DOI: 10.3745/KIPSTA.2009.16-A.2.125

버전동기화 기반의 센서 네트워크 응용 소프트웨어 변경 관리 시스템의 구축 사례

김 재 철[†] · 김 주 일[††] · 정 기 원[†††] · 이 우 진[††††]

요 약

본 논문에서는 센서 네트워크의 중단 없이 노드의 응용 소프트웨어에 대한 결함 수정이나 기능 변경 및 개선 등을 효과적으로 관리할 수 있도록 지원하는 버전동기화 기반의 센서 네트워크 응용 소프트웨어 변경 관리 시스템을 제시한다. 응용 소프트웨어 변경 관리 시스템은 응용 소프트웨어 개발 환경인 NADE, 노드 관리 서버 및 노드 에이전트로 구성된다. NADE는 노드에 설치할 응용 소프트웨어를 개발하기 위한 Eclipse 기반의 개발환경으로 버전 관리 도구인 CVSNT를 연동하여 응용 소프트웨어에 대한 버전 관리를 수행한다. 노드 관리 서버는 NADE에서 개발한 응용 소프트웨어의 버전과 노드에서 수행되고 있는 응용 소프트웨어의 버전을 비교하여 동기화시킴으로써 노드에서 수행되는 응용 소프트웨어가 항상 최신의 버전으로 유지될 수 있도록 관리하며, 노드 에이전트는 노드에 탑재되어 노드의 정보를 서버에 주기적으로 전송하고, 수정된 노드의 정보를 저장 및 업데이트하는 기능을 수행한다. 제안한 버전동기화 기반의 센서 네트워크 응용 소프트웨어 변경 관리 시스템을 구현하면, 개발자들이 노드의 응용 소프트웨어에 대한 결함을 수정하거나 기능 변경 및 소프트웨어를 개선하여 응용 소프트웨어의 버전이 변경되면 자동으로 센서 노드의 응용 소프트웨어가 업데이트 되므로, 센서 네트워크를 중단하거나 지연시키지 않고 효과적으로 센서 네트워크 시스템의 실행을 관리할 수 있으며, 센서 네트워크 응용 소프트웨어의 변경 관리에 대한 가시성을 향상시킬 수 있을 것으로 기대한다.

키워드 : 센서 네트워크, 소프트웨어 변경 관리, 버전동기화

A System for Change Management of Sensor Network Applications based on Version Synchronization

Jaecheol Kim[†] · Juil Kim[††] · Kiwon Chong[†††] · Woojin Lee[††††]

ABSTRACT

This paper proposes a change management system of sensor network applications based on version synchronization that supports to effectively manage defect correction of applications, change of functions for applications or improvement of applications without suspending the sensor network. The proposed change management system consists of the NADE which is an application development environment, the Node Management Server, and the Node Agent. NADE is an Eclipse-based development environment for developing applications which are installed into nodes. NADE is also connected with CVSNT which is a version management tool and performs application version management using the CVSNT. Node Management Server manages nodes to maintain latest versions of applications by synchronizing versions of applications which are performed on the nodes with the versions of applications which are developed in the NADE. Node Agent which is loaded into the node periodically sends the version information of the application to the server, and stores and updates the version information of the application. Through the proposed change management system, applications of nodes are automatically updated when versions of applications are changed by correcting defects, changing functions or improving applications. Therefore, the user can effectively manage the execution of sensor network system without suspending or delaying the sensor network. Also, visibility of change management for sensor network applications will be improved.

Keywords : Sensor Network, Software Change Management, Version Synchronization

1. 서 론

센서 네트워크[1, 2]는 모든 사물에 전자태그 및 센서를 부착해서, 사물과 환경을 인식하고, 네트워크를 통해 실시간 정보를 구축 및 활용토록 하는 것으로서 궁극적으로는 모든 사물에 computing 및 communication 기능을 부여하여 anytime,

† 정 회 원 : 숭실대학교 컴퓨터학과 박사과정
†† 준 회 원 : 숭실대학교 컴퓨터학과 박사과정
††† 종신회원 : 숭실대학교 컴퓨터학부 교수
†††† 정 회 원 : 세종대학교 정보통신공학과 초빙교수

논문접수 : 2008년 6월 13일
수 정 일 : 1차 2008년 10월 20일
심사완료 : 2008년 11월 14일

anywhere, anything 통신이 가능한 유비쿼터스 환경을 구현하기 위한 것이다. 센서 네트워크는 물리적 세계와 디지털 세계를 연결할 수 있는 특징 때문에 많은 분야에 응용될 수 있다. 예를 들어, 홈 네트워크에서 집안의 침입 감지 및 가스 센서를 이용해 가스 안전 모니터링 등을 수행할 수 있고 산업현장에서는 위치 인식 서비스나 물류 관리 등에 사용될 수 있다. 또한 지능형 환경 모니터링으로 강수량 측정이나 산불 감시 등에 쓰일 수 있으며 각종 의료시스템이나 과학 분야에도 사용될 수 있어 그 가능성은 무궁무진하다.

이러한 센서 네트워크 환경의 실현을 위해서 센서 네트워크가 중단되지 않고 지속적으로 실행되어야 하는 것은 매우 중요한 사항 중의 하나이다. 이를 위해 노드의 응용 소프트웨어에 대한 결함 수정이나 기능 변경 및 개선을 수행한 후에도 센서 네트워크의 중단 없이 해당 노드에 응용 소프트웨어의 변경 사항을 효과적으로 반영하도록 관리할 수 있어야 한다.

기존에는 센서 네트워크를 구성하는 각 무선 센서 노드들의 응용 소프트웨어에 대한 변경이 발생하면 이를 반영하기 위하여 유선 통신을 이용한 소프트웨어 다운로드 방식을 사용하였다. 즉, 각처에 설치되어 동작중인 무선 센서 노드들을 중단시키고 시리얼 라인 혹은 패러럴 라인의 유선 통신을 이용하여 각 무선 센서 노드로 프로그램을 다운로딩 하는 방식이었다. 그러나 다운로드 방식은 센서 노드의 응용 소프트웨어의 재구성을 위해 센서 네트워크의 수행을 일시적으로 중단 시켜야 한다는 문제점을 가지고 있다. 이러한 다운로드 방식의 문제점을 해결하기 위해 노드의 응용 소프트웨어가 변경되면 센서 네트워크의 수행을 중단시키지 않으면서 동적으로 응용 소프트웨어를 재구성하기 위한 기법들[3, 4]이 연구되었는데, 이러한 기법들은 응용 소프트웨어를 동적으로 재구성하는 방법만을 제시할 뿐, 응용 소프트웨어의 체계적인 버전 관리를 통한 센서 노드의 관리 방법을 제시하지는 않고 있다. 또한 센서 네트워크 응용 소프트웨어를 효율적으로 관리하기 위한 방법들[9, 10]도 연구되었지만, 이러한 방법들도 센서 네트워크 응용 소프트웨어를 효율적으로 업데이트 하기 위한 방법들은 제안하지만 주기적인 버전 체크를 통해 변경사항이 생겼을 때 자동으로 업데이트 하는 버전 관리 기반의 체계적인 노드 관리 방법을 제시하지는 않는다.

이에 따라, 본 논문에서는 센서 네트워크 환경이 실현되었을 때 노드를 효율적으로 관리할 수 있도록 지원하는 버전동기화 기반의 응용 소프트웨어 변경 관리 시스템의 구축 사례를 보인다. 제안하는 응용 소프트웨어 변경 관리 시스템은 서버와 다른 버전의 응용 소프트웨어를 가진 노드에 대하여 서버와의 무선 통신을 통해 응용 소프트웨어를 동적으로 재구성함으로써 센서 네트워크의 중단 없이 서버와 노드간의 버전을 동기화시키도록 지원한다. 따라서 노드의 응용 소프트웨어에 대한 결함을 수정하거나 기능 변경 및 소프트웨어를 개선한 경우에 센서 네트워크를 중단하거나 지연시키지 않고 효과적으로 센서 네트워크 시스템의 실행을 관리할 수 있으며, 센서 네트워크 응용 소프트웨어의 변경 관리에 대한 가시성을 향상시킬 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 기존의 센서 네트워크 응용 소프트웨어를 관리하기 위한 방법들에 대하여 살펴보고, 3장에서는 응용 소프트웨어 변경 관리 시스템의 구조와 동작 방법을 설명한다. 4장에서는 응용 소프트웨어의 버전 정의 규칙과 응용 소프트웨어 변경 관리 시스템을 구성하는 응용 소프트웨어 개발환경, 노드 관리 서버 및 노드의 에이전트를 제시한다. 5장에서는 제시한 응용 소프트웨어 변경 관리 시스템의 구현 결과를 보이며, 6장에서는 구현한 시스템에 대하여 평가하고, 7장에서는 결론 및 향후 연구를 제시한다.

2. 관련 연구

현재 효율적으로 센서 네트워크 응용 소프트웨어를 관리하기 위한 여러 방법들이 연구되고 있다. [9]에서는 센서 네트워크에서 효율적인 소프트웨어의 관리를 위해 필요한 3가지 기법으로 feature models, profile-based reconfiguration, assistance operations을 제안한다. Feature model을 통해 서로 다른 모듈 간의 관계를 알고 특정 시나리오에서 실제로 필요한 기능을 선택할 수 있으며, profile-based reconfiguration에서는 profile을 통해서 특정 목적에 맞는 노드의 어플리케이션을 생성하기 위해 필요한 코드 프래그먼트와 템플릿을 선택하는 방법을 제공하며, assistance operations에서는 고장감내를 향상시키거나 여러 노드에 어플리케이션을 배포하기 위해 필요한 기능들을 제공한다. 센서 네트워크에서 강건하고 효율적인 소프트웨어 관리를 위해서는 이러한 3가지 기법이 필요하다고 논문에서는 이야기 하고 있다.

센서 네트워크 응용 소프트웨어 관리를 위한 또 다른 연구인 [10]에서는 센서 네트워크 소프트웨어 업데이트 기법을 조사하여 소프트웨어 업데이트 도구를 위한 모델 제시하였다. 제시하는 모델을 구성하는 3가지 컴포넌트는 센서 노드에서의 실행 환경, 네트워크에서의 소프트웨어 배포 프로토콜, 업데이트의 최적화 컴포넌트이다. 논문에서는 이러한 3가지 컴포넌트를 고려하여 도구를 만들어야 센서 네트워크 소프트웨어를 효율적으로 업데이트 할 수 있다고 말한다. 센서 노드에서의 실행 환경은 Monolithic 환경, Modular 환경, Virtual Machine이 있을 수 있으므로 이러한 환경을 고려해야 하며, 잘 설계된 업데이트 배포 프로토콜은 target selection, update delivery, completion verification의 3가지 컴포넌트를 가져야 한다고 말한다. 또한 센서 네트워크는 자원이 한정되어 있으므로, 업데이트의 최적화를 위해 업데이트 크기를 최적화해야 한다고 말한다.

이러한 기존의 연구들은 센서 네트워크 응용 소프트웨어를 효율적으로 업데이트 하기 위한 방법들은 제안하지만, 응용 소프트웨어를 업데이트 해야 하는 시기를 결정하는 방법에 대해서는 언급하지 않는다. 즉, 기존 연구들의 방법에는 주기적인 버전 체크를 통해 변경사항이 생겼을 때 자동으로 업데이트 하는 의사결정 부분이 없어서 사용자가 소프

트웨어의 업데이트 시기를 직접 결정해야 한다.

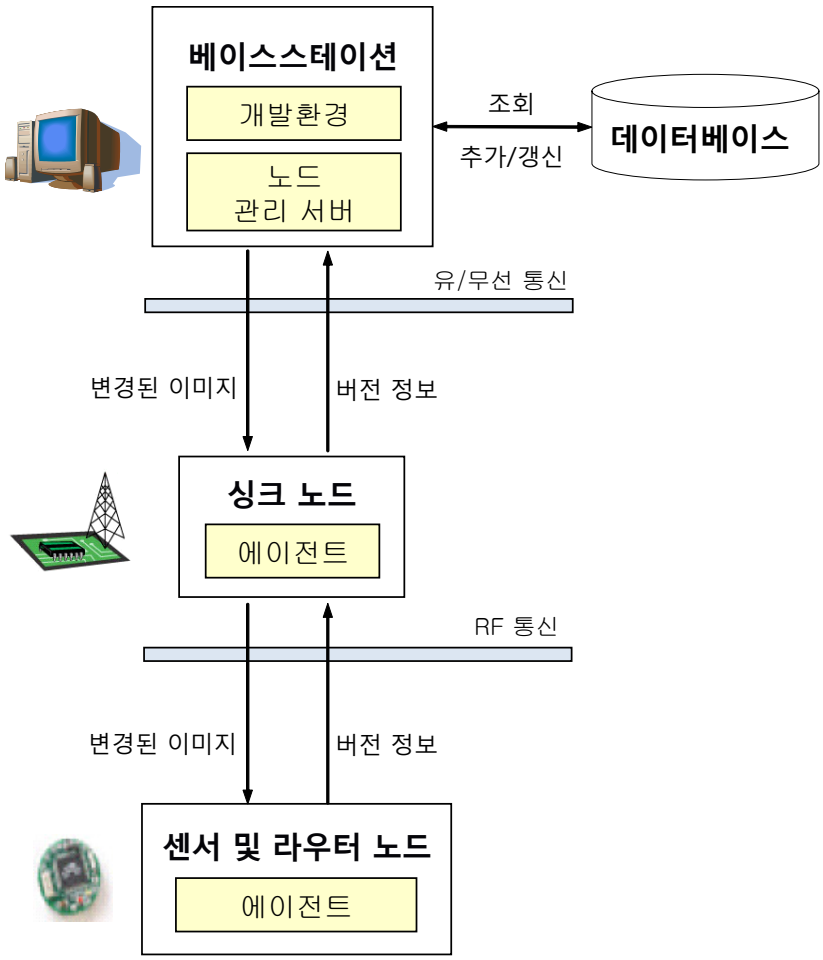
이러한 부분을 보완하기 위하여 본 논문에서는 버전동기화 기반의 센서 네트워크 응용 소프트웨어 변경 관리를 위한 시스템 구축 사례를 보인다. 본 논문에서 제안하는 시스템은 응용 소프트웨어를 업데이트 하는 것뿐만이 아니라 버전 관리를 통하여 소프트웨어의 업데이트 시기를 결정하고 자동으로 업데이트할 수 있도록 지원한다. 본 논문에서는 응용 소프트웨어를 업데이트 하는 방법보다는 주기적인 버전 체크를 통하여 버전이 변경되었을 경우에 자동으로 응용 소프트웨어를 업데이트 한다는 것에 초점을 두었다.

3. 시스템 아키텍처

(그림 1)은 본 논문에서 제시하는 응용 소프트웨어 변경 관리 시스템의 전체 구조를 보여준다. 응용 소프트웨어 변경 관리 시스템은 베이스스테이션에 존재하는 개발환경을 통하여 노드 응용 소프트웨어의 버전을 관리하고, 이를 바탕으로 노드 관리 서버에서는 각 노드가 수행하고 있는 응용 소프트웨어의 버전과 버전정보 저장소에 등록되어 있는 버전간의 버전동기화 기능을 제공한다.

응용 소프트웨어 변경 관리 시스템은 특정지역의 데이터를 수집하는 노드의 응용 환경의 변화로 인해 응용 소프트웨어의 변경이 필요한 경우나 노드의 성능개선을 위해 응용 소프트웨어를 수정할 경우에 새로운 버전의 응용 소프트웨어를 각 노드가 실시간으로 바로 사용할 수 있도록 동적으로 관리해 주는 시스템이다. 응용 소프트웨어가 수정되어 새로운 버전이 작성되면, 시스템은 각 노드의 버전 정보와 비교하여 버전이 일치하지 않는 노드로 최신 버전의 응용 소프트웨어를 동적으로 다운로드 함으로써 서버에서 작성한 응용 소프트웨어의 버전과 각 노드들이 사용하고 있는 버전을 동기화시킨다.

센서 네트워크를 구성하는 노드들의 가장 큰 역할은 자신이 속해 있는 해당 지역의 특정 환경에 대한 데이터를 감지(sensing)하여 서버로 전송하는 것이기 때문에 수집해야 하는 정보가 변경될 경우 응용 소프트웨어 변경 관리 시스템을 통하여 해당 노드가 필요한 정보를 수집할 수 있도록 응용 소프트웨어를 실시간으로 변경할 수 있다. 또한 여러 이유로 노드들의 응용 소프트웨어를 수정한 경우에도 센서 네트워크의 수행을 중지하지 않고 실시간으로 노드들의 응용



(그림 1) 센서 네트워크 응용 소프트웨어 변경 관리 시스템의 아키텍처

소프트웨어를 변경하여 유지할 수 있다. 이러한 응용 소프트웨어 변경 관리 시스템은 실시간으로 노드들의 응용 소프트웨어를 변경할 수 있으므로, 네트워크를 구성하는 노드들의 응용 소프트웨어가 변경된 경우에 구축된 노드들을 다시 수거하여 해당 노드에 각각의 프로그램을 적재하는 기존의 비효율적인 방식에서 벗어날 수 있다.

4. 시스템의 설계

본 장에서는 응용 소프트웨어 변경 관리 시스템에서 센서 네트워크 응용 소프트웨어의 버전 관리를 위해 버전을 정의하는 방법을 설명하고, 응용 소프트웨어 개발 환경과 노드 관리 서버 및 각 노드에서의 버전 관리를 위한 에이전트의 구조를 설명한다.

4.1 응용 소프트웨어의 버전 정의 규칙

버전 관리에 있어서 버전 넘버는 응용 소프트웨어의 버전 정보를 가장 함축적으로 표현하는 것이다. (그림 2)는 응용 소프트웨어 변경 관리 시스템에서의 버전 관리를 위한 버전 정의 규칙을 보여준다. 본 논문에서는 기존에 사용되고 있는 소프트웨어 버전 정의 규칙[5]을 센서 네트워크의 응용 소프트웨어에도 그대로 적용한다. 단, 센서 네트워크의 응용 소프트웨어는 각 노드의 역할을 수행하기 위한 응용 프로그램과 응용 프로그램에서 사용되는 운영체제 모듈을 포함하여 생성되므로, 응용 소프트웨어의 버전을 응용 프로그램의 버전과 운영체제 모듈의 버전으로 나누어서 관리하도록 한다. 응용 프로그램과 운영체제의 버전 정보를 별도로 관리하는 이유는 운영체제의 버전은 변경되지 않고 노드가 수행하는 역할 및 성능의 변화로 응용 프로그램만 변경되어야 하는 경우나 운영체제가 변경되어 응용 소프트웨어의 버전이 달라지는 경우가 발생하기 때문이다. 따라서 본 논문에서 제시하는 응용 소프트웨어 변경 관리 시스템은 각 노드에 설치되는 응용 소프트웨어를 구성하는 응용 프로그램과 운영체제 모듈 각각에 대하여 (그림 2)와 같은 버전 정의 규칙을 적용하여 버전 관리를 수행한다.

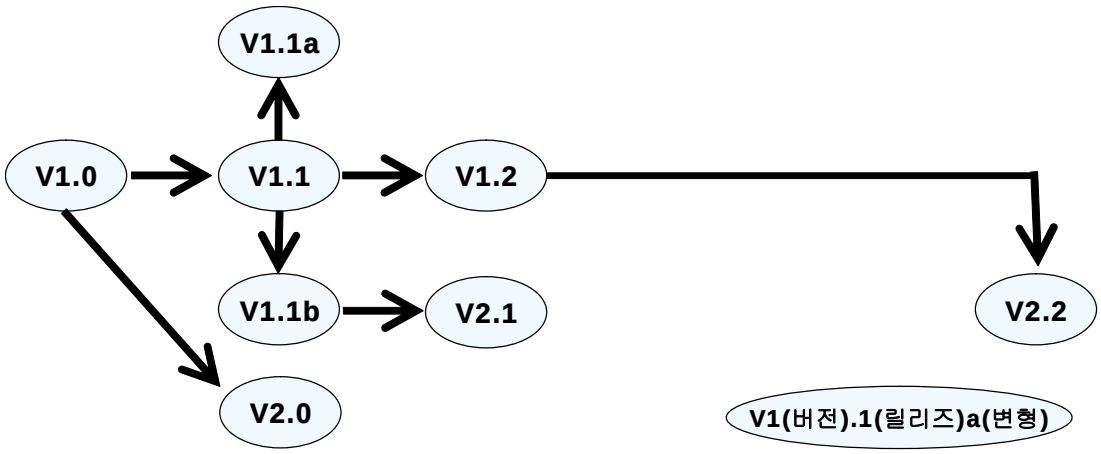
버전 넘버는 일반적으로 크게 버전(version), 릴리즈(release), 변형(variant)의 세 부분으로 구성된다. 버전은 기존 응용 소프트웨어와의 기능적인 차이를 나타내며 1부터 시작하는 정수를 사용한다. 릴리즈는 실제 노드로 배포되는 횟수로, 배포했던 버전을 수정한 후 다시 배포하는 횟수를 나타내며 0부터 시작하는 정수를 사용한다. 변형은 응용 소프트웨어의 기능 면에서는 차이가 없지만 처리속도, 성능 등과 같은 비기능적인 부분에서 차이가 있을 경우를 나타내며, 소문자 알파벳 a부터 시작한다.

응용 소프트웨어 변경 관리 시스템에서는 CVSNT[6]를 연동하여 버전 관리를 수행하는데, 응용 소프트웨어 개발 프로젝트에 태그를 부여하여 이를 버전 정보로 관리한다. 응용 소프트웨어 개발 후 노드에 배포하기 위한 이미지 파일을 생성하기 직전에 프로젝트에 태그를 부여하여 최종적인 버전이 되도록 한다.

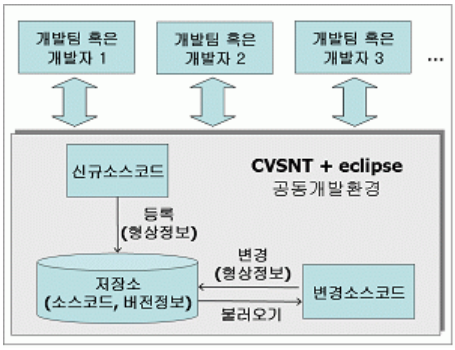
4.2 버전 관리를 위한 응용 소프트웨어 개발환경

본 논문에서는 응용 소프트웨어 변경 관리 시스템의 일부로 노드에 설치할 응용 소프트웨어를 개발하기 위한 Eclipse[7] 기반의 개발환경(NADE: Node Application Development Environment)을 제안한다. NADE는 크게 이미지 빌더와 이미지 다운로더로 구성된다. 이미지 빌더는 응용 소프트웨어의 소스 코드 생성기능, 응용 소프트웨어를 노드에 설치하기 위한 ROM 이미지 파일 생성기능, 수정된 소스코드를 재 컴파일하는 기능 및 생성된 소스코드의 버전을 입력하는 기능으로 구성되며, 이미지 다운로더는 노드에서 응용 소프트웨어를 수행하기 위하여 이미지 빌더가 생성한 ROM 이미지 파일을 노드로 다운로드 하는 기능을 수행한다.

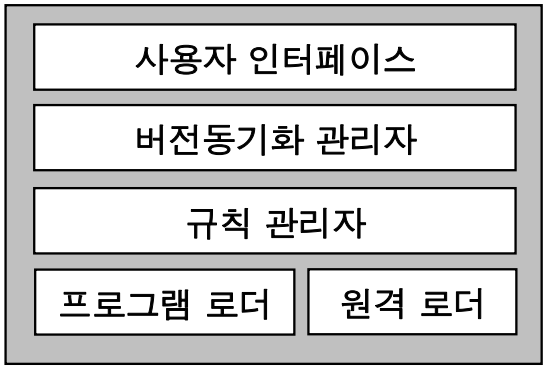
(그림 3)은 응용 소프트웨어의 버전 관리를 위한 NADE의 구조를 보여준다. NADE에서는 버전 관리 도구인 CVSNT를 연동하여 응용 소프트웨어에 대한 버전 관리를



(그림 2) 버전 정의 규칙



(그림 3) 노드의 응용 소프트웨어 변경 관리에 용이한 공동개발환경(NADE)의 구조



(그림 4) 노드 관리 서버의 구조

수행한다. 개발팀 혹은 개발자들은 신규로 생성되는 소스코드와 그 소스코드의 버전 정보를 저장소에 저장하고, 모듈의 변경이 필요할 경우 해당 모듈을 다시 불러와서 수정한 후에 리버전(reversion) 과정을 통해 소스코드를 수정하고 버전 정보 변경을 수락(commit)한다. 이러한 버전 관리 도구는 각 모듈을 CVSNT 서버에 저장하여 개발자들이 클라이언트로 접속하여 모듈의 변경에 대한 공동 작업이 가능하고, 별도로 모듈을 개발하기 위한 가지치기를 지원함으로써 수정에 대한 복구를 용이하게 한다.

4.3 노드 관리 서버

노드들이 수행하는 응용 소프트웨어의 버전을 실시간으로 관리하기 위해서는 NADE에서 개발한 응용 소프트웨어의 버전과 노드에서 수행되고 있는 응용 소프트웨어의 버전을 비교하여 동기화시킴으로써 노드에서 수행되는 응용 소프트웨어가 항상 최신의 버전으로 유지될 수 있도록 관리하기 위한 서버가 필요하다. 이러한 역할을 수행하도록 하기 위하여 본 논문에서는 노드 관리 서버를 제안한다. 노드 관리 서버는 베이스스테이션에 존재하면서 NADE에서 개발한 응용 소프트웨어의 버전과 각 노드들이 사용하고 있는 버전을 동기화시킨다.

(그림 4)는 노드 관리 서버의 구조를 보여준다. 노드 관리 서버는 사용자 인터페이스, 버전동기화 관리자, 규칙 관리자, 프로그램 로더, 원격 로더의 다섯 개의 모듈로 구성되는데, 각 모듈의 역할은 다음과 같다.

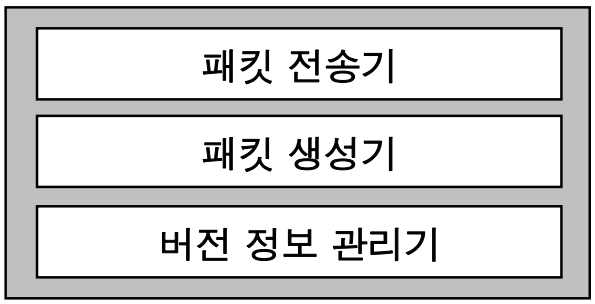
- 사용자 인터페이스: 사용자가 응용 소프트웨어 변경 관리 시스템을 사용할 수 있도록 하기 위한 사용자 인터페이스이다.
- 버전동기화 관리자: 버전동기화 관리자는 서버에서 개발한 응용 소프트웨어의 버전과 각 노드에서 수행되고 있는 응용 소프트웨어 사이의 버전을 동기화시키기 위한 핵심적인 역할을 수행한다. 각 노드로부터 전달된 버전 정보 및 서버에서 개발된 응용 소프트웨어의 버전 정보를 등록/수정하며, 버전 정보의 비교를 통하여 버전 동기화를 결정하고, 각 노드의 버전 동기화를 위한 규칙을 결정한다.
- 규칙 관리자: 규칙 관리자는 노드의 응용 소프트웨어를 재구성하기 위한 규칙을 관리하는 역할을 수행한다. 각 노드의 재구성 시점을 결정하는 사전규칙(사용자모드, 주기모드, 즉시모드)을 정의한다. 즉시모드(direct mode)는 항상 최근의 버전을 유지하고자 하는 노드에 대해 응용 소프트웨어가 변경되어 등록되면 자동으로 노드에 새로운 버전을 업데이트해주는 모드이며, 주기모드(period mode)는 어느 특정 주기 간격으로 변경해주는 모드이고, 사용자모드(user mode)는 반드시 사용자의 수락을 통해서만 변경해주는 모드이다.
- 프로그램 로더: 프로그램 로더는 타겟 노드의 응용 환경에 맞게 이미지 빌더에 의해 생성된 최초의 응용 소프트웨어 이미지를 타겟 노드에 적재한다.
- 원격 로더: 원격 로더는 변경된 응용 소프트웨어의 이미지를 무선통신을 통해 타겟 노드에 적재하는 기능을 수행한다.

4.4 노드 정보를 주기적으로 서버로 전달하기 위한 에이전트

서버에서 개발한 응용 소프트웨어와 노드에서 수행되는 응용 소프트웨어 간의 버전 동기화를 위해서는 각 노드에서 수행되고 있는 응용 소프트웨어의 버전 정보를 서버로 보내야 한다. 이를 위해 본 논문에서는 노드에 탑재되어 노드의 정보를 서버에 주기적으로 전송하고, 수정된 노드의 정보를 저장 및 업데이트하는 기능을 수행하는 에이전트를 제안한다.

(그림 5)는 각 노드에 탑재되어 있는 에이전트의 구조를 보여준다. 노드의 에이전트는 패킷 전송기, 패킷 생성기, 버전 정보 관리기의 세 개의 모듈로 구성되며, 각 모듈의 역할은 다음과 같다.

- 패킷 전송기: 패킷 전송기는 패킷 생성기가 생성한 각 노드의 응용 소프트웨어의 버전 정보를 포함하는 패킷을 서버로 전송한다.
- 패킷 생성기: 패킷 생성기는 현재 노드에서 실행되고 있는 응용 소프트웨어의 버전 정보를 서버로 전송하기 위한 패킷을 생성한다. 버전 정보 관리기로부터 응용 소프트웨어의 버전 정보를 받아 버전 정보를 포함하는 패킷을 생성한다.
- 버전 정보 관리기: 버전 정보 관리기는 각 노드에 설치되어 있는 응용 소프트웨어의 버전 정보를 관리한



(그림 5) 노드 에이전트의 구조

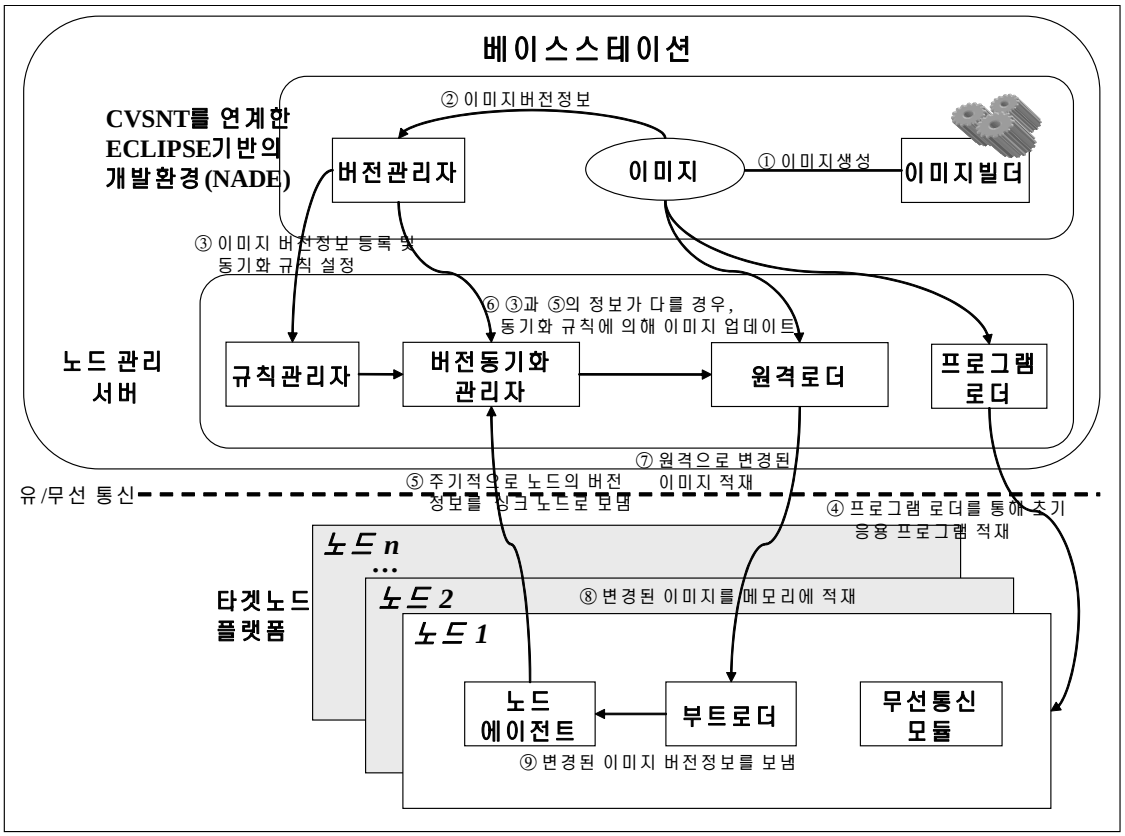
다. 응용 소프트웨어를 서버로부터 노드로 다운로드 하면 버전 정보 관리기는 다운로드 받은 응용 소프트웨어의 버전 정보를 등록/수정한다.

5. 시스템의 구현

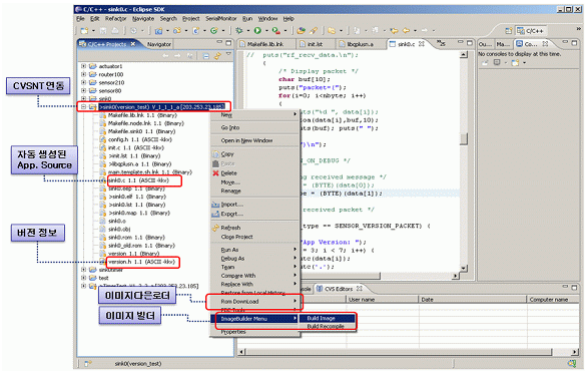
(그림 6)은 논문에서 제안하여 구현한 응용 소프트웨어 변경 관리 시스템을 통해서 노드에서 실행되는 응용 소프트웨어의 버전 관리를 수행하는 절차를 보여준다.

응용 소프트웨어 변경 관리 절차를 자세히 설명하면 다음과 같다.

- NADE에서 개발한 응용 소프트웨어로부터 이미지 빌더를 통하여 타겟 노드의 응용 환경에 맞게 타겟 노드에 설치할 이미지를 생성한다.
- 생성된 이미지에 대한 버전 정보가 NADE의 버전 관리자에 의해 생성된다.
- 노드 관리 서버의 버전동기화 관리자는 각 노드의 버전 정보를 저장하고, 규칙 관리자를 통하여 동기화 규칙을 설정한다.
- 최초로 생성된 응용 소프트웨어에 대한 이미지는 프로그램 로더에 의해 각 노드에 적재된다.
- 프로그램 로더에 의해 초기 응용 소프트웨어가 적재된 타겟 노드에는 원격으로 응용모듈을 재구성하기 위해 부트로더와 무선통신 모듈이 기본적으로 갖춰지며, 버전동기화 에이전트는 적재된 응용 소프트웨어의 버전 정보를 저장하고, 주기적으로 노드의 버전 정보를 싱크 노드로 전송한다.
- 응용환경의 변화에 의해 이미지가 변경되면 3번과 5번에서의 응용 소프트웨어의 버전 정보를 비교하여 버전이 다를 경우 설정한 동기화 규칙에 의해 타겟 노드의 응용 소프트웨어 이미지 변경 요청을 한다.
- 동기화 규칙에 의해 변경이 요청되면 원격 로더에 의해 새 이미지를 타겟 노드의 부트로더에 전송한다.
- 부트로더에 의해 새 이미지를 재구성한 후 완료되면 5번 절차부터 반복하여 수행한다.



(그림 6) 노드의 응용 소프트웨어 버전 관리 절차



(그림 7) NADE 실행 화면

(그림 7)은 응용 소프트웨어 변경 관리 시스템의 일부로 베이스스테이션에 설치되어 노드에 설치할 응용 소프트웨어를 개발하고 개발한 응용 소프트웨어에 대한 버전 관리를 수행하는 NADE를 구현하여 실행한 화면을 보여준다. 본 논문에서는 Nano-Qplus[8] 운영체제 기반의 센서 네트워크를 위한 응용 소프트웨어를 개발하여 각 노드에 설치하고, 응용 소프트웨어 변경 관리 시스템을 통하여 버전 관리를 수행하도록 하였다.

각 노드에서 실행되고 있는 응용 소프트웨어의 버전 관리를 위해서는 각각의 센서 및 라우터 노드에 존재하는 에이전트는 노드에서 실행되고 있는 응용 소프트웨어의 정보를 주기적으로 싱크 노드로 전달하고, 싱크 노드에 존재하는 에이전트는 각 노드들로부터 전달받은 정보를 노드 관리 서버로 전송해야 한다. 이를 위해서 에이전트는 각 노드에서 실행되고 있는 응용 소프트웨어에 대한 버전 정보를 포함하는 패킷을 생성하여 주기적으로 전송한다.

(그림 8)은 센서 노드 및 싱크 노드에서 응용 소프트웨어의 버전 정보를 전송하기 위해 에이전트가 생성한 패킷의 구조를 보여준다. 응용 소프트웨어 변경 관리 시스템에서는 각 노드에 설치되는 응용 소프트웨어의 버전을 각 노드의 역할을 수행하기 위한 응용 프로그램과 응용 프로그램에서 사용되는 운영체제 모듈로 구분하여 관리하므로, 버전 정보를 포함하는 패킷에는 응용 프로그램의 버전과 운영체제의 버전을 구분하여 저장한다.

(그림 9)는 라우터 노드에서 응용 소프트웨어의 버전 정보를 전송하기 위해 에이전트가 생성한 패킷의 구조를 보여준다. 라우터 노드는 자신의 응용 소프트웨어에 대한 버전 정보뿐만이 아니라 싱크 노드로 전송해야 하는, 다른 노드

NODE ID	PACKET TYPE	J_ID	J_APP_V	J_APP_B	J_APP_A	J_OS_V	J_OS_B	J_OS_A

✓ NODE ID: 송신노드 ID (1byte)
✓ PACKET TYPE: 송신 패킷의 타입 (1byte)
✓ J_ID: 버전 식별자 (1byte)
✓ J_APP_V: 응용 프로그램의 버전 (1byte)
✓ J_APP_B: 응용 프로그램의 릴리즈 (1byte)
✓ J_APP_A: 응용 프로그램의 변형 (1byte)
✓ J_OS_V: 운영체제의 버전 (1byte)
✓ J_OS_B: 운영체제의 릴리즈 (1byte)
✓ J_OS_A: 운영체제의 변형 (1byte)

(그림 8) 센서 노드 및 싱크 노드에서의 버전 정보 전송을 위한 패킷 구조

NODE ID	PACKET TYPE	DEST NODE ID	SOURCE NODE ID	HOP COUNT	ROUTER NODE ID	…	ROUTER J_ID	ROUTER APP VERSION	ROUTER OS VERSION	ORG DATA

✓ NODE ID: 송신노드 ID (1byte)
✓ PACKET TYPE: 송신 패킷의 타입 (1byte)
✓ DEST NODE ID: 최종 목적지 노드 ID (1byte)
✓ SOURCE NODE ID: 송신노드 ID (1byte)
✓ HOP COUNT: 거쳐온 노드 개수 (1byte)
✓ ROUTER NODE ID: 라우터 노드 ID (1byte)
✓ ……: HOP COUNT – 1 만큼의 ROUTER NODE ID {(HOP COUNT – 1) * 1byte}
✓ ROUTER J_ID: 버전 식별자 (1byte)
✓ ROUTER APP VERSION: 응용 프로그램의 버전, 릴리즈, 변형 (3byte)
✓ ROUTER OS VERSION: 운영체제의 버전, 릴리즈, 변형 (3byte)
✓ ORG DATA: 수신된 패킷 (가변적)

(그림 9) 라우터 노드에서의 버전 정보 전송을 위한 패킷 구조

로부터 전달받은 버전 정보도 포함해서 패킷을 생성해야 한다. 따라서 센서 노드나 싱크 노드에서의 패킷 구조와는 다른 패킷 구조를 가진다.

6. 평 가

이 장에서는 논문에서 제안하는 버전동기화 기반의 응용 소프트웨어 변경 관리 시스템을 구현하여 본 결과 발견된 문제점 및 센서 네트워크에서 중요한 메모리와 에너지 효율 측면에서 제안하는 시스템을 평가한 결과를 설명하고, 문제점을 해결하기 위한 해결책을 제시한다.

6.1 구현 후에 발견된 문제점

응용 소프트웨어 변경 관리 시스템을 구현하여 Nano-Qplus 기반의 센서 네트워크 환경에 적용하여 본 결과 크게 2가지의 한계점을 가지고 있음을 발견하였다. 첫째는 업데이트 후에 새로운 응용 소프트웨어를 실행시키기 위하여 재시작을 해야 한다는 것이다. 응용 소프트웨어의 업데이트가 동적으로 이루어지므로 업데이트를 위해 시스템을 중단시키지는 않지만 응용 소프트웨어 전체를 업데이트 하므로 새로운 응용 소프트웨어를 실행시키기 위해서는 재시작을 해야 한다. 따라서 기존 응용 소프트웨어의 수행 중에 저장된 상태 정보가 사라지게 된다. 둘째는 한 번에 하나의 노드에 대한 응용 소프트웨어만 업데이트가 가능하다는 것이다. 논문에서 제시하는 시스템은 응용 소프트웨어 전체를 업데이트 하므로 여러 노드의 응용 소프트웨어에서 공통으로 사용하는 모듈을 변경하였다고 하더라도 한 번에 여러 노드에 변경된 모듈만 업데이트 할 수 없고, 각 노드마다 응용 소프트웨어 전체를 업데이트 해야 한다.

6.2 메모리 효율

원격 업데이트 시 사용되는 메모리와 운영될 때 사용되는 메모리와 다른 영역이다. 일반 어플리케이션이 사용하는 메모리가 따로 있고, 업데이트 시는 부트로더에 있는 메모리를 사용한다. 따라서 본 논문에서 제시하는 시스템을 통하여 응용 소프트웨어를 업데이트 할 때 센서 네트워크 응용 소프트웨어의 메모리 효율에는 영향을 미치지 않는다.

6.3 에너지 효율

본 논문에서는 응용 소프트웨어를 업데이트 할 때, 응용 소프트웨어 전체를 하나의 이미지로 만들어서 업데이트 하므로, 한 번 업데이트 시에 노드는 전체 전력의 10-20% 정도의 전력을 소비한다. 따라서 본 논문에서 제시하는 시스템을 통하여 응용 소프트웨어를 업데이트 할 경우에는 노드의 에너지 효율이 많이 떨어지는 것을 알 수 있다. 그러므로 제시하는 방법은 응용 소프트웨어를 자주 변경하는 경우에는 비효율적이지만, 센서 네트워크의 특성 상 응용 소프트웨어에 오류가 발생하는 경우를 제외하고는 기능 변경을 자주 하지는 않기 때문에 제시하는 기법을 사용하여 응용 소프트웨어의 변경 관리를 수행하는데 큰 무리는 없을 것으로 본다.

6.4 해결책

응용 소프트웨어의 업데이트 후에 재시작을 하지 않기 위해서는 응용 소프트웨어 전체를 한번에 업데이트 하는 것이 아니라 부분 별로 점진적으로 업데이트 하는 방법을 사용해야 한다. 또한 한 번에 여러 노드에 대한 응용 소프트웨어를 업데이트 하기 위해서는 응용 소프트웨어 전체를 업데이트 하는 것이 아니라 모듈별로 업데이트가 가능하도록 해야 한다. 이러한 문제점들을 해결하기 위해서는 관련 연구에서 제시하는 방법들을 수용하여 모듈별로 변경 관리를 수행하여 업데이트 하도록 제안하는 시스템을 개선해야 할 필요가 있다.

제안하는 시스템의 에너지 효율을 높이기 위해서도 역시 관련 연구에서 제시하는 방법들을 수용하여 모듈별로 버전 관리를 수행하여 업데이트 할 수 있도록 해야 한다. 응용 소프트웨어를 모듈별로 업데이트를 하게 되면 업데이트 크기가 작아져서 응용 소프트웨어 전체를 업데이트 하는 것보다 전력 소비를 줄일 수 있다. 이렇게 에너지 효율을 높임으로써 응용 소프트웨어의 잦은 변경도 효율적으로 관리할 수 있게 된다.

7. 결 론

센서 네트워크는 물리적 세계와 디지털 세계를 연결할 수 있는 특징 때문에 홈 네트워크, 위치 인식 서비스나 물류 관리와 같은 산업 현장, 지능형 환경 모니터링, 의료시스템이나 과학 분야와 같은 많은 분야에 응용될 수 있다. 이러한 센서 네트워크 환경의 실현을 위해서 센서 네트워크가 중단되지 않고 지속적으로 실행되어야 하는 것은 매우 중요한 사항 중의 하나이다. 이를 위해 노드의 응용 소프트웨어에 대한 결함 수정이나 기능 변경 및 개선을 수행한 후에도 센서 네트워크의 중단 없이 해당 노드에 응용 소프트웨어의 변경 사항을 효과적으로 반영하도록 관리할 수 있어야 한다.

이에 따라, 본 논문에서는 센서 네트워크 환경이 실현되었을 때 노드를 효율적으로 관리할 수 있도록 지원하는 버전동기화 기반의 응용 소프트웨어 변경 관리 시스템을 제시하였다. 응용 소프트웨어 변경 관리 시스템은 응용 소프트웨어 개발 환경인 NADE, 노드 관리 서버 및 노드 에이전트로 구성된다. NADE는 노드에 설치할 응용 소프트웨어를 개발하기 위한 Eclipse 기반의 개발환경으로 버전 관리 도구인 CVSNT를 연동하여 응용 소프트웨어에 대한 버전 관리를 수행한다. 노드 관리 서버는 NADE에서 개발한 응용 소프트웨어의 버전과 노드에서 수행되고 있는 응용 소프트웨어의 버전을 비교하여 동기화시킴으로써 노드에서 수행되는 응용 소프트웨어가 항상 최신의 버전으로 유지될 수 있도록 관리하며, 노드 에이전트는 노드에 탑재되어 노드의 정보를

서버에 주기적으로 전송하고, 수정된 노드의 정보를 저장 및 업데이트하는 기능을 수행한다.

본 논문에서 제안한 버전동기화 기반의 응용 소프트웨어 변경 관리 시스템은 다음과 같은 장점을 가진다. 첫째, 응용 소프트웨어 변경 관리 시스템은 서버와 다른 버전의 응용 소프트웨어를 가진 노드에 대하여 서버와의 무선 통신을 통해 응용 소프트웨어를 동적으로 재구성함으로써 센서 네트워크의 중단 없이 서버와 노드간의 버전을 동기화시키도록 지원한다. 따라서 노드의 응용 소프트웨어에 대한 결함을 수정하거나 기능 변경 및 소프트웨어를 개선한 경우에 센서 네트워크를 중단하거나 지연시키지 않고, 효과적으로 센서 네트워크 시스템의 실행을 관리할 수 있게 된다. 둘째, 노드 관리 서버에서 각 노드에 설치되는 응용 소프트웨어에 대한 버전 정보 몇 변경 이력을 저장하여 관리하므로 개발자들은 노드에서 실행되고 있는 응용 소프트웨어의 상태를 서버를 통해서 쉽게 파악할 수 있으며, 응용 소프트웨어에 대한 변경 및 관리가 용이해지고, 변경 관리에 대한 가시성을 향상시킬 수 있다. 셋째, 응용 소프트웨어에 대한 버전이 변경되면 서버에서는 자동으로 해당 노드로 변경된 소프트웨어를 업데이트하므로, 서버와 노드 간의 버전동기화가 이루어져 잘못된 버전의 응용 소프트웨어로 인하여 센서 네트워크 시스템이 오류를 일으키는 것을 막을 수 있다.

향후에는 본 논문에서 제시하는 시스템의 문제점과 에너지 효율 부분을 극복하기 위하여 관련 연구들에서 제시하는 기법들을 검토하고 반영하여 더욱 효율적인 버전동기화 기반의 응용 소프트웨어 변경 관리 시스템을 개발하도록 연구할 것이다.

참 고 문 헌

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless Sensor Networks: A Survey,” Computer Networks, 38(4):393 - 422, 2002.

[2] Shigeru Fukunaga, Tadamichi Tagawa, Kiyoshi Fukui, Koichi Tanimoto, and Hideaki Kanno, “Development of ubiquitous sensor network,” Oki Technical Review, Vol.71, No.4, pp.24-29, 2004.

[3] Balasubramanian, J., Natarajan, B., Schmidt, D.C., “Middleware Support for Dynamic Component Updating,” On the move to meaningful internet systems 2005: CoopIS, DOA, and ODBASE, 2005.

[4] Sivaharan, T., Blair, G., Coulson, G., “GREEN: a configurable and re-configurable publish-subscribe middleware for pervasive computing,” Lecture Notes in Computer Science 3760, pp.732-749, 2005.

[5] Ivan Sommerville, Software Engineering 5th edition, Addison-Wesley, 1996.

[6] http://www.cvsnt.org

[7] Eric Clayberg and Dan Rubel, Eclipse: Building Commercial- Quality Plug-ins, Addison Wesley, 2004.

[8] Kwangyong Lee et al., “A Design of Sensor Network System based on Scalable & Reconfigurable Nano-OS Platform,” IT-SoC2004, 2004.

[9] Wolfgang Schr¨oder-Preikschat, R¨udiger Kapitza, J¨urgen Klein¨oder, Meik Felser, Katja Karmeier, Thomas Halva Labella, and Falko Dressler, “Robust and Efficient Software Management in Sensor Networks,” 2nd IEEE/ACM International Conference on Communication System Software and Middleware, 2007.

[10] Chih-Chieh Han, Ram Kumar, Roy Shea, Mani Srivastava, “Sensor Network Software Update Management: A Survey,” ACM International Journal on Network Management, Vol.15, No.4, 2005.

김 재 철

e-mail : tbjckim@yahoo.co.kr

1994년 경원대학교 독어독문학과(학사)

2004년 숭실대학교 정보과학대학원(공학석사)

1995년~현 재 한냉(주) 정보지원팀 팀장

2005년~현 재 숭실대학교 컴퓨터학과 박사과정

관심분야 : 유비쿼터스 컴퓨팅, 엔터프라이즈 아키텍처, 프로젝트 관리

김 주 일

e-mail : sespop@empal.com

2004년 한밭대학교 컴퓨터공학과(학사)

2006년 숭실대학교 컴퓨터학과(공학석사)

2006년~현 재 숭실대학교 컴퓨터학과 박사과정

관심분야 : 유비쿼터스 컴퓨팅, 임베디드 시스템, 웹 서비스

정 기 원

e-mail : chong@ssu.ac.kr

1967년 서울대학교 전기공학과(학사)

1981년 미국 알라바마 주립대학(헌츠빌) 전산학과(공학석사)

1983년 미국 텍사스주립대학(알링턴) 전산학과(공학박사)

1990년~현 재 숭실대학교 컴퓨터학부 교수

관심분야 : 프로세스, 모델링, 방법론, 실시간 응용, 전자거래 등

이 우 진

e-mail : bluewj@empal.com

2000년 숭실대학교 컴퓨터학부(학사)

2002년 숭실대학교 컴퓨터학과(공학석사)

2007년 숭실대학교 컴퓨터학과(공학박사)

2007년~2008년 한국정보통신대학교 공학부 박사후과정생

2009년~현 재 세종대학교 정보통신공학과 초빙교수

관심분야 : 유비쿼터스 컴퓨팅, 임베디드 시스템, SOA, 프로덕트 라인