Journal of Korea Multimedia Society Vol. 14, No. 7, July 2011(pp. 949-963)

단일 홉 무선 센서 네트워크를 위한 코드 뱅킹 기반의 고속 병렬 소프트웨어 업데이트 기법

박영균†, 남영진††

요 약

무선 센서 네트워크상에서 센서노드의 소프트웨어 변경, 패치 등의 작업을 위해서는 일반적으로 센서노드별로 ISP(In System Programming)를 이용하여 프로그램 해야 한다. 이는 네트워크 내에 존재하는 노드의 수가 소량인 경우 크게 문제되지 않지만, 대량의 노드로 구성된 경우에는 프로그래밍을 위한 시간적인 측면과 투여되는 인력적인 측면에서 매우 높은 비용이 소요된다. 본 논문에서는 현재 많이 사용하고 있는 IEEE 802.15.4 기반의 단일 홉 무선 센서 네트워크 환경에서 노드의 수와 무관하게 고속 병렬로 센서노드 내부의 프로그램을 무선으로 업데이트 할 수 있는 기법을 제시하고 실험을 통하여 그 성능을 검증한다.

A Code Banking-based High-speed Concurrent Software Update Method for Single Hop Wireless Sensor Networks

Young Kyun Park†, Young Jin Nam††

ABSTRACT

Generally, It is indispensible to use an ISP(In System Programming) tool for upgrading, patching, or changing the system software of the each sensor nodes in wireless sensor networks. While under a small number of nodes, the upgrading task is not a serious burden, however if there are a large number of nodes to be updated, the task is almost impossible to do for the given constrains such as limited budgets and resources. Based on this observation, in this paper we have proposed a novel upgrading scheme based on a single hop in IEEE 802.15.4 PAN(Personal Area Network)s. Simulation results have shown the scheme outcomes the conventional methods in the performance measures.

Key words: Wireless Sensor Networks(무선 센서 네트워크), Wireless Software Update(무선 소프트웨어 업데이트), Code Banking(코드 뱅킹)

1. 서 론

근래의 무선 센서 네트워크(wireless sensor network)는 생활의 편의성을 도모하기 위한 다양한 형태로 구성되고 있다. 이러한 센서 네트워크를 구성하는 센서노드(sensor node)는 비교적 사람이 접근하기 쉬운 건물 내에 설치되는 경우도 있지만 목적에 따라 원격지나 사람이 접근하기 힘든 지역 혹은 위험한 지역의 정보를 수집하고 감시하기 위한 목적으로 설치되기도 한다. 경우에 따라서는 조밀한 지역에 대

※ 교신저자(Corresponding Author) : 남영진, 주소 : 경북 경산시 진량읍 내리리15 대구대학교 정보통신대학 공학7호관 7510A(712-714), 전화 : 053)850-6586, FAX : 053)850-6589, E-mail : yjnam@daegu.ac.kr
접수일 : 2011년 3월 3일, 수정일 : 2011년 6월 2일
완료일 : 2011년 6월 15일

† 정회원, 대구대학교 컴퓨터 · IT공학부
(E-mail : superhornet@daegu.ac.kr)
†† 정회원, 대구대학교 컴퓨터 · IT공학부
※ 본 연구는 교육과학기술부와 한국연구재단의 지역혁신인력양성사업과 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음(NIPA-2011-C1090-1131-0009)

량의 센서노드가 설치될 수도 있고 비용을 고려해 최소 비용으로 최적의 구조로 설치되는 경우도 있다. 하지만 어느 경우이건 단순 배터리 교체에서부터 소프트웨어의 변경, 수정, 재설치 등 센서노드의 유지, 보수가 필요하다. 센서노드의 소프트웨어에 관련된 유지 및 보수의 경우, 사람이 직접 모든 센서노드를 수거해 다시 프로그램을 할 수 밖에 없는데 각각의 노드를 수거하는 과정에서부터 새로 프로그램 하는 과정까지 시간적, 인적 비용이 많이 들 수밖에 없다. 이러한 비용은 관리해야 하는 센서노드의 수가 많아지면 심각할 정도로 문제가 된다.

이러한 문제점을 해결하기 위해 센서노드의 소프트웨어를 무선으로 효과적으로 업데이트하기 위한 다양한 연구[1-8]들이 진행되어 왔다. 각각의 센서노드를 순차적인 방법으로 업데이트 작업을 진행할 경우, 모든 소프트웨어 업데이트 과정이 종료될 때까지의 소요시간은 센서노드 수에 비례하여 증가하는 문제가 존재한다. 본 논문에서는 무선 센서 네트워크 내에 존재하는 센서노드의 수에 크게 영향을 받지 않으면서 순차적인 업데이트 방법에 비해 획기적으로 업데이트 시간을 줄일 수 있는 무선 병렬 업데이트 기법을 제시하고 구현을 통해 성능을 검증한다. 성능 검증을 위해서 본 논문에서는 센서 네트워크를 단일 홉 성형 토폴로지로 구성하며, 8051 MCU를 사용하는 센서노드를 이용하여 노드 수에 따른 순차적인 소프트웨어 업데이트 시간과 제안 기법의 소프트웨어 업데이트 시간을 측정하여 비교한다.

본 논문은 2장에서 기존의 소프트웨어 업데이트 기법과 성능시험에 사용되는 8051 MCU의 메모리 뱅킹 구조를 설명하고, 3장에서 소프트웨어를 무선으로 병렬 업데이트 하는 기법을 제안한다. 4장에서는 제안기법을 구현하여 노드 수에 따른 소프트웨어 업데이트 소요시간 측정을 통해 성능을 검증하고, 끝으로 5장에서 결론 및 향후 연구방향을 기술한다.

2. 관련연구 및 배경지식

2.1 단일 홉 무선 센서 네트워크

무선 센서 네트워크는 유비쿼터스 컴퓨팅 구현을 위한 다수의 경량, 저전력 노드로 구성된 네트워크이며 기본적으로 센서노드와 베이스스테이션(base-station)으로 구성된다. 일반적으로 이러한 무선 센서 네트워크는 적용 현장에 센서노드를 설치하여 모니터링 및 감시에 활용된다. 각각의 센서노드는 장비하고 있는 센서를 이용해 주위 환경 및 상태 등을 감지한다. 센서노드는 감지한 센서 데이터를 싱크노드로 전송하고 최종적으로 모니터링 시스템 및 관리자로 전달된다. 기존의 유무선 네트워크와 마찬가지로 단일 홉, 멀티 홉으로 구성될 수 있으며, 애드혹(ad-hoc) 망 구축도 가능하다. 단일 홉 망 구축을 위해서는 IEEE 802.15.4 MAC 프로토콜만으로도 구성이 가능하며, 멀티 홉 망을 구성하기 위해서는 지그비(Zigbee)와 같은 프로토콜 스택이 필요하다. 차후 언급하겠지만 본 논문에서 제안하는 기법은 단일 홉 네트워크를 대상으로 하기 때문에 단일 홉 무선 센서 네트워크에 대해서 살펴본다. 그림 1은 단일 홉 무선 센서 네트워크의 일반적인 구성을 나타낸다. 베이스스테이션은 PC 혹은 모니터링 시스템과 싱크노드로 구성되며 각각의 센서노드는 싱크노드와 무선으로 통신한다.

무선 센서 네트워크를 구성하는 센서노드는 일반적으로 처리 능력과 메모리, 전원 등 하드웨어 사양이 제한적이다. 대부분의 센서노드는 상시 전원이 아닌 배터리로 동작하며, 마이크로프로세서의 처리 능력도 한계가 있어 복잡하거나 큰 메모리를 필요로 하는 프로그램은 수행하지 못한다. 따라서 이같이 제한적인 센서노드를 구동하는 센서노드용 운영체제는 기본적으로 저전력 통신 특성을 갖춰야하며 메모리 영역을 효율적으로 관리할 수 있어야 한다. 대표적인 센서노드 및 센서 네트워크를 위한 운영체제로는 TinyOS, SOS, MANTIS, T-Engine Micro Kernel, PEEROS[9] 등이 있다.

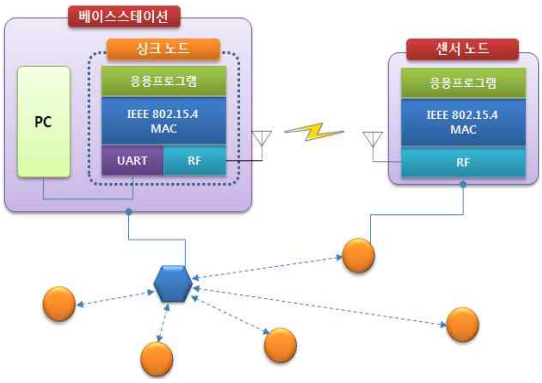

그림 1. 단일 홉 무선 센서 네트워크의 일반적인 구성

2.2 기존 소프트웨어 업데이트 기법

일반적으로 네트워크를 통한 소프트웨어 업데이트는 인코딩, 이미지 전송, 디코딩 단계로 나눌 수 있다. 그림 2는 소프트웨어 업데이트의 호스트가 되는 베이스스테이션과 소프트웨어 업데이트 대상 노드에서 각 단계별 진행 내용을 도식화한 내용이다.

인코딩 단계는 그림 2에서와 같이 베이스스테이션에서 각 노드에 업데이트할 이미지를 전송하기 위해 패킷화 하여 전송할 데이터를 생성하는 단계이다. 이때, 이미지 전체를 전송하는 업데이트 기법(전체 이미지)도 있고 현재 구동 중인 소프트웨어와 비교해서 다른 부분만 패치형태로 전송하는 기법(델타 패치) 등 다양한 방식이 존재한다. 이미지 전송 단계는 인코딩 단계에서 생성된 이미지를 유무선 네트워크를 통해 업데이트 대상 노드로 전송하는 과정이다. 디코딩 과정은 업데이트 대상 노드가 베이스스테이션이 전송한 이미지를 외부 메모리에 모두 저장하고 프로그램 메모리에 새로운 코드를 복사하여 재시작하는 과정이다. 일반적으로 노드를 시작하는 시작코드와 외부 메모리에 저장된 새로운 소프트웨어 이미지를 프로그램 메모리에 복사하는 업데이트 코드는 응용프로그램이 동작하는 메모리 영역과 별도로 구분되어 관리된다.

기존의 소프트웨어 업데이트 기법은 전송하는 이미지의 유형, 멀티 홉 지원 여부, 업데이트 대상 범위 등 업데이트 성능을 판단 할 수 있는 몇 가지 기준에 따라서 표 1과 같이 구분 지을 수 있다. 업데이트를 위해 전송하는 데이터 측면에서 보면 대부분의 소프트웨어 업데이트 기법은 업데이트 할 새로운 소프트웨어를 컴파일 한 전체 이미지(HEX)를 전송한다. 하지만 완전히 새로운 소프트웨어로 교체하는 것이 아닌 부분적인 변경일 경우에도 전체 이미지를 전송해야 하므로 오버헤드가 존재한다. 노드의 전력소모나 네트워크 부하 측면에서 전송데이터는 업데이트 성능을 판단하는 하나의 기준이 될 수 있다. 전송 프로토콜 또한 업데이트 시간에 영향을 줄 수 있는 요인으로 단순 CSMA/CA를 통해 업데이트 하는 기법이 있는 반면 업데이트 할 이미지의 버전을 광고하고 요청하는 노드에 이미지를 전송하는 기법, 이웃하는 노드에 이미지를 전송하는 기법, 부모가 업데이트 한 뒤 자식노드에 이미지를 전송하는 기법 등이 있다. 전송 프로토콜과 함께 업데이트 성능에 영향을 미치는 기준으로 업데이트 범위와 지원 홉을 들 수 있다. 멀티 홉으로 구성된 전체 네트워크를 대상으로 업데이트가 가능한 기법은 기능적인 측면에서는 단일 홉, 선택된 노드만 업데이트 하는 기법에 비해 우수하다고 할 수 있을지 모르지만, 업데이트 시간적인 측면에서는 그렇지 못한 경우가 있다.

업데이트할 소프트웨어의 이미지 일부를 업데이트 하는 방법에는 표 1과 같이 Incremental[10,11], TinyCubus[12], Trickle[13]등이 있으며, 소프트웨어의 전체 이미지를 전송하는 기법에는 XNP[14], MNP[15], Deluge[16], MOAP[17], Infuse[18], Sprinkler[19], Aqueduct[20] 등이 있다.

2.3 8051 MCU의 메모리 구조 및 코드 뱅킹

8051 호환 MCU는 메모리 어드레싱 영역이 64KB로 제한되어있다. 하지만 프로그램 크기가 64KB를 초과할 경우 코드 뱅킹(code banking) 이라는 기능을 사용할 수 있는데, 전체 내부 플래시 메모리를

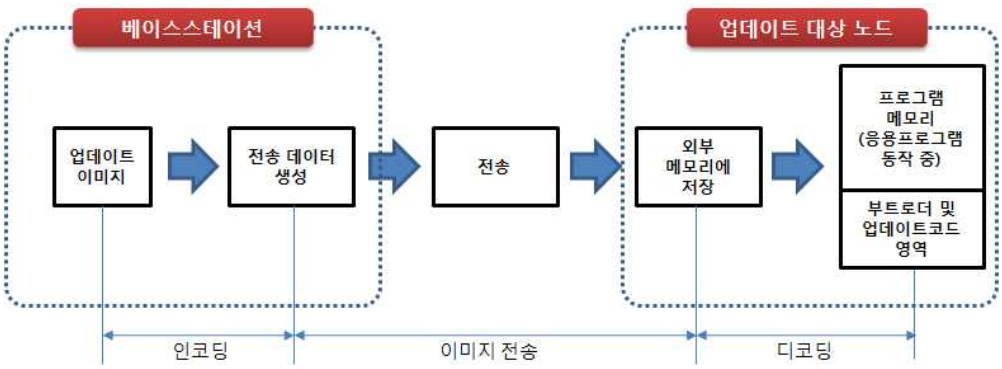

그림 2. 네트워크를 통한 일반적인 소프트웨어 업데이트 과정

표 1. 기존 무선 소프트웨어 업데이트 기법

기 법 명	전송데이터	전송프로토콜	지 원 홉	업데이트범위
Incremental[10, 11]	델타 패치	--	단일 홉	전체 네트워크
TinyCubus[12]	모듈 단위 업데이트	--	멀티 홉	선택된 노드
Trickle[13]	Maté 스크립트	adv-req-data	멀티 홉	전체 네트워크
XNP[14]	전체 이미지	--	단일 홉	전체 네트워크
MNP[15]	전체 이미지	adv-req-data	멀티 홉	전체 네트워크
Deluge[16]	전체 이미지	adv-req-data	멀티 홉	전체 네트워크
MOAP[17]	전체 이미지	이웃 노드 간 전달	멀티 홉	전체 네트워크
Infuse[18]	전체 이미지	부모노드-자식노드(TDMA)	멀티 홉	전체 네트워크
Sprinkler[19]	전체 이미지	부모노드-자식노드(TDMA)	멀티 홉	전체 네트워크
Aqueduct[20]	전체 이미지	--	멀티 홉	선택된 노드

32KB의 뱅크들로 나누고 프로그램 이미지를 각 뱅크에 분할하여 저장한다[21]. 이 후 코드를 실행되는 중에 필요한 코드가 저장된 뱅크가 현재 맵핑된 메모리 주소 공간 외에 즉, 다른 뱅크에 있으면 필요한 뱅크의 주소를 맵핑하여 수행하도록 한다. 그림 3은 8051 MCU의 메모리 구조를 나타낸다. 코드뱅킹 모드를 사용하게 되면 기본적으로 뱅크 0과 뱅크 1이 어드레싱 되도록 맵핑된다.

첫 번째 뱅크 0에는 시작코드(startup code), 뱅크 스위칭 코드(bank switching code), 전체 프로그램의 메인코드 및 인터럽트 서비스 루틴이 포함된 이미지가 저장되어야 한다. 뱅크 1~뱅크 n-1의 영역에 나머지 코드가 저장되며 이 뱅크들이 앞서 언급한 바와 같이 필요에 따라 변경되어 사용하게 된다.

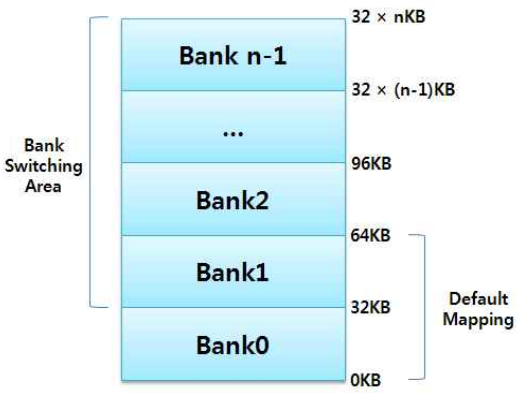

그림 3. 8051 MCU의 일반적인 메모리 뱅킹의 일반적 구조

3. 제안 기법

본 논문에서 제안하는 무선 병렬 소프트웨어 업데이트 기법은 단일 홉 거리 내에 위치한 대량의 노드를 대상으로 하며 순차적으로 센서노드의 소프트웨어를 업데이트할 때 수행되는 공통적인 작업을 전체 노드가 병렬로 처리하는 기법을 제안한다. 제안 기법을 적용하기 위해서는 IEEE 802.15.4 MAC을 이용해 통신하며, 코드 뱅킹기법을 제공하고, 64KB이상의 외부 메모리가 부착된 센서노드에 적용할 수 있다. 제안 기법을 적용하여 소프트웨어를 업데이트 할 시에는 기존의 응용프로그램은 잠시 동작을 멈추고 소프트웨어 업데이트가 종료된 후 다시 동작한다.

소프트웨어 업데이트 과정은 크게 세 단계로 나눌 수 있다. 첫째, 각 노드의 외부 메모리를 삭제하는 단계, 둘째, 업데이트 할 이미지를 전송하는 단계, 셋째, 데이터 오류 검사 및 센서노드 리셋 단계로 이루어진다. 그림 4는 제안 기법을 수행하기 위한 구성도이다. 사용자와 인터페이스하며 업데이트를 위한 명령과 소프트웨어 이미지의 전송을 위해 베이스스테이션이 필요하며, 이는 업데이트 대상이 되는 센서노드와 IEEE 802.15.4 무선 통신한다.

베이스스테이션은 명령 전송과 소프트웨어 이미지의 선택 및 패킷화를 위한 PC 그리고 이러한 명령과 소프트웨어 이미지를 무선으로 전송하기 위한 IEEE 802.15.4 통신이 가능한 '싱크노드'로 칭하는 노드로 구성된다. 프로그래머는 PC와 UART로 통신하

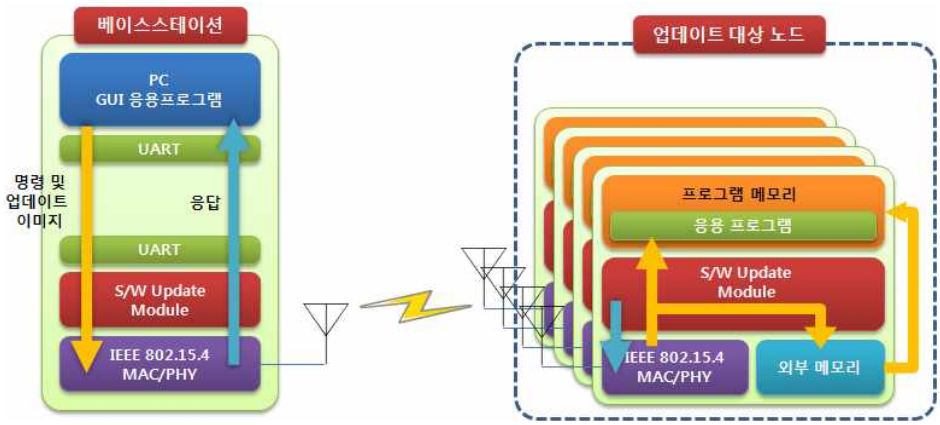

그림 4. 제안기법의 구성도

면서 베이스스테이션과 센서노드 간 명령, 데이터, 응답의 중계역할을 수행한다. 제안 기법은 멀티 홉 네트워크에의 적용은 불가능 하나, 베이스스테이션이 반드시 업데이트 대상이 되는 망에 포함될 필요는 없으며, 망 외부에서 업데이트 대상이 되는 망과 채널을 맞춰 업데이트가 가능하기 때문에 단일 홉 범위에 속하지 않는 노드들에 대해서는 따로 업데이트가 가능하다.

그림 5는 제안 기법의 대상이 되는 센서노드의 범위를 나타낸다. 베이스스테이션을 기준으로 통신이 가능한 거리 내에 위치하고 프로그래머와 동일한 채널을 사용하는 센서노드는 모두 제안기법을 이용한 소프트웨어 업데이트의 대상이 될 수 있다. 그 외에 베이스스테이션과 다른 채널을 사용하거나 통신이 불가능한 지역에 위치한 센서노드는 제안 기법으로 소프트웨어를 업데이트 할 수 없다. 제안 기법은 MAC 계층의 상위에 네트워크 계층 혹은 응용프로그램 계층에 위치하며 모든 명령 및 소프트웨어 이미지의 전송은 IEEE 802.15.4 MAC 계층의 데이터 프레임의 페이로드(payload)에 담아 전송한다.

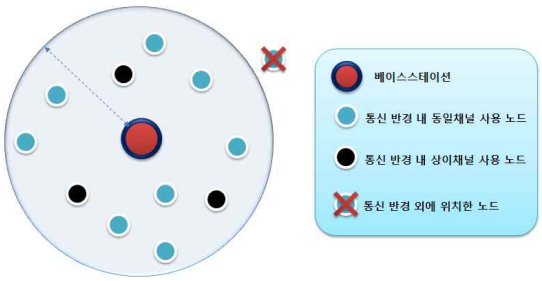

그림 5. 제안 기법 적용가능 센서노드의 범위

3.1 제안 기법을 위한 명령 및 소프트웨어 업데이트 프레임 정의

제안 기법의 수행을 위해서는 베이스스테이션과 소프트웨어 업데이트 대상이 되는 센서노드간의 약속된 명령처리 루틴이 필요하다. 이에 필요한 명령은 표 2와 같이 베이스스테이션에서 업데이트 대상 노드로 전송되는 명령과 업데이트 대상 노드에서 베이스스테이션으로 전송되는 응답 메시지로 정의하였다.

표 2의 명령 및 응답은 소프트웨어 업데이트 프레임(SWUpdateFrame)으로 구성되어 IEEE 802.15.4 표준의 *MCPS-DATA.request* 프리미티브를 통해 MAC 계층에 전송을 요청한다. 소프트웨어 업데이트 프레임은 일반 명령을 위한 프레임과 코드 전송을 위한 프레임으로 나눠진다. 두 프레임 모두 802.15.4 표준의 데이터 프레임을 그대로 사용하므로 최대 길이는 802.15.4에 정의된 길이와 같다.

그러나 본 논문에서 제안하는 업데이트를 위한 두 종류의 프레임은 그림 6, 7과 같이 각각 20바이트, 88바이트를 차지한다. 그림 6은 일반 명령 및 응답을 전송할 때의 프레임 구조이며 그림 7은 *REQ_WRITE_CODE* 명령을 전송할 때의 프레임 구조이다. 두 프레임 모두 수신한 프레임이 소프트웨어 업데이트를 위한 패킷임을 구분하기 위해 첫 번째 1바이트를 S/W Protocol로 사용하고 송신노드의 주소(source address) 및 수신노드의 주소(destination address)는 64비트 주소를 사용한다. 표 2에 정의된

표 2. 베이스스테이션과 업데이트 대상 노드 간 명령 및 응답

명령 및 응답	설 명	전송주체	전송 유형
REQ_ERASE_EEPROM	외부 메모리 삭제 요청	베이스 스테이션	브로드캐스트
REQ_ERASE_CHECK	외부 메모리 삭제 결과 요청	베이스 스테이션	유니캐스트
RES_ERASE_CHECK	외부 메모리 삭제 결과 응답	업데이트 대상 노드	유니캐스트
REQ_WRITE_CODE	외부 메모리에 이미지 쓰기	베이스 스테이션	브로드캐스트
REQ_CALC_CHECKSUM	체크섬 계산 요청	베이스 스테이션	브로드캐스트
REQ_RESET_CHECK	리셋 결과 요청	베이스 스테이션	유니캐스트
RES_RESET_CHECK	리셋 결과 응답	업데이트 대상 노드	유니캐스트

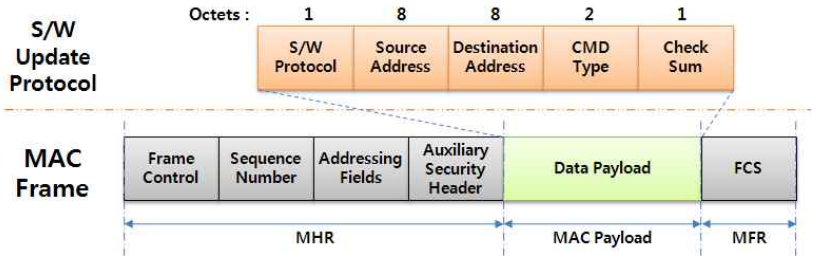

그림 6. 일반 명령 및 응답의 전송 프레임 구조

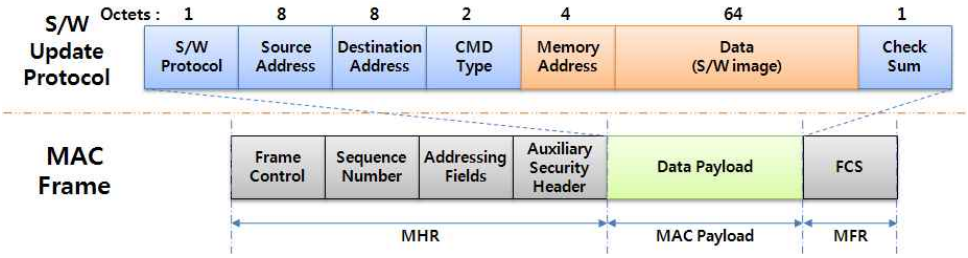

그림 7. REQ_WRITE_CODE 명령의 전송 프레임 구조

명령 및 응답과 대응되는 CMD Type필드는 2바이트로 구성되며 프레임의 마지막에 오류확인을 위한 체크섬(checksum)을 붙인다. 체크섬은 소프트웨어 업데이트를 위한 명령 및 데이터 전체를 1바이트 단위로 더해 계산한다. 실제 업데이트 할 소프트웨어의 이미지가 전송되는 *REQ_WRITE_CODE* 명령의 경우 이미지를 64바이트 씩 분할하여 해당 데이터가 적재될 메모리의 주소와 함께 전송된다. 그림 5에서 볼 수 있듯이 CMD Type필드 뒤에 4바이트의 메모리 주소와 Data필드가 추가된 구조를 가진다. 업데이트할 새로운 이미지의 길이가 64바이트의 배수가 되지 않는 경우에는 나머지 길이만큼 0xff를 채워서 보내도록 한다.

3.2 제안 기법의 동작 알고리즘

본 절에서는 베이스스테이션과 업데이트 대상이 되는 센서노드 측에서 수행되는 제안 기법의 동작 알고리즘을 설명한다. 그림 8은 베이스스테이션의 제안 기법 수행 절차를 나타낸 것이다. 센서노드 각각을 순차적으로 업데이트할 때 수행하는 공통적인 작업에 대해 해당 명령을 전체 노드에 브로드캐스팅하여 동시에 처리하도록 하는 부분으로, 전체 센서노드의 소프트웨어 업데이트 소요시간을 획기적으로 줄일 수 있는 핵심 부분이라 할 수 있다.

베이스스테이션이 각 센서노드로부터 업데이트 결과를 받아 업데이트 성공여부를 판단하는 과정에서 만약 업데이트 실패의 응답을 받으면 사용자는

```
Procedure Basestation_SoftwareUpdate
  // Node = {TargetNode(i) | 0<=i<n} : a set of sensor nodes to be updated
  // NewSWImage = HEX file of newly updated software image
  // T_EXT_MEM_ERASE = required time to erase underlying external memory in each sensor node
  // T_RESPONSE = required time to respond to REQ_ERASE_EXT_MEM
  // T_IFS = Inter-frame-spacing time
  // T_CALC_CHECKSUM_RESET = required time to calculate checksum or reset
input : Node, NewSWImage, time information
output : SWUpdateFrame

broadcasts REQ_ERASE_EXT_MEM to Node;
delay(T_EXT_MEM_ERASE);

foreach TargetNode(i) ∈ Node do
   unicast REQ_ERASE_CHECK to TargetNode(i);
   delay(T_RESPONSE);
   If (RES_ERASE_CHECK == 1) then TargetNode(i).status ← ERASE_SUCCESS;
   Else TargetNode(i).status ← ERASE_FAIL;
end
repeat
   broadcast REQ_WRITE_CODE to Node;
   delay(T_IFS);
until EOF(NewSWImage);
broadcast REQ_CALC_CHECKSUM with criteria check-sum to Node;
delay(T_CALC_CHECKSUM_RESET);
foreach TargetNode(i) ∈ Node do
   If (TargetNode(i).status != ERASE_FAIL) then
      unicast REQ_RESET_CHECK to TargetNode(i);
   If (RES_RESET_CHECK == 1) then TargetNode(i).status ← UPDATE_SUCCESS;
   Else TargetNode(i).status ← UPDATE_FAIL;
end

end Basestation_SoftwareUpdate
```

그림 8. 베이스스테이션의 제안 기법 동작 알고리즘

해당 노드를 차후 개별적으로 업데이트 할 수 있다.

그림 9는 업데이트 대상 노드가 수행하는 제안 기법의 알고리즘이다. 베이스스테이션이 전송하는 소프트웨어 업데이트 프레임을 수신한 후, 명령에 따라 외부 메모리 삭제, 업데이트 할 코드의 이미지 저장, 체크섬 계산 및 리셋 과정을 수행하며, 각 과정의 상태를 저장하여 베이스스테이션으로 응답한다.

그림 10은 제안 기법이 동작하는 예를 나타낸다. 베이스스테이션을 구성하는 PC와 프로그래머 는 UART로 통신하며 베이스스테이션과 업데이트 대상이 되는 센서노드는 IEEE 802.15.4 MAC으로 통신한다.

소프트웨어 이미지의 전송은 각 센서노드의 하드웨어 정보가 복사된 페이지는 덮어 쓰지 않도록 전송되어야 하며, 소프트웨어 이미지의 전송이 완료되면 센서노드가 스스로 내부 프로그램 메모리의 내용을 변경하고 리셋 하도록 명령을 줘야한다. 하지만 만약 소프트웨어 이미지 패킷을 완전하게 받지 못한 상태에서 리셋 되면 그 노드는 사용자가 직접 다시 프로그램 해줘야 하므로 수신한 소프트웨어 이미지 패킷이 정상적인지 체크섬을 통해 확인한다.

3.3 비정상적인 업데이트 상황에 대한 대처

센서노드를 업데이트하는 과정 중에 네트워크 불

```
Procedure Node_SoftwareUpdate()
input : SWUpdateFrame
output : Response for S/W Update Protocol
RESET_FLAG ← 1;
parse SWUpdateFrame;

switch(SWUpdateFrame.CMD_Type)
   case REQ_ERASE_EXT_MEM :
      If ( ERASE_EXT_MEM() == SUCCESS ) then STATUS ← ERASE_DONE;
      Else
         STATUS ← ERASE_FAIL;
         RESET_FLAG ← 0;
      break;
   case REQ_ERASE_CHECK :
      response RES_ERASE_CHECK(STATUS) to Basestation;
      break;
   case REQ_WRITE_CODE :
      store SWUpdateFrame.Data to external memory at EXT_MEM_INDEX;
      EXT_MEM_INDEX ← EXT_MEM_INDEX+64;
      break;
   case REQ_CALC_CHECKSUM :
      csum_criteria ← check-sum value in REQ_CALC_CHECKSUM;
      csum_temp ← calculate_checksum();
      If ( csum_criteria == csum_temp ) then
         RESET_FLAG ← 1;
         copy SWUpdateFrame.Data from external memory to program memory;
         reset;
      Else RESET_FLAG ← 0;
      break;
   case REQ_RESET_CHECK :
      response RES_RESET_CHECK(RESET_FLAG) to Basestation;
      break;
end switch

end Node_SoftwareUpdate
```

그림 9. 센서노드의 제안 기법 동작 알고리즘

안정, 센서노드의 베터리 부족 등의 문제로 정상적으로 업데이트 과정을 마치지 못하는 경우가 있다. 이와 같이 업데이트에 실패하는 경우는 크게 두 가지로 나눌 수 있다. 첫 번째는 업데이트 대상 노드 중 몇몇의 센서노드가 베이스스테이션이 브로드캐스팅하는 업데이트할 소프트웨어의 이미지가 담긴 패킷의 일부를 수신하지 못해 업데이트에 실패하는 경우이다. 이 경우에는 실패한 센서노드만 선택적으로 업데이트 할 수 있다. 업데이트에 실패하는 두 번째 경우는 네트워크 의 불안정으로 업데이트 이미지 전송 중에 중단되는 경우이다. 극단적인 예로 베이스스테이션이 이미지를 전송하는 중에 정전 등의 이유로 이미지 전송이 중단된 경우를 들 수 있다.

앞서 기술한 제안기법을 보면 각 센서노드는 베이스스테이션으로부터 업데이트 이미지를 수신하고 프로그램 메모리로 복사하기 전에, 수신한 이미지의 체크섬을 계산하여 비교하는 과정이 있다. 베이스스테이션이 보낸 체크섬과 센서노드가 계산한 체크섬이 다르면 외부 메모리에 저장해둔 새로운 소프트웨어의 이미지를 프로그램메모리로 복사하지 않고 업데이트를 중단하는데, 이러한 장치는 비정상적인 업데이트상황에 대비하여 기존 소프트웨어 코드를 보

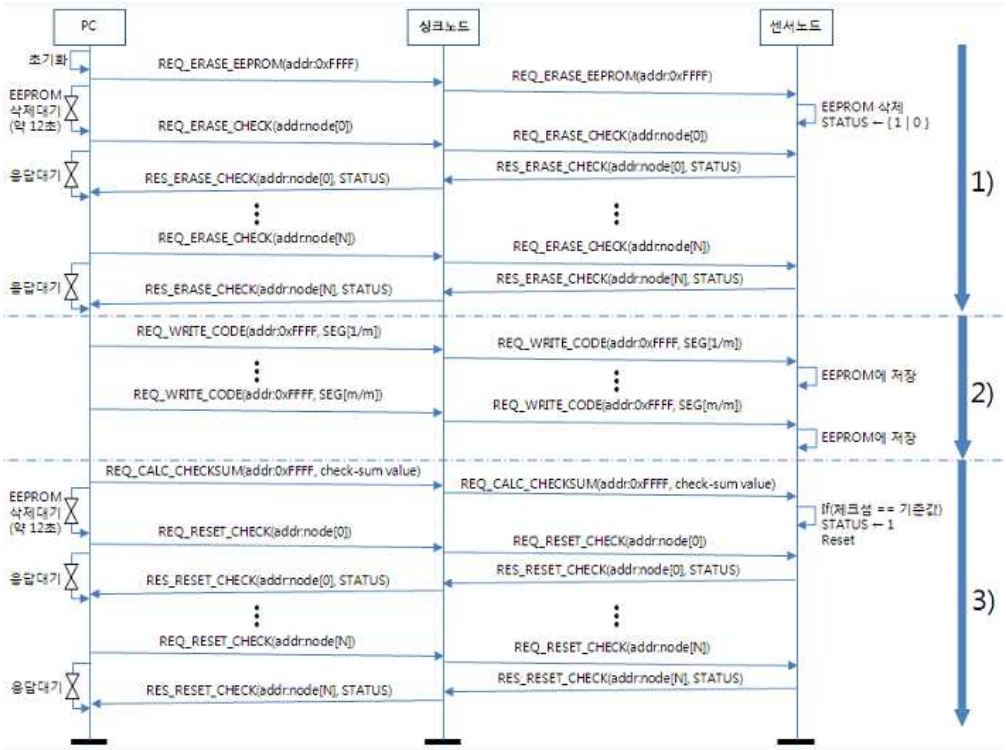

그림 10. 제안 기법의 수행의 예 ; 1) 각 노드의 외부 메모리 삭제 단계 2) 업데이트 이미지 전송 단계, 3) 데이터 오류 검사 및 센서노드 리셋 단계

호하기 위함이다. 따라서 어떤 이유로 업데이트에 실패하더라도 센서노드는 기존의 프로그램이 계속 동작하기 때문에 업데이트 재시도에 문제가 없다.

4. 구현 및 성능평가

4.1 제안 기법의 구현

제안하는 무선 병렬 소프트웨어 업데이트 기법의 구현을 위해 8051 MCU와 IEEE 802.15.4 표준을 따르는 RFIC, 64KB 외부 메모리로 구성된 센서노드를 사용했다. 통신을 위한 MAC 프로토콜은 IEEE 802.15.4 표준 중 데이터 송수신 기능만 구현된 라이브러리를 이용했다. 추가로 소프트웨어 업데이트의 편의를 위해 PC에서 동작하는 GUI 응용프로그램을 개발했으며, 기존 상용제품[21]의 디바이스프로그래머를 참고하였다. 상용제품은 하나의 센서노드를 대상으로 ISP 및 무선 업데이트가 가능한 프로그램이다. 프로그래머로부터 센서노드로 전송되는 이미지는 C언어로 작성 및 컴파일 후 만들어지는 이미지 파일(약 36KB)이며, 이미지 내에는 8051 MCU를 위한 기본 펌웨어, IEEE 802.15.4 MAC, 소프트웨어 업데이트 프로토콜이 포함되어 있으며 센서노드에 부착된 7세그먼트를 동작시키는 응용 프로그램이 포함되어 있다.

4.2 PC GUI 응용 프로그램 구현

사용자가 센서노드의 소프트웨어 업데이트를 편리하게 할 수 있도록 프로그래머와 UART로 통신하는 베이스스테이션 GUI응용 프로그램을 개발하였다. 그림 11은 개발한 PC GUI 응용 프로그램의 동작 화면 이다. 외부 메모리 삭제 대기 시간 및 소프트웨어 이미지를 전송할 때 패킷과 패킷 사이의 휴면 시간, 리셋 확인 대기 시간 등을 옵션 창을 통해 지정할 수 있도록 구현하였다.

사용자는 GUI 응용프로그램을 통해 업데이트할 센서노드의 주소를 지정하고 새로운 응용프로그램 이미지를 선택하여 센서노드를 업데이트 할 수 있다. GUI 응용프로그램은 선택된 이미지 파일을 인텔

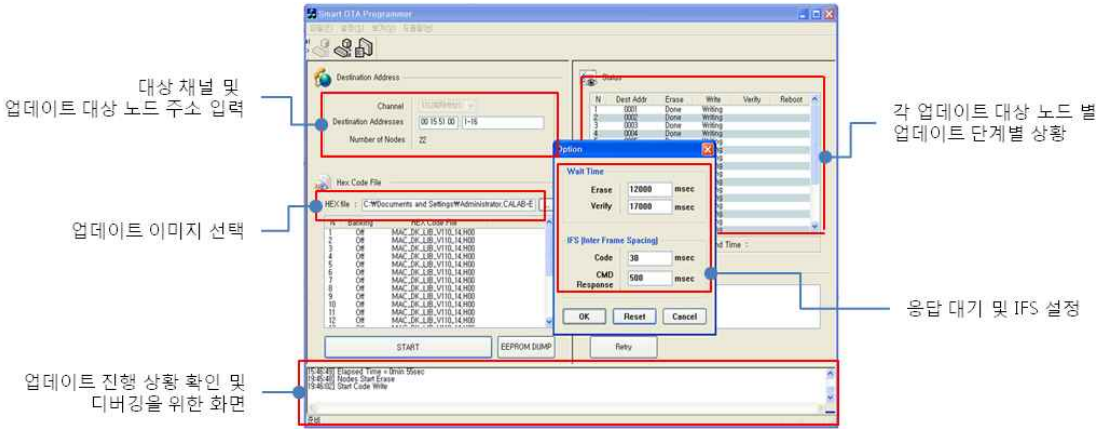

그림 11. 베이스스테이션 GUI 응용 프로그램

HEX파일형식으로 읽어 들이고 해당 코드가 적재되는 메모리 주소 순서대로 정렬해둔다. 이후 사용자가 업데이트를 시작하면 정렬된 이미지를 64바이트 씩 나눠 프로그래머로 UART를 통해 전송하며 프로그래머는 이를 각 센서노드로 브로드캐스팅 한다.

4.3 업데이트 대상 노드에서의 소프트웨어 업데이트 기법 구현

제안 기법은 센서노드 상에 포함된 IEEE 802.15.4 MAC을 이용하는 상위 네트워크 프로토콜 형태로 구현되었다. 소프트웨어 업데이트를 위한 각 명령 및 반환되는 데이터는 MAC계층에서 데이터 패킷형태로 구현하였다. 각 센서노드의 주소는 정적으로 할당하였으며, 모든 센서노드로의 브로드캐스트 전송을 위해서는 IEEE 802.15.4 표준에서 정한 주소(0xFFFF)를 이용하였다. 각 센서노드로 업데이트가 되는 이미지는 동일하며, 이미지의 크기는 8051 MCU에서 지원하는 4개의 뱅크 중에서 3개의 뱅크 크기를 넘지 않도록 한다. 마지막 뱅크는 시작 코드와 외부 메모리에 저장된 새로운 이미지를 MCU의 프로그램 메모리로 복사하고 노드를 재시작하는 코드가 위치하므로 마지막 뱅크를 덮어쓰는 크기의 프로그램은 업데이트 할 수 없다.

4.4 성능평가

제안 기법과 순차적 업데이트 기법의 성능을 비교하기 위해 표 3과 같은 하드웨어를 이용해 성능 실험을 진행하였다.

성능평가는 제안 기법 및 순차적 업데이트 기법에 대해 동일한 조건으로, 단일 노드 및 다수의 노드를 대상으로 이미지 크기별 업데이트 시간을 측정하여 비교한다. 노드는 1대에서부터 255대까지를 증가시키고, 이미지 2KB, 4KB, 6KB, 8KB, 16KB, 32KB, 64KB 크기별로 실험한다. 이 때, 소프트웨어 업데이트에 걸리는 시간만을 비교하기 위해 순차적 업데이트 기법의 경우 센서노드를 회수하고 각 센서노드를

표 3. 성능평가 환경 - 베이스스테이션과 업데이트 대상 노드

	베이스스테이션		업데이트 대상 노드
	PC	싱크 노드	센서노드
운영체제	32bit Windows XP	Firmware	Firmware
하드웨어 사양	Pentium4	MCU : 8051 RF : LM2455 External Memory : ATMEL 24c512	MCU : 8051 RF : LM2455 External Memory : ATMEL 24c512
통신/인터페이스	RS-232(Baudrate:115,200)	RS-232(Baudrate:115,200) IEEE 802.15.4 MAC	IEEE 802.15.4 MAC

프로그래머와 연결하는 시간은 무시한다.

성능평가에 사용된 노드는 표 3과 같이 LM2455 RF를 사용한다. 이 RF는 +10dBm으로 전송할 경우 46.4mA의 전류를 소모하며, 수신시에는 19.5mA의 전류를 소모한다. 응용서비스에 따라 전력소모에 따른 생명주기가 달라지겠지만 제안 기법을 이용하여 센서노드를 업데이트하는 동안의 전력 소모를 추정해보면 그림 12와 같다. 제안 기법은 베이스스테이션의 싱크노드가 업데이트 이미지를 전송하며 각 센서노드는 이미지를 수신한 후 결과만 전송하므로 대부분의 전류를 업데이트 이미지를 수신하는데 소모한다. 소모전류는 업데이트 이미지 크기가 2KB일 때 14.68nA, 256KB일 때 1,878nA로 나타난다.

제안 기법의 성능을 검증하기에 앞서 상용제품의 무선 업데이트에 소요되는 시간을 측정해보았다. 실험에 사용된 데이터는 데이터 전송 소요 시간 측정을 목적으로 하여 실제 프로그램 코드가 아닌 2KB부터 32KB까지 임의의 데이터로 작성한 것이다. 결과는 그림 13과 같이 하나의 센서노드에 대해 32KB크기의 이미지를 업데이트 하는데 평균 67초가 소요된다. 64KB 외부 메모리를 삭제하는데 14초 정도가 소요되고 체크섬 확인은 평균 7.5초 정도 소요된다.

이 경우 하나의 센서노드만을 대상으로 하기 때문에 망 내의 전체 노드를 업데이트하기 위해서는 식 1 만큼의 시간이 소요된다. 각각의 센서노드는 새로운 소프트웨어로 업데이트 할 때, 외부 메모리를 삭제하는 시간 T_{Erase}, 업데이트 할 이미지를 수신하여 센서노드의 외부 메모리에 저장하는 시간 T_{Write}, 외부 메모리에 저장된 이미지의 체크섬을 계산하는 시간 T_{CRC} , 외부 메모리의 이미지를 프로그램 메모리로 복사하고 재시작 하는 시간 T_{Reset}을 소요한다. 따

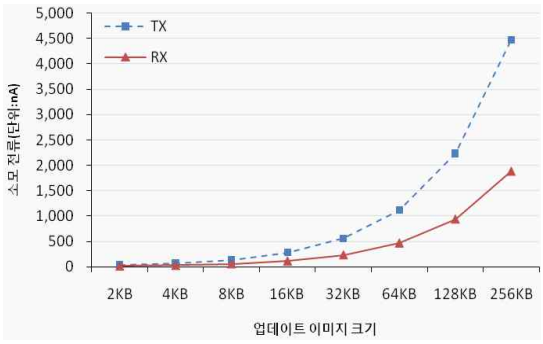

그림 12. 이미지 크기별 업데이트 소모전류

그림 13. 상용제품을 이용한 이미지 크기별 무선 업데이트 소요시간

라서 순차적 업데이트 기법으로 전체 노드가 업데이트 되는 시간 $T_{sequential}$은 식 1과 같이 각 노드가 소요하는 시간에 전체 센서노드 수 N배 만큼 소요된다. 외부 메모리에 이미지를 저장하는데 걸리는 시간 T_{Write}는 한 번에 전송할 수 있는 데이터 크기의 제한으로 인해 전체 이미지를 64바이트로 나눠 보내므로 식 2와 같이 나타낼 수 있다. 식 2에서 T_{IFS}는 이미지를 담아 보내는 패킷간의 유휴시간이며, S_{Image}는 업데이트 할 새로운 이미지의 크기 이고, $T_{Rom\ Write}$는 실제 센서노드가 수신한 이미지를 외부 메모리에 쓰는 시간이다.

$$T_{sequential} = N(T_{Erase} + T_{Write} + T_{CRC} + T_{Reset}) \quad (1)$$

$$T_{Write} = T_{IFS}((S_{Image}/64) - 1) + T_{Rom\ Write} \quad (2)$$

제안 기법의 경우 각 센서노드를 업데이트 할 때 수행되는 공통의 과정을 병렬로 처리하므로 외부 메모리 삭제 시간, 외부 메모리에 이미지를 저장하는 시간, 저장한 이미지의 체크섬 계산 시간은 각 노드별로 동시에 진행된다. 다만, 각 과정별 결과를 확인하기 위해 베이스스테이션이 각 센서노드로 결과를 요청하고 응답하는 시간이 추가된다. 제안 기법을 이용해 전체 센서노드를 업데이트 하는 시간은 식 3과 같다.

$$T_{proposed} = T_{Erase} + T_{Write} + T_{CRC\_Reset} + N(T_{ChkErase} + T_{ChkReset}) \quad (3)$$

식 3에서 T_{CRC_Reset}은 센서노드가 외부 메모리에 저장된 이미지의 체크섬을 계산 후 정상적으로 수신한 이미지 이면 바로 프로그램 메모리로 이미지를 복사하고 재시작하는 시간이다. 센서노드에 체크섬을 계산하는 명령을 전송하고 재시작 하는 명령을 따로 전송하는 순차적 기법과 달리, 제안기법은 체크

섬 계산 명령을 전송할 때, 베이스스테이션이 전송한 이미지의 체크섬을 함께 전송하므로 각 센서노드는 계산한 체크섬과 베이스스테이션이 전송한 체크섬의 비교를 통해 업데이트 및 재시작을 결정할 수 있다. $T_{ChkErase}$과 $T_{ChkReset}$시간은 각각 외부 메모리 삭제를 정상적으로 완료했는지, 센서노드가 수신한 이미지로 정상적으로 업데이트 했는지를 확인하는 시간이다. 이 는 베이스스테이션이 각 센서노드와 일대일 통신으로 진행하므로, 순차적 업데이트 기법과 달리 $T_{ChkErase}$, $T_{ChkReset}$ 이 두 시간에 대해서만 센서노드의 배수만큼의 시간이 소요된다.

제안 기법과 순차적 업데이트 기법의 소요 시간을 비교하여 제안 기법의 성능을 검증한다. 우선, 하나의 센서노드를 대상으로 업데이트 하는데 걸리는 시간을 각 단계별로 측정해보았다. 64B바이트씩 나눈 이미지 패킷을 전송하는 사이에 0.03초의 휴면 시간을 두었다. 반복해서 실험을 수행한 결과 36KB의 새로운 응용프로그램 이미지를 전송하고 리셋 하는데 평균 48초 정도가 소요되었다. 상용제품을 이용해 업데이트 할 때보다 19초 정도 줄어든 것을 확인 할 수 있는데 이는 EEPROM 삭제 대기 시간과 이미지 패킷 전송 사이의 휴면시간을 최적화 한 결과라 할 수 있다.

다음으로, 센서노드 수를 늘여서 22개 노드를 대상으로 동일한 실험을 수행하였다. 이미지가 업데이트 대상이 되는 노드 전체에 브로드캐스팅 되기 때문에 이미지를 전송하고 EEPROM에 쓰는 시간은 대상 노드가 수십 개가 되더라도 하나 일 때와 변함없다. 단, 하나의 센서노드를 업데이트하는 경우에 비해 EEPROM 삭제확인 및 리셋확인 명령을 각 노드별로 수행할 때, 센서노드 하나당 약 0.12초 정도의 부가적인 오버헤드가 존재함을 볼 수 있었다.

이러한 실험을 바탕으로 순차적 업데이트 기법과 제안 기법의 센서노드 수에 따른 업데이트 예상 소요 시간을 비교해 보았다. 제안 기법의 경우 대상 센서노드가 많아지더라도 이미지는 전체 노드에 동시에 전송하므로 노드 당 EEPROM 삭제 확인 및 리셋 확인 명령을 주고받는 시간만 더 늘어날 뿐이다. 이론적으로 하나의 PAN(personal area network)은 최대 255개의 노드로 구성할 수 있는데 이를 대상으로 업데이트 할 경우 그림 14와 같이 제안 기법은 순차적 업데이트에 비해 99.37%의 시간을 줄일 수 있을 것으로 기대한다.

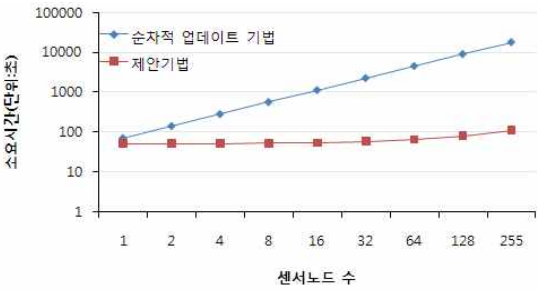

그림 14. 순차적 업데이트 기법과 제안기법의 255개 센서노드에 대한 업데이트 예상 소요 시간 비교

그림 15는 업데이트 이미지 크기와 노드 수에 따른 업데이트 시간을 비교한 것이다. 점선으로 표시된 그래프가 순차적 업데이트 기법의 업데이트 시간을 나타내고 실선이 제안기법의 업데이트 시간을 나타낸다. 그림에서 알 수 있듯이 이미지 크기에 관계없이 앞서 실시한 실험들과 같은 패턴을 보인다. 순차적 업데이트 기법과 제안기법 모두 이미지 크기가 커져도 데이터 송신 패킷량이 많아짐에 따라 업데이트 시간이 늘어 날 뿐 EEPROM 삭제 및 체크섬 확인, 재부팅 시간은 큰 차이가 없다.

5. 결론 및 향후 연구계획

본 논문에서는 다수의 센서노드를 대상으로 무선 병렬 소프트웨어 업데이트 기법을 제안하고 성능을 검증해 보았다. 제안 기법의 구현 및 성능 검증을 위해 메모리 뱅킹 기능이 제공되는 8051 MCU를 이용했지만 프로그램 메모리를 분할하고 특정영역을 보호할 수 있는 기능을 별도로 구현한다면, 타 MCU에도 적용 가능할 것으로 예상한다. 제안 기법은 32개의 노드를 업데이트 할 때 제안 기법은 1분 정도의 시간이 소요되는데, 이는 순차적으로 업데이트 할 때보다 상대적으로 97.4%의 시간이 줄어 든 수치이다. 이론적으로 하나의 PAN을 구성하는 최대 노드 수인 255개의 노드를 대상으로 업데이트 한다고 가정할 때, 제안 기법은 순차적으로 업데이트 하는 방법 대비 99.37%의 시간을 줄일 수 있을 것으로 기대된다.

현재 기존의 XNP, Deluge 등의 소프트웨어 업데이트 기법을 적용할 수 있는 하드웨어를 확보하지 못해 제안 기법과 비교 분석은 향후 진행할 계획이며, 전체 이미지가 아닌 변경된 이미지에 대한 정보만 전송하여 업데이트 시간을 줄일 수 있는 방향으로

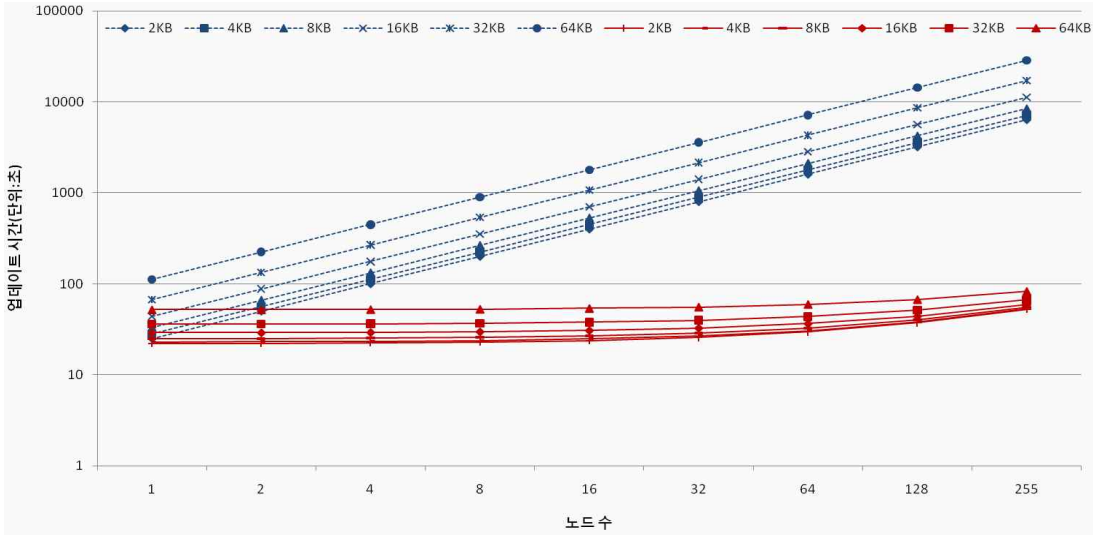

그림 15. 업데이트 이미지 크기에 따른 순차적 업데이트 기법과 제안기법의 시간 비교

연구할 계획이다. 더불어 단일 홉 네트워크를 대상으로 설계된 제안 기법을 멀티 홉으로 구성된 네트워크에 적용하기 위한 연구도 진행한다. 기존에 연구된 멀티 홉 지원 무선 소프트웨어 업데이트 기법의 경우 업데이트 시간을 줄이기 위해 일정량의 이미지를 수신하면 자식 노드로 해당 이미지를 전송하는 파이프라인 기법을 사용한다. 하지만 이러한 파이프라인 기법으로 인한 소요시간 감소 효과를 얻기 위해서는 최소 4홉 이상으로 네트워크가 구성되어야 하고, 각 노드는 충분한 거리를 두고 설치되어야 한다. 재해 상황 대비 및 긴급 메시지 전송을 고려한 네트워크의 경우 다양한 라우팅 경로를 생성할 수 있도록 하나의 노드의 RF범위 내에 하나 이상의 노드를 배치하는데, 이러한 망에 파이프라인 기법으로 업데이트 할 경우 버전 비교, 동일 버전일 경우 업데이트 메시지 거부, 누락된 패킷의 재전송 요청 등으로 인해 제안 기법 보다 상대적으로 노드의 전력소모가 늘어나고 업데이트 시간이 늦어지는 경우가 발생할 것으로 예상된다.

제안 기법으로는 멀티 홉 네트워크를 지원할 수 없다. 하지만 업데이트 명령 및 이미지의 전송을 담당하는 베이스스테이션이 업데이트 대상이 되는 망에 포함될 필요가 없어 독립적으로 망 외부에서 채널을 맞추면 망 혹은 노드를 선택적으로 업데이트가 가능하기 때문에 멀티 홉 네트워크에서도 부분적으로는 적용이 가능하다. 향후 멀티 홉 네트워크를 지원하기 위해 현재의 제안 기법을 앞서 언급한 멀티 홉에서의 문제점들을 고려하여 개선할 계획이다.

참 고 문 헌

[1] 이한선, 정광주, “무선 센서 네트워크를 위한 신속한 코드 전송 기법,” 한국정보과학회 학술발표논문집, Vol.33, No.1, pp. 283-285, 2006.

[2] 김성호, 김종근, “무선 센서 네트워크에서의 목표 노드 리프로그래밍 프로토콜,” 한국정보과학회 학술 심포지움 논문집, Vol.1, No.1, pp. 13-18, 2007.

[3] 이동호, 김창훈, 홍춘표, “OSEK 기반 무선 센서 노드용 운영체제 및 소프트웨어 업데이트 시스템 설계,” 한국멀티미디어학회 추계학술발표논문집, pp. 92-96, 2008.

[4] Q. Wang, Y. Zhu, and L. Cheng, “Reprogramming Wireless Sensor Networks: Challenges and Approaches,” *IEEE Network*, Vol.20, No. 3, pp. 48-55, 2006.

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

[6] M. Kuorilehto, M. hannikainen, and T. Hamalainen, “A Survey of Application Distribution in Wireless Sensor Networks,” *EURASIP*

Journal on Wireless Communications and Networking, Vol.2005, No.5, pp. 744-788, 2005.

[7] S. Brown and C. Sreenan, "Software Update Recovery for Wireless Sensor Networks," Proceedings of the 1st International Conference, SENSAPPEAL, Vol.29, pp. 107-125. 2009.

[8] G. Yoo and E .Lee, "Self-Healing Methodology in Ubiquitous Sensor Network," *International Journal of Advanced Science and Technology*, 2009.

[9] A. Kumar, D. Janakiram, and G. Kumar, "Operating Systems for Wireless Sensor Network: A Survey Technical Report," 2007.

[10] N. Reijers and K. Langendoen, "Efficient Code Distribution in Wireless Sensor Networks," Proceedings of the 2nd ACM International Conference Wireless Sensor Networks and Applications pp. 60-67, 2003.

[11] J. Jeong and D. Culler, "Incremental Network Programming for Wireless Sensors," Proceedings of the 1st Annual IEEE Computer Society Conference Sensor and Ad Hoc Communication and Networks, pp. 25-33, 2004.

[12] P. Marrón, A. Lachenmann, D. Minder, J. Hähner, R. Sauter, and K. Rothermel, "TinyCubus: A Flexible and Adaptive Framework for Sensor Networks," Proceedings of the 2nd European Workshop on Wireless Sensor Networks, pp.278-289, 2005.

[13] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," Proceedings of the 1st Symposium on Networked Systems Design and Implementation, pp.15-28, 2004.

[14] Crossbow Technology, Inc., "Mote In-Network Programming User Reference," pp.1-8, 2003.

[15] S. Kulkarni and L. Wang, "MNP: Multihop Network Programming for Sensor Networks," Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, pp. 7-16, 2005.

[16] J. Hui and D. Culler. "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," ACM SenSys, pp. 81-94, 2004.

[17] T. Stathopoulos, J. Heidemann, and D. Estrin, "A Remote Code Update Mechanism for Wireless Sensor Networks," Center for Embedded Network Sensing. Technical Reports, 2003.

[18] S. Kulkarni and M. Arumugam, "Infuse: A TDMA Based Data Dissemination Protocol for Sensor Networks," *International Journal of Distributed Sensor Networks*, Vol.2, No.1, pp. 55-78, 2006.

[19] V. Naik, A. Arora, P. Sinha, and H. Zhang, "Sprinkler: A Reliable and Energy Efficient Data Dissemination Service for Extreme Scale Wireless Networks of Embedded Devices," IEEE Transactions on Mobile Computing, Vol.6, No.7, pp. 762-776, 2007.

[20] L. Phillips, "Aqueduct: Robust and Efficient Code Propagation in Heterogeneous Wireless Sensor Networks," Master's thesis, Univ. Colorado, 2005.

[21] RadioPulse, http://www.radiopulse.co.kr.

박 영 균

2007년 2월 대구대학교 전산통계학과 학사
2009년 8월 대구대학교 컴퓨터정보공학과 석사
2009년~현재 대구대학교 전산공학 박사과정

관심분야 : 임베디드 시스템, WSN, 분산 파일시스템

남 영 진

1992년 2월 경북대학교 전자공학과 학사
1994년 2월 포항공과대학교 전자전기공학과 석사
2004년 2월 포항공과대학교 컴퓨터공학과 박사

2004년~현재 대구대학교 컴퓨터·IT공학부 부교수
관심분야 : 임베디드 시스템, 저전력 스토리지