무선 센서 네트워크 환경에서의 에너지 효율성을 고려한 태스크 스케줄링 기법

준회원 이 진 호*, 최 훈**, 종신회원 백 윤 주**°

Task Scheduling Technique for Energy Efficiency in Wireless Sensor Networks

Jinho Lee*, Hoon Choi** *Associate Members*, Yunju Baek**° *Lifelong Member*

요 약

무선 센서 네트워크에서의 센서 노드는 배터리로 동작하기 때문에 에너지에 대한 제약이 있다. 따라서 무선 센서 노드를 위한 효율적인 전력 관리 기법과 스케줄링 기법 설계가 중요한데, 본 논문에서는 운영체제 레벨의 소비 에너지를 줄이기 위한 알고리즘을 제안한다. 효율적인 배터리 사용을 위해 상황에 따라 필요한 컴포넌트에만 전원을 인가함으로써 센서노드의 에너지 소비를 줄일 수 있다. 본 논문에서는 제안한 알고리즘을 시뮬레이션 한 결과 기존의 duty 사이클과 비교하여 최대 56%의 에너지가 절약되었음을 알 수 있다.

Key Words : Energy-Aware, Sensor Node, Sensor Networks, Task Scheduling

ABSTRACT

A wireless sensor node is typically battery operated and energy constrained. Therefore it is critical to design efficient power management technique and scheduling technique. In this paper, we propose an OS-level power management technique for energy saving of wireless sensor node, it is called EA-SENTAS (Energy-Aware Sensor Node TAsk Scheduling). It can decrease the energy consumption of a wireless sensor node to use task scheduling technique that shut down components or use low power mode of each component when not needed. Simulation results show that EA-SENTAS saves energy up to 56 percent to compare with conventional duty cycle.

Ⅰ. 서 론

무선 센서 네트워크는 많은 수의 센서 노드들이 배치된 센서 필드와 외부 망을 연결하는 싱크로 구성된다. 사용자는 싱크를 통하여 센서 필드에 질의를 전달하거나 센서 필드로부터 수집된 데이터를 전달 받는다. 데이터 전달과정에서 센서 노드들은 기존에 미리 설치된 네트워크를 사용하지 않고, 자가 구성(Self-organizing)이 가능한 무선 ad-hoc 네트워크를 구성하여 싱크에게 데이터를 전달한다. 이러한 무선 센서 네트워크는 군사, 홈 네트워크, 환경 감시, 공장 관리, 그리고 재난 감시 등의 다양한 분야에 적용할 수가 있다.

이러한 네트워크의 센서 노드는 기존의 컴퓨팅 플랫폼과는 다르게 계산 능력, 메모리와 배터리 등 모든 자원이 극도로 제한적이다. 각 노드는 데이터 전송과 라우터의 기능을 동시에 한다. 만약 어떤 노드의 배터리가 방전되어 더 이상 그 노드를 사용할

※ 이 논문은 교육인적자원부 지방연구중심대학육성사업 (차세대물류IT기술연구사업단)의 지원에 의하여 연구되었음.
* LG전자 MC 사업본부
** 부산대학교 컴퓨터공학과 임베디드시스템 연구실 (yunju@pusan.ac.kr) (° : 교신저자)
논문번호 : KICS2006-05-233, 접수일자 : 2006년 5월 25일, 최종논문접수일자 : 2006년 9월 18일

수 없다면 네트워크의 구성을 다시 해야 하게 되고 그에 따라 전체 네트워크의 노드들은 많은 에너지를 소모하게 된다. 따라서 전원 관리는 중요한 이슈이다[1, 2].

센서 노드의 에너지 효율성을 높이기 위하여 에너지 효율성을 고려한 하드웨어 설계, 효율적인 라우팅 프로토콜 및 MAC 프로토콜 설계에 대한 연구가 진행되고 있다. 그 중에 OS레벨에서의 기법은 이종의 센서노드와의 호환성이나 개발 시간에 있어 매우 효과적인 방법이라 할 수 있다. 그래서 우리는 센서 노드의 효율적 에너지 사용을 위해 OS레벨의 전력 소모 기법을 제안한다.

일반적으로 무선 센서 노드는 duty cycle이라는 미리 정의 되어진 일정에 따라 동작한다. 노드의 모든 장치는 태스크의 동작과는 관계없이 항상 켜져 있다. 이것은 에너지를 비효율적으로 소비하고 결국 전체 시스템의 수명을 줄이는 결과를 가져온다. 이러한 불필요한 에너지의 소비를 방지하기 위하여 각 컴포넌트들의 전력 모드를 적절히 조합하는 방법을 사용한다. 컴포넌트들의 다양한 전력 모드를 적절히 사용하지 못하게 되면 중요한 이벤트를 감지하지 못하거나 다른 센서 노드와의 통신 문제 그리고 태스크의 응답시간 지연 등 무선 센서 노드의 기능을 제대로 수행하지 못하는 문제가 발생할 수 있다. 이러한 문제점을 최소화하기 위해 5가지 시스템 상태 모델을 제시하고 제시한 시스템 상태 모델을 최적으로 사용할 수 있는 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. 2장은 센서 노드의 시스템 구조를 설명하고, 관련 연구에 대한 소개와 문제점을 설명한다. 3장에서 새로운 센서 노드 시스템 상태 모델을 제안하고, 제안하는 알고리즘을 설명한다. 4장은 실험 결과를 제시하고, 마지막으로 5장에서는 결론과 향후 연구 과제를 기술한다.

Ⅱ. 배 경

2.1 센서 노드의 구조

그림 1은 무선 센서 노드의 구조를 나타내고 있다. 서 노드의 시스템은 크게 센서 보드, 마이크로컨트롤러, 외부 메모리, 통신부, 그리고 전원부로 이루어져 있다. 센서 보드에는 다수의 센싱 유닛(sensing unit)들이 존재하고 이러한 센싱 유닛들을 온도나 습도 등 일상생활에서 지속적으로 발생하는 이벤트센를 감지하는 알파 센서와 소리, 움직임 등 짧은

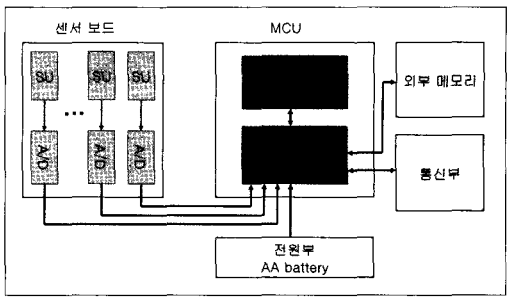


그림 1. 무선 센서 노드 구조

순간 발생하는 이벤트를 감지하는 베타 센서로 구분할 수 있다. 알파 센서는 발생하는 이벤트가 실시간으로 처리될 필요가 없고, 이러한 센서와 관련된 작업은 주로 주기적으로 발생한다. 반면 베타 센서는 해당 이벤트들이 반드시 실시간(real time)으로 처리되어야 한다. 센싱 유닛들은 특정 이벤트를 감지하고 이를 마이크로컨트롤러로 보낸다. 그리고 통신부는 이웃 노드들과 통신을 하기 위해 사용되고 데이터를 받으면 이를 마이크로컨트롤러에게 알리고 데이터를 처리할 수 있게 한다.

2.2 기존의 연구 : Duty cycle기반 알고리즘

Duty cycle 기반 알고리즘은 일반적으로 일정 시간동안 sleep하였다가 잠시 깨어나서 해당하는 이벤트에 대응하는 일을 수행하는 방식이다. 센서 네트워크에서는 센서 노드의 소비 에너지를 줄이기 위한 기법으로 주로 duty cycle을 사용한다.

그림 2는 duty cycle의 동작을 설명하고 있다. 노드가 깨어나서 sleep하였다가 다시 깨어나기 전까지를 하나의 사이클이라고 하자. 이 사이클은 일정하게 반복하게 된다. 여기서 노드가 데이터를 처리하거나 이웃 노드와 통신을 하기 위해 깨어나는 것은 한 사이클에서 1%로 가정한다. 이러한 duty 사이클은 소비 에너지를 줄이기 위해 사용하고 있지만 여기에서도 불필요한 에너지 낭비가 발생하게 된다. 그림 2에서 보는 바와 같이 처리할 태스크가 없음에도 불구하고 모든 컴포넌트를 켬으로써 에너지 낭비가 발생하는 문제점이 발생한다.

2.3 기존의 연구 : Greedy 알고리즘[3, 4]

OS 레벨에서의 전원 소비 관리 기법에 대한 기존의 연구사례는 매우 드문데, Sinha 등은 Greedy 알고리즘이라고 불리는 방식을 제안하였다[3]. 그림 3은 불연속적인 이벤트가 발생할 경우 Greedy 알고리즘에서의 노드 상태를 나타내고 있다. 이 알고리

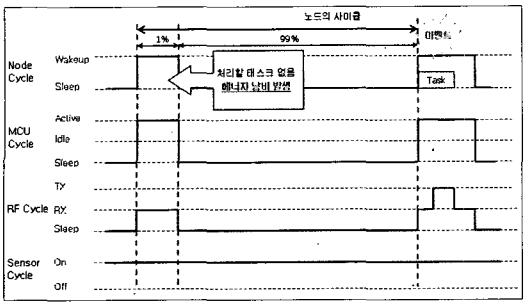


그림 2. Duty cycle 기반 알고리즘

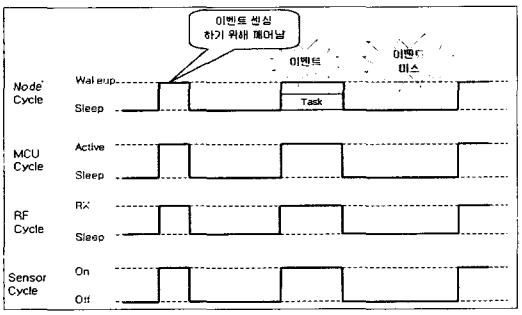


그림 3. Greedy 알고리즘

즘은 이벤트 발생 시간을 예측하여 센서 노드가 깨어나 이벤트를 감지하고 해당 데이터를 처리하는 방식을 사용한다. 하지만 이러한 기법은 랜덤 이벤트 발생에 대해서는 제대로 예측을 하지 못하며, 그 결과 많은 이벤트 미스(miss)가 발생하고, 불필요하게 자주 깨어나기 때문에 에너지 낭비가 발생하게 된다.

그림 3은 Greedy 알고리즘을 설명하고 있다. 각각은 컴포넌트의 상태를 나타내며 이벤트를 감지하기 위해 깨어나고 이벤트를 감지하지 못하면 다시 sleep 상태가 된다. 그리고 이벤트를 감지하면 해당 이벤트를 처리하고 sleep 상태로 전환한다. 만약 sleep하고 있을 때 이벤트가 발생하게 되면 해당 이벤트에 대한 미스가 발생하게 된다. 그리고 센서 노드가 언제 깨어날지 모르기 때문에 이웃 노드와의 통신에 문제가 발생하게 되어 현재의 센서 네트워크에 적용하기에는 무리가 있다.

Ⅲ. 에너지 효율을 위한 스케줄링 기법

3.1 센서노드를 위한 상태 모델

표 1은 duty 사이클에서 발생하는 에너지 낭비를 최소화하기 위한 센서 노드 시스템 상태 모델이다. 이 시스템 상태 모델은 노드의 소비 에너지를 최소

표 1. 시스템 상태 모델

State	마이크로컨트롤러	RF	α	β	소비전력
S1	Active	Rx	On	On	20.4
S2	Active	Rx	Off	On	15.9
S3	Idle	Rx	Off	On	11.1
S4	Power-Save	Rx	Off	On	8.0
S5	Power-Save	Power -down	Off	On	0.6

화할 수 있는 상태를 제안한 것이다. 이 상태 모델은 센서 노드의 컴포넌트들의 다양한 소비 전력에 대한 상태 조합으로 구성 되어있고 소비 전력은 MICA2의 각 컴포넌트들[5-10]을 참조하였다. S1은 알파 센서와 관련된 작업을 하기 위해 사용하는 모델로 가장 많은 소비 전력을 보이고 있다. S2는 작업을 처리하거나 이벤트 발생을 대기하고 있는 모델이며, S3과 S4는 처리할 태스크가 없을 경우 선택될 수 있는 모델이다. 그리고 마지막으로 S5는 sleep 사이클에서 사용하는 모델로 가장 적은 소비 전력을 보이고 있다.

이러한 상태모델을 센서 노드에 적용하면 duty cycle기반 알고리즘은 그림 4와 같이 동작한다. 우선 처리할 태스크가 없다면 마이크로컨트롤러를 idle이나 sleep 상태로 변경하여 소비 에너지를 줄일 수 있다. 그리고 이벤트가 감지되어 처리해야 할 태스크가 발생한다면 이를 처리하기 위해 마이크로컨트롤러를 active 상태로 변경하여 태스크를 처리하게 된다. 그리고 얻어진 데이터를 이웃 노드에게 전송하기 위해 RF를 Tx모드로 변경하여 데이터를 전송하게 된다. 그리고 태스크 완료 후 idle time이

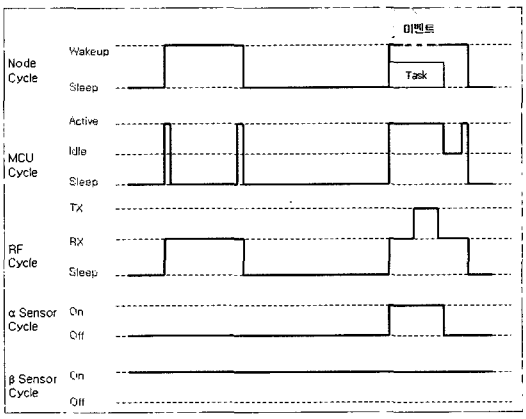


그림 4. 수정된 duty cycle 기반 알고리즘

존재한다면 다시 마이크로컨트롤러의 상태를 IDLE 나 sleep 상태로 변경 할 수 있다. 그리고 알파 센서는 관련 태스크 실행시 전원을 켜고 태스크를 완료하게 되면 알파 센서의 전원을 끔으로써 에너지 낭비를 막을 수 있다.

3.2 시스템 상태 변경 오버헤드 및 상태 변화

그림 5은 상태 변경 시 발생하는 지연 시간을 나타낸다. 지연 시간은 태스크 실행의 시작을 지연시키기 때문에 노드의 시스템 상태를 결정할 때에는 이러한 시간을 고려하여야 한다. 즉, 상태 변경을 할 경우와 하지 않을 경우 중 소비 에너지가 적은 상태를 선택하여야 한다.

이것을 수식 (1)과 같이 나타낼 수 있고 이 수식으로 절약되는 소비 에너지를 계산할 수 있다. 그리고 이 수식의 결과 값은 주어진 시간 Ti에 의해 변하기 때문에 Ti를 수식 (2)와 같이 유도할 수 있다. 결국 이 Ti 의 값은 노드의 상태를 변경하기 위해 필요한 최소 시간을 의미하고 있으며 이것을 사용하여 최적의 노드 상태를 결정할 수 있다.

수식 (1)과 (2)의 모델을 사용하였을 경우 시스템의 상태 변화가 세 종류로 일어날 수 있다.

첫 번째는 알파 센서와 관련된 태스크를 위한 상태 변화로 센서 노드가 깨어나서 알파 센서와 관련된 태스크가 발생하면, 노드 상태를 S1로 변경하게 되는 것이다. 실행할 태스크가 없는 경우 S2, S3,

$$E_{save,j} = P_1 T_i - \frac{P_1 + P_j}{2}(\tau_{d,j} + \tau_{u,j}) - P_j(T_i - \tau_{d,j} - \tau_{u,j})$$
$$= (P_1 - P_j)T_i - \left(\frac{P_1 - P_j}{2}\right)\tau_{d,j} - \left(\frac{P_1 - P_j}{2}\right)\tau_{u,j} \quad (1)$$

$$T_i \geq \frac{1}{2}\left[\tau_{d,k} + \left(\frac{P_0 + P_k}{P_0 - P_k}\right)\tau_{u,k}\right] \quad (2)$$

그리고 S4중 소비 에너지를 줄일 수 있는 상태로 변경한다. 그리고 이벤트가 감지 될 경우 이를 처리하기 위해 노드의 상태를 S2로 변경한다. S3과 S4는 상태 변경으로 발생하는 지연 시간으로 해당 태스크를 실행하는데 지연시간이 발생하게 된다(그림 6).

두 번째 경우는 센서 노드가 깨어나서 처리할 태스크가 없는 경우로 일정 시간을 대기한 후 아무런 이벤트가 발생하지 않으면 노드의 상태를 S2, S3, S4중 소비 에너지를 줄일 수 있는 상태로 변경한다 (그림 7).

세 번째 경우는 노드가 S3, S4인 상태에서 duty

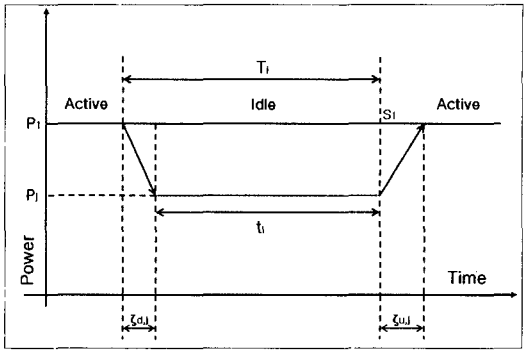


그림 5. 상태 변경에 따른 지연 시간 발생

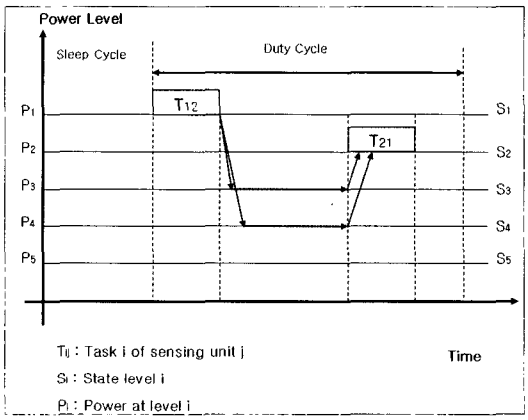


그림 6. 센서 노드 상태 변화 기법 (1)

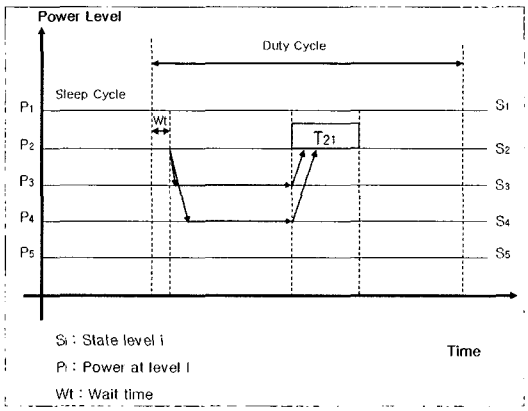


그림 7. 센서 노드 상태 변화 기법 (2)

사이클이 끝나고 sleep 상태인 S5로 들어가기 위한 과정을 설명하고 있다. 이와 같이 노드가 sleep 상태로 들어가기 위해서는 노드 상태 변경을 할 때 타이머 인터럽트가 duty 사이클이 끝나는 시간에 인터럽트를 발생하도록 설정한다. 타이머 인터럽트를 사용하여 노드의 상태를 S2로 변경 후에 S5로

변경하는 이유는 다른 컴포넌트들을 제어하기 위해서는 CPU가 필요하기 때문이다(그림 8).

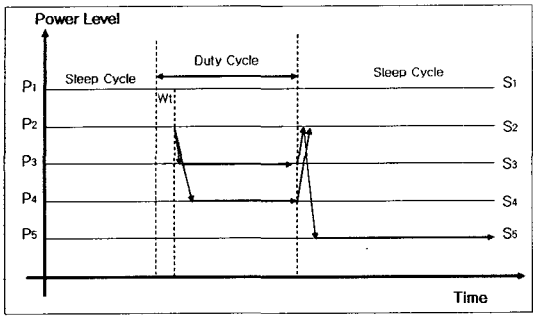


그림 8. 센서 노드 상태 변화 기법 (3)

3.3. EA-SENTAS(Energy-Aware Sensor Node Task Scheduling)

3.1, 3.2절에서 언급한 내용을 토대로 본 논문에서는 EA-SENTAS라고 부르는 새로운 알고리즘을 제안한다. 제안하는 알고리즘은 크게 두 부분으로 나눠진다.

하나는 duty 사이클이고 나머지 하나는 sleep 사이클이다. 센서 노드가 duty 사이클일 때에는 ready-list에서 태스크를 선택하여 실행한다. 그리고 태스크 실행 중에 이벤트가 감지되어 인터럽트가 발생하게 되면 현재 실행하고 있는 태스크와 새롭게 생성된 태스크 중 우선순위가 높은 것을 선택하여 실행하게 된다. 그리고 처리할 태스크가 존재하지 않으면 decide_state 함수를 호출하여 노드의 상태를 변경한다. Decide_state 함수는 우선 duty 사이클이 끝날 때까지의 시간을 계산하여 시스템 상태 결정 기법에 따라 노드의 상태를 S2에서 S4중 적절한 상태로 변경시킨다. 그리고 duty 사이클이 끝나는 시점으로 타이머 인터럽트를 발생시켜 sleep 상태로 진입할 수 있게 한다.

```
BEGIN
 1. do forever
 2. if(DUTY_CYCLE || READY_LIST에 실행되기를 기다리는
      태스크가 존재)
  1) READY_LIST에서 실행될 태스크를 하나 선택하여 태스크 실행
  2) if( 이벤트 감지로 인한 인터럽트 발생 )
     (1) if(새로운 태스크의 우선순위가 현재 태스크 보다 높다)
           새로운 태스크를 READY_LIST 제일 앞에 삽입
           새로운 태스크 실행
     (2) else
           새로운 태스크를 우선순위에 맞게 READY_LIST에 삽입
           기존 태스크 실행
  3) if(READY_LIST에 대기 중인 태스크가 없음)
     (1) 일정 시간대기
     (1) decide_state()
 3. else
  1) Rf, 마이크로컨트롤러 sleep 모드로 변경
  2) sleep during SLEEP_CYCLE
 4. end do
END
```

```
Function decide_state()

BEGIN
 1. Duty 사이클이 끝날 때까지의 delay 시간 계산
 2. delay 시간과 시스템 상태 결정 기법(수식 2)을 사용하여 변경
    가능한 상태 검사
    1) S4로 상태 변경 검사
       (1) 마이크로컨트롤러를 sleep모드로 변경
    2) S3로 상태 변경 검사
       (1) 마이크로컨트롤러를 idle 모드로 변경
    3) S2로 상태 변경 검사
       (1) 마이크로컨트롤러는 active 모드
       (2) α 센서 off
 3. Duty 사이클이 끝나는 시점으로 타이머 인터럽트를 발생
    1) 센서 노드 상태를 S2로 변경,
       마이크로컨트롤러를 active 모드로 변경
    2) 센서 노드 상태를 S5로 변경,
       Rf, 마이크로컨트롤러를 sleep 모드로 변경
END
```

EA-SANTAS의 동작 : decide_state()

Ⅳ. 실험 및 성능 평가

본 논문에서 제시한 알고리즘은 MICA2를 대상으로 하였고, PARSEC 시뮬레이터를 이용하였다. 사용된 이벤트들은 랜덤으로 생성한 이벤트, 확률 분포로 생성한 이벤트, 그리고 확률 분포와 랜덤으로 생성한 이벤트들의 조합으로 구성하였다. 랜덤으로 생성한 이벤트와 확률 분포로 생성한 이벤트를 한 시간에 50~1000개를 발생시켰을 경우와 이 두 가지의 이벤트를 조합하여 랜덤으로 생성한 이벤트의 비율을 10~90%로 변경하였을 경우에 대하여 실험하였다. 실험에서 비교할 알고리즘들은 greedy 알고리즘, duty 사이클, EA-SENTAS이다. Duty 사이클과 EA-SENTAS는 항상 센서를 켜 놓고 있으나 greedy 알고리즘은 노드가 깨어날 경우에만 센서를 켜 놓는다. 그래서 보다 나은 실험을 하기 위하여 센서를 항상 켜 놓은 greedy 알고리즘을 같이 비교 실험하였다. Duty 사이클은 전체 사이클의 1%이고 전체 사이클은 100초라 가정하여 실험하였다.

그림 9와 그림 10은 확률적 주기성 이벤트를 발생시킨 경우에 대한 실험 결과이다. 에너지 소비가 가장 큰것은 duty 사이클이고, EA-SENTAS는 가장 적은 에너지 소비를 보이고 있다. 이벤트 수가 700개를 넘어서면서 EA-SENTAS의 에너지 소비량이

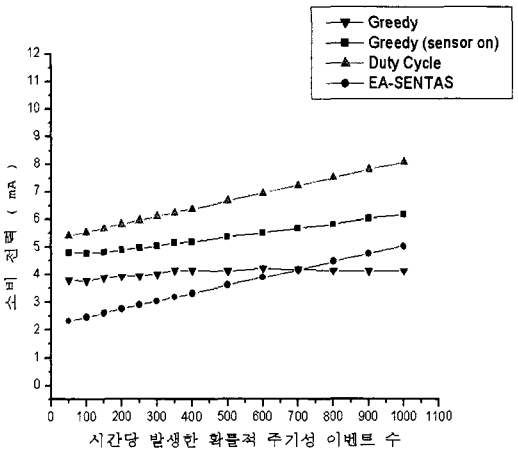


그림 9. 확률적 주기성 이벤트 수에 따른 소비전력

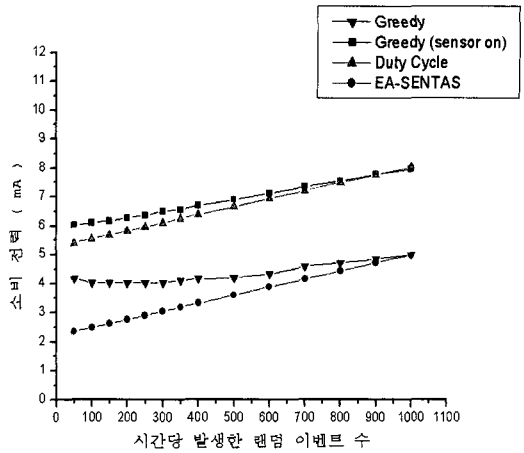


그림 11. 랜덤 이벤트 수에 따른 소비 전력

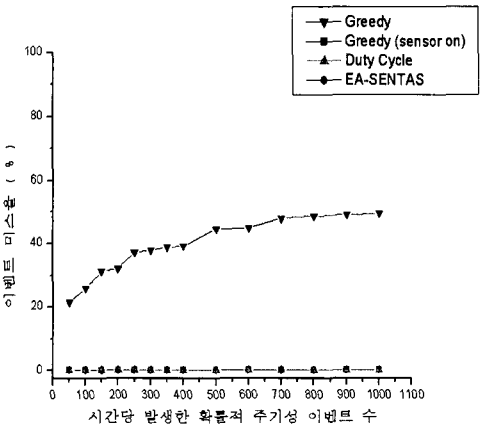


그림 10. 확률적 주기성 이벤트 수에 따른 이벤트 미스율

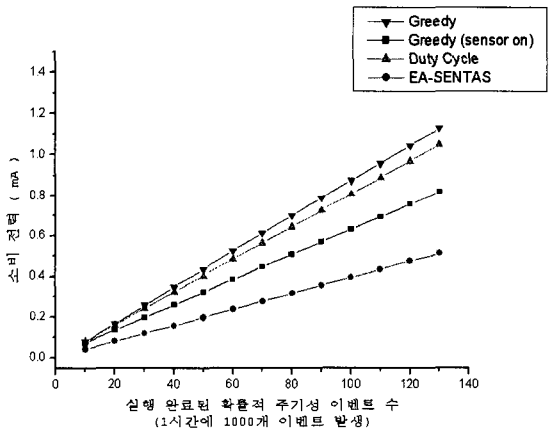


그림 15. 실행 완료된 확률적 주기성 이벤트 수에 따른 소비 전력

greedy보다 많아지는데 EA-SENTAS의 처리 태스크 수가 월등히 많아지기 때문이다. EA-SENTAS와 duty 사이클의 이벤트 미스는 거의 없으며 greedy는 최소 20%, 최대 50%의 이벤트 미스율을 보이고 있다. 발생 이벤트 수가 많아져도 EA-SENATS의 이벤트 미스율은 크게 증가하지 않는 것으로 보아 상태 전위 오버헤드가(약 4ms) 시스템에 큰 영향을 미치지 않는 것을 알 수 있다.

그림 11과 그림 12는 랜덤 이벤트를 발생시켰을 때의 실험이며, greedy (sensor on)의 에너지소비가 가장 큼을 알 수 있다. 이는 이벤트가 랜덤하게 생성되었을 경우 예측을 제대로 하지 못하여 불필요하게 많이 깨어나기 때문이다. 이벤트 발생 수가 1000개일 때 EA-SENTAS와 greedy 알고리즘은 거의 같은 소비 에너지를 보이고 있으나 처리 이벤트 수는 EA-SENTAS가 greedy보다 2배 많다. Greedy의 소비 에너지가 크게 증가하지 않는 것은 높은 이벤트 미스율로 인해 실제 처리 이벤트수가 많이 증가하지 않기 때문이다. 그림 10과 비교하면 Greedy 알고리즘의 미스율은 확률적 주기성 이벤트의 경우보다 높음을 알 수 있다.

그림 13과 그림14는 랜덤 이벤트와 확률적 주기성 이벤트를 혼합하여 300개의 이벤트를 만들어 랜덤 이벤트의 비율에 따른 소비 전력을 나타낸다. Greedy (sensor on)은 랜덤 이벤트 수가 증가함에 따라 소비 전력이 증가한다. EA-SENTAS와 duty 사이클은 이벤트의 종류가 소비 전력에 영향을 미치지 않고 있다. Greedy 알고리즘의 이벤트 미스율이 랜덤 이벤트의 수가 증가 될수록 이벤트의 발생 시간을 예측하기가 힘들어져 증가하게 된다.

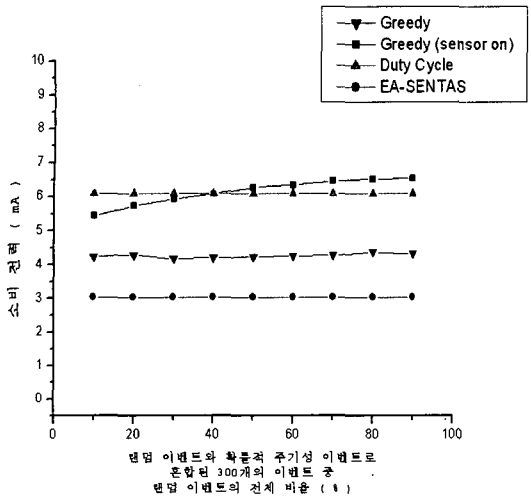


그림 13. 랜덤 이벤트 비율에 따른 소비 전력

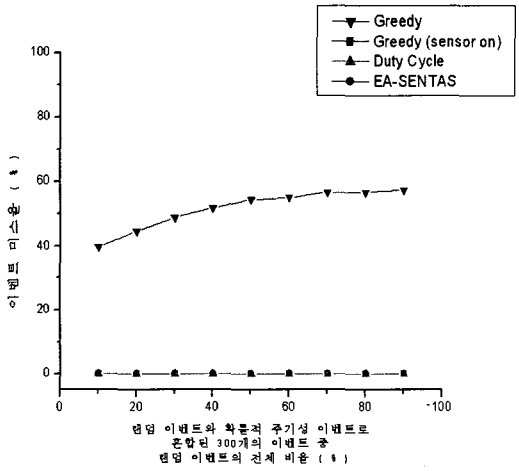
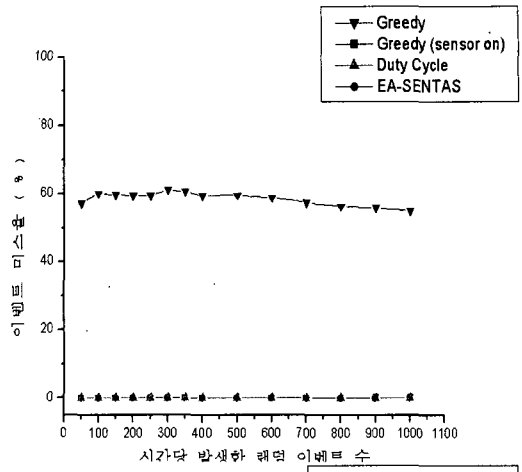


그림 14. 랜덤 이벤트 비율에 따른 이벤트 미스율

그림 15는 확률적 주기성 이벤트를 한 시간에 1000개 발생시켜서 작업이 완료된 이벤트 수에 따른 소비 전력을 나타내고 있다. Greedy 알고리즘의 소비 에너지가 가장 높으며, EA-SENTAS가 가장 적은 소비 전력을 보여주고 있다. 그림 16은 랜덤 이벤트를 한 시간에 1000개 발생시켜서 실행 완료된 이벤트 수에 따른 소비 전력을 보여주고 있다. 실행 완료된 이벤트 수에 따른 소비 전력은 greedy 알고리즘이 가장 높게 나타나고 있음을 보여준다. Duty 사이클은 그 다음으로 높은 소비 전력을 나타내고 있으며, EA-SENTAS는 가장 적은 소비 전력을 보이고 있다.



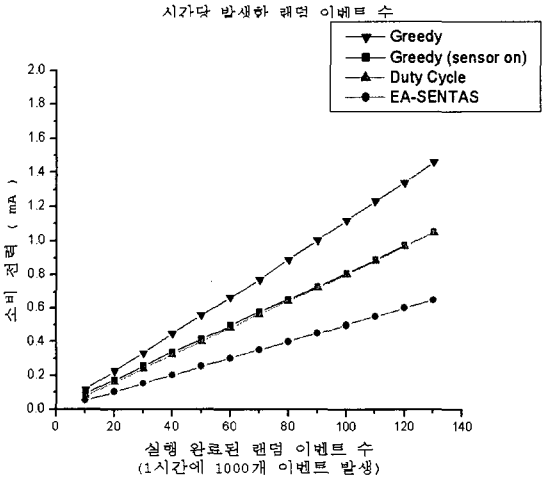


그림 16. 실행 완료된 랜덤 이벤트 수에 따른 소비 전력

V. 결 론

본 논문에서는 무선 센서네트워크에서 센서노드의 에너지 소비를 줄일 수 있는 새로운 태스크 스케줄링 기법을 제안하였다. 제안하는 기법은 처리할 태스크가 없을 때 노드의 상태를 변화시켜 에너지 소비를 줄이는 방법으로서 기존의 방식에 비해 에너지 효율적인 방법이다. 제안하는 EA-SENTAS는 기존의 duty cycle 기반 알고리즘보다 56%, greedy 알고리즘보다 40%의 에너지 이득이 있음을 실험을 통해서 알 수 있었다.

향후에는 실제 센서 네트워크상에서 이웃 노드와 함께 동작할 경우 발생할 수 있는 문제점을 보완하는 등 추가적인 연구가 진행 되어야 할 것이다.

참 고 문 헌

[1] L.Doherty, B.A. Warneks, B.E. Boser, K.S.J Pister, "Energy and Performance Considerations for Smart Dust," International Journal of Parallel Distributed Systems and Networks, 2001.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey," Computer Networks, vol. 38, pp. 393-422, 2002.

[3] A. Sinha and Chandrakasan, "Dynamic Power Management in Wireless Sensor Network", in Proc. IEEE Design & Test of Computers, March-April 2001, pp. 62-74

[4] A. Sinha and A.Chandrakasan, "Operating System And Algorithmic Techniques for Energy Scalable Wireless Sensor Networks," in Proc, of Mobile Data Management, Second International Conference, MDM 2001, Hongkong, Jan. 8-10, 2001.

[5] Crossbow Technology, Inc. Mica2 datasheet, 2003.

[6] Atmel Corp., ATmega128L Datasheet, 2004.

[7] Atmel Corp., AT45DB041B Datasheet, 2003.

[8] Chipcon AS, SmartRF CC1000 Datasheet, 2004.

[9] Crossbow Technology, Inc., MPR / MIB User's Manual, 2004.

[10] Crossbow Technology, Inc., MTS/MDA Sensor and Data Acquisition Board User's Manual, 2005.

[11] PARSEC User Manual For PARSEC Release 1.1, 1999

이 진 호 (Jinho Lee) 준회원

2003년 2월 경성대학교 컴퓨터공학과 학사

2006년 2월 부산대학교 컴퓨터공학과 석사

2006년 2월~현재 LG전자 MC 사업본부

<관심분야> RF, 임베디드 시스템, RTOS

최 훈 (Hoon Choi) 준회원

2005년 2월 : 부산대학교 컴퓨터공학과 학사

2005년 2월~현재 : 부산대학교 컴퓨터공학과 석사

<관심분야> 센서 네트워크, 임베디드 시스템

백 윤 주 (Yunju Baek) 종신회원

1990년 2월 한국과학기술원 전산학과 학사 졸업

1992년 2월 한국과학기술원 전산학과 석사

1997년 2월 한국과학기술원 전산학과 박사

1999년~2002년 NHN 기술연구소 소장

2003년~현재 부산대학교 컴퓨터공학과 조교수

<관심분야> 임베디드시스템, 센서네트워크, 컴퓨터구조