3

무선 센서 네트워크 운영체제

연세대학교 ▌최석원* · 신효정* · 윤찬민 · 최학수 · 차호정**

1. 서 론

무선 센서 네트워크 기술은 소형 MCU(Micro-Controller Unit) 및 무선 통신 칩을 바탕으로 하는 저가의 센서 노드를 대량으로 배포하여 데이터 수집 및 중요 이벤트를 감지하기 위한 기술이며 최근 건축, 과학, 해양 등과 같은 다양한 분야에 적용 되고 있다[1]. 기술 영역의 확대는 센서 응용의 특성을 다양화 시켰으며, 센서 노드 플랫폼도 여러 형태로 개발 되고 있다. 센서 노드 플랫폼은 대량 배포라는 특징에 의해 초소형 저비용 하드웨어로 제작되며[2,3] 센서 응용은 하드웨어 자원의 제약으로 인해 적용 분야의 요구사항에 최적화 되어 개발되어야 한다. 이는 응용 개발자에게 하드웨어에 대한 많은 지식을 요구하게 되며 결국 센서 응용 개발에 어려움을 야기한다. 무선 센서 네트워크를 위한 운영체제(이하 센서 운영체제)는 센서 노드의 자원 제약을 극복하고, 다양한 분야의 응용개발을 쉽게 하기 위하여 개발되었다. 초기에는 자원 제약의 극복, 장치사용의 독립성, 유연성 보장 등의 기본적인 목적에 맞춘 소형 운영체제 형태로 개발되었으나 최근에는 기술 상용화에 대한 요구를 충족시키기 위하여 운영체제의 안정성 보장에서부터 보안, 무선 네트워크 기술, 파일 시스템, 저전력 관리 기술 등의 운영체제로서의 전반적인 기능은 물론 사용자 편의 및 대규모 센서 배포를 위한 여러 GUI 툴과 같은 실용화 기술에 대한 연구가 활발하게 진행 중이다. 본 논문에서는 센서 운영체제의 핵심 기술을 소개하고, 대표적인 센서 네트워크 운영체제들에 대한 분석을 통해 각각의 운영체제들에서 이러한 기술이 어떻게 적용되었는지 알아본다.

2. 센서 운영체제 기술

전형적인 센서 노드 플랫폼은 8-bit 또는 16-bit의 마이크로 컨트롤러를 중심으로, IEEE 802.15.4와 같은 저전력 무선 통신 칩을 사용하며, 배터리나 태양전지 판과 같은 제한된 용량의 전원으로 동작한다. 또한 센서 플랫폼에 사용되는 저 성능 마이크로 컨트롤러들은 대부분 100kB 이하의 코드 메모리 영역과 20kB 이하의 RAM 영역을 가진다. 다음은 이 같은 제약이 있는 환경에서 효율적인 자원관리와 응용 지원을 위해 필요한 센서 운영체제의 핵심 기술에 대해 설명한다.

2.1 Event driven vs. Multi-threading

대부분의 범용 운영체제는 멀티 스레드 기반의 작업 수행 모델을 지원하고 있으나, 초기 센서 운영체제는 시스템 리소스의 제약을 해결하기 위해 이벤트 기반의 작업 수행 모델을 사용하였다[4]. 이벤트 기반의 모델은 네트워크, 시스템 타이머, 외부 센서 등을 통해서 이벤트가 발생할 경우 이벤트 핸들러를 통해 작업을 수행하는 프로그래밍 모델이며 스레드 기법에 비해 메모리 사용량이 적고 구현이 간단한 특징을 갖는다. 이벤트 핸들러는 실행 도중 다른 이벤트가 발생하더라도 실행중인 작업을 멈추지 않고 끝날 때까지 수행한다(run-to-completion). 이와 같은 특징 때문에 이벤트 핸들러 실행 중에 우선순위가 높은 또 다른 작업이 지연되는 경우가 발생하며, 하나의 이벤트 핸들러에서 발생되는 지연은 전체적인 시스템 성능의 저하를 가져오기도 한다. 한편, 멀티 스레드 기법이 스케쥴링 등의 오버헤드를 극복하며 다양한 센서 운영체제들에 의해 채택되고 있다[5-8]. 멀티 스레드 모델이 가지는 스케줄링 및 스위칭 오버헤드는 variable timer, 싱글스택 커널과 같은 기법을 이용하여 해결하고 있다[9].

2.2 프로그래밍 언어

초기 센서 네트워크 운영체제는 이벤트 기반의 수행모델을 가지고 있어 새로운 프로그래밍 언어를 사

† 이 논문은 2007년도 한국과학재단의 국가지정연구실사업(NRL)으로 수행된 연구임(No. M10500000059-06J0000-05910).

* 학생회원
** 종신회원

용하였다. TinyOS에서 사용하는 nesC가 대표적인 경우이다[10]. nesC는 C 언어의 문법을 사용하면서 이벤트 핸들러 단위로 프로그래밍 하도록 되어 있어, 이벤트 기반의 운영체제의 응용개발에 최적화 되어 있다. 또한 컴포넌트 단위 프로그래밍이나 와이어링 등과 같은 개념을 이용하여 코드 재 사용성을 극대화 시켰다. nesC와 같은 새로운 개념의 프로그래밍 문법은 이벤트 기반의 운영체제 개념을 프로그래밍 언어로 표현 하였다는 장점을 가지고 있으나 프로그램의 크기가 커질 경우 프로그래밍 복잡도가 늘어난다는 단점이 있다. 한편, 응용의 개발 비용을 줄이고 유연성을 제공하기 위해 가상머신(VM, Virtaul Machine)과 스크립트 언어가 제안되었다[11]. 스크립트 프로그래밍은 프로그램이 간단하고 완벽한 커널의 추상화가 가능하여 처음 등장 시 많은 주목을 받아왔으나 VM이 가지는 코드 및 실행 오버헤드가 큰 단점으로 지적 받고 있다.

최근에 개발되고 있는 센서 운영체제는 대부분 응용 개발을 위한 프로그래밍 언어로써 C 언어를 지원한다[6,7,12]. 특히, 센서 운영체제의 경향이 멀티 스레드 모델로 옮겨가면서 스레드 문법을 표준 C 구문으로 표현 할 수 있도록 하는 추세이다.

2.3 시스템 모듈화

센서 네트워크는 한번 프로그래밍되어 배포된 센서 노드의 지속적인 관리가 필요하고, 이를 위하여 네트워크 프로그래밍 기술이 필요하다[13]. 네트워크 프로그래밍은 배포된 전체 노드를 재 프로그래밍 하는 기술이며, 재 프로그래밍 되는 코드의 크기에 따라 네트워크 및 전력 오버헤드가 큰 차이를 가지므로 배포되는 코드의 크기를 줄이기 위한 다양한 방법이 연구되고 있다. 이러한 기술 중 운영체제 관점에서 연구되고 있는 것이 시스템 모듈화이다. 초기 센서 네트워크 운영체제는 커널과 응용이 하나의 이미지로 컴파일 되어 네트워크 프로그래밍 이미지가 하나로 배포되었다. 따라서 커널 이미지가 이미 배포된 응용이미지에 중복됨으로 오버헤드를 증가시켰다. 이러한 오버헤드를 줄이기 위해 운영체제 수준에서 고려된 것이 시스템 모듈화이고, 이를 이용하여 각각의 응용 또는 기능 별로 프로그래밍 및 배포가 가능하다.

2.4 시스템 보호 기법

센서 네트워크에서 사용되는 센서 하드웨어는 일반적으로 MMU(Memory Management Unit)가 없다. 따라서 잘못된 응용코드의 수행은 시스템 안정성에 좋지 않은 영향을 주고, 커널을 보호하기 위한 다양한 기법이 연구되고 있다. 대표적인 방법으로 메모리 보호 기법, 응용 코드 검사, 스택 보호 기법 등이 있다. 메모리 보호 기법은 응용의 코드가 시스템이 사용하는 주요 메모리 공간을 침범하여 변경할 수 없도록 방지하는 기법이다. 응용 코드 검사 기법은 응용과 커널이 실행코드 단계에서 완전히 분리된 경우에 사용 가능한 시스템 보호 기법으로, 컴파일 단계에서 확인 가능한 오류를 찾는 정적 코드 검사와 응용 실행 중에 확인 가능한 오류를 찾는 동적 코드 검사가 있다. 스택 보호 기법은 멀티 스레드를 사용하는 운영체제에서 필수적인 시스템 보호 기법으로 응용 수행 중 스택 영역 침범으로 인해 커널이 손상되는 것을 방지한다. 스레드의 스택 사용량을 고정시키고 오버플로우 되는 것을 체크하는 방법과, 스레드가 사용할 스택사용량을 컴파일 단계에서 예측하고, 그 예측된 양만큼을 할당하는 방법이 있다.

2.5 네트워크 추상화

네트워크 추상화는 네트워크 응용의 일관성 있는 개발을 위한 필수 요소이며 커널의 독립성 보장을 위해서도 필요하다. 범용 운영체제의 경우에는 OSI 7 레이어에 기반한 TCP/IP 스택 등의 표준을 이용하여 네트워크 추상화를 구현하였다. 하지만 센서 노드는 부족한 시스템 리소스에도 불구하고 요구되는 네트워킹 기법이 다양하므로 최근 네트워크 추상화의 중요성을 인식하고 각 운영체제 별로 활발한 연구를 진행하고 있다[15-19].

네트워크 추상화 기법으로 가장 먼저 제시된 것은 UC Berkeley의 SP(Sensornet Protocol) 이다. SP는 MAC 레이어와 상위 네트워크 레이어 부분을 SP라 불리는 구조를 통해 분리시킴으로써 미리 준비된 API를 이용하여 네트워크 프로토콜 개발을 가능하게 하였다[15,16]. 연세대학교는 기존 기능 위주의 네트워크 추상화 구조가 아니라 프로그래머 관점에 입각하여 구성한 네트워크 추상화 구조인 NSL(Networking Support Layer)을 제안하였다. 센서 네트워크 프로그래머를 커널, 네트워크, 응용 프로그래머로 구분하고, 네트워크 스택을 3단계 구조로 정의하여 응용 의존적으로 변경되는 레이어를 최소화 하는 방법이다[17]. Swedish Institute의 Chameleon은 네트워크 추상화를 위해 운영체제에 패킷 헤더 변환 기능을 가진 Chameleon 스택과 표준화된 네트워크 API를 정의하는 Rime 스택을 제안하였다. 이를 이용하여 LR-LAN, WLAN, Ethernet 등 다양한 네트워크 프로토콜을 사용하여도 하나의 시스템 API로 프로그램을 작성할 수 있다[18]. 최근

에는 기존 인터넷 망과의 연동을 위해 IP프로토콜을 센서 네트워크에 구현하려는 움직임이 일어나고 있다. 특히 IPv6를 기반으로 하는 6LoWPAN 표준을 센서 운영체제에서 지원하여 기존 네트워크 인프라에 센서 네트워크를 접목시키고자 하는 연구가 활발히 진행 중이다[19].

2.6 시스템 개발 및 모니터링 도구

일반적으로 개발 환경이 매우 제한된 센서 네트워크 시스템의 경우, 빠르고 쉬운 개발 프로세스를 위한 각종 도구들의 개발은 중요하다. 센서 네트워크의 개발 환경을 개선하기 위한 많은 연구들이 진행되어 왔으며 지금까지 진행된 연구들은 크게 통합 개발 환경, 디버깅 도구, 모니터링 도구의 세가지 주제로 구분할 수 있다.

통합 개발 환경(IDE)은 TinyOS를 지원하는 Crossbow 사의 Moteworks가 있다[30]. 버전 2.0까지 발표된 상태이며 MoteConfig와 TinyOS의 개발언어인 nesC를 지원하는 Programmer's Notepad[31] 환경으로 구성된다. MoteConfig는 별도의 프로그램 다운로드 과정 없이 사용자가 모트의 ID, RF 채널, RF 송신 세기 등을 편리하게 변경할 수 있는 프로그램이며 Programmer's Notepad는 통합 개발 환경의 손쉬운 구축을 위한 가볍고 강력한 오픈소스 프로젝트이다. NanoQplus의 통합 개발 환경인 NanoEsto[32]는 오픈 소스 IDE인 Eclipse [33]의 플러그 인 형태로 제작 되었으며 Eclipse에 이미 익숙한 사용자들이 쉽게 적응 할 수 있다는 장점이 있다. RETOS에서 제공하는 센서 네트워크 모니터링 도구로는 RETOS Monitoring Tool(RMTool)이 있다 [34]. RMTool은 컴포넌트 기반의 네트워크 모니터링 툴로서 개발자로 하여금 네트워크의 상태를 손쉽게 진단, 분석하고 하고 동적으로 재구성하는 기능을 제공한다. Moteview[35]는 TinyOS를 지원하는 네트워크 모니터링 툴로서 Motework 2.0에 포함되어 있다. Moteview는 다수의 센서 데이터 스트림을 손쉽게 그래프화 하고 분석할 수 있는 기능을 제공하고 네트워크 토폴로지 및 각종 상태 등을 GUI기반으로 출력한다.

디버깅을 도구로는 TinyOS를 지원하는 소스 레벨 디버거인 Clairvoyant이 있다[36]. Clairvoyant는 GDB에서 제공하는 표준 디버깅 명령어와 WSN을 위한 특별한 명령어들을 제공하며 특히 센서 네트워크의 디버깅이 무선으로 가능하다는 것이 특징이다. NanoEsto 또한 Eclipse에서 제공되는 GUI 디버깅 기능을 바탕으로 AVR계열의 MCU에서 사용되는 JTAG과 GDB를 통해 소스 레벨 디버깅을 지원하고 있다.

3. 센서 운영체제 소개

3.1 TinyOS

TinyOS[4]는 UC Berkeley에서 개발되었으며 센서 네트워크의 자원 제약적 특성에 적합한 응용을 개발할 수 있는 센서 네트워크 운영체제이다. TinyOS는 다양한 하드웨어 및 시스템 기능을 모듈의 형태로 지원하며 프로그램은 모듈화 된 컴포넌트를 연결하고 컴포넌트 사이에 이벤트를 주고받게 함으로써 전체 기능을 수행한다. 모든 시스템 컴포넌트와 응용 코드는 TinyOS 프로그래밍 언어인 nesC[10]로 개발되며, nesC 컴파일러는 컴포넌트를 서로 조합하고 커널을 포함하여 한 개의 바이너리를 생성한다. TinyOS는 이벤트 기반(event-driven), 그리고 단일(monolithic) 커널 센서 운영체제의 특징을 가진다. 최근 개발된 TinyOS 2.0[16]의 경우는 커널 컴포넌트를 명확히 정의하고, 네트워크 레이어의 추상화를 통해 기존의 TinyOS 1.0보다 유연한 운영체제로 업그레이드 되었다. TinyOS는 지금까지 개발된 센서 네트워크 운영체제 중 가장 많은 사용자를 보유하고 있으며 세계 각지의 개발자들이 시스템 커널, 드라이버, 시뮬레이션 툴, 개발 툴킷 등의 분야에서 도움을 주고 있다.

3.2 SOS

SOS[14]는 모듈 기반의 센서 운영체제로써 UCLA의 CENS연구팀이 개발하였다. 커널을 정적인 모듈과 동적 모듈 두 개의 영역으로 구분하여 개발 및 배포 가능하다. 동적 모듈은 커널과 독립적으로 개발되고 링크되어 시스템이 배포된 후 수정이 비교적 자유롭다는 장점을 가진다. 가상 메모리(Virtual Memory)를 지원하지 않는 센서 네트워크에서 동적 모듈을 지원하기 위해 SOS는 각 모듈을 상대 주소 참조(Position Independent Code)방식을 사용한다. 동적 모듈과 정적 커널은 상호 간 통신을 위해 이벤트를 사용하며 우선순위 기반의 이벤트 스케쥴링 기법을 제공한다. 또한, 동적 메모리 할당 및 메모리 보호 기법 역시 지원하고 TinyOS 와 같은 이벤트 모델로 응용을 개발하지만, 개발 언어는 C를 채택하였다.

3.3 MANTIS

MANTIS[5]는 콜로라도 대학에서 개발된 멀티 스레드 지원 및 빠른 프로토타이핑을 목표로 개발된 센서 운영체제이다. MANTIS의 스레드는 다섯 단계의 우선순위를 가지고 있으며 선점 가능하여 작은 단위로 스케쥴링을 수행할 수 있다. 기본 스케줄링 기법으로 미리 정의된 단위 시간을 기준으로 하는 라운드 로빈

스케쥴링을 사용하였다. MANTIS 네트워크 스택은 제로 카피를 지원하여 불필요한 메모리 작업을 줄이고 보다 빠른 IO 성능을 보인다. MANTIS 는 TinyOS 와 같이 커널과 응용이 함께 하나의 바이너리로 컴파일 되고 실행 시 응용의 스레드는 커널의 메모리와 별도의 스택에서 작업을 수행한다. 세마포어 등 기본적인 동기화 기법을 지원한다.

3.4 Contiki

Swedish Institute of Computer Science에서 개발된 Contiki[6]는 멀티태스킹 지원을위한 센서 운영체제이다. 커널은 이벤트 드리븐 방식이며 응용 프로그램이 런타임에 동적으로 적재되며 protothreads[20]라는 light-weight 스레드를 사용하여 멀티 태스킹 응용 개발 환경을 제공한다. 프로세스 단위의 선점형 멀티 스레드와 이벤트를 통한 메시지 전달 방식의 프로세스 간 통신을 지원하며 GUI 시스템을 지원한다. Contiki는 MSP430과 AVR계열의 마이크로 컨트롤러를 포함하여 다양한 종류의 플랫폼에 이식 되어 있다. 2007년 4월에 바이너리 이미지의 동적 적재기능과 Rime 통신 스택, 그리고 Cooja라 불리는 네트워크 시뮬레이터가 포함된 버전 2.0이 발표되었다.

3.5 t-kernel

t-kernel[21]은 University of Virginia에서 개발된 센서 운영체제로 64kB의 가상메모리를 지원하고 선점형 스케쥴링과 운영체제 차원의 보호 메커니즘 제공을 특징으로 갖는다. 사용자가 작성한 응용 프로그램의 바이너리 이미지를 load-time에 직접 변경하여 운영체제 보호기법, 가상 메모리, 선점형 스케쥴링을 포함하는 프로그램을 생성한다. 응용 프로그램의 바이너리를 직접 변경하므로 사용자가 어떠한 언어로 프로그램을 작성하여도 상관없으며 nesC로 작성된 TinyOS 바이너리 코드 또한 최소한의 수정을 거쳐 실행 가능하다. t-kernel의 응용 프로그램은 인터럽트 제어를 포함한 하드웨어 제어에 관해 모든 권한을 가지고 있다. 사용자가 작성한 인터럽트 루틴을 포함 시킬 수도 있으며 각종 레지스터나 핀 조작 등이 모두 응용 프로그램에서 직접 가능하다. 따라서 새로운 디바이스를 위한 드라이버를 작성할 경우 t-kernel 내부를 변경할 필요가 없고 모두 응용 프로그램 내에서 해결이 가능하다. 현재 Mica2계열의 모트(Mica2, XSM, ExScal 등)에 이식이 되어 있다. 베타버전이 곧 릴리즈 될 예정이다.

3.6 LiteOS

LiteOS는 University of Illinois at Urbana-Champaign 에서 개발 되었으며 센서 네트워크에서 UNIX와 비슷한 운용 및 개발 환경을 제공하는 것이 주된 특징이다[22]. C 프로그래밍은 물론 C++ 기반의 객체 지향 프로그래밍 환경을 제공하고 스레드의 런타임 동적 적재를 지원하며 UNIX/Linux 사용자들에게 친숙한 파일 시스템이 제공된다. LiteOS의 개발 환경은 PC에서 동작하는 쉘 프로그램이 중요한 역할을 한다. 이 쉘 프로그램은 ls, mkdir, ps, man과 같은 UNIX 명령어들을 구현하였으며 센서 네트워크의 파일 시스템 혹은 프로세스 실행 등을 관리한다. 응용 프로그램을 모트에 설치하기 위해서는 단순히 복사 명령을 수행하여 실행 이미지를 모트에 저장하면 되고, 파일 시스템에서 현재 디렉토리를 변경하는 것으로 현재 명령을 수행중인 모트를 변경할 수 있다. LiteOS의 커널은 현재 최대 8개의 스레드를 지원하고 온라인 디버깅과 동적 메모리 할당, 그리고 스레드의 현재 상태를 snapshot으로 저장하고 복구하는 기능을 제공한다. 2007년 10월에 MicaZ 모트 기반의 버전 0.2가 발표 되었다.

3.7 Nano-RK

Nano-RK는 Carnegie Mellon University에서 개발된 센서 네트워크를 위한 예약 기능 기반의 실시간 운영체제이며 FireFly 센서 네트워크 플랫폼과 MicaZ 모트에서 동작한다[23]. Nano-RK는 다양한 기능과 실시간성을 지원하는 경량의 리소스 매니지먼트 커널을 포함하며 태스크의 데드라인을 맞추기 위해 CPU, 네트워크, sensor, actuator등의 자원 예약기능을 통한 고정 우선순위 기반의 선점형 멀티 스레딩을 제공하고 있다. 각 스레드들은 자신이 필요한 자원을 예약할 수 있으며 커널은 각 스레드에게 CPU 및 네트워크 패킷 등의 자원 제공을 정확한 시간에 보장한다. 또한 가상 에너지 예약(virtual energy reservation)이라는 개념을 통하여 시스템 및 스레드 레벨의 에너지 관리 기법도 제공한다. 2007년 10월에 pre-beta버전의 모든 소스가 공개 되었으며 정식 베타버전이 곧 공개될 예정이다.

3.8 RETOS

연세대학교 모바일 임베디드시스템 연구실에서 개발된 RETOS[7]는 멀티스레딩(Multithreading), 시스템의 신뢰성(Resiliency) 그리고 확장성(Extensibility)의 세 가지 특징에 초점이 맞추어져 개발된 센서 운영체제이다. RETOS는 기존의 멀티스레딩 센서 운영체제의 단점인 컨텍스트 스위칭으로 인한 오버헤드를 줄이기 위해 전통적인 방식의 고정 주기 시분할 방식을

사용하는 대신 가변 주기 타이머를 소프트웨어적으로 동작시켜 불필요한 인터럽트는 줄이는 반면 우선순위가 높은 태스크를 신속하게 처리하는 특징을 가지고 있다.

대부분의 기존 센서 운영체제들은 MMU가 없는 디바이스로 구성된 센서 네트워크 시스템에서의 안정성을 고려하지 않고 있다. 이러한 요소는 사용자의 응용 프로그램 작성 오류를 통해 전체의 시스템의 안정성을 해칠 수 있는 문제의 원인이 될 수 있다. RETOS는 이러한 잠재적 요소를 운영체제의 신뢰성을 무너뜨리는 요소로 인식하고 두 가지 기법을 통하여 운영체제로서의 안정성을 유지하고 있다. 하나는 듀얼 모드 운용 기법으로서 커널과 응용의 수행 공간을 완벽하게 분리하고 상호 간의 권한을 검사하여 접근을 허용하는 방법이다. 다른 하나는 정적/동적 코드검사를 통해 응용이 하드웨어 영역을 직접적으로 접근하는 것을 막는 방법이다.

센서 하드웨어의 리소스의 제약으로 인하여 다양한 응용이 필요로 하는 운영체제의 모든 기능을 한번에 설치하는 것은 불가능하다. RETOS는 이러한 점을 고려하여 커널의 기능을 확장 또는 축소할 수 있는 모듈 탑재 기능을 제공한다. 모듈 기능을 통해 실시간 업데이트, 실시간 모니터링 툴, 여러 라우팅 프로토콜 등 추가적으로 필요한 기능을 동적으로 설치하여 이용할 수 있다. 커널과 응용의 공간이 분리된 RETOS 커널은 응용 프로그램과 모듈 프로그램의 주소 재배열 기법을 통해 여러 개의 모듈과 응용을 동적으로 로딩하는 기능을 제공한다. 응용 개발자는 여러 모듈의 재사용과 멀티 응용의 로딩을 통해 복잡한 센서 응용을 쉽게 구현할 수 있다.

센서 운영체제가 가져야 할 특징을 잘 정의하여 구현된 RETOS는 범용 운영체제에 가장 가까운 모습을 가지고 있으며 실제 센서 네트워크 응용 개발을 통해 운영체제로서의 역할과 성능을 검증 받고 있다. 현재 RETOS는 1.2버전이 배포되고 있으며 MSP430과 ATmega128 마이크로 컨트롤러가 탑재된 하드웨어에서 동작이 가능하며, 새로운 모듈 기능과 하드웨어 그리고 개발 프로그램 도구의 지속적인 지원을 하고 있으며 계속 업데이트가 되고 있다.

3.9 Nano Q plus

한국 전자 통신 연구원(ETRI)에서 개발한 NanoQplus [12]는 멀티 스레드 모델을 기반으로 설계되었으며 프로세스간의 선점이 가능한 스케줄링을 지원한다. 기본 개발 언어로 C 언어를 사용할 수 있는 nanoQplus는 GUI를 지원하는 사용자 개발 툴 NanoEsto 2.0를 함께 제공하고 있다. 이 툴은 사용자가 개발 단계에서 필요로 하는 커널의 설정과 응용의 개발은 물론 JTAG 디버거의 사용을 지원하여 디버깅의 어려움을 줄이고 있다. 또한 EEPROM 관리 도구를 제공하여 대규모 센서 네트워크 응용 개발 시 각 센서 노드의 정보 관리를 EEPROM을 통해 쉽게 할 수 있는 도구를 갖추고 있으며 MySQL 을 통한 대규모 센서 데이터 관리와 실시간 모니터링이 가능하다. 또한 커널의 이미지를 실시간 업그레이드가 가능하도록 지원하여 센서의 대규모 배포 시 겪는 커널과 응용의 업그레이드 문제 해결 방안을 가지고 있다.

4. 평가 분석

다음은 기존 센서 운영체제들이 설계 관점에서 어떤 특징을 가지고 있는지 분석한다. 운영체제의 설계 관점을 크게 시스템 특징 및 기능 제공, 안정성 그리고 사용성을 기준으로 나누어 비교한다.

4.1 시스템 특징 및 기능성

센서 운영체제를 분류하는 대표적인 방법은 응용 프로그램 작성 모델이다. TinyOS의 등장과 함께 센서 네트워크의 자원 제약적인 특징이 부각되며 이벤트 모델을 기본으로 하는 운영체제들이 개발되었다. TinyOS와 SOS 등이 그 대표적인 운영체제로, 이벤트 큐 및 이벤트 스케줄러를 통해 태스크 간 통신 및 스케줄링을 수행한다. 이벤트의 중요도에 따라 우선 순위를 정하여 우선순위 기반의 스케줄링을 수행하기도 한다. 한편, 이벤트 기반의 운영체제가 갖는 "긴 태스크의 자원 선점 문제" 등의 단점을 보완하기 위해서, 센서 운영체제는 스레드를 지원하기 시작하였다. 이벤트 기반의 운영체제는 이벤트 핸들러의 작성요령에 따라 시스템 전체의 성능에 영향을 줄 수 있으며, I/O 등의 시스템 요청 작업 수행 시 마다 이벤트 핸들러를 나누어 작성하여 작업의 일관성이 떨어진다는 단점이 있다[8]. 반면, 멀티 스레드 기반의 운영체제는 하나의 태스크가 CPU 등의 제한된 자원을 독점하는 현상을 방지할 수 있으며 태스크를 일관성 있고 직관적으로 작성할 수 있어 개발의 편의성이 높다고 평가된다.

시스템 콤포넌트 및 응용프로그램의 모듈화, 추상화는 시스템의 확장성을 높여 추가적인 시스템 기능 개발 또는 응용 프로그램의 개발을 용이하게 한다. TinyOS는 시스템 기능, 하드웨어 드라이버, 네트워크 레이어 등의 시스템 구성요소들이 콤포넌트 단위로 분류되고 개발되어 소스 레벨의 모듈화를 제공한다. SOS

와 Contiki는 커널과 모듈의 영역을 나누어 모듈(응용)의 개발이 커널 개발과 독립적으로 이뤄지게 하며, 동적으로 코드가 설치되고 재설치가 가능하기 때문에 전체 센서 네트워크의 기능 변화가 자유롭다. RETOS는 센서 시스템을 커널, 커널 모듈, 응용으로 세분화시키고 하나의 센서 시스템에 여러 커널 모듈 및 응용을 자유롭게 설치 할 수 있도록 하였다. 멀티 응용의 사용을 지원하기 위해 파일 시스템을 제공하며, 확장성 있는 네트워크 사용을 위한 포트 시스템을 제공한다. 이 같은 특징은 RETOS에 범용 운영체제의 특징을 부여하며, 센서 네트워크에 필요에 따라 자유롭게 센서 응용을 설치하여 사용하도록 하여 센서 네트워크 전체를 일종의 인프라 스트럭쳐로 발전된 개념을 부여하였다.

기존 센서 운영체제 들은 센서 네트워크의 자원 제약적 특징을 들며, 운영체제의 크기가 작음을 장점으로 내세웠다. 운영체제의 크기는 작을수록 네트워크를 통한 커널 업데이트 등의 동적 코드 전파 시 유리하며, 남는 시스템 자원을 다양한 용도로 활용할 수 있다. 한편, 센서 하드웨어가 발전함에 따라, 노드에 장착되는 CPU 의 성능, 메모리의 크기 등이 크게 개선될 것으로 예측하고 있어, 시스템의 크기는 에너지 관리 등의 이슈보다 덜 중요하다고 분석하기도 한다. 최근에 제작된 운영체제일수록 시스템 크기를 감수하고, 시스템 보호기법, 보안 기법, 효율적인 자원 관리 기법 등, 시스템의 안정성을 높이고 사용자의 편의성을 높이는 복잡한 기능들을 제공하고 있다.

센서 네트워크에서 스토리지 자원 보다 중요한 자원은 에너지이다. 센서 운영체제는 시스템 운용에 드는 에너지를 최소화해야 할 뿐 아니라 다양한 작업을 위해 부가적으로 사용되는 에너지가 효율적으로 관리되도록 전력 관리 기법을 제공한다. 초기에는 마이크로 콜트롤러, 라디오 통신 칩 및 부가 센서 장비를 켜고 끄는 기본적인 함수를 제공했던 반면, 최근에는 작업을 수행을 감시하고 에너지 사용 패턴을 분석하여 개발자에게 보고하고, 이 정보에 근거하여 작업 수행에 반영하는 등의 좀 더 적극적인 전력 관리 기법을 제공한다.

센서 네트워크 응용을 개발함에 있어 실시간(real-time)성에 대한 요구가 늘어나고 있다. 실시간성은 각 태스크의 작업이 정해진 시간 내에 완료할 수 있음을 보장하는 것이다. 센서 네트워크 운영체제는 실시간성 지원에 제약이 많아 저 수준의 실시간(soft real-time) 지원이 먼저 이뤄졌다. 스레드 스케쥴러, 세머포어 등의 스레딩 기술로 높은 우선순위의 태스크가 다른 태스크 수행 중간에도 수행 권한을 획득하여 작업을 완료할 수 있도록 한다. 센서 환경의 자원 제약적인 특징에 따라 일부 태스크에만 실시간성 기능을 추가하기도 한다. MANTIS, LiteOS, RETOS 등 스레드를 지원하는 운영체제들이 이 같은 방법으로 저 수준의 실시간성을 지원하고 있다. 이벤트 지원 운영체제의 경우에도 우선순위 기반 이벤트 스케쥴러를 지원하여 중요한 작업을 먼저 완료하도록 하고 있으나, 여기서는 저 수준 실시간성 지원으로 분류하지 않았다. TinyOS와 같은 이벤트 기반 운영체제에서도 실시간성 지원을 하기 위해 여러 연구진에서 추가로 연구[24,25]가 진행 되었으며, t-kernel와 Nano-RK는 각 태스크의 수행 시간을 분석하여 스케쥴링에 적용할 수 있게 하여 실시간성을 지원하고 있다.

4.2 안정성

시스템 다운에 대해 전체 시스템을 재 실행하는 소극적 안정성 확보를 넘어, 영속적으로 운용 가능한 시스템을 향한 시스템 보호 기법이 센서 운영체제에 적용되었다. 일반적으로 MMU를 탑재하지 않는 센서 하드웨어에서 메모리 보호 기법은 중요한 시스템 기능으로 연구되었다. TinyOS, SOS, RETOS는 SFI(Software-based Fault Isolation) 기법[26]을 기반으로 응용 프로그램의 잘못된 메모리 접근을 방지하는 메모리 보호 기법을 센서 네트워크에 도입하였다[27-29]. SFI 기반 기법은 응용 프로그램의 메모리 쓰기 접근 마다 잘못된 영역인지 판단하는 코드를 삽입하여 동적으로 메모리 접근을 통제하는 방식을 따른다. 따라서 메모리 쓰기가 많은 응용은 추가 되는 코드로 인한 오버헤드가 많은 단점이 있다. t-kernel은 하드웨어의 지원이 없이 소프트웨어 기법으로 가상 메모리 시스템을 구현하여 응용에게 64kB의 메모리 공간을 제공한다. 따라서 응용의 메모리 공간은 커널과 완전히 독립적으로 유지되고 응용의 메모리 접근은 커널에 의해 모두 제어될 수 있어 안정성이 매우 높다는 장점이 있다. 하지만 페이지 교환과 느린 플래시 메모리의 속도, 바이너리 Re-writing에 의한 많은 양의 추가적인 코드 등 오버헤드가 많은 편이다.

RETOS 역시 커널과 응용 영역을 분리하여 관리하고 있으며 SFI 기법을 적용하여 응용의 잘못 된 수행으로부터 커널을 안전하게 보호한다. 다른 운영체제의 시스템 보호 기법들이 시스템 전체를 감시하는 것과는 달리 RETOS는 전체 시스템을 커널, 커널 모듈, 응용 프로그램으로 나눈다. 커널은 신뢰할 수 있다고 가정하고, 추가 개발되는 커널 모듈과 응용 프로그램을

감시 대상으로 하여 시스템 보호 기법의 오버헤드를 줄였다. 또한, 응용 개발이 안정화 되어 작성된 코드가 안전하다고 판단되면 시스템 보호 코드를 제거할 수 있도록 한다.

메모리 보호 기법과 함께 스택 범람 등을 방지하는 스택 보호 기법 또한 센서 운영체제의 안정성을 더해 준다. Contiki는 개발자로 하여금 스레드가 사용하는 스택의 크기를 결정하도록 하고 있다. 개발자의 잘못된 스택 크기 결정은 시스템의 중대한 결함으로 이어 질 수 있기 때문에 시스템 차원의 스택 분석 기능이 필요하다. RETOS는 응용 프로그램의 응용 빌드 시 함수 콜 그래프를 분석하여 스택 사용량을 계산하여 할당한다. 재귀 호출을 제외하고 응용 프로그램의 스택 분석은 멀티 스레드 지원으로 인한 스택 낭비를 줄이고 스택 범람으로 인한 시스템 결함을 방지한다. 또한, 스택 내부의 반환 주소(return address) 등의 중요 시스템 정보를 함수의 잘못된 메모리 접근으로 보호하는 기능을 제공한다.

Nano-RK의 경우 다양한 상황에서 Fault Handling 기법을 제공하고 있다. 실시간 커널의 특성상 태스크가 타이밍 조건을 위반한 경우 이를 적절히 처리할 수 있는 기법을 제공하고 있으며, 태스크의 스택 무결성을 보장하고 어떤 태스크가 자원을 과다 사용하는지 추적하고 관리한다. 또한 하드웨어 적으로 Watchdog Timer를 사용하고 갑작스런 센서 노드의 재시작 혹은 배터리의 전압이 낮은 경우 등 다양한 상황에서의 Fault Handling 기법들이 제공된다.

4.3 사용 편의성

센서 운영체제는 TinyOS의 nesC를 제외하면 주로 C 언어를 이용하여 응용 프로그램을 작성한다. 개발 편의를 위해 다양한 개발 툴킷을 제공한다. TinyOS는 개발의 편의성을 제공하기 위해 GUI로 구성된 개발 툴을 제공하며, 시뮬레이션 툴 등 다양하고 도구를 제공한다. NanoQplus 또한 센서 응용의 개발 효율성을 높이기 위해 NanoEsto라는 통합 개발 환경을 제공한다. 커널, 응용 그리고 응용이 사용하는 데이터를 관리하며, 개발과 설치 기능을 제공한다. 개발자의 편의를 도모하기 위해 LiteOS는 UNIX/Linux 사용자에게 친숙한 인터페이스를 센서 노드 개발 환경에

표 1 센서 운영체제 비교

		TinyOS	SOS	MANTIS	Contiki	t-kernel	LiteOS	NanoRK	RETOS	Nano-Qplus
시스템 특징 및 기능성	Model	E	E	T	T-like Pro.T*	△ P.S*	T	T	T	T
	Module	O Static	O Dyn.	O	O	X	O	X	O Dyn.	O Static
	Kernel Size	소	중	대	대	중	대	소	중	중
	P.M.	O	O	O	X	X	X	O	O	O
	Realtime	-	-	X	△	O	△	O	△	△
	D.Mem allc	X	O	O	X	O	O	X	O	O
안정성	Protection	O	O	X	X	O	X	O	O	O
	Stack	X	O	X	X	O	X	O	O	O
	Security	△	△	X	X	X	X	X	X	X
사용성	Hardware	다양	다양	AVR	다양 MSP AVR	AVR	AVR	AVR	보통 MSP AVR	보통 MSP AVR
	Language	nesC	C	C	C	Any	C, C++	C	C	C
	Platform	Linux	Linux	Linux window	Linux	Linux	Linux	Linux	Linux	Windows
	Community	큼	보통	보통	보통	적음	적음	적음	적음	적음
	편의성	Tools Simul.	-	Monitor shell	-	-	shell	-	-	IDE
	Flexibility	낮음	좋음	낮음	좋음	낮음	좋음	낮음	좋음	낮음

Dyn.(module): Dynamic
T: Threading model
P.S*: Preemptive Scheduling
E: Event-driven model
Pro.T*: Proto-threads
△ (realtime) : soft realtime support

도입하여 개발자의 편의성을 높였다. LiteOS의 쉘을 이용하면 센서노드에 프로그램을 다운로드 하는 것은 단순한 복사 명령으로 이루어지고 다른 센서 노드에 접근하는 것은 현재 디렉토리를 변경하는 것만으로 가능하다. 위 운영체제가 각자의 언어와 개발환경에 의존하는 것과는 달리 t-kernel은 응용 프로그램의 바이너리를 직접 로딩하고 변경하는 구조로 응용개발에 사용된 프로그래밍 언어에 의존적이 않다는 특징을 갖는다. 개발자는 센서 노드의 MCU에 맞는 컴파일러를 사용할 수 있으며, 원하는 프로그래밍 언어를 자유롭게 선택해 개발할 수 있다.

MCU, 라디오 칩, 센서 등의 분야에서 지원 하드웨어가 다양해짐에 따라 센서 운영체제는 유연한 구조를 갖추어 여러 장비를 지원하고 있고 운영체제의 기능성을 높여, 시스템의 각 기능을 추상화하고 확장 가능한 구조를 제공하도록 노력하고 있다. 시스템의 유연성은 새로운 장비, 센서의 장착, 새로운 알고리즘의 추가에 개발자들이 쉽게 대처할 수 있도록 해준다. TinyOS는 업계와 학계의 가장 큰 사용자 커뮤니티를 가지고 있으며 OS 코어, 네트워크 프로토콜, 시뮬레이션 등 다양한 워킹 그룹을 통해 자유 개발자의 참여를 받고 있다.

5. 결 론

센서 운영체제는 센서 노드의 하드웨어를 추상화하고 시스템 자원을 효과적으로 관리하며 센서 응용 프로그램 개발을 위한 개발 환경을 제공한다. 센서 네트워크의 자원제약적인 특징에 따라 초기에 개발된 센서 운영체제는 안정성, 저전력 기능, 시스템 최적화 등의 기술 지원에 초점을 맞추고 있다. 이후 센서 응용의 사용 범위가 확대됨에 따라, 실시간 기법, 시스템 유연성, 보안 등의 요구 사항이 증대되었고, 센서 운영체제는 이 같이 센서 응용이 직접 다루기 어려운 기술을 지원하게 되었다. 이와 함께 다양한 알고리즘과 다양한 하드웨어를 지원하기 위해 유연성 있는 구조를 채택하고, 개발의 편의성을 도모하기 위한 다양한 툴을 지원하고 있다.

아직 센서 운영제제는 표준화 된 기능 정의가 이뤄지지 못하였고, 응용 개발자의 다양한 요구 사항에 맞추며 발전하고 있는 단계이다. 센서 네트워크의 영역 확장과 기술 발전으로 센서 운영체제는 지속적으로 변화해 나갈 것이다. 기존의 인터넷 망, 블루투스, WiFi 등의 이 기종 망과의 연결이 이뤄지고, 이를 위해 센서 운영체제 기술의 표준화가 필요하게 될 것이다. 센서 네트워크의 저변 확대와 기술 발전을 위해 기술 변화와 요구 사항의 변화에 신속하게 적응하여 지원하는 것이 중요하다.

참고문헌

[1] R. Kay, F. Mattern, "The Design Space of Wireless Sensor Networks," IEEE Wireless Communications, Vol. 11, Issue: 6, December, 2004, pp. 54-61.

[2] Crossbow, http://www.xbow.com/

[3] MoteIV, http://www.sentilla.com/moteiv-endoflife.html

[4] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, Kristofer Pister, "System architecture directions for network sensors," In Proc. of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS), Cambridge, MA, November 2000.

[5] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, R. Han, "MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms," ACM/Kluwer Mobile Networks & Applications, Special Issue on Wireless Sensor Networks, vol. 10, no. 4, August 2005.

[6] A. Dunkels, B. Grönvall, T. Voigt, "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors," In Proc. of the First IEEE Workshop on Embedded Networked Sensors (EmNets), Tampa, Florida, November 2004.

[7] H. Cha, S. Choi, I. Jung, H. Kim, H. Shin, J. Yoo, C. Yoon, "RETOS: Resilient, Expandable, and Threaded Operating System for Wireless Sensor Networks," In Proc. of the Information Processing in Sensor Networks, Massachusetts, USA, April 2007.

[8] R. Behren, J. Condit, E. Brewer, "Why events are a bad idea(for high-concurrency servers)," In Proc. of the 9th Workshop on Hot Topics in Operating Systems(HotOS), Lihue, Hawaii, 2003.

[9] H. Kim, H. Cha, "Multithreading Optimization Techniques for Sensor Network Operating Systems," In Proc. of the 4th European conference on Wireless Sensor Networks(EWSN), Delft, Netherlands, January 2007.

[10] D. Gay, P. Levis, R. Behren, M. Welsh, E. Brewer, D. Culler, "The nesC Language: A Holistic Approach to Network Embedded Systems," In Proc. of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation(PLDI), San Diego, CA,

June 2003.

[11] P. Levis, D. Culler, "Maté: A Tiny Virtual Machine for Sensor Networks." In Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X), 2002.

[12] nano Q plus http://qplus.or.kr/

[13] P. Levis, N. Patel, D. Culler, S. Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks." In Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004).

[14] C. Han, R. Rengaswamy, R. Shea, E. Kohler, M. Srivastava, "SOS: A dynamic operating system for sensor networks," In Proc. of the Third International Conference on Mobile Systems, Applications, And Services(Mobisys), Seattle, WA, June 2005.

[15] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, I. Stoica, "A Unifying Link Abstraction for Wireless Sensor Networks," In Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems(SenSys), November 2-4, 2005.

[16] C. Ee, R. Fonseca, S. Kim, D. Moon, A. Tavakoli, D. Culler, S. Shenker, I. Stoica, "A Modular Network Layer for Sensornets," In the Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation(OSDI '06), Seattle, WA, November 2006, USENIX Association, pp. 249-262.

[17] S. Choi, H. Cha, "Application-Centric Networking Framework for Wireless Sensor Nodes," 3rd Annual International Conference on Mobile and Ubiquitous Systems, August 2006.

[18] A. Dunkels, F. Österlind, Z. He, "An adaptive communication architecture for wireless sensor networks." In Proceedings of the Fifth ACM Conference on Networked Embedded Sensor Systems(SenSys 2007), Sydney, Australia, November 2007.

[19] 6lowpan, http://www.ietf.org/html.charters/6lowpan-charter.html

[20] A. Dunkels, O. Schmidt, T. Voigt, "Protothreads: Simplifying Event-Driven Programming of Memory-Constrained Embedded Systems," In Proc. of the 4th ACM Conference on Embedded Networked Sensor Systems(Sensys), Boulder, Colorado, November 2006.

[21] L. Gu, J. Stankovic, "t-kernel: Provide Reliable OS Support for Wireless Sensor Networks," In Proc. of the 4th ACM Conference on Embedded Networked Sensor Systems(Sensys), Boulder, Colorado, 2006.

[22] Q. Cao, T. Abdelzaher. Demo Abstract: LiteOS, A Lightweight Operating System for C++ Software Development in Sensor Networks, In Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems(ACM Sensys), 2006.

[23] A. Eswaran, A. Rowe, R. Rajkumar, "Nano-RK: an energy-aware resource-centric RTOS for sensor networks," In Proc. of the 26th IEEE International Real-Time Systems Symposium(RTSS), 10 pp, December 2005.

[24] C. Duffy, U. Roedig, J. Herbert, C. J. Sreenan, "Adding Preemption to TinyOS," In Proc. of the Fourth Workshop on Embedded Networked Sensors (EmNets), Cork, Ireland, June 2007.

[25] C. Duffy, J. Herbert, "Achieving Real-Time Operation in TinyOS," In Proc. of Performance control in wireless sensor networks at the 2006 IFIP Networking, Coimbra, Portugal, May 2006.

[26] R. Wahbe, S. Lucco, T. E. Anderson, S. L. Graham, "Software-based fault isolation," In Proc. of the 14th ACM Symposium on Operating System Principles(SOSP), Asheville, NC, USA, December 1993.

[27] J. Regehr, N. Cooprider, W. Archer, E. Eide, "Memory Safety and Untrusted Extensions for TinyOS," In the Technical Report UUCS-06-007, University of Utah, June 2006.

[28] protection on SOS) R. Kumar, E. Kohler, M. Srivastava, "Software-Based Memory Protection In Sensor Nodes," In Proc. of the Third Workshop on Embedded Networked Sensors(EmNets), Cambridge, MA, 2006.

[29] H. Kim, H. Cha, "Towards a Resilient Operating System for Wireless Sensor Networks", In Proc. of the 2006 USENIX Annual Technical Conference, Massachusetts, USA, June 2006.

[30] Moteworks 2.0, Crossbow Inc., http://www.xbow.com

[31] Programmer's Notepad, http://www.pnotepad.org

[32] In-geol Chun, Chae-deok Lim, "NanoEsto Debugger: The Tiny Embedded System Debugger," The 8th International Conference on Advanced Communication Technology(ICACT), Feb 2006.

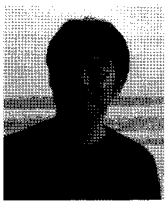
[33] Eclipse, http://www.eclipse.org

[34] I. Jung, H. Cha, "RMTool: Component-Based Network

Management System for Wireless Sensor Networks," 2007 IEEE Consumer Communications and Networking Conference(CCNC), Las Vegas, Jan 2007

[35] Turon, M., "MOTE-VIEW: A Sensor Network Monitoring and Management Tool," The Second IEEE Workshop on Embedded Networked Sensors, 2005 (EmNetS-II). vol., no., pp. 11-18, 30-31 May 2005

[36] Jing Yang, Mary Lou Soffa, Leo Selavo, Kamin Whitehouse, "Clairvoyant: A Comprehensive Source-Level Debugger for Wireless Sensor Networks." The 5th ACM Conference on Embedded Networked Sensor Systems(Sensys), Nov 2007



최 석 원

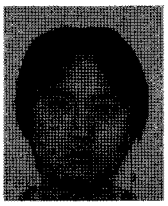
1999 광운대학교 컴퓨터과학과 학사
2005 연세대학교 컴퓨터과학 석사
2005~현재 연세대학교 컴퓨터과학 박사과정
관심분야 : 무선 센서 네트워크, 저전력 운영체제
E-mail : sukwon@cs.yonsei.ac.kr

신 효 정

2005 연세대학교 컴퓨터과학 학사
2007 연세대학교 컴퓨터과학 석사
2007~현재 연세대학교 컴퓨터과학과 박사과정
관심분야 : 무선 센서 네트워크, 운영체제
E-mail : hjshin@cs.yonsei.ac.kr

윤 찬 민

2006 연세대학교 컴퓨터과학 학사
2006~현재 연세대학교 컴퓨터과학 석사과정
관심분야 : 무선 센서 네트워크, 실시간 운영체제
E-mail : cmyoon@cs.yonsei.ac.kr



최 학 수

2007 연세대학교 컴퓨터과학 학사
2007~현재 연세대학교 컴퓨터과학 석사과정
관심분야 : 무선 센서 네트워크, 디바이스 드라이버 관리 기법
E-mail : haksoo@cs.yonsei.ac.kr

차 호 정

1985 서울대학교 컴퓨터공학 학사
1987 서울대학교 컴퓨터공학 석사
1991 영국 맨체스터대학 컴퓨터과학 박사
현재 연세대학교 컴퓨터과학과 교수
관심분야 : 운영체제, 임베디드시스템, 무선 센서 네트워크
E-mail : hjcha@cs.yonsei.ac.kr