논문 2009-04-16

무선 센서 네트워크상에서 코드뱅킹 및 델타이미지 기반의 효율적인 센서노드 소프트웨어 업데이트 기법

(An Efficient Software Update Technique with Code-Banking & Delta-Image for Wireless Sensor Networks)

남 영 진*, 남 민 석, 박 영 균, 김 창 훈, 이 동 하
(Young-Jin Nam, Min-Seok Nam, Young-Kyun Park, Chang-Hoon Kim, Dong-Ha Lee)

Abstract : Software update has been regarded as one of fundamental functions in wireless sensor networks. It can disseminate a delta-image between a current software image operating on a sensor node and its new image in order to reduce an update image(transmission data) size, resultantly saving energy. In addition, code-banking capability of micro-controllers can decrease the update image size. In order to maximize the efficiency of the software update, the proposed scheme exploits both the delta-image and the code-banking at the same time. Besides, it additionally delivers a recovery delta-image to properly handle abnormal conditions, such as message corruptions and unexpected power-off during the update.

Keywords : Wireless sensor networks, Software update, Delta-image, Code-banking

Ⅰ. 서 론

무선 센서 네트워크는 일반적으로 원격지나 직접 정보를 얻어올 수 없는 위험하거나 고립된 지역 또는 가혹한 환경에서의 다양한 정보를 얻기 위해 사용된다. 이러한 네트워크를 구성하는 센서 노드들은 대체로 컴퓨팅 능력 및 이용가능 메모리 그리고 배터리 측면에서 자원 제약적인 특징을 가지며, 한번 배치되면 장기간 동안 사람의 간섭 없이 동작하여야 한다. 그러나 센서노드상의 동작하는 소프트웨어는 버그 수정이나 패치, 노드 작업 변경 또는 보안상의 문제로 업데이트가 필요하다. 이렇게 무선 센서 네트워크상에서 센서노드의 소프트웨어를 업데이트 하는 것을 무선기반 소프트웨어 업데이트라 하며 이를 위하여 현재까지 다방면의 연구가 진행되어 왔다[1-3]. 특히, 델타이미지 업데이트 기법은 센서노드 상에 동작하는 현재 소프트웨어 이미지와 새로이 업데이트 될 이미지의 델타이미지를 이용하여 업데이트함으로써 업데이트 이미지 전송에 소모되는 에너지와 대역폭의 사용을 줄일 수 있다[4,5].

한편, 무선 센서 네트워크를 위한 무선기반 소프트웨어 업데이트 기법은 여러 가지 제약된 자원을 가지는 센서노드들의 하드웨어적 특성을 고려하여 설계되어야 하며, 이때 주로 고려대상이 되는 요인은 마이크로컨트롤러에 내장된 내부 플래시 메모리와 외부 메모리의 사용 측면이다. 특히, 마이크로컨트롤러 측면에서 8051 기반의 마이크로컨트롤러는 메모리 어드레싱 영역이 64KB로 제한되어 있어, 프로그램 크기가 64KB를 초과하는 경우 코드뱅킹 기능이 사용 된다. 코드뱅킹 기능은 전체 내부 플래시 메모리를 어드레싱 가능한 크기인 32KB의 뱅크들로 나누고 실행 이미지를 각 뱅크에 적절히 분할하여 저장한다[8]. 이러한 특성은 소프트웨어 업데이트 시 뱅크별로 이미지를 업데이트 할 수 있는 장점을 가져다줄 수 있다.

* 교신저자(Corresponding Author)
논문접수 : 2009. 01. 14., 수정일 : 2009. 02. 20., 2009. 07. 30., 채택확정 : 2009. 09. 04.
남영진, 남민석, 박영균, 김창훈 : 대구대학교
이동하 : 대구경북과학기술원
※ 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업 (NIPA-2009-C1090-0902-0045)과 지식경제부 및 한국산업기술진흥원의 지역혁신인력양성사업으로 수행된 연구결과임

본 논문은 8051 마이크로컨트롤러가 탑재된 센서노드 상에서 코드 뱅킹 기능을 이용하여 에너지 및 대역폭 사용면에서 효율적인 델타이미지 업데이트 기법을 제안한다. 또한, 소프트웨어 업데이트 과정에서 메시지 전송 실패 혹은 전원불량 등으로 인해 업데이트 작업이 비정상적일 경우 기존 이미지로 복구하여 실행 이미지의 정상 동작을 보장하는 기법도 함께 제안한다.

Ⅱ. 배경지식 및 관련연구

1. 8051 마이크로컨트롤러의 뱅킹 구조

무선 센서 네트워크를 구성하는 각 센서노드들은 다양한 종류의 마이크로컨트롤러가 탑재되어 동작될 수 있다. 특히 8051 기반의 마이크로컨트롤러는 메모리 어드레싱 영역이 64KB로 제한되어 있어, 프로그램 크기가 64KB를 초과하는 경우 코드 뱅킹 기능이 사용 된다. 코드뱅킹 기능은 전체 내부 플래시 메모리를 어드레싱 가능한 크기인 32KB의 뱅크들로 나누고 실행 이미지를 각 뱅크에 적절히 분할하여 저장한다[8]. 이러한 특성은 소프트웨어 업데이트 시 뱅크별로 이미지를 업데이트 할 수 있는 장점을 가져다줄 수 있다. 그림 1은 일반적인 8051 마이크로컨트롤러의 뱅킹 구조를 나타낸다. 여기서 뱅크 0은 항상 어드레싱 가능 영역에 맵핑되며, 노드 부팅 시 초기 설정 값으로 뱅크 0과 1이 64KB 어드레싱 영역에 맵핑된다. 나머지 뱅크들은 함수 호출 등에 의한 필요시마다 뱅크 1과 교체되며 동작하게 된다.

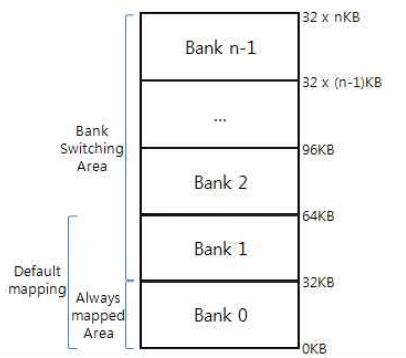

그림 1. 8051 마이크로컨트롤러의 뱅크 구조
Fig. 1. A bank structure of the 8051 micro-controller

2. 델타 이미지 기반 소프트웨어 업데이트

현재까지 무선 센서 네트워크를 위한 다양한 소프트웨어 업데이트 기법들이 개발되어 왔다. 특히, 업데이트 될 새로운 이미지와 기존의 이미지 사이의 차이가 나는 부분인 델타 이미지를 이용한 업데이트 기법을 델타 이미지 기반 소프트웨어 업데이트 기법이라 하며, 이러한 방법에 대한 프레임워크는 일반적으로 그림 2와 같이 이미지 인코딩, 이미지 송수신, 디코딩 및 리프로그래밍 순으로 이루어지며[4,5], 일반적으로 이러한 델타 이미지 기반 소프트웨어 업데이트 방법은 모든 노드들이 단일 채널을 서로 공유하는 네트워크상에서 업데이트 시 노드상의 기존 이미지와 업데이트 될 새로운 이미지 사이의 델타 이미지를 전송함으로써 전송되는 이미지의 크기를 감소시켜 채널 이용 상의 경쟁을 줄이고 전송 시 소모되는 에너지를 줄일 수 있다는 이점이 있다.

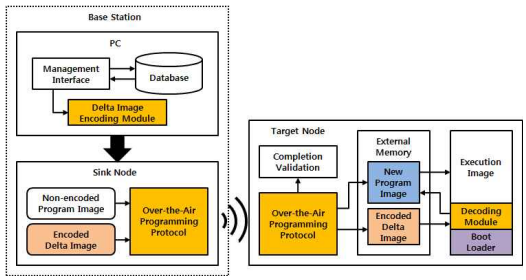

그림 2. 델타 이미지 기반 소프트웨어 업데이트 프레임워크
Fig. 2. A delta-image based software update framework

3. 기존 소프트웨어 업데이트 기법

현재까지 연구된 대표적인 소프트웨어 업데이트 기법으로는 XNP, MNP, MOAP, Deluge, Sprinkler, Infuse, Aqueduct, Incremental, Trickle, TinyCubus, Hermes, P. Rickenbach et al. 등이 있으며[1,2,5,9,10], 이들 중에서 델타 이미지 기반 소프트웨어 업데이트 기법은 Incremental, Hermes, P. Rickenbach 등에 의한 방법이다.

Incremental[9]과 Hermes[10]는 Rsync 알고리즘을 기반으로 하여 델타 이미지를 생성하는데, Rsync 알고리즘은 구현상에서는 간단한 이점이 있으나, 특정 블록 단위로 이미지를 비교하여 델타 이미지를 생성하기 때문에 그 특정 블록 내에 1바이트라도 다른 부분이 존재하면 그 블록 전체를 업데

이트해야만 하는 단점이 있다. 반면 P. Rickenbach 등에 의한 방법[5]에서 사용된 VCDIFF기반의 방법은 Rsync 알고리즘을 이용한 방법에 비해 인코딩 시 많은 계산 량이 요구된다는 단점이 있으나 델타 이미지의 크기 상에서 상당히 작은 용량으로 이미지의 생성이 가능하다. 본 논문에서는 이러한 VCDIFF기반의 델타이미지 생성 방법을 이용함과 동시에 구현별 이미지 업데이트지 생해 분이 존재 마이크로알고리즘은 코드뱅킹 특성을 이용하여 이미지 전송량을 최소화하고 업데이트 과정 중 타겟 노드은 실행 프로그램 이미지전송 정상적으로 업데이트 되지 않을 경우 복구 이미지를 이용하여 타겟 노드 상의 특정 뱅크 이미지를 업데이트 이전의 이미지로 복구할 수 있는 방법을 다룬다.

Ⅲ. 제안 기법

본 논문에서는 8051마이크로컨트롤러의 코드 뱅킹 특성을 사용하여 센서노드 상에서 델타이미지 기반의 뱅크별 코드 업데이트를 함으로써 이미지 전송에서의 에너지 소모 및 대역폭 사용량을 최소화하며, 업데이트 중 예상치 못한 상황 발생으로 정상적인 업데이트가 되지 않았을 경우 대처 방안에 대하여 제안한다.

1. 제안 기법의 구조

그림 3은 제안된 기법의 기본 구조로서 업데이트를 위한 이미지의 관리를 위한 모듈과 업데이트 이미지의 전송을 위한 통신 모듈, 그리고 타겟노드를 업데이트 하기위한 업데이트 모듈로 구성된다. PC측의 업데이트 이미지 관리 모듈은 이미지 간 유사도를 바탕으로 뱅크 별 이미지의 인코딩 여부를 판단하고 인코딩하여 델타 이미지와 복구 이미지를 생성한다. 생성된 이미지들은 싱크노드의 업데이트 통신 모듈을 통해 타겟노드로 전송된다. 업데이트 이미지를 수신 받은 타겟노드는 노드 업데이트 모듈에 의해서 업데이트되며, 이상상태 발생 시 복구 이미지를 이용함으로써 기존 이미지로 복구된다.

1. 업데이트 이미지 관리 모듈

일반적으로 델타 이미지를 이용하여 차이가 있는 부분만을 업데이트 하게 되면 원래 이미지를 업데이트 하는 것 보다 효율적이다. 그러나 인코딩 시에는 인코딩으로 인한 추가 코드 발생으로 약간의 오버

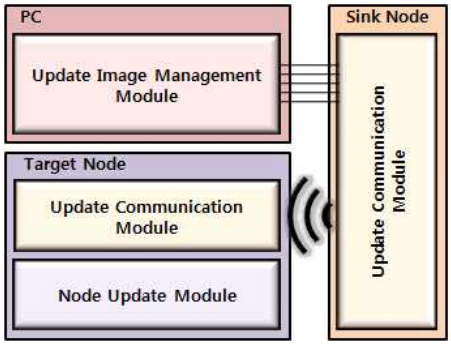

그림 3. 제안된 기법의 전체 구조
Fig. 3. An overall architecture of the proposed technique

헤드가 발생하는데, 기존 이미지와 새로운 이미지들의 유사성이 매우 다른 경우에는 추가 코드로 인한 오버헤드가 상대적으로 커지게 되어 원래 이미지를 전송하는 경우 보다 효율이 떨어지는 경우가 발생할 수 있다. 전송의 효율성을 위한 제안 기법에서의 업데이트 이미지 관리 모듈은 그림 4와 같다. 데이터베이스에서는 네트워크 내부의 모든 센서노드에 대한 뱅크 별 이미지 정보를 관리 인터페이스에 제공하며, 관리 인터페이스는 이러한 정보를 바탕으로 타겟노드의 이미지와 새로운 업데이트 이미지 사이의 유사도를 측정한다. 측정된 유사도에 따른 인코딩 이미지 전송량을 측정하고, 업데이트 이미지의 인코딩 여부를 판단한다. 이미지의 인코딩에 사용되는 이미지 인코더는 VCDIFF 기반의 Xdelta 알고리즘[6,7]을 이용하며, 델타 이미지와 복구 이미지를 생성하여 싱크노드의 업데이트 통신 모듈에 전달한다.

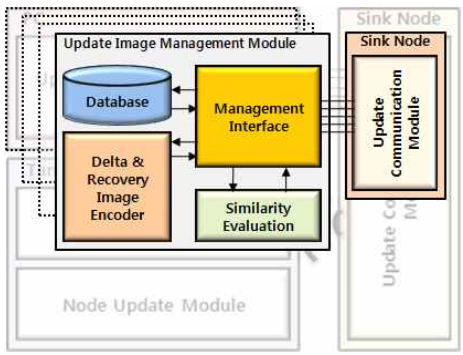

그림 4. 업데이트 이미지 관리 모듈의 구성
Fig 4. The update image management module

2. 업데이트 통신 모듈

소프트웨어 업데이트를 위해 네트워크 내의 모든 노드들은 업데이트 통신 모듈이 필요하며 제안된 기법에서의 업데이트 통신 모듈은 싱크와 타겟 노드 간에 업데이트 이미지를 송수신한다. 싱크노드측의 업데이트 통신 모듈은 업데이트 이미지 관리 모듈로부터 전달 받은 업데이트 이미지를 패킷화하고 통신 프로토콜을 통하여 타겟노드로 송신하며, 타겟노드로부터의 이미지 재전송 요청을 받으면 업데이트 이미지를 다시 패킷화하고 재전송하게 된다. 반면, 타겟노드측의 업데이트 통신 모듈은 싱크노드로부터의 업데이트 패킷을 수신하고 패킷 내의 코드 부를 추출하여 노드 업데이트 모듈에 전달한다. 패킷 수신이 끝나면 노드 업데이트 모듈에 의해 관리되는 외부메모리에 저장된 코드들이 정상적인 이미지인지 확인 과정을 거치며. 비정상적일 경우 싱크노드로 이미지 재전송 요청을 수행한다.

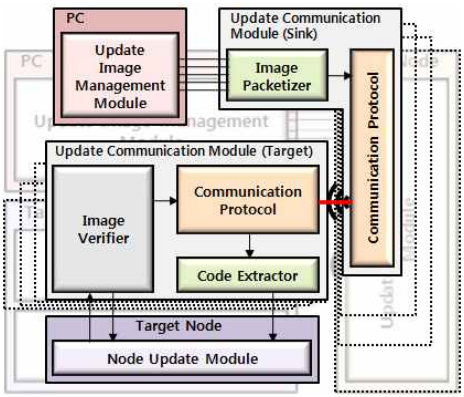

그림 5. 업데이트 통신 모듈의 구성
Fig. 5. The update communication module

3. 노드 업데이트 모듈

노드 업데이트 모듈은 제안된 기법에서 타겟노드상에 동작하는 실행 이미지를 새로운 업데이트 이미지로 전환시키며, 업데이트 시 이상상태가 발생하면 복구 이미지를 이용하여 타겟노드를 복구한다. 그림 6에서 보이는 바와 같이 노드 업데이트 모듈은 타겟노드상의 업데이트 통신 모듈과 데이터를 주고받는다. 업데이트 통신 모듈에서 업데이트 컨트롤러로 새로운 업데이트 코드가 전달되면 업데이트 컨트롤러는 외부 메모리에 저장한다. 업데이트 컨트롤러는 코드의 수신이 끝나면 이미지 검증을 위하여 타겟노드상의 업데이트 통신 모듈로 이미지의 내용을 전달한다. 업데이트 통신 모듈에서 정상적인 이미지라는 결과가 도착하면 인코딩된 이미지의 경우 이미지 디코더에서 디코딩하고 내부 플래시 메모리의 해당 뱅크 영역에 복사한다. 만약 업데이트 중 이상상태 발생으로 업데이트가 중단되면 업데이트 컨트롤러는 복구이미지를 이용하여 노드를 복구시킨다.

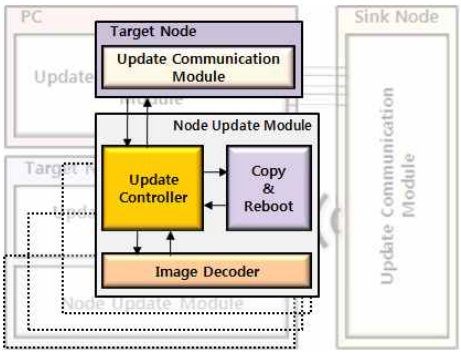

그림 6. 노드 업데이트 모듈의 구성
Fig. 6. The node update module

2. 제안 기법의 업데이트 메커니즘

본 논문에서는 업데이트 이미지의 전송량을 최소화하기 위한 목적을 위하여 코드 뱅킹기능과 델타이미지를 이용한다. 내부메모리는 32KB 단위의 뱅크로 나뉘며, 노드상에 동작하는 코드는 뱅크별로 나뉘어져 저장되는데 업데이트 시 해당 뱅크의 이미지만 업데이트하게 됨으로써 업데이트 이미지의 전송량 및 업데이트 과정에서의 컴퓨팅 량을 줄일 수 있다. 또한 뱅크별 코드 이미지들은 업데이트 시 전송량의 효율을 판단하여 델타이미지를 이용하여 업데이트 할 것인지 원 이미지를 이용하여 업데이트할 것인지 결정한다.

1. 업데이트 이미지의 판정

본 논문에서는 업데이트 이미지의 전송량을 최소화하기 위한 목적을 위하여 업데이트 시 업데이트 이미지 관리 모듈에서 원 이미지를 이용하였을 경우와 델타이미지를 이용하였을 경우의 전송량을 측정하여 두 경우에서 이미지 전송량 측면에서 효율적은 방법을 이용하여 센서노드를 업데이트 하게 된다. 델타 이미지 전송 시 이미지 전송량 TR_d는 새로운 이미지와 타겟노드의 기존이미지 사이의 다른 부분 D_{nt}와 인코딩 시 추가되는 코드 E_a에 의하여 식 (1)과 같이 정리되며, 전송 시 총 이미지 전송량 TR_T는 업데이트 중단 시 복구를 위한 복구이미지 전송량 TR_r이 식 (2)와 같이 정리된다.

$$TR_d = D_{nt} + E_a \quad (1)$$

$$TR_T = TR_d + TR_r \quad (2)$$

따라서 원래의 새로운 이미지에 대한 전송량을 S_n에 대하여 업데이트 시 전송되는 총 이미지 전송량이 $TR_T < S_n$인 경우 델타 이미지를 이용하여 업데이트 하게 되며, 이외의 경우에는 원 이미지를 이용하여 업데이트 하게 된다.

2. 뱅크별 이미지 구성 및 업데이트 순서

센서노드는 통신을 위해 통신 프로토콜이 필요하며 또한 소프트웨어 업데이트 시 통신 프로토콜은 업데이트가 진행되는 동안 데이터 수신을 위해 업데이트로부터 보호되어야 한다. 제안된 업데이트 방법에서 이 통신 프로토콜이 위치한 뱅크의 이미지는 다른 코드들이 위치한 뱅크 이미지가 모두 업데이트 된 후에 마지막으로 업데이트되게 된다. 그림 7은 제안된 업데이트 기법에서의 메모리 구조로써 내부 플래시 메모리는 총 n개의 뱅크로 나누어지며, 뱅크 0에는 업데이트 통신 모듈과 이미지 디코더를 포함하는 코드가 위치하고, 마지막 뱅크에는 외부 메모리에서 새로운 업데이트 이미지를 내부 플래시 메모리로 복사하고 재부팅하는 기능의 코드와 업데이트 컨트롤러를 포함하는 코드가 위치하게 되는데 이 마지막 뱅크의 코드 이미지는 업데이트를 위해 반드시 필요한 코드 이미지로써 업데이트 대상에서 제외되는 부분이다. 이들 이외의 뱅크들은 뱅크 스위칭에 의해 스위칭되는 코드들이 위치한다. 그리고 외부 메모리는 새로운 프로그램 이미지가 저장될 공간과 델타 이미지가 저장될 공간, 그리고 복구 이미지가 저장될 공간으로 나뉜다. 여기서 새로운 프로그램이 저장될 공간의 크기는 내부 플래시 메모리의 뱅크 크기인 32KB가 된다.

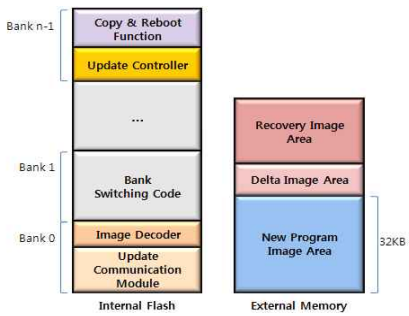

그림 7. 노드상의 메모리 초기화 구조

Fig. 7. Initialized memory structure at a sensor node

3. 이상상태의 복구

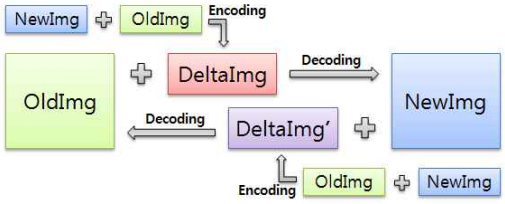

그림 8. 델타 이미지(DeltaImg)와 복구 이미지(DeltaImg')의 관계

Fig. 8. The relation between the delta image(DeltaImg) and the recovery image(DeltaImg')

네트워크를 구성하는 센서노드들은 제한된 컴퓨팅 능력을 가지며 전원을 배터리로 공급하게 됨 으로써 작동 시 데이터 전송 실패, 계산 오류 혹은 전원불량 등으로 예기치 못한 상황이 발생할 수 있으며, 업데이트 시 이러한 상황이 발생하면 센서노드는 정상적인 동작이 불가능하게 된다. 제안된 기법에서의 노드 복구 방법은 뱅크스위칭에 의해 스위칭 대상이 되는 뱅크 이미지들을 대상으로 하며 이 뱅크 이미지들은 업데이트 시 이상상태가 발생하여 업데이트가 정상적으로 되지 않을 경우 복구 이미지를 이용하여 기존의 이미지로 복구가 가능하다. 복구 이미지의 생성 과정은 델타 이미지 생성 과정을 역으로 수행하여 생성되며, 이들의 관계는 그림 8과 같이 서로 역의 관계가 성립하여, 기존 이미지와 델타 이미지를 디코딩하면 새로운 이미지가 생성되며, 새로운 이미지와 복구 이미지를 디코딩하면 기존 이미지가 생성되게 된다.

한편, 복구동작을 수행하기 위해서는 직전까지 업데이트된 뱅크의 수만큼 복구 이미지가 필요하다. 복구를 위한 복구이미지의 수 N_R과 업데이트 되는 뱅크의 수 N_U는 다음과 같은 관계를 갖는다.

$$N_R = N_U - 1 \quad (3)$$

이는 가장 마지막에 업데이트 되는 뱅크 0의 경우 업데이트가 비정상적이면 디코딩 모듈의 정상 동작을 보장할 수 없게 되며 복구이미지를 통한 복구가 불가능함으로 뱅크 0의 복구이미지는 불필요하기 때문이다. 이 복구 이미지 수 N_R은 구현 시 필요한 외부 메모리 크기 S_{EMneed}와 밀접한 관계가 있다. 하나의 뱅크에 대한 델타 이미지와 복구이미지의 크기를 각각 S_D, S_R이라 하면 다음의 관계식이 도출된다.

$$S_{EMneed} = 32KB + S_D + (S_R \times N_R) \qquad (4)$$

따라서 업데이트 되는 뱅크의 수를 낮게 조정함으로써 구현 시 필요한 외부 메모리의 크기를 줄일 수 있다.

4. 이미지 수신 및 노드 업데이트

타겟 노드상에서 업데이트 이미지는 그림 9에서와 같이 두 가지 경우로 수신될 수 있다. 첫 번째는 원 이미지를 이용하여 업데이트 되는 경우로써, 타겟 노드는 일반적인 업데이트 기법에서처럼 수신된 이미지를 외부 메모리의 새로운 프로그램 이미지 저장 공간에 저장하고, 복사 및 리부팅 코드에 의해서 노드상의 내부 플래시 메모리의 이미지는 새로운 코드 이미지로 교체되며 재부팅함으로써 업데이트는 끝이 난다. 다음으로 두 번째 경우는 델타이미지를 이용하여 업데이트 되는 경우로써. 수신된 델타 이미지와 복구 이미지를 외부메모리에 저장하고, 저장한 이미지가 정상적으로 수신되었는지 검사한 다음 새로운 코드 이미지로 디코딩하여 내부 플래시 메모리 영역에 복사한다. 업데이트에 해당되는 뱅크들의 업데이트가 끝나면 전체 업데이트가 정상적으로 완료되었는지를 검사한 후 재부팅되며 업데이트는 종료된다.

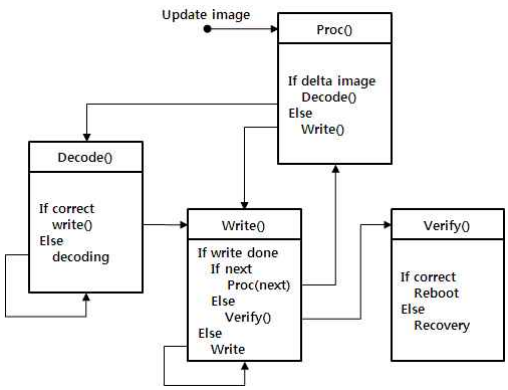

그림 9. 타겟노드의 상태 다이어그램
Fig. 9. A state diagram of a target node

한편, 만약 싱크노드에서 이상상태 발생으로 업데이트 이미지가 일부만 수신된 후 더 이상의 업데이트 수신이 없으면 타겟노드는 일정 기간 동안 다음 이미지의 수신을 기다리며, 이 기간 동안 다음 이미지의 수신이 이루어지지 않으면 복구 이미지를 이용하여 직전까지 업데이트 된 뱅크를 기존 상태로 복구한다. 한편, 업데이트 중 타겟노드에서 일시적인 전원 미공급 등의 이상상태가 발생하면, 타겟노드는 부팅되자마자 업데이트 완료 여부를 검사하고 업데이트가 완료 되지 않은 경우 직전 업데이트 과정 이 후의 부분부터 업데이트를 재개한다. 만일 업데이트를 재개할 수 없는 경우는 직전 업데이트에서 변경된 부분을 복구 이미지를 통하여 복구한 다음 리부팅을 함으로써 기존 이미지로의 복구가 완료 된다.

Ⅳ. 성능 평가

제안된 방법에 대한 성능 평가를 위하여 본 논문에서는 8051기반 RF SoC MG2455와 64KB 외부 EEPROM인 AT24C512 및 7-세그먼트를 탑재한 센서 노드를 대상으로 하여, KEIL μVision 버전 2.40a 상에서 부분적인 구현을 통하여 이미지 전송량 측면에서 테스트하였다. 구현된 플랫폼의 내부 플래시 메모리는 총 4개의 뱅크로 이루어져 있으며 본 성능 평가에서는 뱅크 2는 이용되지 않는다. 뱅크 0에는 업데이트를 위한 통신프로토콜과 함수 콜 및 뱅크 스위칭을 위한 점핑 테이블 등이 포함되었으며, 뱅크 3에는 이미지 복사 및 재부팅을 위한 코드들이 포함되었다. 또한 데이터 전송 프로토콜은 한 번의 전송 당 64바이트를 전송한다. 실험을 위해 그림 10과 같이 7-세그먼트를 제어하는 간단한 프로그램들을 이용하였는데, Seg_0과 Seg_1은 각각 0번과 1번 세그먼트만을 제어하는 프로그램들이며, Seg_All은 7-세그먼트 전체를 제어하는 프로그램으로써, 뱅크 1에 포함되었다.

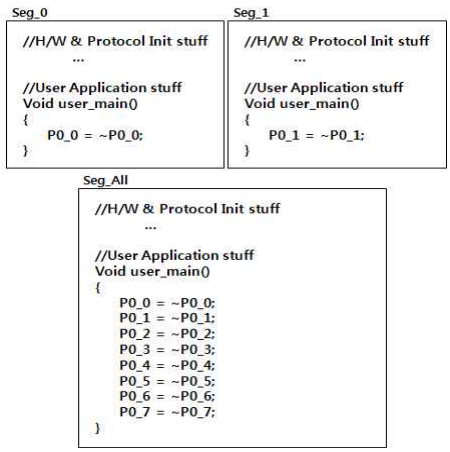

그림 10. 성능 평가를 위한 테스트 프로그램
Fig. 10. Test programs for performance evaluation

또한 Seg_0과 Seg_1은 실제적으로 프로그램 상에서 하나의 인수만 변경된 경우로써 표 1에서 보이는 바와 같이 이미지 크기가 동일한 반면, Seg_All은 약간의 코드가 추가되어 이미지 크기가 약간 더 크다.

표 1. 응용프로그램의 Hex 이미지 크기 및 실 데이터 전송량

Table 1. The hex image size and the actual amount of data dissemination

Application		Hex 이미지 크기	데이터 전송량
Seg_0	H00	25013 byte	25024 byte
	H01	12401 byte	12416 byte
Seg_1	H00	25013 byte	25024 byte
	H01	12401 byte	12416 byte
Seg_All	H00	25013 byte	25024 byte
	H01	12418 byte	12480 byte

표 2는 그림 10의 코드들이 컴파일 된 후의 실행 이미지들인 Seg_0에 대한 Seg_1 및 Seg_All 에 대한 이미지 유사도를 나타내는데, 각 이미지별 유사도가 매우 높게 나타난 것은 프로그램 상에서 하드웨어 초기화 및 S/W 업데이트 프로토콜 관련 코드가 전체 코드의 대부분을 차지하기 때문이다.

표 2. 실행 이미지 간 유사도

Table 2. The similarity between executing images

비교대상	이미지 유사도
Seg_0 vs. Seg_1	0.99997936%
Seg_0 vs. Seg_All	0.99836782%

표 3은 타겟노드 상에 동작하는 실행이미지 Seg_0에 대하여 Seg_1(Case 1)과 Seg_All(Case 2)을 업데이트 하기위한 각 이미지 크기 및 실제 데이터 전송시 전송 감소율을 나타내며 괄호안의 수치는 실제 전송되는 이미지의 크기이다. 제안된 업데이트 기법을 이용하면, Seg_1으로의 업데이트 시(Case 1)에는 이미지 전송프로토콜 및 점핑 테이블이 포함된 뱅크 이미지(H00)는 변동 사항이 없으므로 업데이트 할 필요가 없게 되며, 응용프로그램이 포함된 뱅크 이미지(H01)는 생성된 델타이미지 크기가 68바이트로써 함께 전송되는 복구 이미지(68바이트)와 함께 256바이트의 전송만으로 업데이트가 가능하여 원 이미지를 이용할 경우에 비해 97.94%의 이미지 전송 감소율을 보인다. 또한 Seg_All으로의 업데이트 시에도 뱅크 0과 뱅크 1에 대한 전송량 감소율은 각각 98.98%, 94.87%로 전송량 측면에서 상당히 효율적임을 알 수 있다.

표 3. 경우별 성능 평가표 (단위:byte)

Table 3. Performance of each case (unit:byte)

Case		코드 차이	델타 이미지 크기	복구 이미지 크기	전송량 감소율
1	H00	0	-	-	100%
	H01	1	68 (128)	68 (128)	97.94%
2	H00	6	112 (128)	112 (128)	98.98%
	H01	129	275 (320)	262 (320)	94.87%

V. 결 론

본 논문에서는 무선 센서 네트워크상에서 소프트웨어 업데이트 시 업데이트 이미지의 전송량을 최소화하여 이미지 전송과정에서 수반되는 대역폭 이용 및 에너지 소모량을 줄이기 위한 기법을 제안하였다. 제안된 기법은 코드 뱅킹 특성을 이용하여 업데이트가 필요한 뱅크의 이미지만 업데이트 하게 되며, 업데이트 될 뱅크 이미지를 VCDIFF 기반의 델타 이미지 압축 기법을 이용하여 이미지 전송량을 감소시킨다. 또한 제안된 방법은 노드상의 실행 이미지가 비정상적으로 업데이트 될 경우 특정 뱅크 이미지를 업데이트 이전의 이미지로 복구하는 기능을 제공함으로써 타겟노드의 정상동작에 대한 강건성을 보장한다. 마지막으로 제안된 기법의 성능을 입증하기위해 실제 8051기반 센서노드 상에서 성능을 검증하였다.

참고문헌

[1] Q. Wang, Y. Zhu and L. Cheng, "Reprogramming Wireless Sensor Networks: Challenges and Approaches," IEEE Network, Vol. 20, No. 3, pp. 48-55, May/Jun. 2006.

[2] S. Brown, "Updating Software in Wireless Sensor Networks: A Survey," Technical Report UCC-CS-2006-13-07.

[3] M. Kuorilehto, M. hannikainen and T.

Hamalainen, "A Survey of Application Distribution in Wireless Sensor Networks," EURASIP Journal on Wireless Communications and Networking, 2005.

[4] J. Koshy and R. Pandey, "Remote Incremental Linking for Energy-Efficient Reprogramming of Sensor Networks," Proc. EWSN 2005.

[5] P. Rickenbach and R. Wattenhofer, "Decoding Code on a Sensor Node," 4th International Conference on Distributed Computing in Sensor Systems (DCOSS), Jun. 2008.

[6] D. Korn, J. MacDonald, J. Mogul and K. Vo, "The VCDIFF Generic Differencing and Compression Data Format," RFC 3284 (Proposed Standard), Jun. 2002.

[7] J. MacDonald, "File System Support for Delta Compression," Masters thesis. Department of Electrical Engineering and Computer Science, University of California at Berkeley, 2000.

[8] http://www.keil.com/support/man/docs/lx51/

[9] J. Jeong and D. Culler, "Incremental Network Programming for Wireless Sensors," Proc. 1st Annual IEEE ComSoc. Conf. Sensor and Ad Hoc Commun. and Networks (SECON), pp. 25-33, 2004.

[10] R. Panta and S. Bagchi, "Hermes: Fast and Energy Efficient Incremental Code Updates for Wireless Sensor Networks", To appear in 28th Annual IEEE Conference on Computer Communications (INFOCOM), Apr. 2009.

저 자 소 개

남 영 진(Young-Jin Nam)

1992년 2월 : 경북대학교 전자공학과 학사
1994년 2월 : 포항공과대학교 전자전기공학과 석사
2004년 2월 : 포항공과대학교 컴퓨터공학과 박사
2004년~현재 : 대구대학교 컴퓨터·IT공학부 조교수
관심분야 : 임베디드 시스템, 저전력 스토리지
Email : yjnam@daegu.ac.kr

남 민 석(Min-Seok Nam)

2008년 2월 : 대구대학교 전산통계학과 학사
2008년~현재 : 대구대학교 컴퓨터정보공학과 석사과정
관심분야 : 임베디드 시스템, WSN
Email : msnam@daegu.ac.kr

박 영 균(Young-Kyun Park)

2007년 2월 : 대구대학교 전산통계학과 학사
2007년~현재 : 대구대학교 컴퓨터정보공학과 석사과정
관심분야 : 임베디드 시스템, WSN
Email : superhornet@daegu.ac.kr

김 창 훈(Chang-Hoon Kim)

2001년 2월 : 대구대학교 컴퓨터정보공학부 학사
2003년 2월 : 대구대학교 컴퓨터정보공학과 석사
2006년 8월 : 대구대학교 컴퓨터정보공학과 박사
2006년 9월 : 대구대학교 정보통신공학부 BK21 연구교수
2007년 8월~현재 : 대구대학교 컴퓨터·IT 공학부 전임강사
관심분야 : 암호시스템, 임베디드 시스템, RFID/USN 보안
Email : kimch@daegu.ac.kr

이 동 하(Dong-Ha Lee)

1985년 : 경북대학교 전자공학과 학사
2001년 : 경북대학교 전자공학과 석사
2005년 : 경북대학교 전자공학과 박사
2006년 : 경북대학교 경영학과 T-MBA 수료
1987년~2005년 : (주)LG전자 S/W연구실장
2004년~현재 : 영남대학교 겸임교수
2005년~현재 : 대구경북과학기술원 연구부장
관심분야 : 영상처리, DTV시스템, 임베디드소프트웨어
Email : dhlee@dgist.ac.kr