MCUP: 자원 제약하의 센서 노드를 위한 다중 수준 코드 갱신 기법

이상호[O1] 민홍[1] 김석현[1] 조유근[1] 홍지만[2]
서울대학교 컴퓨터공학부[1]
숭실대학교 컴퓨터학과[2]
{shyi[O],hmin,shkim,cho}@os.snu.ac.kr[1], jiman@ssu.ac.kr[2]

MCUP: Multi-level Code Update Protocol for Resource-constrained Sensor Nodes

Sangho Yi[O1] Hong Min[1] Seokhyun Kim[1] Yookun Cho[1] Jiman Hong[2]
School of Computer Science and Engineering, Seoul National University[1]
School of Computing, Soongsil University[2]

요 약

무선 센서 네트워크는 다양한 환경에서 자연의 정보를 수집하여 인간이 필요로 하는 형태로 정보를 제공하는 네트워크이다. 이러한 센서 네트워크는 수많은 무선 센서 노드들로 이루어지고, 이들은 원격지에서 동작한다. 각 센서 노드는 자연 환경의 정보를 센싱하고, 무선 네트워크를 통하여 센싱 데이터를 사용자에게 전달한다. 만약 소프트웨어의 코드 갱신을 수행해야 한다면, 이는 원격에서 수행 중에 이루어져야 한다. 이 때에, 각 센서 노드의 자원 제약을 고려하여 보다 자원 효율적으로 동작해야 한다. 본 논문에서는 MCUP이라 불리는 무선 센서 노드를 다중 수준 코드 갱신 프로토콜을 제안한다. 제안한 기법은, 다양한 수준에서 다양한 크기의 코드에 대한 동적 재구성을 지원한다. 본 논문의 시뮬레이션 결과를 통하여, MCUP을 사용하였을 때에 전체 센서 네트워크에서의 코드 갱신이 에너지 효율적으로 이루어질 수 있음을 보인다.
키워드: 동적 재구성, 센서 운영체제, 원격 코드 갱신, 무선 센서 네트워크

ABSTRACT

Wireless sensor networks are sensing, computing, and communication infrastructures that allow us to sense events in the harsh environment. The networks consist of many deployed sensor nodes. Each sensor node senses and transmits the sensed data to the administrator or base station of the networks. The sensor nodes are generally remotely-deployed, and therefore, software update must be done at run-time via communication channel. The software code update protocol should be energy-efficient to maximize lifetime of the sensor nodes. In this paper, we present a MCUP, which is a multi-level code update protocol for resource-constrained sensor nodes. MCUP enables energy-efficient code update by supporting multi-level code management. Our simulation results show that MCUP can reduce energy consumption compared with the existing one-level code update schemes.
Key words: Dynamic reconfiguration, Sensor operating system, Remote code update, Wireless sensor networks

1. 서론

무선 센서 네트워크는 사람과 자연 환경간의 통신 수단으로, 자연의 여러 가지 정보를 센싱하고, 이러한 정보를 수집하여 인간이 필요로 하는 형태로 정보를 가공하여 전달하여 주는 네트워크이다[1]. 이러한 무선 센서 네트워크는 대상 환경에 뿌려진 다수의 무선 센서 노드들로 이루어진다. 각 센서 노드는 비용 효율성의 이유로 매우 제한적인 하드웨어로 구성되어야 한다. 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 구성된다[2]. 이들은 실제로, 산불 감지 시스템[3], 교량 진동 감지 시스템[4], 학교[5], 주차장 등의 다양한 환경에서 사용된 바 있다.

센서 네트워크에서, 각 센서 노드에 탑재되어있는 소

프트웨어는 다양한 이유로 갱신 혹은 수정될 필요가 있다. 예를 들면, 운영체제 혹은 디바이스 드라이버의 업그레이드, 응용 프로그램의 탑재 및 기능 수정, 그리고 디버깅 및 유지 보수 등의 이유로 소프트웨어의 동적 재구성이 필요하다. 하지만, 센서 노드들을 재 취합하여 프로그램을 일일이 새로 탑재시키는 것은 상당히 어려운 작업일 것이다. 이러한 이유로, 소프트웨어의 갱신은 런타임에 무선 네트워크를 통하여 동적으로 이루어져야 하며, 이러한 갱신 기법은 센서 운영체제 수준에서 지원되어야 한다.

현재까지, 센서 노드를 위한 다양한 소프트웨어 동적 재구성 기법들[6-16]이 연구되어왔다. 이들은 크게 전체 이미지 갱신 기법[6,7], 점진적 블록 단위 갱신 기법[8,9], 모듈 및 컴포넌트 갱신 기법[10-12], 함수 단위 갱신 기법[13], 그리고 가상 머신 기법[14-16] 등으로 나눌 수 있다. 이러한 기법들은 갱신 수준 및 단위가 고정적이라는 단점을 가진다. 따라서 다양한 센서 응용 소프트웨어들의 특성에 따른 자원 효율적인 코드 갱신을 수행하기 어렵다. 예를 들면, 동적 모듈 기법은 각 모듈의 일부만을 수정하고자 할 때에 비효율적이고, 함수 단위 갱신 기법은 상당한 수의 함수들을 갱신할 때에 비효율적이다. 또한, 점진적 갱신 기법은 한 블록으로부터의 코드 밀림 현상으로 인하여, 실제로는 비효율적인 경우가 많다. 명령어 단위 갱신 기법은 대규모의 코드 갱신 시에 수행 시간의 부하가 상당히 증가할 수 있다. 따라서 어떠한 하나의 코드 갱신 기법으로는 다양한 센서 응용 소프트웨어들의 에너지 효율적인 코드 갱신을 수행하기 힘들게 된다.

본 논문에서는, 자원 제약하의 센서 노드를 위한 다중 수준 코드 갱신 기법으로, MCUP를 제안한다. MCUP는 다양한 센서 응용들에 대한 에너지 효율적인 코드 갱신을 위하여, 전체 이미지, 모듈, 함수, 그리고 각 명령어 단위 갱신 기능을 지원한다. 이를 활용하여, 갱신 되어야 할 소프트웨어의 형태 및 크기에 따른 적당한 수준의 코드 갱신을 수행할 수 있다. 본 논문의 성능 측정 결과를 바탕으로, 제안한 기법을 사용하면 기존의 고정적인 수준의 코드 갱신 기법들에 비하여 에너지 사용량을 줄일 수 있음을 보인다. 또한, 보다 다양한 선택 가능 항을 제공함으로써, 런-타임에 응용의 성격이 바뀌더라도 적응성 있게 동작할 수 있음을 보인다.

본 논문의 구성은 다음과 같다. 2절에서는 본 논문과 관련된 여러 연구들을 살펴본다. 3절에서는 제안한 기법의 설계 방식에 대하여 자세히 다룬다. 4절에서는 성능 평가 실험 환경 및 결과가 나타나있다. 마지막으로, 5장에서 결론을 보이고, 향후 연구 계획에 대해서도 간략히 살펴본다.

2. 관련 연구

본 절에서는, 무선 센서 노드를 위한 동적 코드 갱신 기법과 관련된 다양한 연구를 보인다. 이를 통하여, 본 논문에서 제안하는 MCUP와 기존 기법들의 관계도 살펴본다.

현재까지 다양한 코드 생신 기법들이 연구되었고, 이들은 설계 및 동작 방식에 따라 분류해볼 수 있다. 다음의 세부 분류에 따라 기존의 관련 연구들을 간략히 설명한다.

2.1. 전체 및 점진적 이미지 갱신

TinyOS의 경우에는, 하나의 바이너리 이미지로 센서 운영체제, 드라이버 및 응용 프로그램 모두가 구성된다. 이러한 운영체제를 위한 가장 단순한 코드 갱신 기법으로, Deluge[6]가 연구되었다. 이 기법은 전체 이미지를 한번에 갱신하는 방법이다. 일반적으로, 센서 운영체제를 이루는 전체 코드 이미지의 크기는 약 30~40KB 정도이다. 따라서 일부분만 수정된 경우에 이 기법을 사용하는 것은 이미지 전송 및 탑재 비용의 측면에서 매우 비효율적으로 동작한다.

위의 문제를 완화하기 위한 방법으로, 전체 이미지 중에서 수정된 내용만 전송하는 기법들[8,9]이 연구되었다. 이들은 블록 단위로 전체 이미지를 분할 후, 각 이미지에 대한 해시 값의 비교를 통하여 바뀐 부분을 알아내어 전송하는 방식으로 동작한다. 이론상으로는 상당히 효율적으로 동작할 것으로 보이나, 실제로는 아래의 그림 1에 보인 예제 상황과 같이 코드 밀림 현상이 발생하게 되고, 이 때문에 실제로 수정된 양보다 훨씬 많은 양의 코드를 갱신해야 한다.

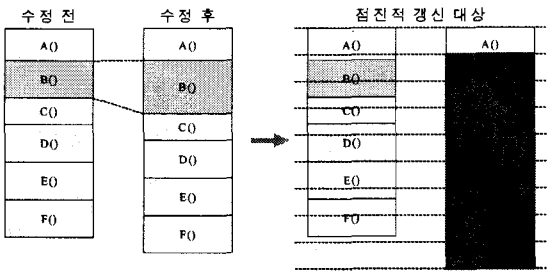

그림 1. 점진적 이미지 갱신 시, 코드 밀림 현상에 의한 문제 발생 예제

2.2. 모듈 및 컴포넌트 갱신

TinyOS는 단 하나의 이미지로 구성되기 때문에, 모듈 단위로 코드를 갱신하기 어렵다는 단점이 존재한다. 이를 개선하기 위하여 동적 모듈의 추가/제거를 지원하는 센서 운영체제인 SOS[10] 및 Contiki[11]가 등장하였다. 이들은 기본 운영체제의 이미지에, 추가적인 모듈을 탑재할 수 있도록 하였고, 이를 통하여 동적인 코드 재구성을 가능하게 하였다. 또한, FlexCup[12] 기법에서는, TinyOS를 이루는 정적 컴포넌트들을 동적인 재구성이 가능한 형태로 수정하였다. 이를 통하여, 기존의 전체 및 점진적 이미지 갱신 기법들 보다 세밀한 코드 갱신을 지원한다. 그러나 각 모듈의 일부분만 수정된 경우에는 이 기법도 역시 비효율적일 수 있다.

2.3. 함수 수준 갱신

좀 더 세밀한 코드 갱신을 위하여, [13]에서 함수 단위 갱신 기법이 제안되었다. 이 기법을 통하여 매우 작은 크기의 코드 갱신도 수행할 수 있으나, 수정된 함수의 크기가 변경되면 함수의 주소가 달라질 수 있다. 이 경우에는 해당 함수를 접근하는 모든 호출 명령어(CALL)들에 대하여 접근 주소 값의 수정에 따른 오버헤드가 발생한다.

2.4. 가상 머신 기반 바이트 코드 갱신

가상 머신 기법은, 갱신해야 할 코드의 크기를 최소화하는 데에 주 목적을 갖고 있다. 대표적인 센서 노드를 위한 가상 머신 기법으로, Mate VM[14], VM*[15], 그리고 DVM[16] 등이 존재한다. 앞의 네이티브 코드를 갱신하는 기법들과 비교하면, 코드의 크기가 작아진다는 장점이 있으나, 바이트 코드의 번역 및 해석, 그리고 실행 비용이 상당히 크다는 단점이 존재한다. 이 기법은 또한, 운영체제 수준의 코드의 갱신은 지원하지 못하는 단점이 존재한다. 게다가, 가상 머신 기법은 네이티브 코드 갱신 기법과는 함께 지원되기 힘들다. 따라서 우리가 제안할 MCUP에서는 가상 머신 기법을 제외하였다.

3. 다중 수준 코드 갱신 기법의 설계

본 절에서는 제안할 MCUP의 설계 및 동작 방식을 내부 구조 및 몇 가지 동작 예제와 함께 자세히 설명한다.

3.1. 무선 센서 노드의 요구사항

새로운 코드 갱신 기법을 설명하기에 앞서, 플랫폼의 요구사항들을 살펴본다. 코드 갱신 기법의 설계 시에 고려할 무선 센서 노드 환경의 특징으로 인한 중요한 요소는 다음과 같다.

▷ 에너지 사용량 고려: 센서 노드는 제한적인 배터리로 동작해야한다. 따라서 무선 통신 비용, CPU 실행 비용, 그리고 메모리 읽기/쓰기, 지우기 비용 등을 고려하여야 한다.

▷ 메모리 사용량 고려: 센서 노드는 매우 작은 크기의 RAM 및 FLASH(ROM) 메모리 공간으로 구성된다. 따라서 공간의 효율성도 고려대상이 된다.

▷ 다양한 소프트웨어 고려: 센서 노드에서 수행될 수 있는 운영체제, 응용 프로그램, 그리고 디바이스 드라이버 수준 등의 다양한 소프트웨어들에 대한 코드 갱신을 지원해야 한다.

위와 같은 요구사항들은 단 하나의 코드 갱신 기법으로는 만족시키기 어렵다. 이를 만족시키기 위하여, 본 논문에서는 다중 수준의 코드 갱신 기법을 설계하게 되었다. 본 논문의 기법은 코드 갱신의 범위를 다중 수준에서 가능하게 함으로써, 다양한 소프트웨어들의 에너지 및 메모리 효율적인 동적 재구성을 가능하게 한다.

3.2. 다중 수준 코드 갱신 기법

그림 2는 본 논문에서 제안하는 MCUP의 설계를 간략히 보인다. MCUP이 지원하는 각 수준별 갱신 기법은 '전체 이미지 갱신' > '모듈 갱신' > '함수 갱신' > '명령어 갱신' 의 포함 관계를 가진다. 다음은 각 수준별 갱신 기법을 실제로 구현하기 위하여 필요한 내부 자료 구조 및 동작 방식을 보인다.

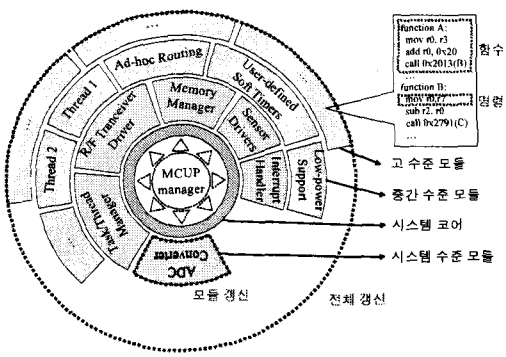

그림 2. 제안한 다중 수준 코드 갱신 기법의 설계도

▷ 전체 이미지 갱신: 일반적으로, 모든 내용을 새로이 구성하고 시스템을 재시작하는 경우에 해당된다.

▷ 모듈 갱신: 하나의 모듈 내부의 기능을 수정한 경우, 혹은 새로운 모듈을 탑재시킬 경우에 해당된다.

▷ 함수 갱신: 모듈의 적은 부분만이 수정된 경우에 사용하기 좋은 방법으로, 함수 단위로 갱신하는 경우에 해당된다.

▷ 명령어 갱신: 함수의 인자 전달 값이 바뀐다거나, 초기 설정 값이 바뀌는 것과 같이, 함수의 일부분 부분만이 수정된 경우에 사용하기 좋은 방법이다. 명령어 단위로 갱신하는 경우에 해당된다.

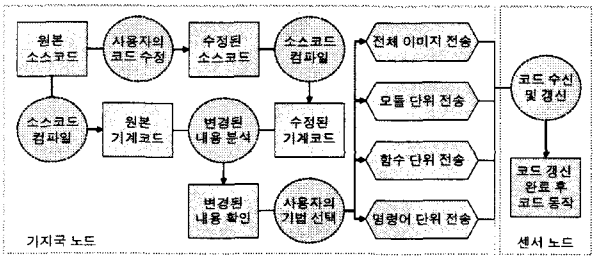

그림 3. MCUP을 사용할 때의 코드 갱신 동작 흐름도

위의 그림 3은 MCUP을 사용하여 코드를 갱신하는 경우에 대하여 세부 동작 흐름을 보인다. 사용자가 코드를 일부 수정하여 센서 노드에 탑재시키려는 경우가 이 예에 해당한다. 아래에 이 과정을 순차적으로 설명한다.

(1) 먼저, 사용자가 코드를 수정한 후, 컴파일 작업을 수행한다. 이 작업은 기지국 노드와 같은 자원이 풍부한 시스템에서 수행된다.
(2) 작성된 이미지와 기존에 센서 노드에 탑재시켰던 이미지와의 차이를 분석한다. 이 작업 역시 기지국 노드에서 이루어지므로, 센서 노드의 자원은 전혀 사용하지 않는다.
(3) 수정된 부분을 사용자가 확인 후, 네 가지 수준의 기법들 중에서 적절한 것 하나를 고른다.
(4) 사용자에 의해 선택된 기법을 사용하여 갱신될 코드를 센서 노드에 전송한다.
(5) 센서 노드는 수신된 코드를 메모리 공간에 탑재한다.

이와 같은 방식으로 동작하여, 네 가지의 코드 갱신 기법 중에 하나를 사용자에 의해 선택하게 된다. 현재, 비용 분석에 기반을 둔 시스템에 의한 자동 선택 방법을 연구 중이며, 이는 향후의 논문에서 다룰 내용이다.

4. 성능 평가

본 절에서는, 제안한 MCUP 기법이 지원하는 네 가지 기법에 대한 성능을 평가한다. 평가 요소로는 에너지 효율성, 수행 시간, 그리고 공간 사용량 등이 된다. 제안한 기법을 평가하기 위하여 몇 가지 예제 상황을 사용하였고, 이는 시뮬레이션을 통하여 측정되었다.

4.1. 시뮬레이션 환경

표 1. 성능 측정에 사용된 값들

항목	크기 (bytes)
전체 이미지	3,040
각 모듈 이미지	320
각 모듈 메타데이터	60
각 함수 이미지	40
각 함수 메타데이터	20
각 명령어	2
각 명령어 메타데이터	4

위의 표 1과 아래의 그림 3은 본 논문에서 사용한 실험 환경에 대한 자세한 정보를 보인다. 총 8개의 모듈이 존재하고, 각 함수 및 모듈의 크기는 위의 표에 보인 바와 같이 가정하였다. 이 가정 내용은 최근에 등장한 여러 센서 운영체제에서 동작하는 커널 및 응용 모듈 소프트웨어의 크기에 기반을 두었다[10,11]. 본 논문의 성능 측정 시뮬레이션을 위하여, 무선 센서 네트워크 시뮬레이터의 하나인 '센서메이커'[17]를 사용하였다. 이는 2004년부터 현재까지, 여러 가지 논문들의 시뮬레이션에 사용된 시뮬레이터이다. 무선 통신 시 발생하는 송수신 비용 및 네트워크 라우팅 시에 발생하는 패킷 헤더 전송 비용 등은 센서메이커의 기본 설정을 사용하였다[1].

다음으로, 시뮬레이션을 수행하기 위하여 다음과 같은 동적 코드 갱신 상황을 가정하였다. 각 센서 노드는 갱신 비율 'α'에 따라 동적인 재구성을 수행하며, 이는 각

1) 센서 노드 개수: 100, 센싱 대상 환경: 100m*100m, 센서 노드의 초기 에너지: 1J, 그 외의 내용은 센서메이커 참조

센싱 라운드에 따른 통계적 확률이다. 만약 α가 0.1 이면, 평균적으로 10번의 라운드에 한 번 정도 코드 갱신을 수행하는 것이 된다. 또한, 갱신 시점에 변경되는 소스코드의 비율을 나타내는 'β'를 사용하였다. 이는 각 명령어 단위로 적용되며, β가 0.01 이면 평균적으로 100개의 명령어 중에 하나가 갱신되는 상황을 떠올리면 된다. 마지막으로, 코드 갱신의 인접성을 나타내기 위한 요소로, δ를 사용하였다. 만약 δ가 0.5 면, 갱신된 코드 바로 뒤의 코드가 갱신될 확률이 0.5 라는 것이 된다. 위의 α와 β, 그리고 δ를 통한 성능 평가 방법은 좋다고 말할 수는 없다. 이는 향후의 연구에서 보다 세밀한 모델링 및 실제 구현 후의 실험을 통하여 보완할 것이다.

4.2. 시뮬레이션 결과 및 분석

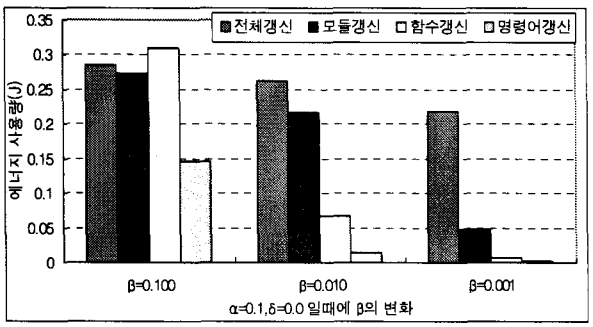

그림 4. β를 변화시킬 때의 에너지 사용량

그림 4는 α를 0.1, δ를 0으로 고정시키고 β를 변화시킬 때, MCUP 기법이 제공하는 코드 갱신 기법들의 에너지 사용량을 나타낸 것이다. 총 100라운드를 수행한 결과이고, 이 결과를 통하여 코드 갱신 비율인 β에 따라 각 코드 갱신 기법의 성능이 달라짐을 확인할 수 있다.

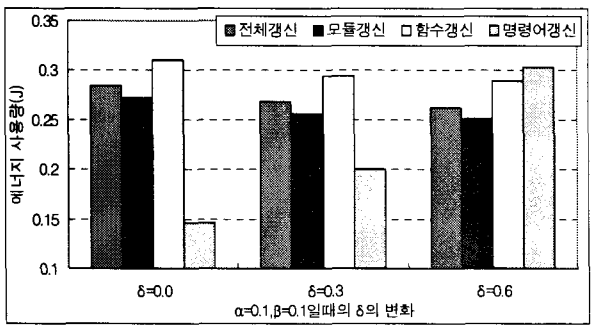

그림 5. δ를 변화시킬 때의 에너지 사용량

그림 5는 α를 0.1, β를 0.1로 고정시키고 δ를 변화시킬 때, MCUP 기법의 에너지 사용량을 나타낸 것이다. 총 100라운드를 수행한 결과이고, 이를 통하여 코드 갱신의 인접성에 따른 수행 결과를 살펴볼 수도 있다.

마지막으로, 그림 6은 α를 0.1, β를 0.001, 그리고 δ를 0.5로 두었을 때에, 최소 비용의 코드 갱신을 수행하는 기법을 사용자가 선택하여 사용한 상황의 에너지 사용량을 보인다. 이 결과를 바탕으로, 제안한 MCUP 기법을 사용하면 사용자에게 다양한 기법들을 제공함으로써, 보다 에너지 효율적인 코드 갱신 기법을 선택하게 할 수 있음을 알 수 있다. 이러한 방식으로, 실제로 에너지 사용량을 감소시킬 수 있고 전체 센서 네트워크의 수명도 연장시킬 수 있다.

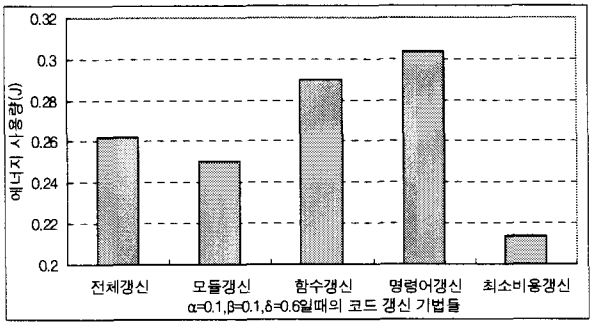

그림 6. 최소 비용을 갖는 코드 갱신 기법을 선택하였을 때의 에너지 사용량 비교

5. 결론 및 향후 연구 계획

무선 센서 네트워크는 다수의 센서 노드들로 구성되고, 각 센서 노드는 원격지에 뿌려진 후에 동작한다. 각 센서 노드에 탑재되는 소프트웨어를 변경하려면 무선 통신을 통하여 동적으로 코드 갱신을 수행해야 한다. 현재까지, 다양한 코드 갱신 기법들이 연구된 바 있으나, 다양한 소프트웨어들에서 모두 에너지 효율적으로 동작하기에는 어려움이 따른다.

본 논문에서는, 자원 제약하의 센서 노드를 위한 다중 수준 코드 갱신 기법으로, MCUP 기법을 제안한다. 이 기법은 다양한 센서 응용들에 대한 에너지 효율적인 코드 갱신을 위하여, 전체 이미지, 모듈, 함수, 그리고 각 명령어 단위 갱신 기능을 지원한다. 이를 활용하여, 갱신 되어야 할 소프트웨어의 형태 및 크기에 따른 적당한 수준의 코드 갱신을 수행할 수 있다. 성능 측정 결과를 바탕으로, 제안한 MCUP 기법을 사용하면 기존의 코드

갱신 기법들에 비하여 에너지 사용량을 줄일 수 있음을 보인다. 또한, 보다 다양한 선택 가능 항을 제공함으로써, 런-타임에 응용의 성격이 바뀌더라도 적응성 있게 동작할 수 있음을 보인다.

현재, 제안한 다중 수준 코드 갱신 기법에서 사용자가 선택을 수행하는 것이 아닌, 시스템에서 자동적으로 선택해 주는 방법을 연구 중에 있다. 이를 위하여 현재 각 수준별 기법의 비용 분석을 수행할 것이다.

6. 시뮬레이션에 사용한 소스코드 다운로드

본 논문에서 사용한 센서메이커 시뮬레이터 및 작성한 소스코드는 다음의 URL에서 다운로드 할 수 있다.

URL: http://ssrnet.snu.ac.kr/~shyi/mcup_kisse07.zip

[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] Jessica D. Lundquist, Daniel R. Cayan, Michael D. Dettinger, Meteorology and Hydrology in Yosemite National Park: A Sensor Network Application, Lecture Note in Computer Science", Vol. 2634, pp.518--528, 2003.

[4] Pedro Jose Marron, Andreas Lachenmann, Daniel Minder, Jorg Hahner, Robert Sauter, Kurt Rothermel, TinyCubus: A Flexible and Adaptive Framework for Sensor Networks, Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN 2005), 2005.

[5] Mani Srivastava and Richard Muntz and Miodrag Potkonjak, Smart Kindergarten: Sensor-based Wireless Networks for Smart Developmental Problem-solving Environments, The 7th Annual International Conference on Mobile Computing and Networking, 2001.

[6] J. W. Hui, D. Culler, The dynamic behavior of a data on the low memory sensor nodes, In Proceedings of the second international conference on Embedded Networked Sensor Systems, 2004.

[7] XNP manual site, http://www.tinyos.net/tinyos-1.x/doc/Xnp.pdf, 2007.

[8] J. Jeong, D. Culler, Incremental network programming for wireless sensors, In Proceedings of the First IEEE Communications Society Conference on Sensor and Ad-Hoc Communications and Networks(SECON), 2004.

[9] N. Reijers, K. Langendoen, Efficient code distribution in wireless sensor networks, In Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, 2003.

[10] Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, Mani B. Srivastava, A Dynamic Operating System for Sensor Nodes, The 3rd International Conference on Mobile Systems, Applications, and Services(MobiSys'05), pp.163-176, 2005.

[11] A. Dunkels, B. Gronvall, T. Voigt, Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors, The 1st IEEE Workshop on Embedded Networked Sensors(EWSN'04), 2004.

[12] P. Jose Marron and M. Gauger and A. Lachenmann and D. Minder and O. Saukh and K. Rothermel, Flexcup: A flexible and efficient code update mechanism for sensor networks, The 3rd European Workshop on Wireless Sensor Networks(EWSN'06), 2006.

[13] J. Koshy and R. Pandey, Remote incremental linking for energy-efficient reprogramming of sensor networks, The 2nd European Workshop on Wireless Sensor Networks(EWSN'05), 2005.

[14] Philip Levis, David Culler, Mate: a Virtual Machine for Tiny Networked Sensors, International Conference on Architectural Support for Programming Languages and Operating Systems, pp.85-95, Oct. 2002.

[15] J. Koshy and R. Pandey, Vm*: Synthesizing scalable runtime environments for sensor networks, The 3rd ACM Conference on Embedded Networked Sensor Systems(SENSYS'05), 2005.

[16] Rahul Balani, Chih-Chieh Han,Ram Kumar Rengaswamy, Ilias Tsigkogiannis, Mani Srivastava, Multi-level Software Reconfiguration for Sensor Networks, EMSOFT'06, Oct. 2006.

[17] 이상호, 이승우, 민홍, 조유근, 홍지만, 센서메이커: 세밀하고 확장성 있는 실험을 위한 무선 센서 네트워크 시뮬레이터, 하계 컴퓨터 통신 워크샵 (SWCC'2007), 2007.