공간 제약적인 센서 운영체제를 위한 효율적인 메모리 할당 기법

(An Efficient Memory Allocation Scheme for Space Constrained Sensor Operating Systems)

이 상 호 † 민 홍 † 허 준 영 † 조 유 근 †† 홍 지 만 †††
(Sangho Yi) (Hong Min) (Junyoug Heo) (Yookun Cho) (Jiman Hong)

요 약 무선 센서 네트워크는 자연 환경의 정보를 수집하고, 수집된 정보를 가공하고, 가공된 정보를 무선 통신을 통하여 사용자에게 실시간으로 전달하는 기능을 가진 설비이다. 이러한 무선 센서 네트워크는 수백 혹은 수천 개의 무선 센서 노드들로 이루어지고, 센서 노드의 플랫폼은 비용 효율성 때문에 매우 제한적인 메모리 공간을 지니며 제한적인 배터리로 동작한다. 따라서 이것들을 동작시키는 센서 운영체제는 공간 제약성을 감내할 수 있어야 하고, 에너지 효율적으로 동작해야 전체 센서 네트워크를 효율적으로 동작시킬 수 있게 된다. 본 논문에서는 공간 제약적인 센서 운영체제를 위한 효율적인 메모리 할당 기법을 제안한다. 제안한 기법을 사용하면, 기존 센서 운영체제들에서 사용되었던 메모리 할당 기법들을 사용하는 것보다 메모리 단편화 문제를 감소시킴과 동시에 공간의 효율성을 증진시킬 수 있다. 본 논문의 비교 실험 결과를 통하여 제안한 기법을 사용하는 것이 기존의 방법보다 메모리 단편화를 상당히 줄일 수 있고, 또한 수행 시간도 나빠지지 않음을 보인다.

키워드 : 메모리 할당 기법, 무선 센서 네트워크, 센서 운영체제, 메모리 관리, 에너지 효율성

Abstract The wireless sensor networks are sensing, computing and communication infrastructures that allow us to monitor, instrument, observe, and respond to phenomena in the harsh environment. Sensor operating systems that run on tiny sensor nodes are the key to the performance of the distributed computing environment for the wireless sensor networks. Therefore, sensor operating systems should be able to operate efficiently in terms of energy consumption and resource management. In this paper, we present an efficient memory allocation scheme to improve the time and space efficiency of memory management for the sensor operating systems. Our experimental results show that the proposed scheme performs efficiently in both time and space compared with existing memory allocation mechanisms.

Key words : Memory Allocation Mechanism, Wireless Sensor Network, Sensor Operating Systems, Memory Management, Energy Efficiency

1. 서 론

최근의 급속한 컴퓨터 시스템 설계 기술 및 무선 통신 기술의 발달은 매우 가볍고 효율적인 컴퓨팅 하드웨어의 발전을 가져오고 있다. 이러한 하드웨어의 급속한 발전을 바탕으로 무선 센서 네트워크 기술을 실생활에서 사용하는 것이 가능해졌고, 무선 센서 네트워크는 연구 분야로서 및 실생활의 사용에서 매우 인기를 끄는 분야가 되었다. 무선 센서 네트워크는 자연의 여러 가지 정보를 센싱하고, 무선으로 통신하며, 사용자가 원하는 형태로 정보를 가공하여 실시간으로 전달하는 네트워크이다[1,2]. 이러한 무선 센서 네트워크는 수백 혹은 수천 개의 무선 센서 노드들로 이루어진다. 각 센서 노드는 비용 적인 측면에서 매우 작은 크기로 구성되어야 전체 네트워크의 비용 효율성을 이루어낼 수 있게 된다. 이러

· 이 논문은 2006년도 두뇌한국21사업에 의하여 지원되었음
† 학생회원 : 서울대학교 컴퓨터공학부
shyi@ssrnet.snu.ac.kr
hmin@ssrnet.snu.ac.kr
jyheo@ssrnet.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
cho@ssrnet.snu.ac.kr
††† 정 회 원 : 광운대학교 컴퓨터공학부 교수
gman@daisy.kw.ac.kr
(Corresponding author임)
논문접수 : 2006년 5월 20일
심사완료 : 2006년 8월 17일

한 센서 노드는 온도나 습도 및 광량 정보 등을 얻어낼 센서와 간단한 계산을 수행할 수 있는 CPU, 무선 통신을 위한 R/F 모듈, 부팅을 위한 작은 크기의 ROM과 메인 배터리 등으로 구성 된다[3-5]. 이러한 제약 사항을 갖는 컴퓨터 시스템에서 동작하는 센서 운영체제는 공간 효율적으로 동작하여야 하며, 센서 노드에서 발생할 수 있는 센싱, 통신 및 데이타 변환 등의 다수의 작업들을 처리해야 하며, 이러한 작업 처리 중에 필요로 하는 메모리 공간에 대한 효율적인 할당 및 반환을 처리할 수 있어야 한다. 따라서 센서 운영체제를 위한 메모리 할당 기법은 메모리 단편화 문제를 최소화하면서 전체 메모리 공간의 활용을 최대화해야 한다.

현재까지, TinyOS[6], Mate-VM[7], SOS[8], MANTIS OS[1], 그리고 Nano-Qplus[2] 등의 다양한 센서 운영체제와 관련한 연구가 진행되었고, 센서 운영체제들에서 다양한 메모리 할당 기법들이 사용되었다. SOS는 동적 모듈의 추가 및 제거 기능을 지원하기 위하여, MANTIS OS와 Nano-Qplus는 다중 쓰레드의 스택 공간을 메모리에 할당 받아야 하기 때문에, 동적 메모리 할당이 필요하다. 그러나 현존하는 센서 운영체제들에서 사용되고 있는 메모리 할당 기법들은, 메모리 공간의 활용 측면에서 비효율적으로 동작하고, 수행 시간도 좋지 않기 때문에, 공간 제약적인 센서 플랫폼에서 사용하기에는 적합하지 않다.

본 논문에서는 공간 및 에너지 제약적인 센서 플랫폼에서 동작하는 센서 운영체제를 위한 효율적인 메모리 할당 기법을 제안한다. 제안한 기법을 사용하면, 기존의 센서 운영체제에서 사용되고 있는 메모리 할당 기법을 사용하는 것보다 메모리 단편화 문제를 절감시킬 수 있다. 본 논문의 실험 결과를 통하여, 제안한 기법을 사용하는 것이 기존의 메모리 할당 기법을 사용하는 것보다 메모리 효율성을 상당히 증진시킬 수 있고, 전체 수행 시간도 나빠지지 않음을 보인다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구를 보이고, 3절에서는 공간 제약적인 센서 운영체제를 위한 효율적인 메모리 할당 기법의 자세한 설명과 함께 그 구현 방법을 보인다. 4절에서는 제안한 기법의 성능 측정 결과를 보이고 이를 평가한다. 마지막으로 5절에서 본 논문의 결론을 보인다.

2. 관련 연구

본 절에서는 동적 메모리 할당 기법과 관련하여 기존의 여러 가지 관련 연구를 보인다. 현재까지 메모리 할당 기법의 공간 및 시간상의 성능을 증진시키기 위하여 다양한 기법들이 연구되어왔다[9-15]. 이러한 기법들은 크게 다음의 네 가지 범주로 나누어볼 수 있다: 순차 적합(Sequential Fit)[9], 격리 적합(Segregated Fit)[10,11], 버디 시스템(Buddy System)[12-14], 그리고 하이브리드 기법(Hybrid Scheme)[15].

2.1 순차 적합 기법

순차 적합 기법은 자유 메모리 공간을 선형 리스트로 관리한다. 이것은 가장 간단한 형태의 메모리 할당 기법으로, 연속적인 리스트 자료 구조를 통하여 다양한 크기의 메모리 블록을 할당 및 반환을 수행할 수 있다. 일반적으로 이 기법을 사용할 경우에는 선형 리스트 자료구조에 경계 태그(Boundary Tags)[9]를 사용하여 인접한 메모리 블록을 합치는데에 걸리는 시간을 최소화한다. 순차 적합 기법에서 사용되는 잘 알려진 기법으로, 최초 적합(Fitst Fit), 다음 적합(Next Fit), 최선 적합(Best Fit), 그리고 최악 적합(Worst Fit)등이 존재한다[16]. 이 기법을 사용하면 내부 단편화(Internal Fragmentation) 문제는 발생하지 않는다. 그러나 이 기법은 선형 리스트를 순차적으로 탐색하기 때문에, 최악 수행 시간은 O(N)이 된다. 또한, 외부 단편화(External Fragmentation)가 발생할 수 있으므로, 이러한 사항을 고려하여 알맞은 탐색 기법을 적용해야 한다.

2.2 격리 적합 기법

격리 적합 기법은 자유 메모리 공간을 크기에 따라 격리된 리스트로 관리한다. 이를 통하여, 순차 적합 기법에서 상당한 문제가 되었던 최악 수행 시간을 크게 절감시켰다. 다음의 두 기법은 격리 적합 기법의 변형이다: 2의 승수 자유 리스트(Power-of-2 free lists), 격리 순차 적합(Segregated Sequential Fit).

2.2.1 2의 승수 자유 리스트

2의 승수 자유 리스트 기법은 메모리 블록들을 모두 2의 승수의 크기로 관리한다. 사용자가 메모리 할당을 요청하면, 요청된 크기와 같거나 혹은 그보다 큰 메모리 블록을 찾아서 사용자에게 할당한다. 따라서 이 기법을 사용할 경우에는 최악 수행 시간이 O(1)으로 감소할 수 있지만, 사용자의 메모리 할당 요청의 크기에 따라, 심각한 내부 단편화 문제가 발생할 수 있다. 최악의 경우에는 약 50% 정도의 메모리 공간이 내부 단편화로 낭비될 수 있다[16].

2.2.2 격리 순차 적합 기법

격리 순차 적합 기법은 기존의 순차 적합 기법에서 사용되던 하나의 선형 리스트를 여러 개로 분할하여 관리한다. 만약 k개로 분할하였다면 최악 수행 시간은 O(N/k)가 된다[11].

2.3 버디 시스템

Peterson 등은 [13]에서 버디 시스템을 제안하였다. 이 기법은 2의 승수 자유 리스트 기법에서 확장된 것으로, 2의 승수의 크기를 갖는 메모리 블록들의 병합 및

분할 기능을 지원한다. 이 방법을 통하여, 2의 승수 자유 리스트에서 문제가 되었던 내부 단편화를 상당히 줄일 수 있고 전체 메모리 공간을 효율적으로 관리할 수 있다. 그러나 이 기법을 사용할 경우에도 최악의 경우 약 25% 정도의 내부 단편화가 발생할 수 있다[17].

2.4 하이브리드 기법

Masmano 등은 [15]에서 실시간 시스템을 위한 새로운 동적 메모리 할당 기법을 제안하였다. 제안한 기법은 계층화 된 격리 적합 기법과 경계 태그 기법 등을 사용하여 최악 수행 시간을 감소시켰다.

3. 센서 운영체제를 위한 메모리 할당 기법의 설계

본 절에서는 무선 센서 네트워크 및 무선 센서 플랫폼의 요구사항과 제약사항을 보인다. 또한, 현존하는 센서 운영체제에서 사용되고 있는 메모리 할당 기법들을 보인다. 이러한 제약사항 및 요구사항, 그리고 기존 메모리 할당 기법 등을 토대로 본 논문의 핵심인 센서 운영체제를 위한 메모리 할당 기법의 설계 및 구현 내용을 자세히 설명한다.

3.1 무선 센서 네트워크의 요구사항

일반적으로 무선 센서 네트워크는 다수의 센서 노드로 구성되고, 비용 효율성 때문에 각 센서 노드는 매우 제한적인 하드웨어들로 구성된다. 대표적인 예로, 미국의 버클리 대학에서 설계한 MICA 시리즈 센서 플랫폼은 8비트 CPU, 4KB RAM, 그리고 두개의 AA배터리 등으로 구성된다[18]. 이러한 센서 플랫폼에서 동작할 센서 운영체제는 제한적인 메모리 공간을 효율적으로 관리하면서 제한적인 배터리 전력을 효율적으로 사용해야한다. 다시 말하면, 메모리 공간을 관리할 효율적인, 그리고 센서 플랫폼에 특화된 메모리 관리 기법이 필요하다고 할 수 있다. 다음은 센서 운영체제를 위한 메모리 관리 기법의 설계 및 구현 시에 반드시 고려해야 할 무선 센서 네트워크의 몇 가지 요구사항들이다.

- 단편화 비율(Fragmentation Ratio): 매우 제한적인 메모리 크기를 갖는 무선 센서 플랫폼에서, 내부 단편화 및 외부 단편화 문제는 전체 메모리 공간의 효율성에 상당한 악영향을 일으킬 수 있다. 따라서 이러한 단편화 비율을 최소화 할 수 있어야 한다.
- 수행 시간(Execution Time): 무선 센서 플랫폼에서의 수행 시간은 곧 생존성 및 수명과 밀접한 관련이 있다. 따라서 메모리 할당 및 반환 작업의 수행 시간을 최소화 한다면, 제한적인 배터리를 보다 효율적으로 사용할 수 있을 것이다.

3.2 현존하는 센서 운영체제들의 메모리 할당 기법

다음의 그림 1은 현존하는 센서 운영체제들[1,2,8]에서 사용되고 있는 메모리 할당 기법들의 자료구조를 보인다.

SOS는 UCLA에서 제작된 운영체제로, 동적 모듈의 추가 및 제거를 지원하기 위하여 동적 메모리 할당 기법을 사용한다. 이 운영체제는 2의 승수 자유 리스트 기법[8]과 유사한 방식을 사용한다. SOS의 전체 메모리 공간은 32개의 16바이트 블록, 16개의 32바이트 블록, 그리고 4개의 128바이트 블록으로 구성된다. 따라서 전체 힙 메모리 공간은 1536바이트가 되고, 나머지 공간은 커널 스택 및 데이타 영역 등의 용도로 사용한다. 격리 적합 기법과 같이, SOS에서 사용자의 메모리 할당 요청에 의하여 적합한 메모리 블록을 찾는 시간은 O(1)이 된다. 따라서 SOS의 메모리 할당 기법은 수행 시간의 측면에서는 빠를 수 있지만, 심각한 내부 단편화 문제를 일으킬 수 있는 단점 또한 존재한다. 그림 1의 (a)는 SOS의 메모리 할당 기법과 관련한 자료 구조를 보인다.

Nano-Qplus는 전자통신연구원(ETRI)에서 제작된 센서 네트워크용 운영체제이고, MANTIS OS는 미국의 콜로라도 대학에서 설계된 운영체제이다. Nano-Qplus와 MANTIS OS는 다중 쓰레드 기반 운영체제로, 쓰레드 들의 스택 공간을 할당하기 위하여 동적 메모리 할당 기법을 사용한다. Nano-Qplus와 MANTIS OS는 단편화 비율을 줄이기 위하여, 순차 적합 기법을 도입하였고, 탐색 방법으로는 최선 적합 정책을 사용하였다. 따라서 단편화 비율은 SOS의 경우에 비하여 상대적으로 적지만, 메모리 할당 요청의 처리에 상대적으로 오랜 시간을 소모한다. 그림 1의 (b)는 Nano-Qplus와 MANTIS OS의 메모리 할당 기법과 관련한 자료 구조를 보인다.

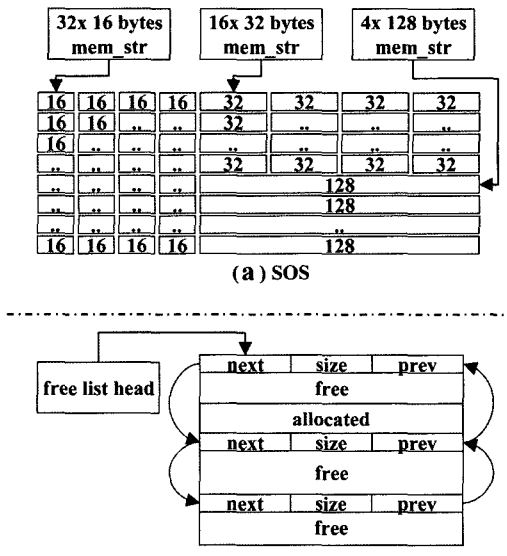


그림 1 센서 운영체제의 메모리 관련 자료구조

3.3 센서 운영체제를 위한 효율적인 메모리 할당 기법의 설계

앞에 보인 요구사항 및 기존 센서 운영체제들에서 사용되고 있는 메모리 할당 기법들의 장단점을 고려하여, 본 논문에서는 여러 가지 제약사항을 갖는 센서 운영체제를 위한 효율적인 메모리 할당 기법을 제안한다. 이 기법은 비트맵을 사용하는 순차 적합, 격리 적합, 그리고 버디 시스템의 여러 가지 기법들을 적응성 있게 활용한다. 순차 적합 기법은 메모리 단편화 비율을 감소시키는 데에 유용하고, 격리 적합 기법은 수행 시간을 감소시킬 수 있다. 또한 버디 시스템은 메모리 관리의 측면에서 유연성을 제공하며, 격리 적합 기법이 가진 단점을 보완해줄 수 있다. 이를 통하여 무선 센서 네트워크의 필수 요구 사항인 메모리 단편화 비율을 최소화 하고, 메모리 할당 및 반납의 수행 시간 또한 나빠지지 않게 한다.

그림 2는 본 논문에서 제안한 기법의 자료 구조 및 물리 메모리, 그리고 사용자의 메모리 할당 요청의 관계를 나타내었다. 그림 2에서, 자료 구조는 비트맵과 k개의 전역 항목들(global entries), 그리고 격리 항목들(segregated entries)로 구성되는 테이블로 구성된다. 각각의 전역 항목은 메모리 블록의 클래스 ID와 주소를 담는다. 테이블은 2의 승수(4, 8, 16, 32, 64, 128바이트) 크기에 해당되는 격리된 항목들로 이루어지고, 각각은 m개의 주소 항목으로 구성된다. 비트맵 구조는 사용자에게 할당된 메모리 영역을 표시하기 위하여 사용되고, 하나의 비트는 4바이트의 실제 물리 메모리와 연결된다. 전역 및 격리 항목들은 자유(사용 가능한) 메모리 블록들의 주소를 관리한다. 예를 들면, 각각의 격리 항목들은 최대 m개의 자유 메모리 블록들을 관리할 수 있고, 나머지 메모리 블록들은 전역 항목들이 관리하게 된다. 다음에는 제안한 메모리 할당 기법의 세부 동작들에 대한 자세한 설명을 보인다.

- 초기화(Initialization): 제안한 기법에서 사용하는 모든 자료 구조는 커널의 데이타 영역에 정적으로 할당되어있다. 따라서 비트맵 및 테이블 항목들은 시스템 부팅 시에 모두 0으로 초기화 되며, 추가적인 자료 구조 할당은 발생하지 않는다.
- 할당(Allocation): 메모리 할당 요청의 크기가 r_{size}일 때, 제안한 기법은 $2^{\lceil \log_2 r_{size} \rceil}$보다 크거나 같은 크기의 메모리 블록을 격리 항목에서 찾는다. 격리 항목에서 적합한 메모리 블록을 찾았다면 해당 블록을 사용자에게 전달한다. 만약 메모리 블록을 찾지 못했다면, 전역 항목에서 찾아보고, 그래도 찾지 못했다면, 비트맵 자료 구조에서 최초 적합 정책으로 찾는다. 마지막으로, 메모리 블록을 사용자에게 할당한 후에는 그에 해당되는 비트맵의 비트를 1로 설정한다.
- 반환(Release): 메모리 반환 요청의 크기가 r_{size}일 때, 제안한 기법은 반환된 메모리 블록을 격리 항목이나 전역 항목에 재등록한다. 그리고 해당되는 비트맵의 비트를 0으로 설정한다.
- 병합 및 분할(Coalescing and Splitting): 버디 시스템과 유사한 방식으로, 제안한 기법은 메모리 공간을 2의 승수 크기로 관리하며, 인접한 메모리 공간에 대한 병합 및 분할 기능을 지원한다. 만약 메모리 할당 요청이 발생했을 경우, 메모리 할당 요청의 크기가 r_{size}일 때, 제안한 기법은 $2^{\lceil \log_2 r_{size} \rceil}$의 크기를 갖는 블록을 찾는다. 찾은 메모리 블록에서, r_{size}의 메모리 공간을 사용자에게 할당시키고, 남아있는 $2^{\lceil \log_2 r_{size} \rceil} - r_{size}$의 메모리 공간을 분할하여 격리 항목 혹은 전역 항목에 등록한다. 반면에, 메모리 반납 요청이 발생했을 경우에 반납되는 메모리 블록의 크기가 r_{size}일 때, 제안한 기법은 $2^{\lceil \log_2 r_{size} \rceil}$의 인접한 메모리 공간을 탐색

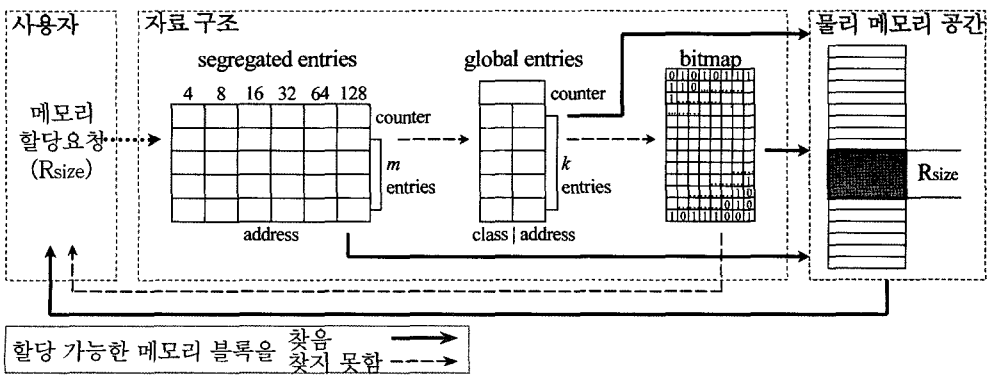


그림 2 제안한 메모리 할당 기법의 자료 구조

후 메모리 공간의 병합을 수행한다. 마지막으로 병합된 메모리 블록을 격리 항목 혹은 전역 항목에 등록한다.

- 자료 구조 관리(Data Structure Management): 제안한 기법은 다음과 같이 3단계의 자료 구조를 갖고 있다: 격리 항목, 전역 항목, 그리고 비트맵. 이러한 자료 구조는 컴퓨터 시스템에서 캐쉬 메모리 계층과 유사하게 동작한다. 예를 들면, 메모리 할당 요청이 발생 했을 때, 적합한 메모리 블록을 찾기 위하여 격리 항목→전역 항목→비트맵의 순서로 탐색을 수행한다. 만약 격리 항목이 비어있다면 전역 항목에서 메모리 블록을 찾는다. 전역 항목에서 찾았다면 해당 블록을 할당하고 나서 격리 항목의 내용을 전역 항목의 내용을 바탕으로 갱신한다. 만약 전역 항목에서도 찾지 못하였다면, 비트맵에서 최초 적합 정책으로 찾고, 격리 항목 및 전역 항목의 내용을 갱신한다.

그림 3은 제안한 기법을 사용하였을 때에 80바이트의 메모리 공간을 할당 및 반환의 예를 보인다. 사용자가 80바이트의 메모리 할당을 요청하였을 때, 제안한 기법은 먼저 $2^{\lceil \log_2 80 \rceil}$을 계산한다. 계산 결과가 128이므로, 128바이트 크기의 메모리 블록을 찾을 것이다. 그 후에는 찾은 128바이트의 메모리 블록을 80바이트와 48바이트의 두개의 메모리 블록으로 분할하고, 80바이트를 사용자에게 할당한다. 남아있는 48바이트의 메모리 공간은

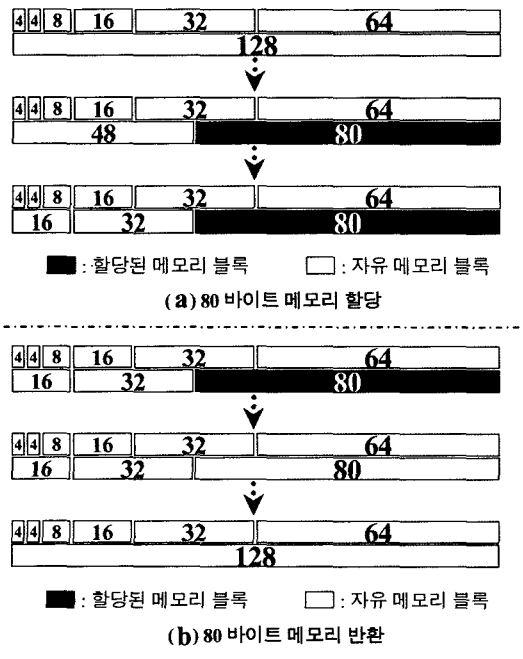


그림 3 80바이트 메모리 공간 할당 및 반환 예제

2의 승수의 크기로 16바이트와 32바이트의 메모리 블록으로 분할되어 적절한 항목에 재등록된다. 만약 사용자가 80바이트의 할당되었던 메모리를 반납할 경우, 제안한 기법은 $2^{\lceil \log_2 80 \rceil}$이 128이므로, 128바이트의 인접한 메모리 공간을 탐색할 것이다. 위의 그림 3에서, 16바이트와 32바이트의 메모리 블록이 존재하므로, 128바이트의 인접 메모리 공간을 찾게 되고, 16, 32, 80바이트의 메모리 블록들을 병합할 것이다. 그 결과로, 128바이트의 메모리 블록이 생성되고, 이것은 적절한 항목에 재등록된다. 이러한 방식으로, 기존의 여러 가지 메모리 할당 기법들을 활용하여 메모리 단편화 문제를 최소화 할 수 있게 된다.

다음의 알고리즘 1은 제안한 기법을 사용하였을 때, r_{size}의 메모리 할당 요청 발생시에 메모리 블록을 찾고, 분할하고, 사용자에게 할당하는 일련의 과정을 보인다. 또한 알고리즘 2는 할당되었던 메모리의 반납 요청이 발생하였을 경우, 인접한 메모리와 병합하여 적절한 항목에 재등록하는 과정을 보인다.

알고리즘 2에서, 인접한 메모리를 병합하는 데에 필요한 시간은 O(1)이면 충분하다. 인접한 메모리를 검사할 때에, 최악의 경우 128바이트의 공간을 탐색하게 된다. 비트맵 자료 구조에서 128바이트의 공간에 해당되는 비트는 32비트이다. 즉, 4바이트 공간만 탐색하면 되므로, 메모리 병합은 매우 짧은 시간에 수행될 수 있다.

4. 성능 평가

본 절에서는 제안한 메모리 할당 기법의 성능 측정 결과를 보인다. 제안한 기법은 Nano-Qplus 운영체제 상에서 구현되었다. 제안한 기법과 현존하는 센서 운영체제들[1]에서 사용되고 있는 메모리 할당 기법을 비교 실험하였고, 실험을 수행한 환경은 다음의 표 1에 보인다[19].

표 1에서, Nano-24 센서 보드는 버클리 대학에서 설계한 MICAZ 센서 플랫폼과 그 구성이 유사하다. 본 논문의 성능 평가에서는 Nano-24 보드를 사용하였고,

표 1 성능 평가를 위한 실험 환경

플랫폼 구성	설명	비고
운영체제	나노 Qplus 1.6.0e	ETRI 제작
센서보드	Nano-24 센서 보드	Octacomm 제작
CPU	ATmega128L	8비트 프로세서
RAM	4KB	데이타+스택 영역
FLASH	128KB	코드 영역
전원	2xAA 배터리	3.0 볼트

1) 본 논문의 실험에서 사용된 운영체제와 커널 버전은 다음과 같다. SOS: sos-july, Nano-Qplus: 1.6.0e, MANTIS OS: 0.9.1b

알고리즘 1. 메모리 할당 및 분할 방법

----- 사용자가 r_{size}의 메모리 할당을 요청한 경우 -----
$2^{\lceil \log_2 r_{size} \rceil}$ 보다 크거나 같은 크기의 메모리 블록을 탐색

IF 적합한 블록을 격리 항목에서 찾은 경우 **THEN**
r_{size}의 메모리 공간을 사용자에게 할당
$2^{\lceil \log_2 r_{size} \rceil} - r_{size}$의 메모리 공간을 2의 승수의 크기로 분할
분할된 메모리 블록을 적절한 항목에 재등록

ELSE IF 적합한 블록을 전역 항목에서 찾은 경우 **THEN**
r_{size}의 메모리 공간을 사용자에게 할당
$2^{\lceil \log_2 r_{size} \rceil} - r_{size}$의 메모리 공간을 2의 승수의 크기로 분할
분할된 메모리 블록을 적절한 항목에 재등록
전역 항목의 내용을 바탕으로 격리 항목을 갱신

ELSE IF 적합한 블록을 비트맵에서 찾은 경우 **THEN**
r_{size}의 메모리 공간을 사용자에게 할당
비트맵의 내용을 바탕으로 전역 및 격리 항목을 갱신

ELSE
메모리 할당 실패

END IF

알고리즘 2. 메모리 반환 및 병합 방법

----- 사용자가 r_{size}의 메모리 반환을 요청한 경우 -----
r_{size}의 메모리를 사용자로부터 반환
인접한 $2^{\lceil \log_2 r_{size} \rceil}$의 메모리 공간을 탐색
IF 병합 가능한 메모리 공간을 찾은 경우 **THEN**
해당 메모리 공간을 병합
병합된 메모리 블록을 적절한 항목에 재등록
END IF

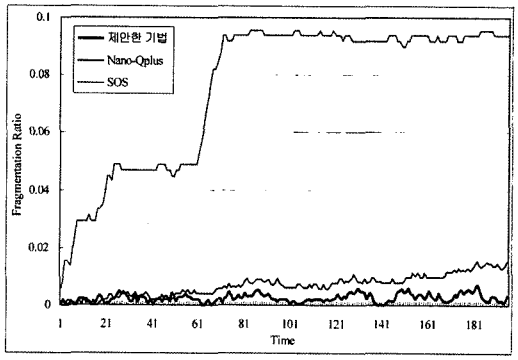
SOS를 Nano-24 센서 보드에서 동작시키기 위하여, 일부의 커널 소스 코드를 수정하였다.

메모리 할당 기법의 성능은 시간과 공간의 두 가지 측면에서 평가되어야 한다. 시간의 측면은 곧 수행 시간을 의미하고, 공간의 측면은 메모리 단편화 비율을 의미한다. 본 논문의 실험에서는, 이러한 시간과 공간에서의 성능을 평가하기 위하여 제안한 기법과 기존의 기법들에 대한 단편화 비율과 전체 수행 시간을 비교하였다. 표 2는 메모리 할당의 크기가 4~8 바이트일 때와 4~16 바이트일 때의 각 메모리 할당 기법에 대한 전체 수행 시간을 보인다. 모든 경우에 대하여, 메모리 할당 및 반환은 200회 이루어졌다. 표 2의 결과에서, 각 숫자는 타임 틱(ticks)을 나타낸다(1 ticks = 16 us). 표 2의 결과에서, SOS의 수행 시간이 가장 빠르다. 그 이유는, SOS는 메모리 공간을 정적으로 관리하기 때문이다. 제안한 기법은 SOS의 결과에는 미치지 못하지만, Nano-Qplus보다는 적은 수행 시간에 동작함을 보인다.

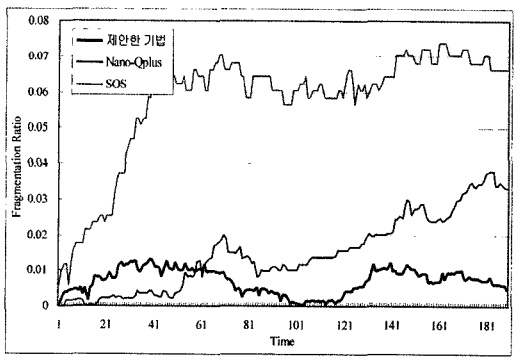
그림 4는 각각의 메모리 할당 기법들에 대한 단편화

표 2 메모리 할당 기법들의 전체 수행 시간

워크로드	제안한 기법	Nano-Qplus	SOS
4~8 바이트	6,073	6,694	5,523
4~16 바이트	6,142	6,511	5,664



(a) 4~8 바이트 메모리 할당 및 반환



(b) 4~16 바이트 메모리 할당 및 반환

그림 4 메모리 할당 기법들의 단편화 비율의 변화

비율을 보인다. 이 그림에서, 세로축은 단편화 비율을 보이고, 가로축은 시간을 나타낸다. 이 결과에서, SOS의 성능이 가장 나쁜 것을 볼 수 있다. SOS는 메모리 할당 요청 시에, 단순히 정적으로 분할되어있는 메모리 블록을 사용자에게 할당한다. 따라서 처리 속도의 측면에서는 빠르지만, 이렇게 하면 심각한 내부 단편화 문제가 발생한다. 제안한 기법과 Nano-Qplus의 단편화 비율은 초기에는 큰 차이가 없으나, 시간이 지남에 따라 둘 간의 차이가 발생하는 것을 볼 수 있다. 제안한 기법의 경우에는 시간이 지나더라도 일정한 수준의 단편화 비율을 보이는데 반해서, Nano-Qplus와 SOS는 시간에 따라 상당히 증가함을 볼 수 있다.

5. 결 론

무선 센서 네트워크는 수백 혹은 수천 개의 무선 센

서 노드들로 이루어지고, 센서 노드의 플랫폼은 비용 효율성 때문에 매우 제한적인 메모리 공간을 지니며, 제한적인 배터리로 동작한다. 따라서 이것들을 동작시키는 센서 운영체제는 공간 제약성을 감내할 수 있어야 하고, 에너지 효율적으로 동작해야 전체 센서 네트워크를 효율적으로 동작시킬 수 있게 된다. 본 논문에서는 공간 제약적인 센서 운영체제를 위한 효율적인 메모리 할당 기법을 제안하였다. 제안한 기법을 사용하였을 때, 기존 센서 운영체제들에서 사용되었던 메모리 할당 기법들을 사용하는 것보다 메모리 단편화 문제를 감소시키고 공간의 효율성을 증진시킬 수 있다. 본 논문의 비교 실험 결과를 통하여, 제안한 기법을 사용하는 것이 기존의 방법보다 메모리 단편화를 상당히 줄일 수 있고, 또한 수행 시간도 나빠지지 않음을 보였다.

만약 제안한 기법이 실제 무선 센서 네트워크의 센서 운영체제들에서 사용된다면, 메모리 공간 효율성을 효과적으로 증진시킬 수 있을 것이다.

6. 소스코드 다운로드

제안한 메모리 할당 기법을 구현한 모든 소스코드와 실험에 사용한 코드는 다음의 URL에서 다운로드 할 수 있다.

참 고 문 헌

[1] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, R. Han, Mantis os: An embedded multithreaded operating system for wireless micro sensor platforms. ACMKluwer Mobile Networks and Applications (MONET) Journal, Special Issue on Wireless Sensor Networks, 2005.

[2] K. Lee, Y. Shin, H. Choi, S. Park, A design of sensor network system based on scalable and recongurable nano-os platform. In Proceedings of IT-Soc International Conference, 2004.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks. IEEE Communications Magazine pp.102-114, 2002.

[4] J.D. Lundquist, D.R. Cayan, M.D. Dettinger, Meteorology and hydrology in yosemite national park: A sensor network application. Lecture Note in Computer Science Vol. 2634, pp.518-528, 2003.

[5] M. Hirafuji, T. Fukatsu, H. Hu, T. Kiura, M. Laurenson, D. He, A. Yamakawa, A. Imada, S. Ninomiya, Advanced sensor-network with field monitoring servers and metbroker. In Proceedings of CIGR International Conference, 2004.

[6] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, D. Culler, The emergence of networking abstractions and techniques in tinyos. In Proceedings of First USENIX/ ACM Symposium on Networked Systems Design and Implementation(NSDI 2004), 2004.

[7] P. Levis, D. Culler, Mate: a virtual machine for tiny networked sensors. In Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems, pp.85-95, 2002.

[8] C.C. Han, R. Kumar, R. Shea, E. Kohler, M.B. Srivastava, A dynamic operating system for sensor nodes. In Proceedings of MobiSys, pp.163-176, 2005.

[9] D.E. Knuth, The art of computer programming, vol. 1: Fundamental algorithms. Addison-Wesley, 1973.

[10] M.K. McKusick, M.J. Karels, Design of a general purpose memory allocator for the 4.3bsd unix kernel. In Proceedings of the San Francisco USENIX Conference, pp.295-303, 1988.

[11] D. Lea, A memory allocator. Unix/Mail, June, 1996.

[12] K.C. Knowlton, A fast storage allocator. Communications of the ACM Vol.8, pp.623-625, 1965.

[13] J.L. Peterson, T.A. Norman, Buddy systems. Communications of the ACM Vol.20, pp.421-431, 1977.

[14] I.P. Page, J. Hagins, Improving the performance of buddy systems. IEEE Transactions on Computers C-35, pp.441-447, 1986.

[15] M. Masmano, I. Ripoll, A. Crespo, J. Real, Tlsf: a new dynamic memory allocator for real-time systems. In: Euromicro Conference on Real-Time Systems(ECRTS'04), 2004.

[16] U. Vahalia, Unix internals: The new frontiers. Prentice Hall, 1996.

[17] M.S. Johnstone, P.R. Wilson, The memory fragmentation problem: solved? ACM SIGPLAN Notices Vol.34, pp.26-36, 1999.

[18] Crossbow, http://www.xbow.com/, (website)

[19] Octacomm, http://www.octacomm.net/, (website)

이 상 호

2003년 고려대학교 전기전자전파공학부 졸업(학사). 2003년~현재 서울대학교 컴퓨터공학부 박사과정(수료). 관심분야는 시스템 및 운영체제, 무선 센서 네트워크, 센서 운영체제, 결함 허용 시스템

민 홍

2005년 한동대학교 경영전산학과 졸업 (학사). 2005년~현재 서울대학교 컴퓨터공학부 석사과정. 관심분야는 시스템 및 운영체제, 무선 센서 네트워크, 센서 운영체제, 임베디드 시스템

허 준 영

1998년 서울대학교 컴퓨터공학과 졸업 (학사). 2002년~현재 서울대학교 컴퓨터공학부 박사과정(수료). 관심분야는 시스템 및 운영체제, 무선 센서 네트워크, 결함 허용 시스템, 임베디드 시스템 보안

조 유 근

정보과학회 논문지 : 시스템 및 이론
제 33 권 제 6 호 참조

홍 지 만

2004년 서울대학교 컴퓨터공학부 박사 졸업. 2004년~현재 광운대학교 컴퓨터공학부 조교수. 관심분야는 운영체제 및 임베디드 시스템, 임베디드 운영체제, 결함 허용 컴퓨팅, 분산 시스템, 무선 센서 네트워크