 정책은 참고 문헌 [4]에서 기술한 정적 전압 조절 (Static Voltage Scaling) 기법과 유사하다. 표 3은 보수적 실행 유휴 시간 분배 정책의 동작 과정을 보여준다. 표 3에서 주파수 조절 계수 값은 소수점 3자리까지만 기술하였고, 시각은 소수점 2자리까지만 기술하였다. 동작 과정의 분석에 사용되는 태스크 집합 및 소수점 자리 기술 방식은 중도적 및 공격적 실행 유휴 분배 정책의 동작 과정에서도 동일하게 적용하였다. 그림 4는 동작 과정의 최종 결과를 보여준다. 시각 0에서 모든 태스크의 최악 실행 시간만을 고려하여 동작 주파수 조정 계수를 0.746으

로 설정한다 (CU-WCET 규칙 1). 마감 시한이 가장 짧은 $Task_1$이 먼저 실행된다 (EDF기반 스케줄링 규칙). 시각 2.67에서 $Task_1$은 최악 실행 시간보다 일찍 완료되며, 발생된 실행 유휴 시간을 $Task_2$와 $Task_3$에게 균등하게 배분한다.

표 3. 보수적 실행 유휴 시간 분배 정책의 동작 과정
Table. 3 Traces of CU-WCET distribution policy

시각	실행 태스크 (실행 대기 중인 태스크)	주파수 조정 계수 (f)	실행 시간 (초)
0	$Task_1$ ($Task_2$, $Task_3$)	0.746 (=$Task(WCET)_1 / Task(D)_1$+ $Task(WCET)_2 / Task(D)_2$ +$Task(WCET)_3 / Task(D)_3$ = 3/8+3/10+1/14)	2.67 (= $Task(ACET)_1 / f$ = 2 / 0.746)
2.67	$Task_2$ ($Task_3$)	0.621 (= $Task(WCET)_1 / Task(D)_1$+ {$Task(WCET)_2 / Task(D)_2$ - ($Task(WCET)_1 / Task(D)_1$ - $Task(ACET)_1 / Task(D)_1$) × 1/2} + {$Task(WCET)_3 / Task(D)_3$ - ($Task(WCET)_1 / Task(D)_1$ - $Task(ACET)_1 / Task(D)_1$) × 1/2}) = 3/8 + {3/10 - (3/8 – 2/8) × 1/2} + {1/14 - (3/8 – 2/8) × 1/2})	1.61 (= $Task(ACET)_2 / f$ = 1 / 0.621)
4.28	$Task_3$ (*None*)	0.546 (= {3/8 - (3/10 – 1/10) × 1/2} + 3/10 + {1/14 - (3/10 – 1/10) × 1/2})	1.83 (= $Task(ACET)_3 / f$ = 1 / 0.546)
6.11	실행 유휴 상태		
8	$Task_1$ (*None*)	0.746 (= 3/8+3/10+1/4)	1.33 (= $Task(ACET)_1 / f$ = 1 / 0.746)
9.33	실행 유휴 상태		
10	$Task_2$ (*None*)	0.746 (= 3/8 + 3/10 + 1/14)	1.33 (= $Task(ACET)_2 / f$ = 1 / 0.746)
11.33	실행 유휴 상태		
14	$Task_3$ (*None*)	0.746 (= 3/8 + 3/10 + 1/14)	1.33 (= $Task(ACET)_3 / f$ = 1 / 0.746)
15.33	실행 유휴 상태		

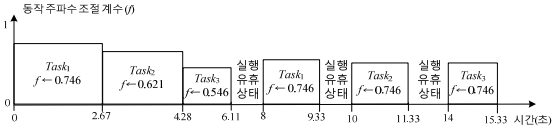

그림 4. 보수적 실행 유휴 시간 분배 정책의 동작 결과
Fig. 4 Outcomes of CU-WCET distribution policy

배분하는 방법은 시간 영역을 주파수 영역으로 변환시켜, 실행 유휴 시간 값을 동작 주파수 조절 계수 값으로 변환하여 배분한다. 먼저 시간 영역에서 실행 유휴 시간을 계산하는 과정은 다음과 같다. 예를 들어 시각 0에서 마감시한이 1이고 최악 실행 시간 양이 0.5인 태스크는 동작 주파수 조정 계수f_1을 0.5 (= 0.5/1 = 1/2)로 설정한다. 그러나 태스크가 0.125초 (= 1/8)에서 실행 완료되는 경우, 실제 실행 시간 양은 0.0625 (= 1/2 × 1/8)가 된다. 따라서 실행 유휴 시간 양은 0.4375 (= 0.5 – 0.0625 = 1/2 – 1/2 × 1/8)가 된다. 시간 영역 대신 주파수 영역에서 실행 유휴 시간을 계산하는 과정은 다음과 같다. 앞서 기술한 바와 같이 최악 실행 시간 양이 0.5인 경우, 동작 주파수 조정 계수 f_1은 0.5이다. 최악 실행 시간이 아닌 실제 실행 시간 양 0.0625를 고려한 경우, 동작 주파수 조정 계수f_2는 0.0625 (= 0.0625/1)로 설정된다. 그리고 주파수f_1과f_2간의 차이 값인 남은 유휴 주파수 양은 0.4375 (=f_1 - f_2 = 0.5 – 0.625)가 된다. 이와 같이 시간 영역을 주파수 영역으로 변환시켜, 남은 유휴 주파수 양을 다른 태스크의 동작 주파수 계수 설정에 사용되도록 하여 실행 유휴 시간을 분배할 수 있다. 이러한 분배 방법은 다음 절에서 기술하는 중도적 실행 유휴 시간 분배 정책에도 적용된다.

표 3의 시각 2.67에서 보는 바와 같이, 실행 유휴 시간을 동작 주파수 조정 계수 영역으로 변환하여 배분한다. $Task_1$의 최악 실행 시간 3을 고려한 경우와 실제 실행 시간이 2인 경우, 주파수 양 1/8 (= {3/8 + 3/10 + 1/4} - {2/8 + 3/10 + 1/4} = 3/8 – 2/8)만큼 차이가 난다. 그리고 1/8만큼의 차이를 시각 2.67에서 균등하게 1/2만큼 나누어 $Task_2$와 $Task_3$에게 배분한 후 (CU-WCET 규칙 2), 태스크의 최악 실행 시간만을 고려하여 동작 주파수 조정 계수를 새로 설정한다 (CU-WCET 규칙 3). 시각 4.28에서는 시각 2.67에 적용된 CU-WCET 규칙들을 적용하여 주파수 조정 계수를 재계산한다. 시각 8과 시각 10, 그리고 시각 14의 경우, 해당 시각에 마이크로프로세서의 사용권한을 선점한 $Task_1$과 $Task_2$, 그리고 $Task_3$ 외에는 실행 대기 중인 태스크가 없다. 따라서 CU-WCET 규칙 1이 적용되어, 주파수 조정 계수는 0.746으로 설정된다.

2.2. 중도적 실행 유휴 시간 분배 정책

중도적 실행 유휴 시간 분배 정책에 대한 3개의 동작 규칙 (MU-WCET, Moderate Unused WCET Rule)은 다음과 같다.

- MU-WCET 규칙 1: 실행 초기에는 태스크의 최악 실

행 시간만을 고려하여 동작 주파수 조정 계수를 설정한다.

- MU-WCET 규칙 2: 현재 실행 중인 태스크를 제외한 다른 모든 태스크로부터 발생한 이전 실행 유휴 시간은 다른 모든 태스크에게 계속해서 균등하게 배분한다. 이 규칙은 현재 실행 중인 태스크는 최악 실행 시간만큼 실행될 것이라는 다소 보수적인 예측을 하지만, 다른 모든 태스크의 다음 실제 실행 시간은 이전 실제 실행 시간과 동일할 것이라는 다소 공격적인 예측을 하는 것이다.

표 4는 중도적 실행 유휴 시간 분배 정책의 동작 과정을 보여준다. 그림 5는 동작 과정의 최종 결과를 보여준다.

표 4. 중도적 실행 유휴 시간 분배 정책의 동작 과정
Table. 4 Traces of MU-WCET distribution policy

시각	실행 태스크 (실행 대기 중인 태스크)	주파수 조정 계수 (f)	실행 시간 (초)
0	$Task_1$ ($Task_2$, $Task_3$)	0.746 (=$Task(WCET)_1 / Task(D)_1$+ $Task(WCET)_2 / Task(D)_2$ +$Task(WCET)_3 / Task(D)_3$ = 3/8+3/10+1/14)	2.67 (= $Task(ACET)_1 / f$ = 2 / 0.746)
2.67	$Task_2$ ($Task_3$)	0.621 (= $Task(ACET)_1 / Task(D)_1$+ {$Task(WCET)_2 / Task(D)_2$ - ($Task(WCET)_1 / Task(D)_1$ - $Task(ACET)_1 / Task(D)_1$) × 1/2} + {$Task(WCET)_3 / Task(D)_3$ - ($Task(WCET)_1 / Task(D)_1$ - $Task(ACET)_1 / Task(D)_1$) × 1/2}) = 3/8 + {3/10 - (3/8 – 2/8) × 1/2} + {1/14 - (3/8 – 2/8) × 1/2})	1.61 (= $Task(ACET)_2 / f$ = 1 / 0.621)
4.28	$Task_3$	0.421 (= {$Task(WCET)_1 / Task(D)_1$ - ($Task(WCET)_2 / Task(D)_2$ - $Task(ACET)_2 / Task(D)_2$) × 1/2} + {$Task(WCET)_2 / Task(D)_2$ - ($Task(WCET)_1 / Task(D)_1$ - $Task(ACET)_1 / Task(D)_1$) × 1/2} + {$Task(WCET)_3 / Task(D)_3$ - ($Task(WCET)_1 / Task(D)_1$ - ($Task(ACET)_1 / Task(D)_1$) × 1/2 - ($Task(ACET)_2 / Task(D)_2$) × 1/2}) = {3/8 - (3/10 – 1/10) × 1/2} + {3/10 -(3/8 – 2/8) × 1/2} + {1/14 - (3/8 – 2/8) × 1/2 - (3/10 – 1/10) × 1/2})	2.38 (= $Task(ACET)_3 / f$ = 1 / 0.421)
6.66	실행 유휴 상태		
8	$Task_1$ (*None*)	0.546 (= {3/8 - (3/10 – 1/10) × 1/2} + 3/10 + {1/14 - (3/10 – 1/10) × 1/2})	1.83 (= $Task(ACET)_1 / f$ = 1 / 0.546)
9.83	실행 유휴 상태		
10	$Task_2$ (*None*)	0.496 (= 3/8 + {3/10 – (3/8 – 1/8) × 1/2} + {1/14 - (3/8 – 1/8) × 1/2})	2.01 (= $Task(ACET)_2 / f$ = 1 / 0.496)
12.01	실행 유휴 상태		
14	$Task_3$ (*None*)	0.296 (= {3/8 - (3/10 – 2/10) × 1/2} + {3/10 –(3/8 – 1/8) × 1/2} + {1/14 - (3/8 – 1/8) × 1/2 - (3/10 – 1/10) × 1/2})	3.37 (= $Task(ACET)_3 / f$ = 1 / 0.296)
17.37	실행 유휴 상태		

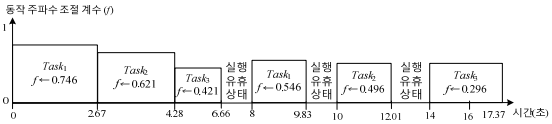

그림 5. 중도적 실행 유휴 시간 분배 정책의 동작 결과
Fig. 5 Outcomes of MU-WCET distribution policy

시각 0과 시각 2.67에서의 동작 과정은 표 3에서 기술한 동작 과정과 유사하다. 시각 0에서 모든 태스크의 최악 실행 시간만을 고려하여 동작 주파수 조정 계수를 0.746으로 설정한다 (MU-WCET 규칙 1). 시각 2.67과 시각 4.28에서는 MU-WCET 규칙 2가 적용되었다. 시각 4.28에서 동작 주파수 조정 계수를 0.421로 설정하는 과정은 다음과 같다. 시각 0에서 실행한 $Task_1$의 실제 실행 시간이 2이기에 남은 주파수 양 1/8 (= 3/8 – 2/8)을 균등하게 1/2만큼 나누어 $Task_2$와 $Task_3$에게 배분한다. 그리고 시각 2.67에서 실행한 $Task_2$의 실제 실행 시간이 2이기에 남은 주파수 양 1/10 (= 3/10 – 2/10)을 균등하게 1/2만큼 나누어 $Task_1$과 $Task_3$에게 배분한다. 이와 같이 배분한 값을 기반으로 하여 시각 4.28에서 동작 주파수의 조정 계수는 0.421 (= 3/8 – (3/10 - 1/10} × 1/2} + {3/10- (3/8 – 2/8) × 1/2} + {1/14 - (3/8 – 2/8) × 1/2- (3/10 – 1/10) × 1/2})로 설정된다. 이러한 중도적 실행 유휴 시간 분배 정책은 참고 문헌 [10]에서 기술한 적응형 전력 조절 기법과 유사하다.

2.3. 공격적 실행 유휴 시간 분배 정책

본 논문에서 제안한 공격적 실행 유휴 시간 분배 정책에 대한 4개의 동작 규칙 (AU-WCET, Aggressive Unused WCET Rule)은 다음과 같다.

- AU-WCET 규칙 1: 마이크로프로세서의 사용 권한을 선점한 최상위 우선순위 $Task_i$는 차상위 우선순위 $Task_j$의 실행 유휴 시간을 활용한다. $Task_j$의 실행 유휴 시간은 $Task(RD)_j$ - $Task(RWCET)_j$ 로 계산

된다.

- AU-WCET 규칙 2: 실행 중인 $Task_j$보다 우선 순위가 더 높은 $Task_i$가 실행 요청되면, $Task_i$의 남은 마감시한 $Task(RD)_i$와 $Task_j$의 실행 유휴 시간 (= $Task(RD)_j$ - $Task(RWCET)_j$)를 비교한다. 만약 $Task_i$의 $Task(RD)_i$가 태스크 $Task_j$의 실행 유휴 시간보다 더 큰 경우에 태스크 $Task_i$가 최악 실행시간으로 수행된다면, 태스크 $Task_j$는 마감 시한을 보장하지 못한다. 따라서 $Task(RD)_i$ 값을 $Task_j$의 실행 유휴 시간으로 변경한다.
- AU-WCET 규칙 3: 마이크로프로세서의 사용 권한을 선점한 $Task_i$의 남은 마감시한 $Task(RD)_i$와 남은 최악 실행 시간 $Task(RWCET)_i$을 이용하여 동작 주파수 조정 계수 f는 $Task(RWCET)_i$ / $Task(RD)_i$ 로 설정된다.

AU-WCET 규칙 1부터 3에서 기술한 바와 같이, 최상위 우선순위 태스크는 차상위 우선순위 태스크의 실행 유휴 시간을 고려하여 남은 마감 시한을 계산한다. 그리고 최상위 우선순위 태스크는 자신의 남은 마감시한 대비 남은 최악 실행 시간 비율만큼 동작 주파수를 공격적으로 최대한 낮게 설정하여 전력 소비를 감소시킴과 동시에 마이크로프로세서의 유휴 상태 (Idle State)를 최대한 줄이고자 한다. 마이크로프로세서의 유휴 상태 (Idle State)에서도 전류가 소비되기 때문에, 유휴 시간을 최대한 활용하기 위하여 마감시한을 만족하는 범위 내에서 태스크를 최대한 느린 속도로 실행시키고자 한다. 본 논문에서 측정한 결과, 구동 전압 3.3V에서 주파수 조정 계수 1일 때 4.8mA, 0.75일 때 3.67mA, 0.5일 때 3.34mA, 0.25일 때 2.25mA, 0.1일 때 1.56mA이다. 이 값은 ATMega128L 마이크로프로세서의 데이터 시트에서 기술한 값과 거의 유사하다 [8]. 또한 참고 문헌 [11]에서 기술한 바와 같이, 태스크의 최악 실행 시간보다 실제 실행 시간이 짧은 경우가 빈번하게 발생하기에, AU-WCET 규칙에서는 현재 시점에서 실행되고 있는 태스크의 마감 시한을 보장하는 범위 내에서 전력 소비를 최소화할 수 있도록 동작 주파수를 최대한 낮게 설정한다.

표 5는 공격적 실행 유휴 시간 분배 정책의 동작 과정을 보여준다. 그림 6은 동작 과정의 최종 결과를 보여준다. 시각 0에서 마감 시한이 가장 짧은 $Task_1$이 실행되며, 직전에 실행된 태스크가 없기에 AU-WCET 규칙 3만 적용된다. 이에 동작 주파수 조정 계수는 0.375 (= $Task(RWCET)_1/Task(RD)_1$ = 3/8)로 설정된다. 그리고 $Task_1$의 실제 실행 시간 양 2를 채우기 위해서는 시간 5.33 (=2/0.375)이 요구된다. 시각 5.33에서 $Task_2$가 실행된다. 이후 시각 5.33과 시각 6.88까지의 동작 과정은 앞서 시각 0에서 기술한 동작 방식과 유사하다.

표 5. 공격적 실행 유휴 시간 분배 정책의 동작 과정
Table. 5 Traces of AU-WCET distribution policy

시각	실행 태스크 (실행 대기 중인 태스크)	주파수 조정 계수 (f)	실행 시간 (초)
0	$Task_1$ ($Task_2$, $Task_3$)	0.375 (=$Task(RWCET)_1$ / $Task(RD)_1$ = 3/8)	5.33 (= $Task(ACET)_1$ / f = 2 / 0.375)
5.33	$Task_2$ ($Task_3$)	0.642 (=$Task(RWCET)_2$ / $Task(RD)_2$ = 3/(10-5.33))	1.55 (= $Task(ACET)_2$ / f = 1 / 0.642)
6.88	$Task_3$	0.14 (= $Task(RWCET)_3$ / $Task(RD)_3$ = 1/(14-6.88))	1.12 (= 8 – 6.88)
8	$Task_3$ ($Task_1$)	0.168 (= $Task(RWCET)_3$ / $Task(RD)_3$ = (1-0.157) / ((16-8) -3))	5 (= ($Task(ACET)_3$ - (8 – 6.88) × 0.14) / f = (1 – 0.157) / 0.168)
13	$Task_1$ ($Task_2$)	1 (= $Task(RWCET)_1$ / $Task(RD)_1$ = 3/(16-13))	1 (= $Task(ACET)_2$ / f = 1 / 1)
14	$Task_{32}$ ($Task_3$)	0.5 (=$Task(RWCET)_2$ / $Task(RD)_2$ = 3/(20-14))	2 (= $Task(ACET)_2$ / f = 1 / 0.5)
16	$Task_3$	0.083 (= $Task(RWCET)_3$ / $Task(RD)_3$ = 1/ (28-16))	12 (= $Task(ACET)3$ / f = 1 / 0.083)

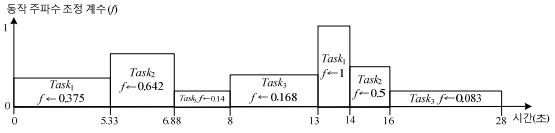

그림 6. 공격적 실행 유휴 시간 분배 정책의 동작 결과
Fig. 6 Outcomes of AU-WCET distribution policy

$Task_1$의 실행 요청이 발생한 시각 8에서는 $Task_3$의 마감 시한 14가 $Task_1$의 마감 시한 16보다 짧기 때문에 $Task_3$은 계속 실행된다. 시각 8에서 $Task_1$에 의해 남은 실행 유휴 시간은 5(= $Task(RD)_1$ - $Task(RWCET)_1$ = (16 - 8) - 3)이다. 그리고 $Task_3$의 남은 마감 시한 $Task(RD)_3$은 6 (= 14 – 8)이다. AU-WCET 규칙 2에 의해 $Task(RD)_3$은 5로 재설정된다. 시각 8까지 실행된 $Task_3$의 실제 실행 시간 양은 0.157 (= 0.14 × (8 –

6.88))이며, 동작 주파수 조정 계수는 0.168로 재설정된다. 시각 13에서 $Task_1$의 남은 마감시한 $Task(RD)_1$은 3 (= 16 - 13)이다. 이에 동작 주파수 조정 계수는 1 (= 3/3)이 된다. 그리고 시각 14와 16에서 AU-WCET 규칙 3이 계속 적용되어 동작 주파수 조정 계수는 0.5와 0.083으로 설정된다.

Ⅲ. 성능 분석

성능 분석에서 ATMega128L의 구동 전압은 3.3V로 설정하고, 동작 주파수 조정 계수 값은 최소 0.1부터 최대 1까지 0.1 단위의 값으로 설정된다. 이에 계산된 동작 주파수 조정 계수 값은 소수점 둘째 자리에서 올림을 하여 0.1단위로 조정하였다. 시뮬레이션에 사용된 주파수 조정 계수 대비 전류 소비량은 주파수 조정 계수가 0.9일 때 9.1mA, 0.8일 때 8.33mA, 0.7일 때 7.1mA, 0.6일 때 6.5mA, 0.5일 때 5.7mA, 0.4일 때 4.7mA, 0.3일 때 4.1mA, 0.2일 때 2.5mA, 0.1일 때 2mA, 그리고 유휴 상태일 때 1.56mA이다. 그리고 표 2에서 기술한 태스크 집합을 사용하였다.

그림 7은 매초마다 소비되는 전류량에 대한 시뮬레이션 결과를 보여준다. 그림 7에서 보는 바와 같이, 보수적 실행 유휴 분배 정책이 다른 실행 유휴 분배 정책보다 초당 사용되는 전류 소비량이 많다. 그리고 공격적 실행 유휴 분배 정책은 가능한 최소의 전력 소비를 유지하는 형태를 보여준다. 그림 8은 그림 7에서 보여준 개별 소비 전류량에 대한 누적 소비 전류량을 보여준다.

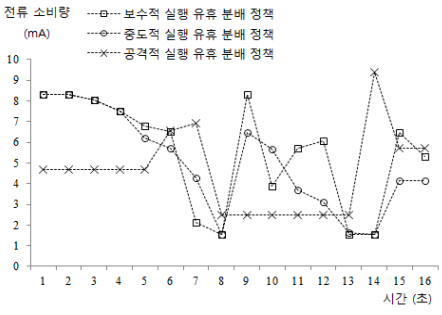

그림 7. 전류 소비량
Fig. 7 Current consumptions

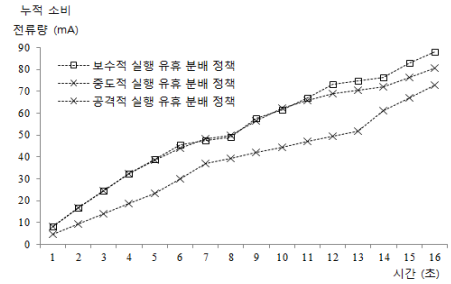

그림 8. 누적 전류 소비량
Fig. 8 Accumulative current consumptions

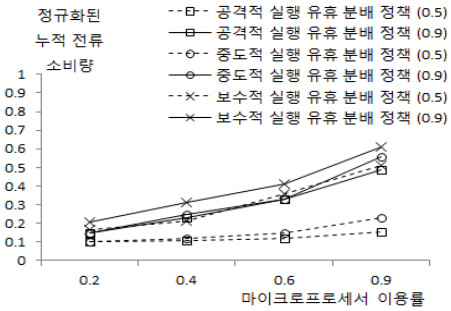

그림 9. 이용률에 따른 전력 소비량
Fig. 9 Current consumptions over varying utilization

그림 9는 마이크로프로세서 이용률과 최악 실행시간 대비 실제 실행시간 비율에 따른 누적 전류 소비량을 정규화한 시뮬레이션 결과를 보여준다. 개별 태스크의 실제 실행시간에 대한 변화는 가우시안 확률 분포를 따른다. 최악 실행시간 대비 실제 실행시간의 평균 비율은 0.5와 0.9의 값을 가지도록 실시간 태스크 집합을 임의로 생성하였다. 생성된 태스크 집합은 EDF 스케줄링 규칙에 따라 마이크로프로세서의 랜덤 이용률이 0.5부터 0.9 이하가 되도록 하였다. 시뮬레이션을 수행한 결과, 공격적, 중도적, 그리고 보수적 실행 유휴 시간 분배 정책 순으로 낮은 전력 소비량을 보여 주었다.

Ⅳ. 결 론

실행 유휴 시간은 센서 노드의 전력 소비를 감소시킬 수 있는 중요한 성능 인자이다. 이에 본 논문에서는 제한된 전원 공급을 사용하는 센서 노드 플랫폼에서 태스크의 최악 실행 시간보다 이른 실행 완료에 의해 발생

되는 실행 유휴 시간을 분배하는 보수적, 중도적, 그리고 공격적 실행 유휴 시간 분배 정책을 제안하고, 성능을 분석하였다. 성능 분석을 수행한 결과, 공격적 실행 유휴 분배 정책에 따른 실시간 전력 관리 스케줄링 기법이 전력 소비 측면에서 우수한 성능을 제공하였다.

감사의 글

본 연구는 미래부가 지원한 2014년 정보통신・방송(ICT) 연구개발사업의 연구결과로 수행되었음

REFERENCES

[1] S. Saruwatari, M. Suzuki, and H. Morikawa, “A compact hard real-time operating systems for wireless sensor nodes,” in *Proceedings of the 6th International Conference on Networked Sensing Systems*, Pittsburgh, pp. 1-8, 2009.

[2] F. Wolf, R. Ernst, and W. Ye, “Path clustering in software timing analysis,” *IEEE Transactions on Very Large Scale Integration (VLSI) systems*, vol. 9, no. 6, pp. 773-782, Dec. 2001.

[3] L. Benini, A. Bogligo, G.D. Micheli, “A survey of design techniques for system-level dynamci power management,” *IEEE Transactions on Very Large Scale Integration (VLSI) systems*, vol. 8, no. 3, pp. 299-316, Jun. 2000.

[4] P. Pillai and K.G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," *in Proceedings of ACM symposium on Operating Systems Principles*, New York, NY, pp. 89-102, 2001.

[5] B. Foo and M. Schaar, “A queuing theoretic approach to processor power adaptation for video decoding systems,” *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 378-392, Jan. 2008.

[6] T.D. Burd and R.W. Brodersen, “Energy efficient CMOS microprocessor design,” in *Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences*, Wailea, Hawaii, pp. 288-297, 1995.

[7] X. Zhang, C-Z. Xu, “Energy-aware modeling and scheduling for dynamic voltage scaling with statistical real-time guarantee,” *IEEE Transactions on Computers*, vol. 56, no. 3, pp. 1-15, Mar. 2007.

[8] ATMEL. ATMega128/L Datasheet. Available: http://www.atmel.com/Images/doc2467.pdf.

[9] M. Healey, T. Newe, and E. Lewis, “Power Management in Operating Systems for Wireless Sensor Networks,” in *Proceedings of IEEE Sensors Applications Symposium*, San Diego, California, pp. 1-6, 2007.

[10] B. Foo and M. Schaar, “A queuing theoretic approach to processor power adaptation for video decoding systems,” *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 378-392, Jan. 2008.

[11] R. Ernst and W. Ye, “Embedded program timing analysis based on path clustering and architecture classification,” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, San Jose, California, pp. 598-694, 1997.

탁성우(Sungwoo Tak)

2003년 2월 미국미주리주립대학교 Computer Science 박사
2004년 ~ 현재 부산대학교 정보컴퓨터공학부 교수 (부산대 컴퓨터및정보통신연구소, 스마트제어센터 겸임 연구원)
※관심분야 : 유무선 네트워크, 위치인식