Journal of Korea Multimedia Society Vol. 13, No. 10, October 2010(pp. 1423-1435)

배터리 작동식의 무선 센서 노드를 위한 에너지 효율적인 실시간 태스크 스케줄링

김동주[†], 김태훈[††], 탁성우[†††]

요 약

무선 센서 네트워크를 구성하는 센서 노드는 배터리 기반의 제한된 전원과 낮은 연산 능력의 초경량 마이크로프로세서, 그리고 제한된 크기의 메모리 자원 등과 같은 하드웨어 사양을 가지고 있다. 이와 같은 제약 사항에도 불구하고 무선 센서 노드는 센싱 데이터의 실시간 처리 및 데이터 송수신 작업을 동시에 병행할 수 있어야 한다. 이에 본 논문에서는 배터리 작동식의 무선 센서 노드를 위한 에너지 효율적인 실시간 태스크 스케줄링 기법을 제안하였다. 제안한 에너지 효율적인 실시간 스케줄링 기법은 태스크의 실제 실행시간이 최악 실행시간보다 작을 경우에 발생되는 태스크의 실행 여유시간을 이용하여, 마이크로프로세서의 동작 주파수를 조절하고 무선 센서 노드의 전력 소비를 줄인다. 제안한 기법의 동작을 시험한 결과, 효율적인 전력 소비를 제공함과 동시에 실시간 태스크의 마감시한이 보장됨을 확인하였다.

Energy-Efficient Real-Time Task Scheduling for Battery-Powered Wireless Sensor Nodes

Dong-Joo Kim[†], Tae-Hoon Kim[††], Sung-Woo Tak[†††]

ABSTRACT

Building wireless sensor networks requires a constituting sensor node to consider the following limited hardware resources: a small battery lifetime limiting available power supply for the sensor node, a low-power microprocessor with a low-performance computing capability, and scarce memory resources. Despite such limited hardware resources of the sensor node, the sensor node platform needs to activate real-time sensing, guarantee the real-time processing of sensing data, and exchange data between individual sensor nodes concurrently. Therefore, in this paper, we propose an energy-efficient real-time task scheduling technique for battery-powered wireless sensor nodes. The proposed energy-efficient task scheduling technique controls the microprocessor's operating frequency and reduces the power consumption of a task by exploiting the slack time of the task when the actual execution time of the task can be less than its worst case execution time. The outcomes from experiments showed that the proposed scheduling technique yielded efficient performance in terms of guaranteeing the completion of real-time tasks within their deadlines and aiming to provide low power consumption.

Key words: Wireless Sensor Node(무선 센서 노드), Task Scheduling(태스크 스케줄링), Dynamic Voltage Scaling(동적 전압 조절 기법)

※ 교신저자(Corresponding Author) : 탁성우, 주소 : 부산시 금정구 장전동 산30번지 부산대학교 정보컴퓨터공학부(609-735), 전화 : 051)510-2387, FAX : 051)515-2208, E-mail : swtak@pusan.ac.kr
접수일 : 2010년 4월 1일, 수정일 : 2010년 9월 8일
완료일 : 2010년 10월 5일
[†] 정회원, 부산대학교 컴퓨터공학과
(E-mail: fsrknigt@nate.com)
[†††] 준회원, 부산대학교 컴퓨터공학과
(E-mail: ninkth@hotmail.com)
[†††] 종신회원, 부산대학교 정보컴퓨터공학부 부교수
※ 이 논문은 국토해양부 첨단도시기술개발사업-지능형국토정보기술혁신 사업과제의 연구비지원(07 국토정보 C04)에 의해 수행되었습니다

1. 서 론

최근 무선 센서 네트워크의 응용 서비스는 단순히 센서 노드들이 환경 데이터를 수집하고 전송하는 수준을 넘어서 센서 노드 스스로 상황 인지 및 위치 기반 서비스, 그리고 군사 감시와 같은 실시간성이 요구되는 분야로 확대되고 있다. 센서 네트워크를 구성하는 센서노드의 핵심 기술 요소는 하드웨어 플랫폼과 소프트웨어 플랫폼 기술로 구성된다. 센서 노드의 하드웨어 플랫폼은 기술적, 그리고 경제적 제약사항을 해결하기 위하여 소형 메모리가 내장된 낮은 성능의 마이크로프로세서를 사용하며, IEEE 802.15.4 기반의 무선 네트워크를 사용한다. 또한 필요에 따라 온도, 조도, 그리고 습도 등을 측정할 수 있는 다양한 종류의 실시간 센서 장비를 지원한다. 센서 노드의 소프트웨어 플랫폼에서는 하드웨어 플랫폼의 효율적인 관리를 지원하는 센서 노드용 운영체제가 요구된다. 센서 노드용 운영체제는 센싱 및 센싱 데이터, 그리고 이와 관련된 이벤트 반응을 일정한 마감시한 내에 처리하는 실시간 태스크 스케줄링을 제공해야 한다. 또한 센서 노드 운영체제는 배터리 기반의 제한된 전력 공급원을 사용하는 하드웨어 플랫폼에서 동작하기 때문에 전력 소모량을 최소화 시킬 수 있는 시스템 소프트웨어 수준의 저전력 태스크 스케줄링 기법도 필요하다.

시스템 소프트웨어 수준의 저전력 설계 기법 연구로는 SVS (Static Voltage Scaling) 기법과 DPM (Dynamic Power Management) 기법, 그리고 DVS (Dynamic Voltage Scaling) 기법이 있으며, 최근의 저전력 기법에 관한 연구들은 주로 마이크로프로세서의 전력 상태를 동적으로 제어하는 방향으로 진행하고 있다[1]. 일반적으로 마이크로프로세서에 입력되는 시스템 부하는 시간에 따라 변하기 때문에 마이크로프로세서를 항상 부하가 최대일 때 요구되는 전력 상태로 동작시킬 필요가 없다. 따라서 마이크로프로세서에 입력되는 작업 혹은 이벤트 부하에 따라 마이크로프로세서에 공급되는 전압과 동작 주파수를 동적으로 조절하여 센서 노드의 전력 소비를 줄이면서 태스크와 이벤트를 처리할 수 있다. 또한 태스크의 마감시한을 만족하는 범위 내에서 프로세서의 동작 주파수를 동적으로 조절하여 최소한의 전력을 소비할 수 있다.

한편, 실시간 센서 노드 플랫폼에서 수행되는 작업은 태스크로 구체화되며, 태스크는 실행요청 시간, 실행 시간, 그리고 마감시한의 속성을 가진다. 태스크를 처리하는 센서 노드 운영체제는 동작 방식에 따라 이벤트 기반 방식과 멀티태스킹 방식으로 구분된다. 국내외에서 많이 사용되고 있는 이벤트 기반의 센서 노드 운영체제인 TinyOS는 선입선출 (FIFO: First-In First-Out) 방식의 비선점형 태스크 스케줄링 정책을 사용하기 때문에, 최상위 우선순위를 가진 사용자 태스크가 즉시 실행이 필요한 태스크임에도 불구하고 우선순위가 낮은 태스크가 획득한 프로세서의 사용권한을 선점하지 못한다. 따라서 실시간 서비스를 요구하는 태스크의 마감시한을 보장할 수 없다. 태스크에게 실시간성을 제공하기 위해서는 선점형 기반의 태스크 스케줄링 정책을 사용하는 운영체제가 요구된다. 일반적으로 멀티태스킹 방식의 운영체제는 선점형 기반의 태스크 스케줄링 정책을 제공하는데 적합하다. 그러나 선점형 태스크 스케줄링 정책은 태스크의 문맥 전환에 따른 오버헤드와 문맥을 저장하기 위한 추가적인 자원 및 연산에 따른 부가적인 전력 소비가 요구된다.

본 논문에서는 배터리 작동식의 무선 센서 노드에서 에너지 효율적인 다중 태스크의 제어 및 실시간성을 제공할 수 있는 에너지 효율적인 실시간 태스크 스케줄링 기법을 제안하였다. 제안한 기법은 일반적으로 태스크가 최악 실행시간보다 빨리 완료되기 때문에 계속해서 발생되는 실행 여유시간을 우선순위가 높은 태스크에게 분배한 후, 마이크로프로세서의 동작 주파수를 조절하여 센서 노드의 전력 소비량을 최소화한다. 본 논문의 구성은 다음과 같다. 2장에서는 저전력 관리 기법 및 기존 센서 노드에서 운영되는 소프트웨어 플랫폼에 대한 문제점을 기술하였다. 3장과 4장에서는 제안한 에너지 효율적인 실시간 태스크 스케줄링 기법인 LRT-DVS (Lazy Real-Time Dynamic Voltage Scaling)를 기술하였다. 5장에서는 제안한 기법의 성능을 분석하였으며, 마지막으로 6장에서는 결론을 기술하였다.

2. 관련연구

실시간 시스템에서 각 태스크의 실제 실행시간은 예상되는 최악 실행시간 (Worst Case Computation Time)보다 작은 경우가 대부분이며, 이로 인해 작업 부하량의 변동에 따른 실행 여유시간이 발생한다

[2]. 이러한 실행 여유시간을 센서 노드의 하드웨어 플랫폼에서 활용할 수 있는 저전력 기법인 DPM 기법, DVS 기법, 그리고 SVS 기법에 대하여 살펴보면 다음과 같다.

만약 하드웨어 플랫폼에 탑재된 주변 장치를 사용하는 태스크가 없다면, DPM 기법은 센서 노드의 하드웨어 플랫폼에서 요구되는 전력 소비를 줄이기 위하여 해당 주변 장치를 저전력 모드로 동작시킨다. DPM 기법은 주변 장치에 대한 전력 제어뿐만 아니라 마이크로프로세서 자체에 공급되는 전력 제어에도 적용되며, 처리해야 할 작업이 없는 경우에 마이크로프로세서를 유휴 상태로 전환한다[3]. DVS 기법은 마이크로프로세서에 공급되는 동작 전압을 조절하여 태스크의 실행에 필요한 전력을 최소화한다. 센서 노드와 같은 내장형 시스템에서 수행되는 작업의 부하량은 계속해서 변하기 때문에, DVS 기법에서는 마이크로프로세서가 항상 최고의 속도로 동작할 필요가 없다는 점에 중점을 두고 있다. SVS 기법은 미리 입력받은 태스크들의 실행 정보를 사용하여 오프라인에서 한 번 결정된 프로세서의 동작 주파수를 계속해서 사용하기 때문에 태스크의 실제 실행시간이 최악 실행시간 (Worst Case Computation Time)보다 더 짧은 경우가 발생하더라도 동작 주파수를 동적으로 재조정하지 않는다[4].

참조 논문 [5]와 [6]에서는 유닉스 운영체제 기반의 노트북 환경에서 프로세서의 공급 전압을 조절하는 태스크 스케줄링 기법의 전력 소비량을 분석하였다. 참조 논문 [7]에서는 DVS 기반의 실시간 태스크 스케줄링 기법인 cc-EDF (cycle-conserving Earliest Deadline First) 기법을 제안하였다. cc-EDF 기법에서는 태스크의 실제 실행시간이 최악 실행시간보다 작은 경우가 발생되면, 태스크의 이른 완료로 발생된 실행여유 시간을 나머지 대기 중인 태스크에게 균등하게 배분한다. 그리고 태스크의 실시간 속성을 위배하지 않는 범위 내에서 마이크로프로세서의 공급 전압을 줄여 전력 소모량을 최소화한다. 참조 논문 [8]과 [9]에서는 DVS 기법에 대한 수학적 모델링을 정립하고, 시뮬레이션 기반의 성능 평가를 수행하였다. 그러나 낮은 연산 능력을 가진 8비트 마이크로프로세서 환경에서 복잡한 수학적 모델링 기반의 DVS 기법을 적용하는 것은 어렵다. 그리고 참조 논문 [10]부터 [12]까지 제안된 DVS 기법은 태스크의 이른 실행 완료에 의해 발생되는 실행 여유시간을 고려하지 않았으며, 또한 제안된 알고리즘의 복잡도 때문에 8비트 마이크로프로세서에 적용하는 것은 어렵다.

한편, 센서 노드의 소프트웨어 플랫폼을 위한 운영체제 연구는 1990년대 말부터 시작되었으며, 이벤트 기반 운영체제인 TinyOS[13]와 멀티태스킹 기반 운영체제인 MANTIS[14]와 DCOS[15], 그리고 A-vrX[16]와 같은 운영체제가 개발되었다. 국내외에서 많이 사용되고 있는 TinyOS에서는 처리해야 할 태스크 혹은 이벤트가 없으면, 시스템을 수면 상태로 전환하여 센서 노드의 전력소모를 줄이는 단순한 DPM 기법을 사용한다. 비선점형 스케줄링 기법에서 실시간 태스크의 최적 스케줄링은 NP-hard 문제임이 이미 증명되었다 [17,18]. 따라서 TinyOS의 비실시간 스케줄링 정책은 무선 센서 네트워크 기반의 실시간 서비스 개발에 중요한 걸림돌이 된다.

미국 콜로라도 대학에서 개발한 MANTIS는 멀티태스킹 기반 센서 노드 운영체제 중에서 많이 알려져 있다. MANTIS의 태스크 스케줄러는 태스크의 우선순위에 따라 태스크를 실행시키며 동일한 우선순위를 가진 태스크들에 대해서는 라운드-로빈 (Round-Robin) 방식으로 처리한다. 그러나 선점형 태스크 스케줄러를 탑재한 센서 노드 운영체제에서는 태스크의 문맥전환에 따른 부가적인 메모리 사용량과 연산 오버헤드에 따른 전력 소비량이 증가할 수 있다. 참고문헌 [19]에서는 앞서 언급한 MANTIS에서 전력 소비를 줄이기 위하여 실행 중인 모든 태스크가 블록 상태로 전환되면, 전력 관리를 담당하는 태스크는 마이크로프로세서의 상태를 유휴 상태로 전환시켜 센서 노드의 전력 소비를 감소시킨다. 이러한 기법의 성능 실험을 수행한 결과, MANTIS의 전력 소비량이 TinyOS의 전력 소비량과 유사함을 보여 주었다.

본 논문에서는 동작 주파수의 동적 조절 기능을 지원하는 초소형 8비트 마이크로프로세서 ATmega 128L을 탑재한 센서 노드 플랫폼에서 실시간 태스크의 마감시한 보장 및 전력 소비를 감소시키는 에너지 효율적인 실시간 태스크 스케줄링 기법을 설계 및 구현하였다.

3. 실시간 센서 노드 플랫폼

그림 1은 본 논문에서 제안한 에너지 효율적인 실

시간 태스크 스케줄링이 실행되는 센서 노드 플랫폼 RT-UNOS (Real-Time Ubiquitous Network Operating System)를 보여준다.

제안한 실시간 센서 노드 플랫폼은 기본적으로 ATmega128L 8비트 마이크로프로세서를 대상으로 설계되었지만, 하드웨어 추상화 계층을 통하여 다른 마이크로프로세서에 대한 이식성도 고려하였다. 그림 1에서 실시간 센서 노드 플랫폼은 하드웨어 플랫폼과 소프트웨어 플랫폼으로 구분되며, 센서 네트워크를 구성하는 하드웨어 플랫폼은 8비트 초경량 마이크로프로세서와 센서 및 통신 장치로 구성된다. 소프트웨어 플랫폼은 실시간 커널 계층, 통신 프로토콜 계층, 그리고 센서 노드의 배포 목적을 수행하기 위한 응용 계층으로 구성된다.

그림 1에서 실시간 커널 계층의 인터럽트 핸들러 모듈, 입출력 및 센서 제어 모듈, 그리고 RF 디바이스 드라이버는 센서 장치 및 IEEE 802.15.4 기반의 통신 모듈과 같은 하드웨어 장치를 제어한다. 타이머 제어 모듈은 상위 응용 계층에 타이머 관련 서비스를 제공한다. 실시간 태스크 관리 모듈은 상위 계층을 구성하는 실시간 태스크의 생성 인터페이스를 제공하여 실시간 태스크의 속성을 명세하고, 실시간 태스크 스케줄링의 가능성 여부를 판단한 후에 제안한 LRT-DVS 스케줄러를 사용하여 저전력 실시간 태스크 스

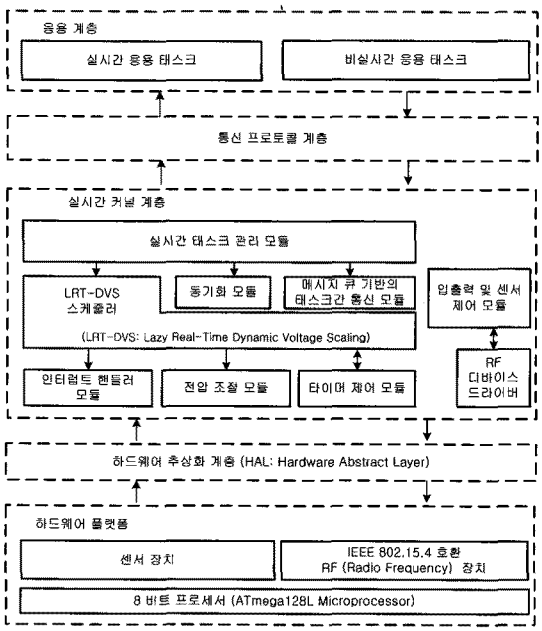


그림 1. 실시간 센서 노드 플랫폼 구조

케줄링을 지원한다. 동기화 모듈은 태스크간의 동기화를 제공하며, 메시지 큐 기반의 태스크간 통신 모듈은 태스크간의 통신을 담당한다. 통신 프로토콜 계층은 센서 노드간의 통신 기능을 제공한다. 마지막으로 응용 계층은 센서 노드의 사용 목적에 따라 여러 개의 실시간 응용 태스크와 비실시간 태스크로 구성된다.

마이크로프로세서에서 사용하고 있는 CMOS 회로의 전력 소모는 구동 전압에 의존한다. 그리고 CMOS 회로의 동작 주파수를 줄이면 마이크로프로세서는 더 낮은 전압에서 동작할 수 있다. CMOS 회로에 공급되는 전압이 V이고, 회로의 커패시턴스 부하가 C일 때, 전력 소모량 P는 식 (1)과 같이 근사화된다.

$$P \approx C \times V^2 \times f \quad (1)$$

식(1)에서 f는 동작 주파수를 나타내며, 주파수 속도는 공급 전압에 비례한다. 따라서 공급 전압을 $1/N$로 낮추면 주파수 속도는 $1/N$ 로 감소한다. 이 때, 마이크로프로세서가 처리해야 할 전체 작업에 대한 실행 시간은 N배 늘어나게 되므로 전체 작업을 완료하는데 필요한 전력은 $1/N^2$로 감소하게 된다. 이와 같이 마이크로프로세서의 동작 속도를 감소하면 처리해야 할 전체 작업의 응답 시간이 느려지는 결과를 가져올 수 있다. 그러나 실시간 태스크는 주어진 마감시한 내에 실행을 완료해야 하는 작업이므로 빠른 응답시간을 요구하지는 않는다. 따라서 태스크의 마감시한을 만족하는 범위 내에서 마이크로프로세서의 공급 전압을 조절하여 해당 프로세서의 전력 소비를 최소화할 수 있다. ATmega128L 마이크로프로세서에서 제공하는 동작 주파수 제어 인터페이스를 사용하여 동작 주파수를 조절하면, 마이크로프로세서의 공급 전압을 낮추어 전력 소비를 줄일 수 있다. 본 논문에서는, 센서 노드 플랫폼에서 실행되는 개별 태스크의 실제 실행시간이 최악 실행시간보다 작은 경우가 대부분이기 때문에 작업 부하량의 변동에 따른 실행 여유시간을 이용한다. 즉, 태스크의 이른 완료에 의해 생성된 실행 여유시간을 우선순위가 높은 태스크에게 할당하면, 실행 여유시간을 할당받은 해당 태스크는 마이크로프로세서의 동작 주파수를 최대한 낮추어 센서 노드의 전력 소비를 최소화함과 동시에 해당 태스크의 마감시한을 보장할 수 있다.

그림 2는 LRT-DVS 스케줄러 기법의 동작 흐름을 보여준다. 먼저 생성되는 태스크의 실시간 속성을 명시하는 요청시간, 실행시간, 그리고 마감시한 값이

실시간 속성 명세 테이블에 저장된다 (그림 2의 과정 1). 실시간 속성 명세 테이블에 저장되는 태스크 T_i는 요청시간 R_i, 최악 실행시간 C_i, 그리고 마감시한 D_i를 속성으로 가지며, $T_i = (R_i, C_i, D_i)$로 표현된다. 태스크의 우선순위는 D_i에 의해 결정되며, D_i가 작을수록 우선순위가 높아진다. LRT-DVS 기법은 EDF (Earliest Deadline First) 정책을 사용한다. 따라서 식 (2)를 사용하여 주어진 태스크 집합의 스케줄링 가능성을 검사한다 (그림 2의 과정 2).

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \tag{2}$$

현재 시각 t에 대하여 태스크 T_i의 상대적인 마감시한 $d_i(t)$는 $D_i - R_i$로 구할 수 있다. R_i와 C_i, 그리고 D_i는 태스크의 생성시점 (그림 2의 과정 1)에 주어지고 시간이 지나도 변하지 않는 반면, RD_i와 RC_i, 그리고 $RSTB_i$는 타이머 인터럽트가 발생하는 시점마다 재계산된다 (그림 2의 과정 A). RD_i는 현재 시각 t에서 태스크 T_i의 절대적인 마감시한 D_i까지 남은 상대적인 마감시한을 나타낸다. RD_i는 $D_i - t + R_i$로 계산된다. 남은 최악 실행시간인 RC_i는 현재 시각 t를 기준으로 하여 태스크 T_i의 남은 실행시간을 나타낸다. RC_i는 C_i 값에서 현재까지 실행한 시간을 뺀 값이다. 실행 여유시간 (RSTB: Residual Start Time Bound)인 $RSTB_i$는 태스크 T_i가 자신의 마감시한 D_i를 초과하지 않는 범위 내에서 태스크의 실행을 최대한 지연시킬 수 있는 시간을 나타낸다. $RSTB_i$는 $RD_i - RC_i$로 계산된다.

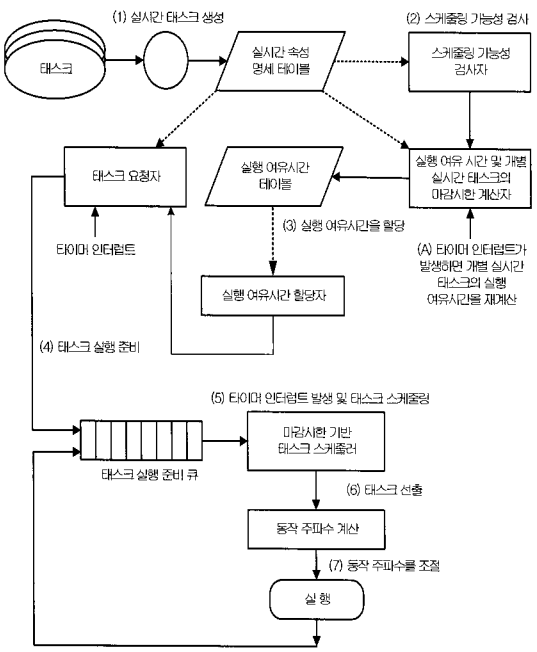


그림 2. LRT-DVS 스케줄러의 동작 흐름도

개별 태스크 T_i의 RD_i와 RC_i, 그리고 $RSTB_i$는 실행 여유시간 테이블에 저장된다 (그림 2의 과정 3). 예를 들어, 태스크 $T_1 = (4, 10, 18)$이 $t = 4$에 시작되었다면, 시각 $t = 9$에서 RC_1는 5이다. 시각 $t = 9$에서 최악 실행시간 C_1이 10이고, 마감시한 D_1이 18인 태스크 T_1의 $RSTB_1$은 8이 되며, 아무리 늦어도 시각 $t = 13$에는 실행이 되어야 마감시한을 보장할 수 있다. 각 태스크의 실행을 위한 동작 주파수 속도는 주파수 조정 계수 α ($0 < \alpha \leq 1$)로 정의된다. α는 태스크 T_i의 남은 마감시한 대비 남은 실행시간의 비율인 $\frac{RC_i}{RD_i}$로 나타낸다.

태스크 요청자는 실시간 속성 명세 테이블의 내용과 실행 여유 시간 테이블에 저장되어 있는 값을 기반으로 하여 각 태스크를 대기 상태에서 실행 준비 상태로 전환한다 (그림 2의 동작 과정 4). 실행 준비 상태가 된 태스크는 실행 준비 큐에 등록되며, 태스크 스케줄러는 마감시한을 기반으로 하여 태스크를 선택한다 (그림 2의 동작 과정 5와 6). 그리고 태스크의 마감시한을 초과하지 않는 범위 내에서 마이크로프로세서의 동작 주파수를 조절하여 선택한 태스크를 실행한다 (그림 2의 동작 과정 7).

4. 저전력 실시간 태스크 스케줄링 기법

이전 태스크의 이른 완료에 의해 발생된 실행 여유시간을 나머지 태스크에게 균등한 배분을 하는 cc-EDF 스케줄링 기법과는 다르게 LRT-DVS 스케줄링 기법은 발생한 시스템의 실행 여유 시간을 최상위 우선순위를 가지는 태스크 T_i에게 모두 할당한 후 재계산된 동작 주파수의 조정계수 α로 태스크 T_i를 실행시킨다. 주파수 조정 계수 α는 문맥 전환 시점마다 재계산된다. 그리고 태스크 T_i가 최악 실행시간 보다 일찍 완료되면, 발생된 실행 여유 시간을 차상위 우선순위를 가지는 태스크에 할당하여 동작 주파수 α가 1보다 낮은 조정계수로 실행할 수 있게 한다. 따라서 LRT-DVS 기법은 최상위 우선순위를 가지는 태스크 T_i보다 낮은 우선순위의 태스크가 최

고 주파수 조정 계수 α=1로 실행되는 경우가 발생하더라도 태스크 T_i를 최대한 낮은 동작 주파수로 실행한다. 이는 태스크들이 일반적으로 최악 실행시간보다 빨리 완료되기 때문에 계속해서 발생되는 실행 여유시간을 현재 최상위 우선순위를 가지는 태스크에게 분배하여 센서 노드의 전력 소비량을 현재 시점에서 최소화한다.

그림 3은 LRT-DVS 스케줄링 기법의 의사코드를 나타낸다. 각 태스크는 남은 마감시한에 의해 정렬되며, RD_i와 RC_i, 그리고 $RSTB_i$는 태스크의 생성시에 초기화되고 (그림 3의 과정 1), 타이머 인터럽트가 발생할 때마다 태스크에서 사용하는 변수 값은 감소된다 (그림 3의 과정 2). 실행할 태스크가 T_i 한 개만 있는 경우, 새로운 태스크 T_i는 태스크 실행 준비 큐 (*Task_Queue*)의 제일 앞에 삽입된 후, Task_Scheduler()에 의해 실행된다 (그림 3의 과정 3과 4). 만약 태스크 실행 준비 큐에 다른 태스크들이 존재하면, 태스크 T_j의 남은 마감시한과 T_i의 남은 마감시한을 비교하여 문맥전환 실행여부를 결정한다 (그림 3의 과정 5). 만약 시스템에 실행할 태스크가 없는 경우, 마이크로프로세서를 수면 상태로 변경한다 (그림 3의 과정 6).

한편 $RSTB_i$값은 새로운 태스크가 생성되는 시점에서 갱신된다 (그림 3의 과정 7). 만약 시스템에 태스크 T_i의 우선순위보다 더 높은 새로운 태스크 T_j가 생성되면 LRT-DVS 스케줄링 기법은 태스크 T_j의 남은 마감시한 RD_j와 태스크 T_i의 실행 여유시간 $RSTB_i$ 를 비교한다. 여기에서 태스크 T_i의 실행은 최대 $RSTB_i$ 시간만큼 최대한 지연시킨 후에 시작하여도 자신의 마감시한을 만족시킬 수 있다. 그러나 태스크 T_j의 RD_j가 태스크 T_i의 $RSTB_i$보다 더 큰 경우에 태스크 T_j가 최악 실행시간으로 수행된다면 태스크 T_i의 $RSTB_i$를 보장하지 못한다. 따라서 $RD_j > RSTB_i$ 이면, LRT-DVS 스케줄링 기법에서는 RD_j의값을 $RSTB_i$ 값으로 변경하고, 태스크 T_j의 실행에사용되는 동작 주파수의 조정 계수 α를 재계산

```
Task_Scheduler()
Task_Queue: 태스크의 남은 마감시한(RD)을 기준으로 오름차순으로 태스크를 정렬하여 저장한 큐
    while(true)
    if Task_Queue is empty then
        Switch Processor_State to Sleep_Mode;          — (6)
    else
        task = Select a task from Task_Queue;          — (4)
        Select_Frequency( );
        Run(task);
    endif
    endwhile
end of Task_Scheduler()

Select_Frequency( )
Ti: 현재 실행할 태스크
    α = RCi / RDi;                                     — (8)
    Set the Process_Frequency to α
end of Select_Frequency( )

Timer_Interrput()
    RCi = RCi - α;
    foreach task j                                     — (2)
        RDj = RDj - 1;
        RSTBj = RDj - RCj;
    endforeach
end of Timer_Interrput()

Task_Create(Ti)
    RDi = Di ; RCi = Ci; RSTBi = RDi - RCi;            — (1)
    if Task_Queue is empty then                        — (3)
        Insert Ti to Task_Queue;
    else
        cTask = Insert_Task_by_Deadline(Ti);
        if cTask == Ti then                            — (5)
            CONTEXT_SWICH();
            Select_Frequency( );
            Run(cTask);
            Select_Frequency( );
        endif
    endif
end of Task_Create()

Insert_Task_by_Deadline(Ti)
{T1, ..., Tk, ..., Ti-1}: 태스크 Ti보다 RD 값이 작은 태스크 집합 (단, RD1<...<RDk<...<RDi-1)
    Sort Task_Queue;
    for j = i-1 to 1 then                              — (7)
        RDj = min(RDj, RSTBj+1);
        RSTBj = RDj − RCj
        if (RSTBj < 0) Reject Ti;
    endfor
    Return T1;
end of Insert_Task_by_Deadline()
```

그림 3. LRT-DVS 스케줄링 기법의 의사코드

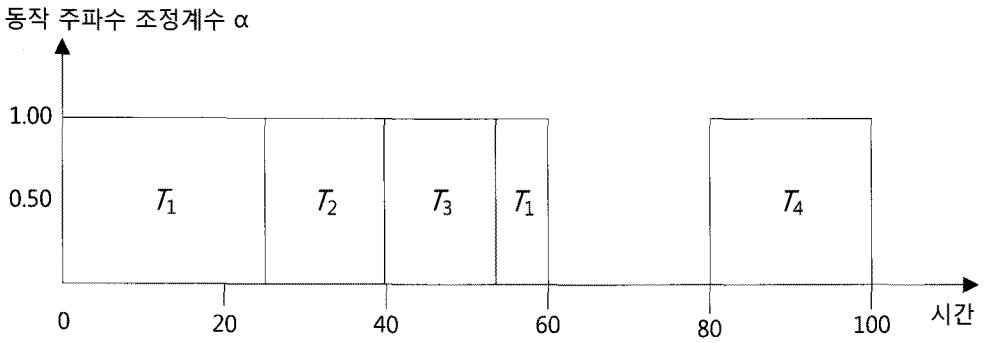


그림 4. EDF 스케줄링의 실행 결과

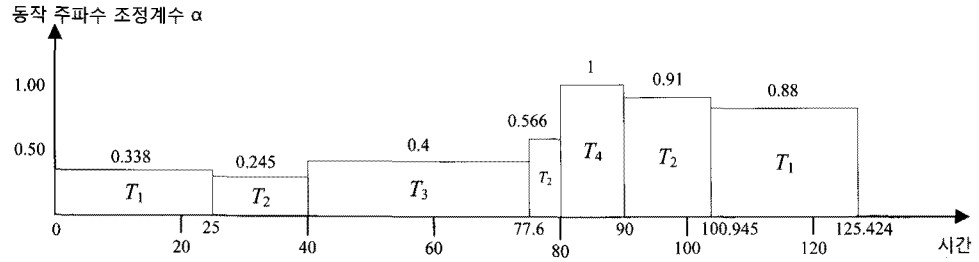


그림 5. LRT-DVS 스케줄링의 실행 결과

한다 (그림 3의 과정 8).

LRT-DVS 스케줄링 기법에 대한 자세한 동작을 살펴보기 위해 표 1에서 기술한 태스크 집합을 실행하였다.

표 1에서 기술한 태스크 집합이 저전력 스케줄링 기법을 적용하지 않은 EDF 스케줄링 기법에서 실행한 결과는 그림 4와 같다. 그리고 표 1에서 기술한 태스크 집합을 저전력 실시간 태스크 스케줄링 LRT-DVS 기법에서 실행한 결과는 그림 5와 같다.

그림 5의 실행 결과를 분석하면 다음과 같다. 시각 t = 0에서 태스크 T_1이 생성되면 LRT-DVS 스케줄링 기법은 태스크 T_1의 상대적인 마감시한 d_1을 초기화한다. 시각 t = 0에서 d_1 = 148 (148 - 0)로 절대적인 마감시한 D_1과 같다. 또한 D_1까지 남은 마감시한과 실행시간, 그리고 실행 여유시간 각각은 RD_1 = 148 (148 - 0), RC_1 = 50 (50 - 0), 그리고 $RSTB_1$ = 98 (148 - 50)로 초기화 된다. 태스크 T_1의 실행에 필요한 전력 소비를 줄임과 동시에 마감시한을 보장하는 마이크로프로세서의 동작 주파수 조정계수 α는 0.338 (50/148)로 계산된다. 이는 태스크 T_1을 0.338의 동작 주파수로 수행하면, 마감시한인 148에 정확히 완료됨을 뜻한다. 그러나 태스크 T_1이 최악 실행시간인 50보다 더 빨리 실행을 완료하면 완료시간은 148보다 더 빠를 수 있다.

시각 t = 25 에서 태스크 T_2가 생성되고, 태스크 T_2의 RD_2와 RC_2, 그리고 $RSTB_2$는 각각 120 (145 - 25), 20, 100 (120 - 20)으로 초기화된다. 이 때, 태스크 T_1보다 짧은 마감시한을 가지는 태스크 T_2의 우선순위가 높기 때문에 태스크 T_2가 태스크 T_1을 선점하게 되고, 선점당한 태스크 T_1의 RD_1과 RC_1, 그리고 $RSTB_1$은 123 (148-25), 41.55 (50 - (0.338 × 25)), 그리고 81.45 (123 - 41.55)로 각각 재계산된다. 그림

표 1. 태스크 집합

구분	요청시간 (R_i)	최악 실행시간 (C_i)	실제 실행시간 (e_i)	마감시한 (D_i)
태스크 T_1	0	50	30	148
태스크 T_2	25	20	15	120
태스크 T_3	40	20	15	85
태스크 T_4	80	20	10	20

3에서 기술한 LRT-DVS 스케줄링 기법의 과정 7에 의하여 태스크 T_2의 RD_2와 $RSTB_2$는 81.45와 61.45로 갱신되며, 동작 주파수 조정계수 α는 0.245 (20/81.45)로 설정된 후에 태스크 T_2가 실행되어 시각 t = 40까지 실행된다.

시각 t = 40에서 생성된 태스크 T_3은 실행 중인 태스크 T_2보다 우선순위가 높기 때문에 태스크 T_3은 태스크 T_2가 획득한 프로세서의 권한을 선점한다. 태스크 T_3의 RD_3과 RC_3, 그리고 $RSTB_3$은 85, 20, 그리고 65로 각각 초기화 된다. 그리고 실행 대기 중인 태스크 T_1의 RD_1과 $RSTB_1$은 108과 66.45로 각각 재계산된다. 선점당한 태스크 T_2의 RD_2와 RC_2, 그리고 $RSTB_2$는 66.45, 16.32, 그리고 50.13으로 재계산된다. 이 때, 태스크 T_3의 RD_3은 현재 실행 대기 중인 태스크들의 실행 여유시간 중에서 최소 $RSTB_2$ 값인 50.13로설정되고, $RSTB_3$은 30.13으로 갱신된다. 또한, 태스크 T_3의 동작 주파수 조정계수 α는 0.4로 설정되며 태스크 T_3의 실행은 시각 t = 77.6에 완료된다. 태스크 T_3의 실행이 완료되는 t = 77.6에서 태스크 T_1의 RD_1과 $RSTB_1$은 70.4와 28.85로 재계산되고, 태스크 T_2의 RD_2와 $RSTB_2$는 28.85와 12.53으로 재계산된다. 태스크 T_1보다 짧은 마감시한을 가지는 태스크 T_2의 우선순위가 더 높기 때문에 태스크 T_2의 동작 주파수 조정계수 α는 0.5656으로 설정되고, 시각 t = 80까지 실행된다.

시각 t = 80에서 생성된 태스크 T_4는 태스크 T_2가 획득한 프로세서 권한을 선점하며, 태스크 T_4의 RD_4와 RC_4, 그리고 $RSTB_4$는 20, 20, 그리고 0으로 각각 초기화된 후, 지금까지 설명한 과정을 계속 반복한다. 시각 t = 90에서 태스크 T_4의 실행이 완료되며, 태스크 T_2의 동작 조정계수 α는 0.91로 설정되고 태스크 T_2가 실행된다. 마지막으로 태스크 T_1의 동작 주파수 조정계수 α는 0.88로 설정되며 시각 t = 125.424에서 태스크 T_1의 실행이 완료된다. 모든 태스크의 실행이 완료되면 마이크로프로세서는 유휴 상태로 전환된다. 제안한 저전력 실시간 센서 노드 플랫폼에서는 처리할 태스크가 없는 경우에 마이크로프로세서를 유휴 상태로 전환시킨다.

5. 성능 분석 및 비교 평가

본 논문에서는 제안한 기법의 성능 분석을 위하여 이벤트 기반 센서 노드 운영체제와 멀티태스킹 기반 센서 노드 운영체제를 각각 대표하는 MANTIS와 TinyOS를 비교 대상으로 사용하였다. 또한 기존의 저전력 실시간 스케줄링 기법인 cc-EDF 기법을 RT-UNOS에 적용시켰다. 센서 노드의 하드웨어 플랫폼으로는 Chipcon사의 CC2420DB 무선 센서 노드를 사용하였다. 성능 평가 요소는 센서 노드의 전력 소모량과 실시간 태스크의 마감시한 초과율이다. 센서 노드의 전력 소비량은 마이크로프로세서와 외부 전원 공급 모듈 간에 흐르는 전류량을 오실로스코프로 측정한 값과 식 (3)을 사용하여 센서 노드 플랫폼에서 소비되는 실제 전력 소비량을 계산하였다. 식 (3)에서 e_i와 α_i는 태스크 T_i의 실제 실행시간과 동작 주파수조정 계수를 나타낸다. 그리고 c_α는 동작 주파수의 조정계수 변화에 따라 소비되는 전류량을 나타낸다. ATmega128L에서 동작 주파수 조정계수 변화에 따라 소비되는 전류량은 표 2와 같다.

$$\sum_{i=1}^{n} e_i \times \alpha_i^2 \times c_\alpha \quad (3)$$

표 3에서 기술한 태스크 집합의 실행 결과는 그림 6과 같다. 그림 6에서 EDF의 전력 소비량은 식 (3)에 의하여 35.75이다. 그리고 SVS 및 cc-EDF와 LRT-DVS 스케줄링 기법에서 요구되는 전력 소모량은 각각 30.15435와 29.204, 그리고 28.15754가 된다. LRT-DVS 스케줄링 기법이 EDF 및 SVS와 cc-EDF 스케줄링 기법보다 전력 소비량이 적다. 센

표 2. ATmega128L에서 동작 주파수의 조정계수에 따라 소비되는 전류량 (mA)

조정 계수 α	마이크로프로세서의 상태	
	실행 중인 태스크가 있는 경우	유휴 상태인 경우
1.00	9.25	4.8
0.75	7.2	3.67
0.50	5.7	3.34
0.25	3.5	2.25

표 3. 전력 소비량 측정 분석에 사용되는 태스크 집합

구분	요청시간 (R_i)	최악 실행시간 (C_i)	실제 실행시간 (e_i)	마감시한 (D_i)
태스크 T_1	0	3	2	8
태스크 T_2	0	3	2	10
태스크 T_3	0	1	1	14

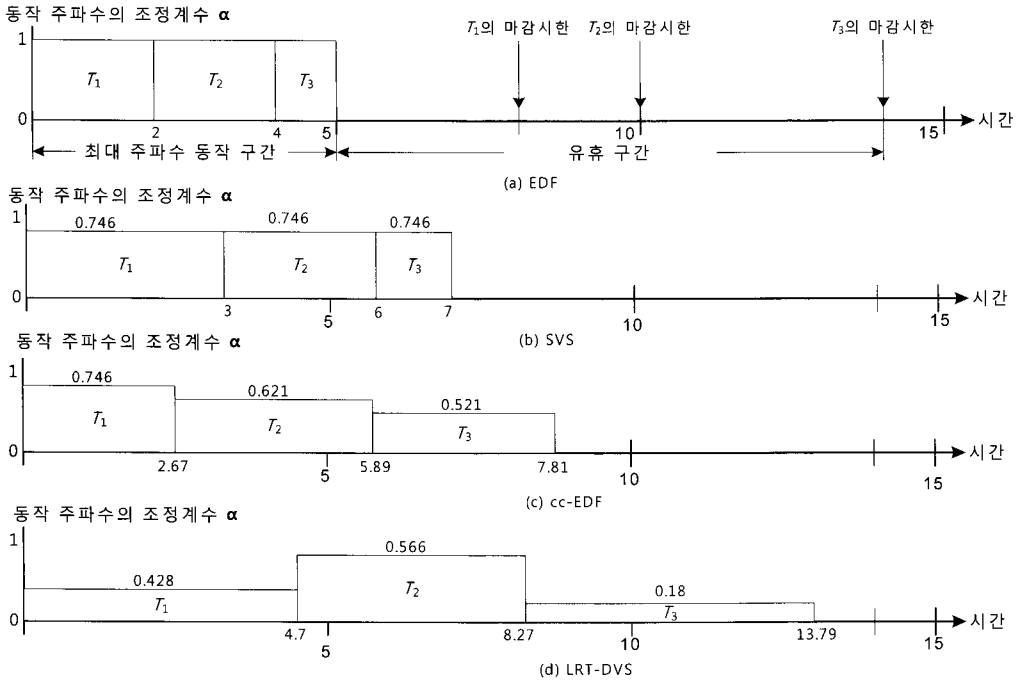
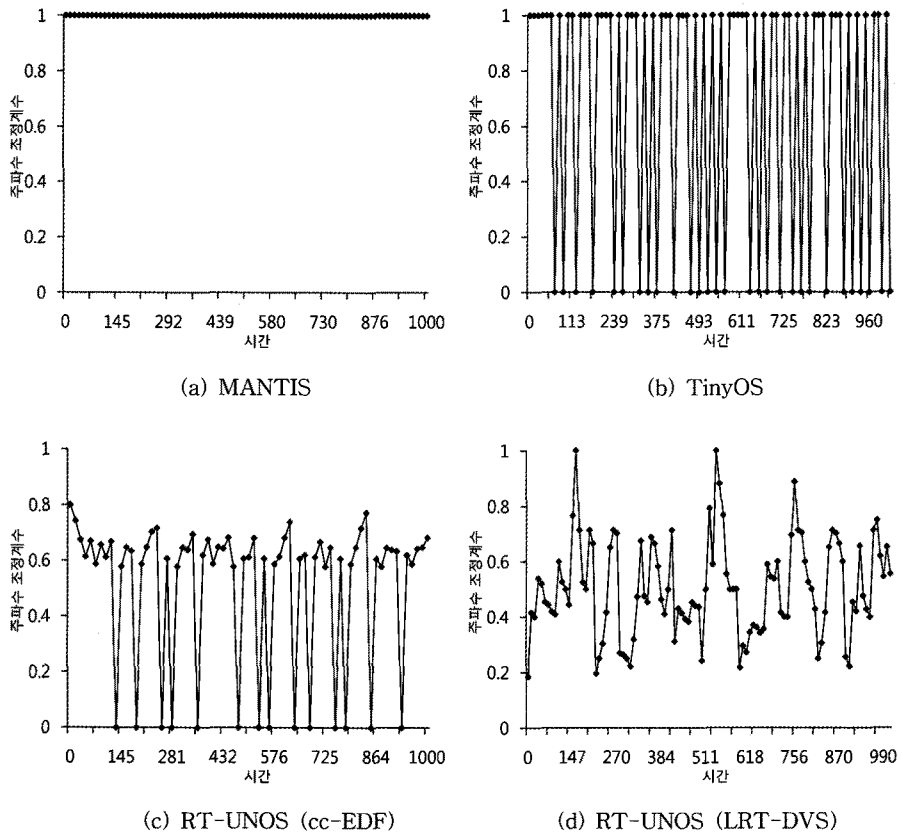


그림 6. 태스크 스케줄링 기법에 따른 실행 결과

서 노드의 동작 시간이 길어질수록 스케줄링 기법에 따른 전력 소비량의 차이는 계속해서 증가한다. 그림 6의 (b)에서 SVS 스케줄링 기법은 오프라인에서 동작 주파수를 미리 결정한다. SVS 스케줄링 기법은 표 3에서 기술한 태스크 집합에 대하여 동작 주파수의 조정계수를 0.746 (= 3/8 + 3/10 + 1/14)으로 설정한다. SVS 스케줄링 기법은 오프라인에서 한 번 결정된 동작 주파수의 조정계수를 계속해서 사용한다. 따라서 태스크 T_1의 실행이 시각 2에 완료되는 동작 주파수의 조정계수 0.746은 그대로 유지되면서, 태스크 T_2와 T_3의 실행에도 적용된다.

그림 6의 (c)에서 cc-EDF 스케줄링 기법은 태스크의 실제 실행시간을 기반으로 하여 동작 주파수의 조정계수를 조절한다. 또한 실행 완료된 태스크의 실행 여유시간을 모든 태스크에게 균등하게 분배한다. 시각 t = 0에서 cc-EDF 스케줄링 기법은 표 3에서 기술한 태스크 집합의 최악 실행시간 및 마감시한 정보를 사용하여 동작 주파수의 조정계수를 0.746 (= 3/8 + 3/10 + 1/14)으로 설정한다. 시각 t = 2.67에서 태스크 T_1의 실행이 완료되며 태스크 T_1의 실제 실행 시간은 2가 된다. 그리고 새로운 동작 주파수의 조정계수를 재계산하기 위하여 차후에 다시 실행이 되는 태스크 T_1의 실행 시간을 2로 가정하고, 태스크 T_2의 실행에 사용되는 동작 주파수의 조정계수를 0.621 (= 2/8 + 3/10 + 1/14)로 설정한다. 시각 t = 5.89에서 태스크 T_3의 실행에 사용되는 동작 주파수의 조정계수를 0.521 (= 2/8 + 2/10 + 1/14)로 설정한 후에 태스크 T_3의 실행이 마감시한 내에 완료된다. 그림 6의 (d)에서 보여주는 LRT-DVS 스케줄링 기법의 실행 결과는 그림 3에서 기술한 동작 절차에 따라 생성된다.

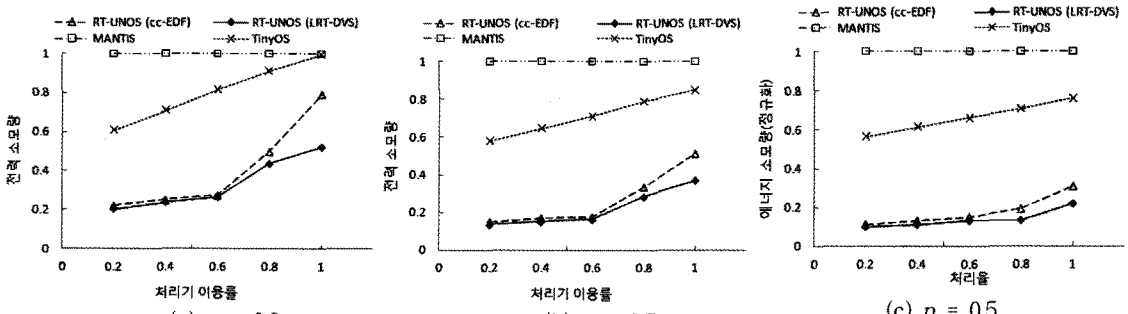
그림 7과 그림 8에서 수행한 실험 환경은 다음과 같다. 하나의 루프 (Loop)를 가지면서 마감시한을 갖



(a) MANTIS (b) TinyOS

(c) RT-UNOS (cc-EDF) (d) RT-UNOS (LRT-DVS)

그림 7. 동작 주파수의 조정 계수 변화



(a) p = 0.9 (b) p = 0.7 (c) p = 0.5

그림 8. 동작 주파수의 조정 계수 변화

는 10개의 더미 (Dummy) 실시간 태스크를 실행하였다. 그리고 태스크 집합의 처리기 이용률 (Processor Utilization)은 0.2부터 시스템의 과부하 상태인 1.0까지 0.2간격으로 변화시켰다. 개별 태스크의 실제 실행시간의 변화는 가우시안 (Gaussian) 확률분포를 따르며, 최악 실행시간 대비 실제 실행시간의 비율은 p로 표현하였고, p는 0.5, 0.7, 0.9의 값을 가지도록 하였다. 실험 결과는 동일 실험에 대하여 30번 반복하여 측정한 평균 결과이며, 매 실험마다 다른 확률 분포를 생성시켜 다양한 태스크 환경을 시뮬레이션 하였다.

그림 7은 처리기 이용률이 0.8이고, 최악 실행시간 대비 실제 실행시간의 비율 p가 0.7에서 센서 노드 플랫폼의 주파수 변화과정을 보여준다. 그림 7의 (b)

에서 MANTIS는 동작 주파수의 조정 계수를 1로 설정하여 마이크로프로세서가 항상 최고 속도로 동작한다. 그림 7의 (b)에서 TinyOS는 처리해야 할 태스크가 없으면, 마이크로프로세서를 유휴 상태로 전환시킨다. 그림 7의 (c)와 (d)는 RT-UNOS에 cc-EDF 기법과 LRT-DVS 기법을 적용한 결과이다. cc-EDF 기법에서는 동작 주파수의 조정 계수가 0.8 근처에서 설정되어 태스크를 처리하지만, LRT-DVS 기법에서는 동작 주파수의 조정 계수가 최소 0.2로 설정하여 센서 노드의 전력 소비를 줄인다.

그림 8은 최악 실행시간 대비 실제 실행시간의 비율 p에 따른 전력 소모량의 정규화 값을 보여준다. 동작 주파수의 조정 계수를 1로 설정하여 마이크로프로세서가 항상 최고 속도로 동작하는 멀티태스킹 기반의 MANTIS인 경우, 처리기 이용률 및 p 값과 상관없이 전력 소모량의 최대값 1을 유지한다. 처리해야 할 태스크가 없으면, 마이크로프로세서를 유휴 상태로 전환하는 TinyOS에서는 cc-EDF 기법과 LRT-DVS 기법을 적용한 RT-UNOS보다 전력 소모량이 많음을 확인하였다. 이는 MANTIS와 TinyOS가 p가 감소할수록 증가되는 실행 여유시간을 충분히 활용하지 못하고 있음을 알 수 있다. 그리고 처리기 이용률의 증가와 p가 감소할수록 cc-EDF기반의 RT-UNOS에서는 태스크에게 균등하게 배분되는 실행 여유시간이 감소하게 되어 동작 주파수의 조정 계수 값이 증가한다. LRT-DVS기반의 RT-UNOS에서는 처리기 이용률과 p 값이 증가할수록 시스템에서 발생되는 실행 여유시간이 감소하게 되지만, 시스템 내의 우선순위가 낮은 태스크들로부터 발생되는 실행 여유시간을 현재 실행 중인 우선순위가 높은 태스크가 전부 활용하기 때문에 cc-EDF 기법보다 낮은 동작 주파수의 조정 계수를 사용할 수 있다. 따라서 LRT-DVS 기법의 전력 소모량이 cc-EDF 기법보다 낮다.

처리기 이용률이 1이고 p 값이 0.9인 그림 8의 (c)에서 비선점형 EDF 스케줄링 기법을 사용하는 TinyOS의 태스크 마감시한 초과율은 38.1%, 선점형 EDF 스케줄링 기법을 사용하는 MANTIS와 cc-EDF 및 LRT-DVS 스케줄링 기법을 사용하는 RT-UNOS의 태스크 마감시한 초과율은 3.72%이다. TinyOS는 선입선출 방식의 비선점형 태스크 스케줄링을 사용하기 때문에 즉시 실행이 필요한 태스크는 자신의 우선순위보다 낮은 태스크가 획득한 CPU 사용권한을 선점할 수 없다. 따라서 TinyOS의 마감시한 초과율은 다른 기법에 비해 매우 높다.

6. 결 론

배터리 기반의 제한된 전력 공급을 사용하는 센서 노드의 설계 및 구현에서, 센서 노드를 구성하는 초경량 마이크로프로세서의 낮은 연산능력과 제한된 크기의 메모리, 그리고 센싱 데이터와 관련된 이벤트 반응을 실시간으로 처리해야 하는 제약 사항을 고려해야 한다. 이에 본 논문에서는 초경량 8비트 마이크로프로세서인 ATmega128L 기반의 센서 노드에서 에너지 효율적인 실시간 태스크 스케줄링 기법인 LRT-DVS를 제안하였다. 제안한 LRT-DVS는 태스크의 실제 실행시간이 최악 실행시간보다 작을 경우에 발생되는 태스크의 실행 여유시간을 이용하여, 마이크로프로세서의 동작 주파수를 조절하고 무선 센서 노드의 전력 소비를 줄인다. 제안한 LRT-DVS의 동작 시험을 수행한 결과, LRT-DVS를 탑재한 RT-UNOS는 TinyOS와 MANTIS보다 25%, 그리고 cc-EDF 보다 15% 향상된 효율적인 에너지 소비를 제공함과 동시에 실시간 태스크의 마감시한 보장이 우수함을 확인하였다.

참 고 문 헌

[1] V. Raghunathan, C. Schurgers, S. Park, and M.B. Srivastava, "Energy-aware wireless microsensor networks," *IEEE Signal Processing Magazine*, Vol. 1, No. 2, pp. 40-50, 2002.

[2] R. Ernst and W. Ye, "Embedded Program Timing Analysis Based on Path Clustering and Architecture Classification," IEEE/ACM International Conference on Computer-aided Design, pp. 598-604, 1997.

[3] T.A. Pering, T.D. Burd, and R.W. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms," International Symposium on Low Power Electronic Design, pp. 76-81, 1998.

[4] H. Aydin, R. G. Melhem, D. Mosse, and P.

Mejia-Alvarez, "Power aware scheduling for periodic real-time tasks. *IEEE Transaction on Computers*, Vol. 53, No. 5, pp. 584 - 600, 2004.

[5] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," Symposium on Operating Systems Design and Implementation, pp. 13-23, 1994.

[6] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power CPU," International Conference on Mobile Computing and Networking, pp. 13-25, 1995.

[7] P. Pillai and K.G. Shin, "Real-Time dynamic voltage scaling for low-power embedded operating systems," ACM symposium on Operating Systems Principles, pp. 89-102, 2001.

[8] X. Zhong and C. Z. Xu, "Energy-aware modeling and scheduling for dynamic voltage scaling with statistical real-time guarantee," *IEEE Transactions on Computer*, Vol. 56, No. 3, pp. 358-372, 2007.

[9] M. Kargahi and A. Movaghar, "Stochastic DVS-based dynamic power management for soft real-time systems," Microprocessors and Microsystems, Vol. 3, No. 2, pp. 121-144, 2008.

[10] A. Manzak and C. Chakrabarti, "Variable voltage task scheduling algorithms for minimizing energy/power," *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 11, No. 2, pp. 270-276, 2003.

[11] W. C. Kwon and T. Kim, "Optimal voltage allocation techniques for dynamically variable voltage processors," *ACM Transactions on Embedded Computing Systems*, Vol. 4, No. 1, pp. 211-230, 2005.

[12] V. Swaminathan and K. Chakrabarty, "Network flow techniques for dynamic voltage scaling in hard real-time systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 23, No. 10, pp. 1385-1398, 2004.

[13] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler, "The emergence of networking abstractions and techniques in TinyOS," USENIX/ACM Symposium on Networked Systems Design and Implementation, pp. 1-14, 2004.

[14] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, and B. Shucker, "MANTIS OS:An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms," *Mobile Networks and Applications Journal,* Vol. 10, No. 4, pp. 563-579, 2005.

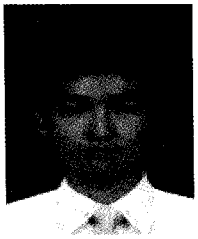
[15] T.J. Hofmeijer, S.O. Dullman, P. G. Jansen, and P. J. Havinga, "DCOS, A Real-time Lightweight Data Centric Operating System," International Conference on Advances in Computer Science and Technology, pp. 259-264, 2004.

[16] P. Ganesan and A.G. Dean, "Enhancing the AvrX kernel with efficient secure communication using software thread integration," Real-Time and Embedded Technology and Applications Symposium, pp. 265-275, 2004.

[17] K. Jeffay and C.U. Martel, "On Non-preemptive Scheduling of Periodic and Sporadic Tasks," IEEE Real-Time Systems Symposium, pp. 129-139, 1991.

[18] L. Georges, P. Muehlethaler, and N.Rivierre, "A Few Results on Non-preemptive Real-Time Scheduling," *INRIA Research Report nRR-3926*, 2000.

[19] C. Duffy, U. Roedig, J. Herbert, and C. Sreenan, "Improving the energy efficiency of the MANTIS kernel," IEEE European Workshop on Wireless Sensor Networks, pp. 261-276, 2007.



김 동 주

2007년 2월 신라대학교 컴퓨터교육과 학사
2009년 2월 부산대학교 컴퓨터공학과 석사
2009년~현재 LG전자 MC 연구소 연구원
관심분야 : 임베디드 시스템 운영체제

김 태 훈

2006년 2월 부산대학교 정보컴퓨터공학부 공학사
2008년 2월 부산대학교 컴퓨터공학과 공학석사
2008년 3월~현재 부산대학교 컴퓨터공학과 박사과정
관심분야 : 무선 네트워크, P2P, 위치인식

탁 성 우

1995년 2월 부산대학교 컴퓨터공학과 공학사
1997년 2월 부산대학교 컴퓨터공학과 공학석사
2003년 12월 미국 미주리주립대 Computer Science박사
2004년 South Dakota State Univ. 부교수
2004년~현재 부산대학교 정보컴퓨터공학부 부교수
2004년~현재 부산대학교 컴퓨터 및 정보통신 연구소 겸임 연구원
관심분야 : 유무선 네트워크, SoC설계, 실시간 시스템, 위치인식