# 공간 제약적인 센서 운영체제를 위한 스택리스 쓰레드 기법

이상호°, 구본철, 민 홍, 허준영, 김용태, 조유근
서울대학교 컴퓨터공학부
{shyi°, bcgu, hmin, jyheo, ytkim, cho}@ssrnet.snu.ac.kr

홍지만
광운대학교 컴퓨터공학부
gman@daisy.kw.ac.kr

# Stackless Thread Scheme for Space Constrained Sensor Operating Systems

Sangho Yi°, Boncheol Gu, Hong Min, Junyoung Heo, Yongtae Kim, Yookun Cho
School of Computer Science and Engineering, Seoul National University

Jiman Hong
School of Computer Science and Engineering, Kwangwoon University

**요 약**

무선 센서 네트워크는 수백 혹은 수천 개의 무선 센서 노드들로 이루어지고, 센서 노드의 플랫폼은 비용 효율성 때문에 매우 제한적인 메모리 공간을 지닌다. 이러한 센서 노드들은 자연의 정보를 수집하고, 이웃 노드들끼리 통신하며, 정보를 가공하여 사용자에게 실시간으로 전달하는 기능을 담당한다. 따라서 무선 센서 네트워크를 위한 센서 운영체제는 공간 효율적이고, 다수의 작업들을 실시간으로 처리할 수 있는 멀티 쓰레드 기법이 필요하다. 본 논문에서는 공간 제약적인 센서 운영체제를 위한 스택리스 쓰레드 기법을 제안한다. 제안한 기법을 사용하면, 기존의 스택기반 쓰레드를 사용하는 것보다 메모리 공간의 사용량을 절감시킬 수 있다. 본 논문의 비교 실험 결과를 통하여, 제안한 기법을 사용하는 것이 기존의 방법보다 메모리 사용량을 상당히 줄일 수 있음을 보인다.

## 1. 서 론

무선 센서 네트워크는 자연의 여러 가지 정보를 센싱하고, 무선으로 통신하며, 사용자가 원하는 형태로 정보를 가공하여 실시간으로 전달하는 네트워크이다[1]. 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. 이러한 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 ROM 과 메인 배터리 등으로 구성 된다[2]. 따라서 센서 운영체제는, 공간 효율적으로 동작하여야 하며, 센서 노드에서 발생할 수 있는 센싱, 통신 및 변환 등의 다수의 작업을 실시간으로 처리하기 위한 멀티 쓰레드 기법을 제공해야 한다.

일반적으로, 멀티 쓰레드 기법은 각 쓰레드 마다의 고유의 스택을 메모리 공간에 할당받아야 한다. 이러한 스택 공간의 할당은, 매우 제한적인 메모리 공간을 갖는 센서 플랫폼에서는 상당한 문제가 된다[3]. 이러한 문제는 실제로 시스템의 메모리 부족 문제 등을 일으키는 주원인이 되고, 전체 무선 센서 네트워크의 오작동을 일으키게 된다. 이러한 멀티 쓰레드 기법의 사용에 있어서의 문제를 확실히 해결하기 위한 방안은 존재하지 않았다. 일반적으로 멀티 쓰레드가 아닌 이벤트 기반 프로그래밍 기법을 사용하여 메모리 공간의 문제를 해결하였으나, 이러한 이벤트 기반 구조는 실시간 작업 처리를 위한 선점 기능을 제공하지 않는 문제가 존재하였다[3].

본 논문에서는 공간 제약적인 센서 플랫폼에서 동작하는 센서 운영체제를 위한 스택리스 멀티 쓰레드 기법을 제안한다. 제안한 기법을 사용하면, 기존의 스택기반 쓰레드를 사용하는 것보다 메모리 공간의 사용량을 절감시킬 수 있다. 또한 본 논문의 실험 결과를 통하여, 제안한 기법을 사용하는 것이 기존의 스택기반 멀티 쓰레드 기법을 사용하는 것보다 메모리 사용량을 상당히 줄일 수 있음을 보인다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구를 보이고, 3절에서는 공간 제약적인 센서 운영체제를 위한 스택리스 쓰레드 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

현재까지, 멀티 쓰레드 기법에서 스택 메모리 공간의 관리를 효율적으로 수행하기 위한 몇 가지의 기법들이 연구되어왔다[4-6].

Tismer 는 [4]에서, 파이썬 인터프리터의 확장성 증진을 위한 스택리스 파이썬을 제안하였다. 이 방법을 통하여, 기존의 정적 스택 할당 방법을 사용한 경우보다 파이썬 인터프리터가 보다 안정적으로 확장 가능하게 동작할 수 있었다.

Adya 등은 [5]에서, 멀티 쓰레드 기법을 위한 동적 스택 영역 확장 기법을 간략히 제시하였다. 이 방법을 사

용하여, 정적인 스택 공간 할당의 문제점인 공간 효율성을 일부 개선시킬 수 있음을 보였다.

Dunkel 등은 [6]에서, C언어에서의 스택리스 쓰레드 프로그래밍 방법을 제안하였다. 그러나 그들이 제안한 방법은 이벤트 기반 프로그래밍의 한 변형이었고, 멀티 쓰레드의 장점인 선점 기능을 지원하지 않았다.

## 3. 스택리스 쓰레드 기법

본 절에서는 공간 제약적인 무선 센서 운영체제를 위한 쓰레드 기법으로, 스택리스 쓰레드 기법을 제안한다.

무선 센서 네트워크를 이루는 다수의 센서 노드들은, 비용 효율성 때문에 매우 제한적인 하드웨어들로 구성된다. 대표적인 예로, 미국의 버클리 대학에서 설계한 MICA 시리즈 센서 플랫폼은, 8비트 CPU, 4KB RAM, 그리고 두개의 AA배터리 등으로 구성된다[7]. 이러한 센서 플랫폼에서 동작할 센서 운영체제는, 제한적인 메모리 공간을 효율적으로 사용하면서, 전체 무선 센서 네트워크의 작업을 실시간으로 수행시킬 수 있는 멀티 쓰레드 기법이 필요하다. 이를 해결하기위하여, 본 논문에서는 전체 메모리 공간을 효율적으로 사용하면서 다중 작업을 실시간으로 수행시킬 수 있는, "스택리스 쓰레드 기법"을 제안한다. 이 기법은 기존의 스택기반 멀티 쓰레드 기법과 여러 가지의 측면에서 큰 차이가 있다.

기존의 스택기반 멀티 쓰레드 기법은, 쓰레드의 생성시에 해당 쓰레드가 사용할 스택 공간을 정적인 크기로 할당하였다. 따라서 쓰레드가 할당 받은 스택 공간을 전부 사용하지 않는다면, 남는 공간은 스택 공간의 낭비로 이어진다. 그러므로 공간 제약적인 무선 센서 플랫폼에서의 스택 메모리 공간의 정적인 할당은, 자원 관리의 측면에서 심각한 문제를 일으킬 수 있다[3].

위의 문제를 해결하기 위하여, 본 논문에서 제안하는 기법은, 쓰레드가 필요로 하는 만큼의 스택 공간의 할당을 동적으로 수행한다. 이는 각 쓰레드의 매 함수 호출마다 이루어지며, 함수가 종료할 때에는 스택을 반납하여 다른 쓰레드가 그 공간을 재사용할 수 있도록 한다. 이 방법을 사용하면, 메모리 공간의 할당 및 반납의 시간상의 부하는 존재하지만, 전체 메모리 공간을 효율적으로 사용할 수 있게 된다. 따라서 공간 제약적인 무선 센서 운영체제에서, 멀티 쓰레드 기법을 사용할 경우의 메모리 부족 문제를 쉽게 해결해줄 수 있다.

표 1. 두 가지 멀티 쓰레드 기법의 비교

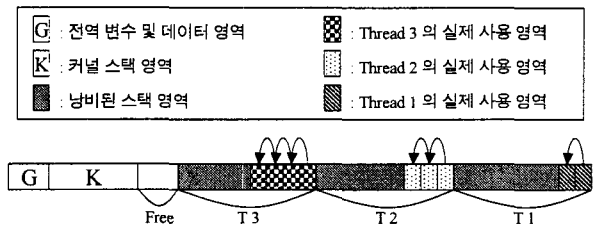
| | 스택기반 쓰레드 기법 | 스택리스 쓰레드 기법 |
|---|---|---|
| 스택 할당 시점 | 쓰레드 생성시 | 함수 호출시 |
| 스택 할당 크기 | 정적 | 동적 |
| 함수 호출 방법 | 일반적인 함수 호출 방법과 동일함 | 스택 할당후 함수 호출<br>함수 종료시 스택 반납 |
| 스택 할당 비용 | 단 한번 발생 | 여러 번 발생 가능 |
| 스택 공간 효율 | 낮음 | 높음 |
| 최대 쓰레드 수 | 스택 크기에 반비례 | 함수 중첩 수준에 반비례 |

표 1은 기존의 스택기반 쓰레드 기법과, 제안한 기법인 스택리스 쓰레드 기법의 기능별 비교를 보인다. 스택리스 쓰레드의 경우, 쓰레드가 사용할 스택 공간이 함수 호출시마다 할당되므로, 수행 시간의 측면에서 잠재적인 부하가 발생할 수 있다. 그러나 전체 스택 공간의 효율성을 증진시킬 수 있고, 최대 쓰레드 수가 스택의 크기가 아닌, 함수의 중첩 수준에 반비례하므로, 기존의 스택기반 쓰레드 기법을 사용하는 경우보다 더 많은 쓰레드를 생성시킬 수 있게 된다. 아래에는, 제안한 스택리스 쓰레드 기법에서, 스택 영역 할당 및 함수 호출, 그리고 스택 영역 반납 과정의 의사 코드를 보인다[1].

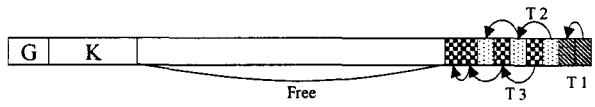
**의사 코드: 스택 영역 할당, 함수 호출, 반납 과정**

1. 스택 포인터 저장 (추후 복원을 위하여)
2. 스택 영역 할당 (불릴 함수가 사용할 스택)
3. 스택 포인터로 할당 받은 스택 영역을 가리킴
4. 함수 인자 전달 (스택에 PUSH)
5. 함수 호출
6. 함수 종료 후에 스택 영역 반납
7. 스택 포인터 복원

위의 의사 코드에서, 스택 영역의 할당량은 불릴 함수가 사용하는 지역 변수의 크기 및 함수 인자들의 개수 등을 사용하여 예상 가능하다. 또한, 컴파일러가 제공하여 주는 어셈블리 코드를 분석한다면, 정확한 스택 영역 크기 또한 알아낼 수 있다. 두 가지 멀티 쓰레드 기법의 실제 작동을 보다 상세히 비교하기 위하여, 다음의 그림 1에는 3개의 쓰레드가 각각 최소 2에서 최대 4수준의 함수를 중첩 호출할 경우, 전체 시스템의 메모리의 사용예를 보인다.




(a) 스택기반 쓰레드 기법의 정적 스택 영역 할당 예




(b) 스택리스 쓰레드 기법의 동적 스택 영역 할당 예

그림 1. 두 가지 멀티 쓰레드 기법의 스택 영역 할당 예

위의 그림 1에 보인 바와 같이 스택리스 쓰레드 기법을 사용할 경우, 스택 영역의 공간 낭비를 최소화 시킬

1) 스택리스 쓰레드의 실제 구현은 나노 Qplus 센서 운영체제 상에서 이루어졌고, C언어와 GCC 인라인 어셈블리를 함께 사용하여 구현하였다.

수 있다. 다음 절에서는 제안한 기법과 기존의 스택기반 쓰레드 기법의 비교 실험을 통하여 스택 공간 할당의 효율성 및 전체 실행 시간의 성능을 평가한다.

## 4. 성능 평가

본 절에서는 공간 제약적인 센서 플랫폼에서 동작하는 센서 운영체제를 위한 스택리스 쓰레드 기법의 성능 측정 결과를 보인다. 제안한 기법과 기존의 스택기반 쓰레드 기법을 비교 실험하였고, 실제 실험을 수행한 환경은 다음의 표 2에 보인다[8].

표 2. 성능 평가를 위한 실험 환경

| 센서 플랫폼 구성 | 설명 | 비고 |
|---|---|---|
| 센서 운영체제 | 나노 Qplus 1.6.0e[2)] | ETRI 제작 |
| 센서 보드 | Nano-24 센서 보드 | Octacomm 제작 |
| CPU | ATmega128L | 8비트 프로세서 |
| RAM | 4KB | 데이터+스택 영역 |
| FLASH MEMORY | 128KB | 코드 영역 |
| R/F | CC2420 | Zigbee(2.4Ghz) |
| 전원 | 2xAA 배터리 | 3.0 볼트 |

본 연구의 실험에서는, 성능 평가를 위하여 쓰레드를 사용하는 예제 응용 프로그램들을 통하여 공간 및 시간상의 효율성을 측정하였다. 또한, 각 쓰레드에 대한 동적 스택 메모리 할당을 위하여, First-fit 알고리즘을 사용하였다. 멀티 쓰레드를 사용하는 LED 깜박임 예제 프로그램과 시리얼 통신 예제 프로그램을 사용하여 비교하였고, 그 결과는 다음의 그림 2에 보인다.

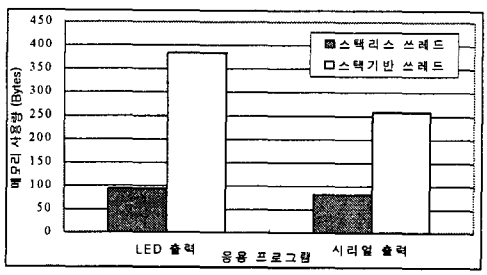



그림 2. 두 가지 기법의 스택 메모리 사용량

그림 2는 각 응용 프로그램별로 사용한 스택 메모리 공간의 크기를 보인다. 위의 결과를 통하여, 스택리스 쓰레드 기법을 사용한 경우에 기존의 스택기반 쓰레드 기법을 사용한 것 보다 메모리 공간의 사용을 현저하게 줄일 수 있음을 알 수 있다. 또한, 다음의 표 3은 각 응용 프로그램의 LED 출력 및 시리얼 출력 함수들을 1,000회 호출하였을 때의 수행 시간을 보인다. 이를 통하여, 스택리스 쓰레드 기법의 함수 호출시에 발생하는 시간상의 부하를 확인할 수 있다.

2) 나노 Qplus 운영체제의 최신 버전은 http://qplus.or.kr 에서 다운로드 할 수 있다.

표 3. 두 가지 기법의 함수 호출 수행 시간

| 예제 프로그램 | LED 출력 (비율) | 시리얼 출력 (비율) |
|---|---|---|
| 스택리스 쓰레드 기법 | 41.8 ms (22.0) | 3453.4 ms (1.1) |
| 스택기반 쓰레드 기법 | 1.84 ms (1.0) | 3120.3 ms (1.0) |

스택리스 쓰레드 기법을 사용할 경우, 기존의 기법을 사용한 경우보다 수행 시간이 증가하는 것을 볼 수 있다. 특별히, LED 출력과 같은 경우에는, 함수 내부에서 처리하는 일의 양이 적기 때문에, 스택리스 기법을 사용할 경우에는 상대적인 부하가 약 22배 정도로 상당하였다. 그러나 시리얼 출력의 경우에는 시리얼 출력 속도가 전체 수행시간을 좌우하는 가장 큰 요소이기 때문에, 그다지 상대적인 부하가 크게 영향을 미치지 않았다. 앞으로의 연구에서는 이러한 성능 저하를 최소화 할 수 있는 방법에 대한 연구를 이어서 진행할 것이다.

## 5. 결론

무선 센서 네트워크는 센서 노드들로 이루어지고, 센서 플랫폼은 비용 효율성 때문에 제한적인 메모리 공간을 지닌다. 이러한 센서 노드들은 자연의 정보를 수집하고, 통신하며, 정보를 가공하여 사용자에게 실시간으로 전달하는 기능을 담당한다. 따라서 센서 운영체제는 공간 효율적이고, 다수의 작업들을 실시간으로 처리할 수 있는 멀티 쓰레드 기법이 필요하다. 본 논문에서는 공간 제약적인 센서 운영체제를 위한 스택리스 쓰레드 기법을 소개하였다. 본 논문의 비교 실험 결과를 통하여, 제안한 기법을 사용하는 것이 기존의 방법보다 메모리 사용량을 상당히 줄일 수 있음을 보였다. 만약 제안한 기법이 실제 무선 센서 네트워크의 응용들에서 사용된다면, 메모리 공간 부족문제를 효과적으로 해결해낼 수 있을 것이다.

## [참고문헌]

[1] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless micro sensor networks, Hawaii International Conference on System Sciences, 2000.

[2] Anna Hac, Wireless Sensor Network Designs, John Wiley and Sons, 2003.

[3] A. Gustafsson, Threads without the Pain, ACM Queue, pp.34-41, 2005.

[4] C. Tismer, Continuations and Stackless Python, In Proceedings of the 8th International Python Conference, 2000.

[5] A. Adya, J. Howell, M. Theimer, W.J. Bolosky, J.R. Douceur, Cooperative task management without manual stack management, In Proceedings of the 2002 USENIX Annual Technical Conference, 2002.

[6] A. Dunkels, O. Schmidt, T. Voigt, Using protothreads for sensor node programming, In Proceedings of the REALWSN'05 Workshop on Real-World Wireless Sensor Networks, 2005.

[7] Crossbow Technology, http://www.xbow.com (web-site).

[8] Octacomm, http://www.octacomm.net (web-site).