무선 센서 운영체제를 위한 지능형 슬랩 할당기

(A Smart Slab Allocator for Wireless Sensor Operating Systems)

민 홍 † 이 상 호 †
(Hong Min) (Sangho Yi)
허 준 영 † 김 석 현 †
(Junyoung Heo) (Seokhyun Kim)
조 유 근 †† 홍 지 만 †††
(Yookun Cho) (Jiman Hong)

요 약 무선 센서 네트워크에서 사용하는 동적 메모리 관리 기법들은 범용 시스템에서 사용되고 있는 기법들을 그대로 적용한 것들이 많기 때문에, 센서 응용에는 부적합한 부분이 있다. 본 논문에서는 센서 응용들의 특성을 살펴보고, 이들의 특성을 모델링함으로써, 기존의 동적 메모리 관리 시스템에서 발생할 수 있는 긴 수행시간과 불필요한 메모리 관리 공간의 문제를 해결할 수 있는 슬랩 할당기를 제안한다. 또한 대표적인 센서 응용 프로그램을 활용한 실험을 통해서 새로이 제안한 방법의 성능을 기존의 시스템과 비교 평가한다.

키워드 : 무선 센서 운영체제, 동적 메모리 관리, 성능평가

· 본 연구는 숭실대학교 교내연구비 지원과 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음(IITA-2008-C1090-0803-0006)
· 이 논문은 제34회 추계학술대회에서 '무선 센서 운영체제를 위한 슬랩 할당기'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 서울대학교 컴퓨터공학부 박사과정
hmin@os.snu.ac.kr
shyi@os.snu.ac.kr
jyheo@os.snu.ac.kr
shkim@os.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
ykcho@os.snu.ac.kr
††† 종신회원 : 숭실대학교 컴퓨터공학부 교수
jiman@ssu.ac.kr
(Corresponding author임)

논문접수 : 2007년 12월 4일
심사완료 : 2008년 5월 28일

Copyright@2008 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.

Abstract Existing dynamic memory allocation schemes for general purpose operating system can not directly apply to the wireless sensor networks (WSNs). Because these schemes did not consider features of WSNs, they consume a lot of energy and waste the memory space caused by fragmentation. In this paper, we found features of WSNs applications and made the model which adapts these issues. Through this research, we suggest the slab allocator that reduces the execution time and the memory management space. Also, we evaluate the performance of our scheme by comparing to one of the previous systems.

Key words : Sensor operating system, Dynamic memory management, Performance evaluation

1. 서 론

무선 센서 네트워크 분야는 최근 들어 하드웨어 시스템의 급속한 발달과 그 응용 분야의 다양성과 특수성 때문에 많은 연구가 수행되고 있다[1]. 국토의 반 이상이 산불로 인해 재로 변한 그리스의 대형 화재[2]는 무선 센서 네트워크를 통한 재난 방지 시스템의 도입이 얼마나 필요한지를 극명하게 보여주고 있다.

일반적으로 무선 센서 네트워크는 수백에서 수천 개의 센서 노드들로 구성된다[3]. 센서 노드들은 생산 비용에 민감하게 영향을 받기 때문에, 비용 효율적으로 설계되어야 한다. 따라서 센서 노드는 낮은 성능의 연산 장치를 가지고 있으며 극단적으로 작은 메모리와 배터리에 의해 전원이 공급되는 특징을 가지고 있다. 게다가, 무선 센서 네트워크는 저 성능의 프로세서를 사용하여, 전력 소모를 최대한 줄이고, 제한된 자원을 최대한 사용해야 하는 제약성을 가지고 있다. 그로 인해, 무선 센서 네트워크를 위한 운영체제는 적은 에너지 소비와 효율적인 자원의 사용이라는 측면을 모두 만족시켜야 한다. 이러한 이유 때문에 시간과 공간의 효율성을 고려한 메모리 관리 시스템이 필요하다.

본 논문에서는 기존의 동적 메모리 관리 시스템에서 간과 했던 메모리 병합과 분리 작업으로 발생하는 추가적인 오버헤드를 줄이고, 각각의 센서 응용에 따라 최적화된 작업을 수행하는 동적 메모리 관리 시스템을 제안한다. 새롭게 제안된 센서 운영체제를 위한 슬랩 할당기는 기존의 동적 메모리 시스템 환경하에서 쉽게 추가 할 수 있고, 응용의 메모리 할당 패턴을 분석하여, 이를 적용함으로써 특정 응용 프로그램에 최적화된 동적 메모리 할당 및 해제를 수행할 수 있다는 장점을 가지고 있다.

본 논문은 다음과 같이 구성되어 있다. 1장의 간략한 서론에 이어, 2장에서는 동적 메모리 할당에 대한 기존의 연구들을 살펴본다. 3장에서는 센서 응용프로그램들

을 각각의 특성에 따라 분류해 보고, 이들의 동적 메모리 패턴에 대한 분석 모델 기법을 제시한다. 4장에서는 3장에서 분석한 응용 프로그램의 패턴 정보를 바탕으로 슬랩 할당기를 구성하고 이를 기존의 시스템에 적용하는 과정에 대해 서술한다. 5장에서는 성능 평가를 통해 본 논문에서 제안된 슬랩 할당기의 유용성을 검증하고, 6장에서는 결론으로 마무리한다.

2. 관련 연구

본 장에서는 무선 센서 운영체제에서 사용하고 있는 동적 메모리 할당 기법에 대한 기존 연구들을 살펴본다.

2.1 TinyOS

TinyOS는 버클리 대학에서 개발한 센서 운영체제로 가장 널리 사용되고 있다[4]. TinyOS는 이벤트 기반 방식으로 수행되며, 메모리 관리시스템에 있어서의 가장 큰 특징은 정적으로 메모리를 관리하기 때문에 동적인 메모리 할당을 사용하지 않는 것이다[5].

2.2 SOS

SOS는 UCLA에서 2005년에 개발한 센서 운영체제로 모듈화를 통해서 시스템 및 응용 프로그램의 개발을 빠르게 할 수 있도록 했다[6]. 각 크기 별로 배열을 사용하여 응답시간을 높이는 장점이 있지만 내부 단편화 비율이 크고, 여분의 공간이 있음에도 불구하고 필요한 크기에 대해 연속적인 할당 요청이 있을 때, 메모리를 할당 받지 못하는 문제가 있다.

2.3 MANTIS OS

MANTIS OS는 콜로라도 대학에서 멀티 쓰레드 기반의 센서 운영체제이다[7]. 태그와 리스트를 사용하여 빈 공간을 관리하고, best-fit 정책을 통해 가장 유사한 크기의 빈 공간을 찾아서 할당한다. 검색 시간이 길다는 것과 매번 메모리를 할당 할 때 마다 불필요한 태그 발생으로 메모리가 낭비된다는 문제가 있다. 또한 메모리 할당 시 필요한 여분의 메모리 분리와 메모리 해제 시 필요한 인접 공간의 병합 작업에 대한 오버헤드가 있다.

2.4 TLSF

TLSF는 스페인의 발렌시아공업대학에서 실시간 시스템을 지원하기 위해 고안한 동적 메모리 할당 기법이다[8]. 메모리 할당과 해제를 O(1)에서 수행하기 위하여 크기 별로 클래스를 나누고, 배열을 사용하여 각 레벨의 빈 공간의 정보를 저장함으로써 검색의 시간을 줄인다는 장점이 있지만, 응용의 특성을 반영하지 못하기 때문에 불필요한 메모리 병합과 분리작업을 추가로 해야 한다는 단점이 있다[9].

3. 센서 응용 프로그램 분석

센서 노드를 위한 응용 프로그램의 동적 메모리 할당

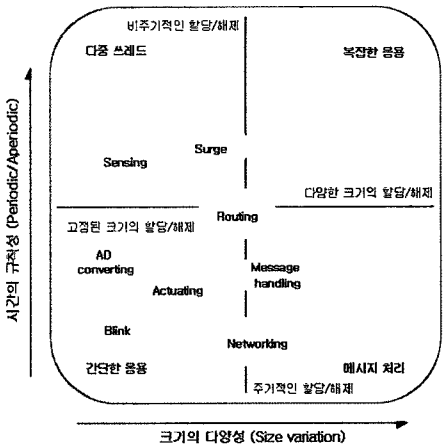


그림 1 동적 메모리 시스템 관점에서의 센서 응용 분류

패턴을 분석하는 것은 효율적인 동적 메모리 할당 시스템을 설계하기 위한 중요한 기반이 된다. 이는 센서 응용 프로그램이 일반 범용 프로그램과는 다른 독특한 특성을 가지고 있기 때문이다.

무선 센서 네트워크에서 응용프로그램은 동적 메모리 관리 시스템 관점에서 다음과 같이 분류할 수 있다.

가로축은 동적 메모리 요청의 크기가 얼마만큼 다양한지에 대한 기준으로 오른쪽으로 갈수록 요구하는 크기가 다양하다는 것을 의미한다. 세로축은 동적 메모리 요청이 얼마나 주기적으로 발생하는지에 대한 기준을 나타낸 것으로, 위쪽으로 갈수록 비주기적으로 메모리 요청이 발생한다는 것을 보여준다.

대부분의 센서 응용 프로그램은 메모리 할당과 해제 작업에 있어서 일정한 크기의 메모리를 주기적으로 할당한다는 것을 알 수 있다. 기존의 동적 메모리 할당 기법들은 이러한 동적 메모리 요구의 패턴에 대한 분석 없이, 범용 시스템에 기반하여 설계가 이루어졌기 때문에, 수행시간과 메모리 활용에 있어서 문제를 드러내고 있다. 따라서 이러한 정보를 충분히 수집할 수 있는 모델이 필요하고, 그 결과를 시스템에 적용하는 새로운 접근 방법이 필요하다.

4. 슬랩 할당기

본 장에서는 앞에서 언급한 센서 응용 프로그램이 갖는 특징을 적절히 활용할 수 있는 무선 센서 운영체제를 위한 슬랩 할당기를 제안한다. 새로운 슬랩 할당기 설계 과정과 기존의 동적 메모리 할당 시스템에 적용하는 방법을 중심으로 설명한다.

4.1 시스템 요구 사항

센서 노드에서 동작하는 시스템을 설계하기 위해서는 다음과 같은 요구 사항을 고려해야 한다[10].

• 효율적인 자원 사용

센서 노드는 자원의 제약성을 갖기 때문에, 새로운 시스템을 적용할 때 과도한 메모리 공간을 사용해서는 안됨

• 빠른 수행시간

슬랩 할당기를 통해 얻을 수 있는 가장 큰 효용은 불필요한 작업의 감소로 응답시간을 단축시키는 것임

• 구현의 간편성

단순한 시스템 설계를 통해 구현 및 적용이 쉬워야 함

• 시스템 안정성

새로운 시스템 적용으로 인해 시스템의 안정성이 위협받아서는 안됨

본 논문에서 새롭게 제안한 슬랩 할당기도 앞에서 언급한 요구사항들을 고려하여 설계하였다.

4.2 슬랩 할당기 설계 과정

응용 프로그램의 동적 메모리 할당 및 해제 패턴을 분석하고 이를 슬랩 할당기 시스템에 적용하기 위해서 다음과 같은 과정을 필요로 한다.

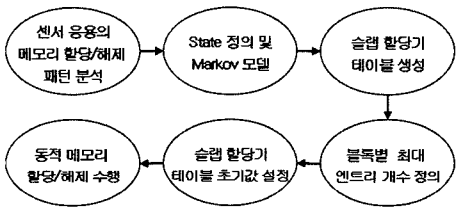


그림 2 제안된 슬랩 할당기 시스템 설계 과정

먼저, 응용 프로그램의 동적 메모리 할당과 해제 패턴을 분석하기 위해서, 응용프로그램 수행과정의 일부를 샘플링 한다. 다음으로 Markov 모델을 사용하기 위하여 요청의 크기에 따라 상태를 정의하고 슬랩 할당기의 테이블을 생성한다. Markov 모델과 슬랩 할당기의 테이블 구조는 다음 절에서 자세히 살펴본다. 응용에 적합한 블록 크기 별 최대 엔트리 개수를 모델을 통해 도출하고, 슬랩 할당기를 초기화한 후에 동적 메모리 시스템에 추가한다.

4.3 Markov 모델

Markov 모델을 통한 시스템 설계 과정을 예제를 통해 알아본다. R/F 통신 프로그램은 무선센서 네트워크에서 사용하는 가장 일반적은 응용으로, 일정 주기마다 카운터 값을 하나씩 증가 시키면서, 카운터의 값을 패킷의 데이타 영역에 저장하여 라디오 모듈을 통해 전송하는 프로그램이다. 메시지 처리 영역에 속하는 네트워크 응용으로, 주기적이면서 전송하는 패킷의 크기에 따라 동적으로 생성할 블록의 크기도 달라진다. Zigbee 프로토콜을 사용하기 때문에 최대 89byte까지 데이타를 전송할 수 있지만, 이 응용에서는 16byte (8byte의 헤더, 8byte의 데이타)를 고정적으로 할당한다.

다음 표는 R/F 통신 프로그램의 블록 크기 별 상태 변화를 정리한 것이다.

표 1 메모리 할당/해제 패턴

할당/해제	+	+	-	+	-	+	-	+	+	-
문자열	A_1	A_2	A_1	A_2	A_1	A_2	A_1	A_2	A_3	A_2
할당/해제	-	+	+	-	+	-	+	+	-	-
문자열	A_1	A_2	A_3	A_2	A_3	A_2	A_3	A_4	A_3	A_2

(a) 128 bytes

할당/해제	+	+	-	-	+	-	+	-	+	-
문자열	B_1	B_2	B_1	B_0	B_1	B_0	B_1	B_0	B_1	B_0

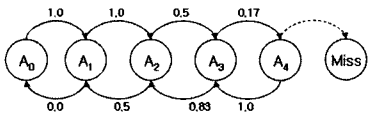
(b) 16 bytes

+ (할당) / - (해제)

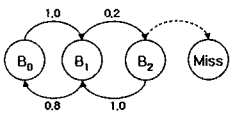
Markov 모델을 적용하기 위해서 상태를 크기에 따른 블록별 메모리 할당 상황으로 정의하면 R/F 통신 프로그램은 다음과 같이 분석할 수 있다[11]. 이 응용 프로그램에서 사용하는 동적 블록의 크기는 각각 128 bytes와 16bytes이다. 각 블록 별로 동적 메모리 할당과 해제 패턴을 분석해 보면 위와 같은 상태 문자열을 확인 할 수 있다. 상태 문자열이 $A_1 \rightarrow A_2$으로 변한 것은 새로운 메모리 할당 요청을 수행하기 위해서 메모리 블록이 2개 필요하다는 것을 의미한다. 반대로 $A_2 \rightarrow A_1$으로 변한 것은 이전에 사용하던 블록이 해제되어서 이제 1개의 블록만 할당되어 있다는 것을 의미한다. 인접한 상태를 괄호로 묶어 보면 (A_0, A_1), (A_1, A_2), (A_2, A_3), …, (A_2, A_3), (A_3, A_2)과 같다. 이것을 바탕으로 상태 전이 확률을 구하고, Markov 상태 그래프를 그려 보면 그림 3과 같다.

128 bytes크기의 블록의 경우 A_3에서 시작하는 총 9개의 문자열 묶음 중에서 7개의 문자열 순서쌍은 A_2 상태로 변환하고, 2개의 문자열 순서쌍은 A_4 상태로 변환한다. 따라서 각각의 전환 확률은 $A_3 \rightarrow A_2$이 78%, $A_3 \rightarrow A_4$이 22%가 된다.

Markov 상태 그래프를 바탕으로 극한 확률은 다음의 공식을 사용해서 구한다[12].



(a) 128 bytes



(b) 16 bytes

그림 3 Markov 상태 그래프

$$\pi_j = \sum_{i=0}^{\infty} \pi_i P_{ij},\ j \geq 0$$

$$\sum_{j=0}^{\infty} \pi_j = 1$$

표 2는 각 블록 크기별 극한 확률을 보여주고 있다.

표 2 블록 크기별 극한 확률

128 bytes		16 bytes	
π_0	0	π_0	0.4
π_1	0.24	π_1	0.5
π_2	0.45	π_2	0.1
π_3	0.27		
π_4	0.04		

128 bytes의 경우 $\pi_4 = 0.04$, 16byte의 경우는 $\pi_2 = 0.1$이기 때문에 슬랩 할당기 테이블의 엔트리 초기값을 각각 4와 2로 설정해주면 응용에 가장 적합한 기능을 수행할 수 있다.

4.4 슬랩 할당기 적용

그림 4는 Markov 모델을 사용하여 R/F 통신 프로그램에 최적화된 동적 메모리 관리 시스템을 보여주고 있다.

슬랩 할당기 테이블은 앞에서 모델링한 결과 값에 따라 각각 16 bytes는 2개의 엔트리를, 128 byte는 4개의 엔트리를 가지고 초기화 된다.

사용자로부터 메모리 할당 요구가 있을 경우, 바로 동적 메모리 할당기로부터 메모리를 할당하는 것이 아니라, 슬랩 할당기 테이블을 참조하여 필요한 메모리 공간을 확보한다. 슬랩 할당기 테이블은 응용 프로그램의 특성에 따라 초기화되기 때문에 대부분의 할당이 동적 메모리 할당기로 넘어가지 않고, 스랩 할당기 레벨에서 처리 된다.

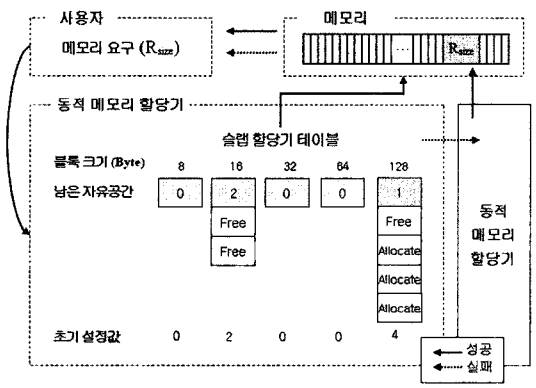


그림 4 슬랩 할당기

5. 성능 평가 및 분석

본 장에서는 MANTIS OS에 슬랩 할당기를 구현하고, 여러 가지 센서 응용 프로그램을 실제로 수행 시켜 보고, 그 결과를 비교함으로써 슬랩 할당기의 성능을 평가한다.

5.1 실험 환경

실험에 사용한 응용 프로그램은 다음과 같다.

- Blink: 센서 노드 상의 LED를 켜고 끄는 응용 프로그램
- CountSender: R/F 모듈을 사용하여 일정 주기마다 카운터 값을 Receiver에게 전송 하는 프로그램
- CountReceiver: R/F 모듈을 사용하여 Sender가 보낸 카운터 값을 전송 받는 프로그램
- Surge: 노드가 측정한 정보를 Base Station으로 전송하는 프로그램
- Routing: 라우팅 테이블을 생성하여 패킷의 전송을 관리하는 프로그램

5.2 실험 결과

다음 그래프(그림 5)는 MANTIS OS에서 동적 메모리 할당기만을 사용할 때와 슬랩 할당기를 적용하여 동적 메모리 요청을 받았을 때의 수행 시간을 비교한 것이다. 각 응용의 메모리 요청 흐름을 분석하여 200회에 걸쳐 할당과 해제를 반복하였다.

간단한 응용에서는 메모리 할당 시 수행하는 분리 작업과 해제 시 수행하는 병합 작업이 빈번하게 이루어지지 않고, 적절한 빈 공간을 검색하는데 필요한 검색 시간이 짧기 때문에, 큰 차이가 발생하지 않지만, 복잡한 응용의 경우에는 슬랩 할당기를 적용한 경우가 약 37% 정도 빠르다는 것을 알 수 있었다.

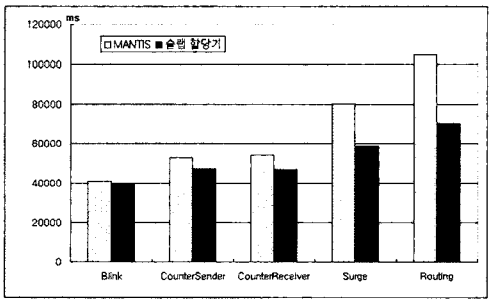


그림 5 수행 시간 비교

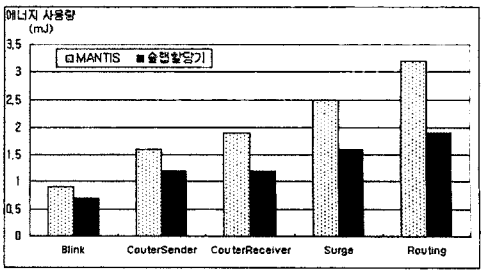


그림 6 에너지 사용량 비교

위 그림 6은 MANTIS의 동적 메모리 관리 부분과 본 논문에서 제안한 슬랩할당기의 에너지 사용량을 비교한 그래프이다. 수행 시간을 비교한 그래프와 마찬가지로 슬랩 할당기를 사용할 경우 복잡한 응용에서 보다 에너지 사용측면에서 효율적이라는 것을 알 수 있다.

다음 표 3은 슬랩 할당기를 통해 동적 메모리 할당을 각각의 센서 응용 프로그램에서 사용했을 때, 미스가 발생하여, 슬랩 할당기 레벨에서 작업을 수행하지 못하고, 기존의 동적 메모리 관리 시스템에서 메모리 할당이 이루어진 확률을 보여주고 있다.

표 3 Miss가 발생할 확률

	Blink	Counter Sender	Counter receiver	Surge	Routing
Miss rate	0%	0%	0.005%	0.005%	0.015%

앞에서도 언급했지만 센서 응용의 경우, 메모리 할당 패턴이 단순하고, 할당하는 크기가 다양하지 않기 때문에 대부분의 메모리 할당과 해제를 예측할 수 있고, 이러한 정보를 활용한 슬랩 할당기를 통해서 99%의 정확성을 가지고 동적 메모리 할당을 수행할 수 있다.

그림 6은 슬랩 할당기의 반영률에 따라서 각 센서 응용프로그램의 수행시간에 어떤 변화가 발생하는지를 나타내주는 그래프이다. 각 비율에 따라 슬랩 할당 테이블의 엔트리 수를 줄여 가면서 수행 시간의 변화를 살펴본 것이다. 각 응용에 따라 반영률에 따라서 수행 시간의 증가에 차이가 있지만 전반적으로 슬랩 할당기의 반영률이 낮아짐에 따라 수행시간이 8%~36% 정도 차이가 발생하는 것을 알 수 있었다.

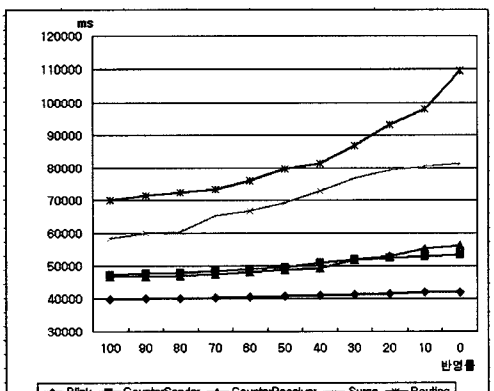


그림 7 슬랩 할당기 반영률에 따른 수행시간

6. 결 론

무선 센서 네트워크 분야는 다양한 분야에서 사용될 수 있고, 실용적인 특성 때문에 많은 연구가 이루어지고 있다. 제한된 자원을 효율적으로 사용하기 위해 많은 연구들이 진행되었고, 많은 발전을 거듭해오고 있다. 그러나 센서 운영체제 상에서 동적 메모리 할당에 대한 기존연구들은 센서 응용 프로그램의 특성에 대한 분석 없이, 기존의 범용 시스템에서 사용하는 기법을 그대로 적용했기 때문에, 불필요한 작업으로 인한 수행시간 증가와 메모리 관리 비용의 증가라는 문제점을 드러내고 있다. 본 논문에서는 센서 응용 프로그램을 그 특성에 따라 분류하고, 각 응용에 적합하도록 설정할 수 있는 슬랩 할당기를 제안하고 있다. 새로운 스랩 할당기는 기존의 시스템 상에 적용하기 쉬울 뿐 아니라, 적절한 모델링 기법을 사용함으로써, 동적 메모리 할당 수행 시간을 약 30%정도 이상 줄일 수 있었다.

참 고 문 헌

[1] A. Mainwaring, "Wireless sensor networks for habitat monitoring," Applications and OS, pp. 88-97, 2002.
[2] 중앙일보 2007년 8월 28일자 기사
[3] W.I. Jeng, "Scalability of a class of wireless sensor networks," Modeling and design of WSN, 2001.
[4] D. Culler, "TinyOS - A Component based operating system for networked sensors," 2000.
[5] P.Levis, "TinyOS: An Operating System for Sensor Networks," 2003.
[6] H. Chih-Chieh, "SOS: A dynamic operating system for sensor networks," The Third International Conference on Mobile Systems, Applications, And Services (Mobisys), pp. 163-176, 2005.
[7] S. Bhatti, "MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms," ACM/Kluwer Mobile Networks & Applications (MONET), Special Issue on Wireless Sensor Networks, pp. 563-579, 2005.
[8] M. Masmano, "Tlsf: a new dynamic memory allocator for real-time systems," Euromicro Conference on Real-Time systems (ECRTS'04), 2004.
[9] P.R.Panda, "Data and memory optimization techniques for embedded systems," Transaction on Design Automation of Electronic Systems (TODAES), pp. 149-206, 2001.
[10] R. Paul, "Dynamic storage allocation: a survey and critical review," International Workshop on Memory Management, 1995.
[11] D. Joseph, "Prefetching using Markov predictors," Transactions on Computers, pp. 121-133, 1999.
[12] S. M. Ross, "Introduction to probability models," fifth edition, 1999.
[13] P.R. Wilson, "Dynamic Storage Allocation: A Survey and Critical Review," 1995.
[14] Crossbow: http://www.xbow.com