센서 네트워크용 초소형 OS

송준근 | 마평수 | 박승민
한국전자통신연구원

요 약

최근 몇 년간 유비쿼터스 서비스를 구현하기 위한 핵심 기술 중 하나인 무선 센서 네트워크에 대한 관심이 높아지고 있다. 무선 센서 네트워크 기술은 물류, 유통, 환경 감시, 홈 오토메이션, 군사 분야 등 다양한 분야에 적용 될 수 있기 때문에 앞으로 관련 분야의 시장 또한 커질 것으로 예상되고 있다 [1].

무선 센서 네트워크는 기존 유선 센서 네트워크나 무선 네트워크 환경과는 많은 차이를 가진다. 우선 극도로 제한된 시스템 자원만을 가질 수 있으며, 열악한 환경 속에서 무선 매체를 통해 유기적으로 동작하여야 하는 특징을 가지고 있다. 적게는 수십 개에서 많게는 수백, 수천 개의 자율적인 하드웨어 노드들로 구성되는 무선 센서 네트워크에서 제한된 자원을 효과적으로 활용하기 위해서는 센서 노드에 적합한 운영체제가 필수적으로 요구된다.

지난 몇 년간 센서 노드 하드웨어의 발달과 더불어 많은 센서 네트워크용 초소형 운영체제가 개발되어왔다. 현재 많이 알려져 있는 센서 네트워크용 OS로는, 가장 활발한 참여를 보이고 있는 TinyOS[3]부터 SOS[4], MANTIS[5], Contiki[6], T-kernel[7] 등이 있으며, 국내 기술로 개발된 Nano-Qplus[8] 등이 존재한다.

본고에서는 우선 센서 네트워크에 대한 배경 지식과 플랫폼 등에 대한 내용을 간단히 다루고, 본론에서 센서 네트워크용 운영체제가 가져야 할 조건과 현재 개발되어 있는 센서 네트워크 OS들의 특징에 대해 간략히 살펴보도록 하겠다. 또한 센서 네트워크 OS와 밀접한 연관성을 가지는 분야에 대해 간단히 살펴보고, 마지막으로 앞으로의 방향에 대해 알아본다.

1. 센서 네트워크 개요

1. 무선 센서 네트워크 개요

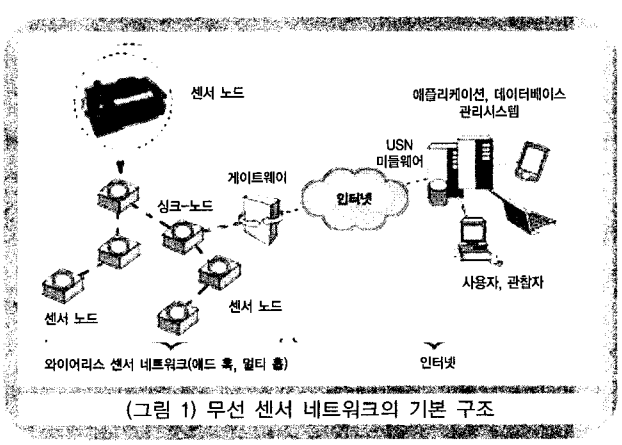
무선 센서 네트워크는 센서로 센싱 가능한 정보를 수집하고, 수집된 정보를 가공, 이를 무선 송수신 장치를 이용해 외부로 전달하는 일련의 시스템을 지칭한다. 더 넓은 의미로는 이와 관련된 모든 기술을 지칭하는데, 센서 노드 하드웨어에 들어가는 운영체제와 미들웨어, 싱크 노드를 통해 전달된 정보를 모니터링 하는 시스템 등도 그 범주에 든다고 할 수 있다.

무선 센서 네트워크는 우리 주변에 쉽게 적용될 수 있는데, 일상생활에서 정보를 수집, 관리 제어하는 간단한 응용부터 물리 공간에서 수집된 광범위한 데이터를 가공하여 작게는 홈 네트워크부터 크게는 U-City까지 사용될 수 있다. 앞으로 센서 네트워크를 위한 관련 기술들은 그 중요성이 증가할 것으로 예상되는데, 일본 총무성 센서 네트워크 기술 관련 조사 연구회에서 2006년 7월 발표된 무선 센서 네트워크의 분야별 일본 시장 동향을 보면 방범, 보안 등 침입 및 도난 방지 응용을 비롯해 교통, 지진 모니터링 등 재해대책이나 환경보전 등의 응용에 사용될 것으로 전망하고 있으며 그 시장 규모도 2010년에는 총 1.2조 엔으로 확대될 것으로 전망하고 있다 [1]. 국내에서도 USN 산업은 2010년까지 기기

분야에서 10조 7,509억 원,서비스 분야에서 4조 7,402억 원을 유발하여 총 15조 4,912억 원의 총생산 유발 효과를 가질 전망이다.

센서 네트워크 분야의 요소 기술로는 크게 사물과 환경 인지에 사용되는 센서 관련 기술과 정보 전달을 위한 네트워크 기술, 전달된 정보를 서비스하기 위한 응용 기술 등으로 구분할 수 있다 [2]. 네트워크상의 정보를 응용, 서비스하기 위한 응용 기술은 크게 미들웨어와 운영체제 두 가지로 구분 할 수 있는데, 현재 센서 네트워크 미들웨어 기술은 아직 초기 단계이며 주로 대학 연구소를 중심으로 개발되어지고 있으며 센서 네트워크 운영체제 또한 여러 연구소에서 활발히 연구되어 지고 있다.

2. 센서노드의 기본 구조



(그림 1) 무선 센서 네트워크의 기본 구조

(그림 1)은 무선 센서 네트워크의 기본 구조를 보여준다. 대부분의 무선 센서 네트워크는 센싱한 정보를 센싱하는 센서 노드, 데이터의 전달을 담당하는 라우팅 노드, 그리고 게이트웨이로의 전달을 담당하는 싱크 노드로 구분할 수 있다. 이런 구분은 논리적인 기능에 따라 나눠지기는 하지만 보통 센싱하는 센싱 기능과 라우팅을 담당하는 라우팅 기능 등이 하나의 노드에서 동작하는 경우가 많다.

수 많은 센서들이 필드(Filed)의 지리적, 환경적 변화를 감지하면 센싱된 데이터를 싱크노드를 통해 게이트웨이로 전달하고 전달된 데이터는 인터넷이나 이종 간의 네트워크로 전달한다. 최종적으로 전달된 데이터는 미들웨어를 통해 응용 프로그램이나 관리 프로그램에 모니터링 등을 위해 사용되게 된다. 최근 이런 인터넷 환경이나 다른 이종 간의 네트워크와의 연동을 위한 기술들이 개발되고 있으며, 6LowPAN같은 IPv6를 센서 네트워크에 도입하기 위한 움직임도 진행되고 있다 [9].

3. 센서노드 플랫폼

센서 노드를 위한 하드웨어 플랫폼은 여러 특징을 가진다. 우선 한번 배치되면 유지 보수가 어렵기 때문에 강인한 구조를 지녀야 하며, 외부에서 무한정의 에너지를 공급 받는 것이 아니기 때문에 저전력으로 설계되어야 한다. 또한 다양한 응용에 맞게 효과적으로 사용될 수 있도록 유연성과 모듈성을 지녀야 하며, 대규모 응용을 위해 가격이 저렴해야 한다 [10].

지난 몇 년간 무선 센서 네트워크 플랫폼에 많은 발전이 있었는데 우선 프로세서를 보면 초기 8Bit 저전력 프로세서에서부터 다양한 응용과 더 나은 컴퓨팅 환경 제공을 위한 32bit 프로세서까지 다양한 Micro Controller Unit (MCU)이 사용되고 있으며, 무선 네트워크 모듈의 경우도 Zigbee, UWB, 블루투스, Z-wave 등이 사용되고 있다 [11].

〈표 1〉 센서 노드 플랫폼

Platform	Year	Microcontroller	Radio	Flash	Sensor Interface
Rene2	2000	Atmel Atmega163, 8KB prog, 1KB RAM	RFM TR1000, 916 Mhz ASK, 40 Kbps, GPIO	24KC256, 32KB, EEPROM, 12C	51-pin Expansion; UART, I2C, SPI, GPIO
Mica	2001	Atmel Atmega128, 128KB Prog,	RFM TR1000, 916 MHz, ASK, 40Kbps, UART	AT45DB041B, 512 KB, SPI	Same
Mica2	2002	Atmega128 128KB prog,	Chipcon CC1000, ASK, 40Kbps, UART	AT45DB041B 512 KB, SPI	Same
Mica2	2002	Atmega128	Chipcon CC1000, 2.4GHz, QPSK, 250Kbps, SPI	AT45DB041B	Same
iMOTE	2004	Zeevo ZV4002 with ARM 7, 64 KB SRAM, 512 KB Flash	ZV4002, Bluetooth		
BTnode	2004	Atmega128	CC1000 & ZV4002 Bluetooth	SST39	
TelosB	2004	TIMSP 430, 60 KB program, 10KB RAM	Chipcon CC2420	STM25P	10pin header, Intergrated temperature, light, amd humiduty
EyesIFX	2005	TI MSP 430	Infineon TDA5250	AT45DB041B	
Fleck	2005	Atmega128 433 MHz, GFSK, 76kbps	Nordic nRF903,	512 KB	31-pin
Schock fish	2006	TI NSO430	Xemics XE1205 868 MHz 153 kbps	512 KB Flash	30-pin Motex
Mote2	2006	Intel PXA271, 256 KB SRAM, 32MB DRAM	Chipcon CC2420	Intel Strata 32 MB direct	31+21-pin top; UART, SPI, I2C, SDIO 40+20-pin bottom; GPIO, USB, JTAG, MSL, CIF

최근 하드웨어 동향은 저전력 MCU와 무선 칩셋을 SoC (System on Chip) 형태로 내장하고 있으며, 초소형 Zigbee 칩셋을 이용한 무선 센서노드의 소형화가 더욱 진전 될 것으로 예상된다.

〈표 1〉은 2000년에서 2006년 사이 발표된 TinyOS용 센서 네트워크 플랫폼을 정리한 것이다. Microcontroller의 경우 8bit MCU인 Atmega128부터 16bit인 TI사의 MSP430을 비롯 ARM 7이나 Intel PXA271까지 용도에 따라 다양한 칩이 사용되는 것을 확인 할 수 있다.

센서 네트워크 플랫폼에 사용되는 MCU에 대해 간단히 살펴보면, Atmega128은 무선 센서 네트워크 초창기부터 많이 사용되었던 하드웨어로 RISC 및 하버드 구조를 가지고 있는 고성능 8bit 마이크로 컨트롤러이다. 프로그램용 코드 메모리와 플래쉬 메모리를 내장하고 있어 쉽게 반복적 프로그래밍이 가능하기 때문에 널리 사용되고 있다 [12]. Texas Instruments에서 개발된 MSP430의 경우 빠른 Wakeup이 가능하며 다양한 전력 관리 모드를 제공하는 16bit MCU로 뛰어난 전력관리 때문에 저전력 센서 노드에 적합하기 때문에 최근 많이 쓰이고 있다 [13]. 그밖에 많이 사용되는 하드웨어로는 8051 Extended MCU기반에 IEEE 802.15.4를 지원하는 Chipcon의 CC2430, Radiopulse의 Mango 등의 단일 칩이 센서 네트워크를 위한 하드웨어 플랫폼에 사용되어지고 있다.

현재 센서 네트워크는 이러한 플랫폼 위에 센서 네트워크 운영체제 및 미들웨어, 모니터링 기술 등이 활발히 개발되고 있으며, 이와 더불어 라우팅, MAC 프로토콜 등 네트워크 관련 부분도 많은 연구가 되고 있다. 본 고에서는 이러한 기술중 핵심이 되는 센서 네트워크를 위한 초소형 OS에 대해 자세히 살펴보도록 하겠다.

II. 센서 네트워크 OS의 요구사항

앞서 언급한 것과 같이 센서 노드는 극히 적은 리소스를 가지고 있기 때문에 센서 네트워크 OS 개발에는 많은 제약사항을 가지고 있다.

먼저 커널의 크기가 제한된다. 센서 노드의 경우 메모리 자체가 제한적이며 별도로 커널과 응용, 코드와 데이터의 구분이 되어 있지 않는 경우도 많다. 그렇기 때문에 응용에서의 많은 메모리 확보를 위해 커널 코드 자체의 사이즈뿐만 아니라 커널이 사용하는 메모리크기 또한 제한될 수밖에 없다. 그래서 커널 자체의 기능이 제한적일 수밖에 없으며, 제한된 상황에서 맞는 센서 노드에 맞는 적합한 구조의 커널이 필요하다.

또 고려해야할 사항으로는 상대적으로 컴퓨팅 파워가 낮은 MCU을 사용하기 때문에 최대한 복잡한 연산을 줄이고 코드를 최적화 하여 MCU의 부하를 줄여야한다는 것이다. 또 각각의 센서 노드는 센싱 적업과 센싱한 데이터를 프로세싱하는 작업 그리고 무선 네트워크로 전달하는 작업을 수행하는데 어느 하나의 기능을 수행하기 위해 너무 많은 시간 MCU를 소비하게 되면 다른 기능에 지장을 줄 수 있기 때문에 설계 단계부터 해당 부분을 고려해야 한다.

대부분의 플랫폼들이 Memory Management Unit이 없는 경우가 많기 때문에 메모리 관련 문제도 많은 부분 신경 써야 한다. 하드웨어적으로 메모리 보호 기능뿐만 아니라 가상 메모리 페이징 등을 지원하지 않고, 데이터와 코드가 별도의 보호 장치 없이 같은 메모리에 존재하는 경우도 있다. 그래서 특히 멀티 쓰레드로 동작하는 센서 노드 운영체제의 경우 각별한 주의가 필요하다.

센서 네트워크가 최대한 라이프 타임을 보장하기 위해 전력 보존 문제 또한 매우 중요하다. 센서 네트워크에서 하나의 노드만 작동을 중단해도 라우팅 패스를 다시 찾는 문제 등 전체 네트워크에 많은 영향을 미칠 수 있고 심한 경우 전체 네트워크의 단절을 가져올 수도 있기 때문에 하나의 노드뿐만 아니라 네트워크 전체의 지속 시간과 효율성을 고려한 전력관리 기술이 필요하다.

위와 같은 요구사항뿐만 아니라 노드 간의 동기화, 효과적인 I/O, 멀티 홉 라우팅 등 센서 네트워크 운영체제를 위한

〈표 2〉 센서 네트워크 운영체제 요구 사항

특 성	요 구 사 항
제한된 자원	센서 노드의 저전력 통신 프로세서 메모리의 효율적 관리 커널 코드, 데이터 크기 제한
에너지 효율 극대화	센서 노드들 간의 시각 동기화 절전 모드 등 저전력 고려
하드웨어적 제약사항 극복	제한된 처리 능력 작은 하드웨어 구조 고려 원시적 I/O접근 방식의 설계
통신 거리의 제약 극복	멀티 홉, 메쉬 네트워크 지원

요구사항은 응용에 따라, 네트워크 규모에 따라 훨씬 많은 제약조건이 따를 수 있다. 최근에는 기능적인 측면뿐만 아니라 응용프로그램의 재사용성을 위한 구조적 문제나 네트워크 계층의 추상화를 통해 기존 네트워크와 융합 네트워크를 구성하는 등의 문제까지 대두되고 있다.

III. 센서 네트워크 운영체제

현재 개발되어 있는 센서 노드를 위한 초소형 운영체제들은 크게 두가지 부류로 나눠질 수 있다. 하나는 이벤트 기반 모델로 TinyOS와 SOS, Contiki등이 여기에 속한다고 할 수 있다. 이벤트 기반의 모델은 하나의 프로그램 흐름을 가지며 각각 이벤트에 따라 별도의 핸들러를 가지고 처리 하게 된다. 나머지 다른 부류는 멀티 쓰레드 환경을 제공하는 운영체제로 MANTIS, NanoQplus, Nano-RT, RETOS 등이 여기에 속한다고 할 수 있다. 전통적으로 성능상의 이점과 제어 흐름 상태 관리가 쉽다는 이유로 이벤트 기반의 프로그래밍 모델이 선호되어 왔다. 하지만 멀티 쓰레드 기반의 운영체제는 여러 일을 효과적으로 처리하고 개발의 복잡도를 낮출 수 있어 보다 손쉽게 개발할 수 있다는 장점이 있다. 본 장에서는 이들 운영체제들 각각의 특징에 대해 자세히 살펴보도록 하겠다.

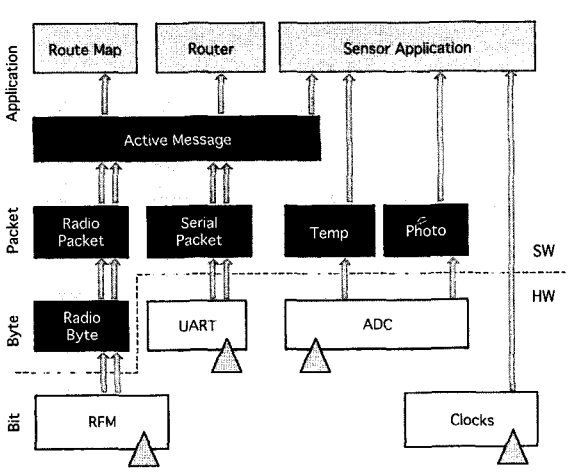
1. Tiny OS

TinyOS는 미국 UC버클리 대학에서 개발된 센서 네트워크를 위한 운영체제로, 현재 전 세계적으로 약 500개 이상의 프로젝트가 진행되고 있으며 가장 큰 센서 네트워크 커뮤니티를 형성하고 있다. 개발자들의 활발한 참여로 빠른 기능 구현 및 업그레이드가 진행되고 있으며 다양한 플랫폼에 구현이 되어 있고 각종 하드웨어 장치 관련 컴포넌트, MAC 프로토콜, 네트워크 프로토콜, 센서 인터페이스 등을 소스 레벨에서 완전 공개하고 있다.

TinyOS는 이벤트 발생 중심의 상태 천이 방식을 채택한 센서네트워크용 운영체제로 동시적인 프로세싱 및 제한된 하드웨어 메모리 공간에서의 효율적인 성능을 지원해주는 운영체제이다. 상태 머신 기반의 구조를 가지며 응용 프로그램은 각 독립적인 컴포넌트를 연결하는 방식으로 이루어진다(그림 2). 이 명령을 처리하는 이벤트 처리기는 그 명령에 따른 상태변화를 일으켜 해당 작업을 처리하게 된다. 간단한 FIFO 스케줄러를 사용하기 때문에 실시간성 등을 고려하기 어렵고 복잡한 응용을 작성하는데 어려움이 있다.

동적 메모리를 할당하지 않지만 nesC(Network Embedded System C)라는 언어를 통하여 컴파일 시간에 컴포넌트가 요구하는 메모리 크기를 정적으로 할당하여 센서 네트워크용 응용 프로그램을 작성할 수 있다. nesC는 안정성을 위해 전체 프로그램에 대한 분석을 통해 최적화를 수행하는 컴포넌트 기반 언어로 TinyOS의 이벤트 기반 동시성 모델을 가능하게 해주지만 프로그래머가 새로 익혀야 하기 때문에 접근하기 어렵다는 단점이 있다.

TinyOS의 큰 강점으로 센싱한 데이터를 효과적으로 관리하기 위한 TinyDB나 센서 네트워크에서의 보안을 위한 TinySec 등의 모듈을 지원함으로써 사용자가 손 쉽게 다양한 기능을 가진 센서 네트워크를 구현할 수 있게 지원하고 있다.



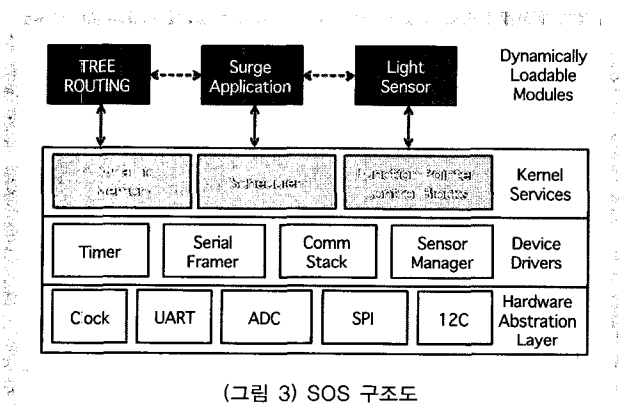
(그림 2) 컴포넌트 기반 TinyOS 응용 예

TinyOS는 작년 11월 기존 1.1버전의 많은 부분을 개선시킨 2.0버전을 발표하였으며 현재 2.0.1 버전까지 발표되어 있다. 2.0.x 버전의 경우 인터페이스와 추상화 과정에 있어 많은 변화가 있어 이전과의 backward complibility는 지원하지 않지만, 코드를 업그레이드 하기가 더 쉬워졌으며 안정성이

향상되었다.

2. SOS

SOS는 UCLA의 Networked and Embedded System Lab에서 만들어진 무선 센서 네트워크용 운영체제이다. 가장 큰 특징으로는 기존 전통 운영체제처럼 사용자 응용 공간과 커널 공간을 분리하여 응용 독립적인 센서 네트워크 운영체제 모델을 채택하였으며, 동적 재구성으로 이미 노드가 초기화 되어 배치된 상황에서 모듈 별 추가, 수정 및 제거를 가능하게 하여 프로그램의 재사용성을 높인 것이다. 또한 사용되지 않는 모듈을 동적으로 메모리에 로드, 언로드 할 수 있기 때문에 자원이 부족한 센서 네트워크 환경에 유리할 수 있다.



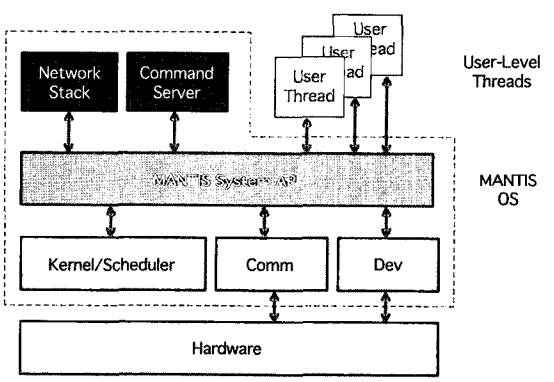
(그림 3) SOS 구조도

(그림 3)은 SOS의 구조를 보여주는 것으로 하드웨어를 추상화 시킨 HAL과 Timer,센서 등의 디바이스 드라이버들 그리고 실제 메모리, 스케줄러 등의 커널 서비스로 이루어져 있으며 그 외에 동적으로 사용자 응용이나 라우팅 등의 모듈 등이 그 위에 동작할 수 있는 구조를 가지고 있다. 2007년 2월에 2.0.0버전이 릴리즈 되었다.

3. Mantis OS

MANTIS OS는 MultimodAl NeTworks of Insitu Sensors의 약자로 콜로라도 대학에서 개발된 최초의 멀티 스레드를 지원하는 센서네트워크용 임베디드 운영체제이다. TinyOS와 달리 초소형 스레드 기반의 멀티 스레드 구조를 채택하여 실제 리눅스 환경과 비슷한 프로그래밍 환경을 제공하기 때문에 센서 네트워크를 처음 접하는 사용자도 쉽게 개발이 가능하다. 하지만 멀티 스레드 환경을 사용하기에 앞서 스택 크기를 사용자가 휴리스틱하게 지정해줘야 한다든지 쓰레드 별로 우선순위를 일일이 지정해줘야 하는 제약사항이 있다.

MANTIS OS는 레이어 기반 운영체제로 멀티 쓰레드 이외에 Preemptive 스케줄링 기법, mutual exclusion을 통한 I/O 동기화, 그리고 하드웨어를 추상화 시키는 디바이스 드라이버 등의 특징을 가지고 있다(그림 4 참조). 또한 어플리케이션 프로그래머가 프로그래밍을 손쉽게 할수록 C언어 방식의 API를 지원하며, 센서네트워크에서의 멀티 홉 통신을 위한 네트워크 스택을 지원한다. 그리고 리모트 쉘 기능을 통하여 운영체제를 컨트롤 하며, 프로그래밍의 재설정 또한 가능하다. MANTIS OS는 커널과 네트워크 스택을 합쳐 500바이트 정도의 메모리만을 차지한다. 2005년 7월 0.9.5 버전 발표 이후 운영체제의 공개적인 업데이트는 하지 않고 있다.



(그림 4) MANTIS OS 구조도

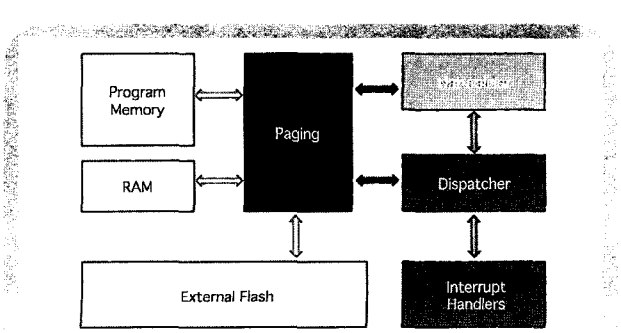
4. Contiki

Contiki 의 가장 큰 특징은 동적 모듈 재할당 기능을 제공한다는 것이다. Contiki는 기본적으로 TinyOS, SOS와 같이 event-driven 모델을 따르고 있지만 Protothread라는 기술을 통해 제한적이기는 하지만 stack reservation없이 멀티쓰래드 같은 코드 간의 동기화를 제공한다 [14]. 하지만 지역 변수를 사용할 수 없고 지정된 구간에서만 blocking이 가능하다는 제약이 있어 Multi-thread 센서 OS라고는 할 수는 없다. 역시 센서 노드용 운영체제를 타켓으로 만들어졌기 때문에 2kb의 RAM과 40km 정도의 ROM을 가지고 구동될 수 있다.

네트워크 스택으로는 uIP와 Rime를 지원하는데 uIP는 TCP/IP스택을 축소하여 임베디드 환경에서 동작할 수 있도록 만든 것으로 인터넷을 통해 통신을 할 수 있도록 해준다 [15]. Rime은 저전력 통신을 위해 구현된 멀티 홉 데이터 전송을 지원하며, 컴파일된 코드가 600byte 미만의 간단한 프로토콜이다 [16]. 또한 Contiki 응용을 테스트해볼 수 있는 Cooja라는 시뮬레이션 툴을 제공한다 [17].

5. t-Kernel

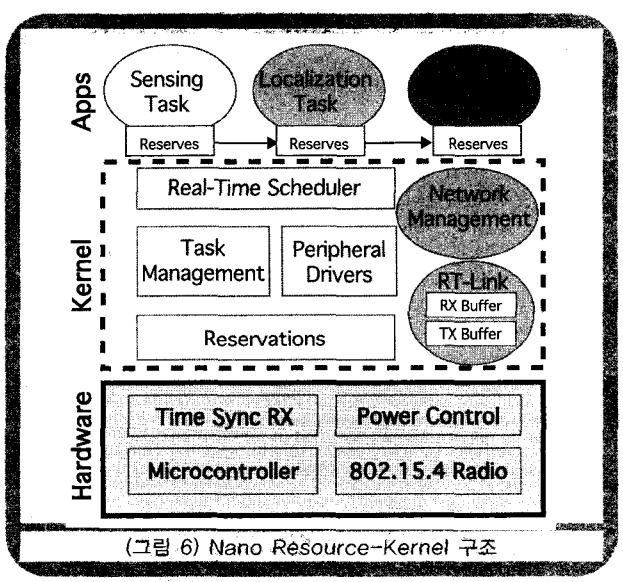
T-Kernel은 2006년 버지니아 공대에서 공개한 센서 네트워크 운영체제로 가장 큰 특징은 소프트웨어 기반의 메모리 보호 기능과 load-time시 코드 수정을 통해 가상 메모리를 지원한다는 것이다 [7]. OS 프로텍션 기능은 운영체제 공간와 응용 프로그램 공간를 구분하고 경계를 검사하여 운영체제 자체의 메모리를 보호하는 기능이다. (그림 5)은 t-kernel을 구성하고 있는 하드웨어, 소프트웨어 컴포넌트를 보여주고 있다. 프로그램 메모리는 커널 공간과 t-kernel에 의해 수정된 코드를 지칭하는 Natin Page를 위한 Naturalized application 공간, 인터럽트 핸들러를 위한 공간으로 구분되어진다. 응용 프로그램은 Naturalizer에 의해 가상 메모리와 메모리 보호 기능을 지원하는 natin이라는 코드 페이지로 전환되며 Paging 컴포넌트를 통해 프로그램 메모리나 외부 메모리에 저장되게 된다. T-Kernel의 또 하나의 특징은 가상 메모리로 최대 64k까지 지원하고 있어 센서 노드의 부족한 RAM을 극복하고자 하였다. 하지만 이런 기능 때문에 코드 사이즈가 증가하고, 런타임시 항상 코드의 주소 변환이나 메모리 스와핑이 요구되기 때문에 런타임 오버헤드가 증가하게 된다.



(그림 5) t-kernel의 구조도

6. Nano-Resource kernel

Nano-RK는 Carnegie Mellon 대학에서 개발된 자원 예약 방식의 실시간 운영체제이다. Nano-RK는 2KB 미만의 RAM과 16KB정도의 ROM의 작은 크기로 양한 기능을 제공하는 Embedded Resource Kernel이다. 고정된 우선순위를 가지고 선점형 멀티테스킹을 지원하며, 가상 에너지 예약을 통해 Task 레벨의 에너지 관리가 가능하다 [18]. 또한 RT-Link [19], WiDom, b-MAC 등 다양한 link layer 프로토콜과 Flooding, DSR같은 라우팅 프로토콜을 제공한다.



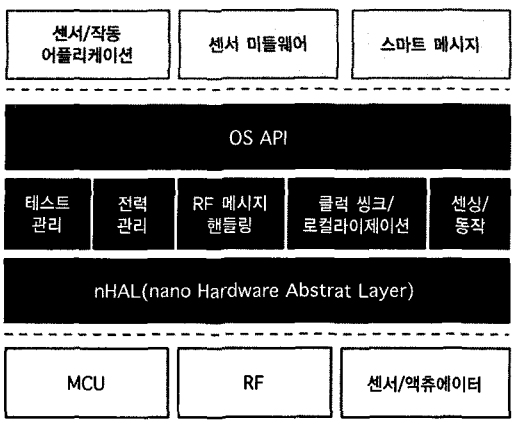
(그림 6) Nano Resource-Kernel 구조

7. NanoQplus

NanoQplus는 한국전자통신연구원에서 개발된 센서네트워크용 초소형 운영체제이다. NanoQplus는 멀티 쓰레드 운영체제로 사용자들이 손쉽게 멀티 쓰레드 응용을 작성할 수 있게 디자인되었다. 또한 완전 선점형 스케줄러를 제공하여 높은 응답성을 제공한다. 낮은 커널 이미지를 제공하기 위해 사용자 응용에 최적화된 커널 Configuration 기능을 통해 적은 메모리 제약을 극복하였다. 또한 EAMR(Energe Aware Multi-path Routing) 프로토콜을 제공하여 센서 네트워크 상황에서 통신을 위한 에너지 소비를 최소화하고 있다.

8. RETOS

RETOS는 Resilient, Expandable and Threaded Operating System의 약자로 연세대학교 모바일 임베디드 시스템 연구



(그림 7) NanoQplus Architecture

실에서 개발된 운영체제이다 [20]. RETOS는 멀티쓰레드 프로그래밍 인터페이스를 제공하며 커널의 동적 재구성을 통한 확장이 가능하다는 특징을 가지고 있다. 또 다른 특징으로 Stack switching을 통한 듀얼 모드를 지원하고 있다. 즉 운영체제의 작동 모드를 커널 모드와 응용 모드로 구분하여 커널 코드가 동작할 때 하나의 커널 스택에서만 동작하도록 만듦으로서 쓰레드 별로 할당된 스택의 증가를 방지하고 있다. 또한 가상 메모리를 지원하지 않는 하드웨어 상황에서 응용 코드에 동적 체크 코드를 삽입함으로써 허용되지 않은 메모리 영역을 침범하는 것을 감지할 수 있다. 하지만 다양한 기능을 제공함으로써 런타임시 오버헤드가 있을 수 있고 상대적으로 코드 사이즈가 커진다는 등의 단점이 있다.

IV. 센서 네트워크 OS 관련 연구

1. MAC, Routing 프로토콜

센서 네트워크에서 운영체제와 별개로 생각할 수 없는 기술 중 하나가 바로 센서 네트워크의 네트워크 부분이다. 센서 네트워크와 관련된 많은 무선 MAC, Routing 알고리즘들이 연구되었는데 대부분 많은 가정을 가진 논문 수준의 알고리즘으로 실제 센서 네트워크 운영체제 구현된 것은 극히 드물다.

현재 대부분의 센서 네트워크 플랫폼의 경우 무선 통신 부분은 IEEE 802.15.4 표준을 지원하는 CC2420 Chip이 가장 많이 사용되고 있다. 해당 RF 모듈을 사용하기 위해선 MAC이 필수적으로 필요하다. MAC이 복잡해짐에 따라 OS와는 별도의 스케줄러를 가지는 경우도 있는데 이때 하나의 MCU로 동작하기 때문에 운영체제와 긴밀한 협조 하에 운영되어야한다. 현재 많은 OS가 간단한 MAC부터 복잡한 MAC까지 독자적인 MAC을 지원하고 있으며 멀티 홉 통신을 위해 그에 맞는 라우팅 프로토콜 또한 지원하고 있다.

〈표 3〉 센서 네트워크 운영체제 비교

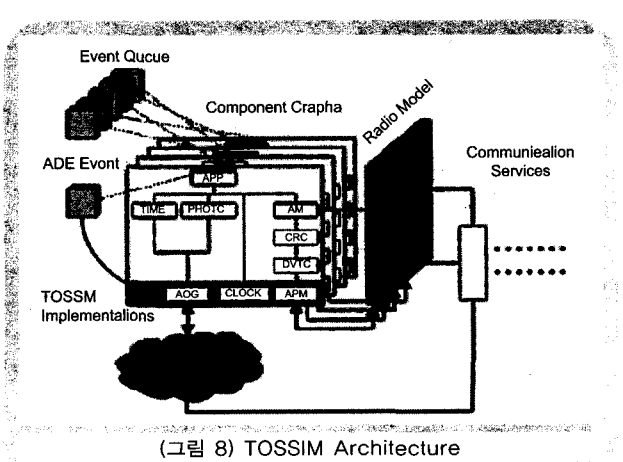
	동적 모듈 로드	실행모델	지원플랫폼	실시간성	네트워크지원	관련도구/툴
TinyOS	미지원	컴포넌트베이스 이벤트 드리븐	Atmega128,3 ARM7,MSP430, PXA271등	X	B-mac, Flooding	TOSSIM, TOSVIS, TinyDB, TinySEC
SOS	지원	모듈베이스 이벤트 드리븐	ARM7 Atmega128	X	UBMAC, Tree Routing	-
MANTIS OS	미지원	멀티쓰레드	Atmega128	Δ	CC1000, CC2420기반 MAC, X-MAC	-
Contiki	지원	Protothread기반 이벤트 드리븐	MSP430 AT91SAM7s	X	UIP, Rime	COOJA
T-Kernel		Natin base 멀티 쓰레드	Atmega128	Δ	-	-
Nano-RK	미지원	Reservation 베이스 멀티 쓰레드	Atmega32 Atmega128	○	Flooding; DSR, RT-Link, WiDom b-MAC	-
NanoQplus	미지원	멀티 쓰레드	MSP430 Atmega128 CC2430	Δ	Zigbee, EAMR	NanoESTO, NanoMON
RETOS	지원	멀티 쓰레드	MSP430 Atmega128 CC2430	Δ	Zigbee, EAMR	NanoESTO, NanoMON

TinyOS의 경우 CC1000을 이용한 B-MAC이 구현되어 있고 CC2420을 이용한 CSMA/CA기반의 MAC이 구현되어 있으며 라우팅 알고리즘으로는 DSDV를 변형한 홉 수를 이용한 distance vector 방식을 사용하고 있다. Nano-RK의 경우 독자적인 TDMA 방식의 RT-Link부분을 제안하고 있으며[19], Mantis OS의 경우 패킷 방식의 Short Preamble을 이용해 에너지 효율성을 제공하는 X-MAC을 제안하였다. Contiki같은 경우 uIP와 Rime같은 라우팅 프로토콜을 제공하고 [15,16], SOS같은 경우 Uncetrain-driven BMAC(UBMAC)이라는 BMAC을 시간 동기화와 결합한 MAC프로토콜과 Tree-routing을 지원한다. NanoQPlus의 경우 다중 경로 라우팅 알고리즘인 EAMR을 제공한다. 1장에서 언급한 센서 네트워크 분야의 요소 기술 중 하나인 네트워크 기술은 이처럼 센서 네트워크 운영체제와 밀접하게 관련이 있으며 아직도 많은 연구가 이루어지고 있다.

2. 시뮬레이터

일반 임베디드 프로그램에서는 타겟 보드를 가지고 호스트에서 프로그램 테스트가 가능하지만 센서 네트워크에서는 한 두 개가 아니라 여러 센서 노드가 유기적으로 통신하는 네트워크 환경을 테스트해야 한다. 실제 대규모 센서 노드를 가지고 테스트하기 앞서, 센서 네트워크 운영체제로 개발한 응용을 테스트하고 동작을 검증하려면 시뮬레이터가 필수적이며, 각각 OS에 맞는 시뮬레이터 또한 개발되어 있다.

TOSSIM은 TinyOS로 개발된 코드를 수행할 수 있는 시뮬레이터로 사용자의 코드를 TOSSIM 프레임워크에 맞게 컴파일 함으로써 수정 없이 손쉽게 시뮬레이션을 수행할 수 있다는 장점이 있다 [27]. TinyOS의 컴포넌트 그래프에 기반한 Discrete-Event Simultor로 라디오, ADC 등이 모델링 되어 있으며 Bit-Level에서 시뮬레이션이 가능 하다(그림 8).



(그림 8) TOSSIM Architecture

NanoQplus의 경우 NanoESTO라는 개발환경에서 오브젝트 파일 기반의 시뮬레이션을 제공한다. Atmega128, CC2420 chip를 가진 센서 노드를 기계어 레벨의 명령어 처리로 에뮬레이션하여 보다 정교한 시뮬레이션이 가능하다 [26].

Contiki는 netsim simulator를 위한 컴파일이 가능하며 그와 별도로 COOJA라는 Contiki OS를 위한 시뮬레이터를 제공한다. COOJA는 Java기반의 크로스-레벨의 시뮬레이션이 가능한 시뮬레이터로 네트워크 시뮬레이션을 해볼 수 있다. 두 가지의 라디오 모델을 지원하며, 응용레벨, OS레벨 하드웨어 레벨 등 레벨에 따른 크로스레이어 시뮬레이션으로 실행 시간을 단축시킬 수 있다 [28].

3. 개발도구

개발자의 입장에서 운영체제와 더불어 중요한 것을 뽑으라면 좀 더 쉽고 효과적으로 응용 프로그램을 개발하기 위한 개발 도구를 들 수 있다. TinyOS의 경우 많은 3rd party들이 존재하기 때문에 개발 도구에 관한 연구도 많이 진행되어 왔다. TinyDT라는 TinyOS 1.x버전을 위한 개발한 Eclipse Plugin이외 많은 Eclipse 기반 plugin들이 존재하며, Gratis라는 컴포넌트 기반 모델링 도구도 존재한다. 상용제품으로는 크로스보우사에서 나온 TinyOS 1.1에 기반한 Software 통합 솔루션인 MoteWorks라는 제품도 출시되어 있다 [25]. 또한 NanoQplus용 개발도구인 NanoEsto는 자체적으로 내장된 시뮬레이션 가능을 제공하며 이를 이용해 프로그램의 정확성 및 안정성을 검증할 수 있다 [26].

V. 결 론

본고에서는 센서 네트워크를 위한 초소형 OS에 대한 소개와 현재 개발되어 있는 운영체제를 비교 분석하여 설명하였

다. 센서 네트워크를 위한 초소형 운영체제 기술은 앞으로 센서 네트워크 시장 성장에 따라 그 중요성이 더욱 부각될 것으로 예상되고 있다. 센서 네트워크 운영체제 관련 기술은 운영체제 이외에도 운영체제와 뗄 수 없는 네트워크 스택부터 개발환경이나 모니터링 등의 툴 또한 중요한 이슈로 떠오르고 있다. 앞서 살펴본 바와 같이 여러 디자인 이슈를 가지고 지금도 많은 센서 네트워크 운영체제들이 개발되고 업그레이드 되고 있어 지속적인 관심이 필요하다.

참 고 문 헌

[1] 박현식, WSN(Wireless Sensor Network) 기술 동향, HN Focus Vol. 11 (2006.7) p. 48-49

[2] U-센서 네트워크, ITFIND 주간기술동향 2006.08.

[3] http://www.tinyos.net

[4] Han, C.C., Kumar, R., Shea, R., Kohler, E., Srivastava, A dynamic operating system for sensor nodes, 2005, 163-176

[5] Bhatti, S., Carlson, J., Dai, H., Deng, J., Rose, Mantis os: An embedded multithreaded operatingsystem for wireless micro sensor platforms. ACMKluwer Mobile Networks and Applications(MONET) Journal, Special Issue on Wireless Sensor Networks, 2005

[6] Adam Dunken, Niclas Finne, Joakim Eriksson, and Thiemo Voigt. "Run-Time Dynamic Linking for Reprogramming Wireless Sensor Networks", SenSys 2006, Boulder, Colorado, USA, November 2006

[7] L. Gu and J. A. Stankovic, t-kernel: Providing Reliable OS Support for Wireless Sensor Networks, SenSys 2006

[9] IPv6 over Low power WPAN WG (6lowpan), (http://www.ietf.org/html.charters/6lowpan-charter.html)

[10] 박승민,전자통신동향분석. 센서 네트워크 노드 플랫폼 및 운영체제 기술동향

[11] USN을 이용한 홈네트워크, TTA Journal 2006.06

[12] Atmel atmega128 datasheet http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

[13] MSP430 Ultra-Low-Power Microcontrollers, Texas Instruments(http://focus.ti.com/mcu/docs/mcuprodoverview.tsp?sectionId=95&tabId=140&familyId=342)

[14] Adam Dunkels, Protothreads: Simplifying Event-Driven Programming of Memory-Constrained Embedded System, ACM Sensys 2006, Boulder, Colorado, Nobember 2006.

[15] Adam Dunkels, Juan Alonso, and Thiemo Voigt, "Making TCP/IP Vialbe for Wireless Sensor Networks", EWSN 2004

[16] Adam Dunkels, RIME - A lightweight Layered Communication Stack for Sensor Networks, EWSN 2007, Netherlands, Jan 2007.

[17] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt, "Cross-Level Sensor Network Simulation with COOJA", SenseApp 2006, Florida, USA, Nov 2006.

[18] A. Eswaran, A. Rowe and R. Rajkumar, "Nano-RK: An Energy-Aware Resource-Centric Operating system for Sensor Networks", IEEE Real-Time Systems Symposium, Dec 2005.

[19] A Rowe, Rahul Mangharam, Raj Rajkumar, "RT-Link: A Time-Synchronized Link Protocol for Energy Constrained Multi-hop Wireless Networks", SECON 2006, Setp 2006.

[20] 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", The Sixth International Conference on Information Processing in Sensor Networks (IPSN 2007), Cambridge, Massachusetts, USA, April 2007.

[22] B-MAC

[23] P. Buonadonna, J. Hill, and D. Culler, "Active MessageCommunication for Tiny Networked Sensors," in Proc. of the 20th Annual Joint Conf. of the IEEE Comput. and Commun. Societies, Anchorage, Alaska,

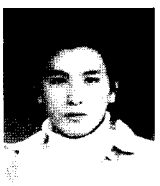
USA, Apr. 2001.

[25] MoteWorks Brochure http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MoteWorks_OEM_Edition.pdf

[26] In-geol Chun, Chae-deok Lim, "NanoEsto Debuger: The Tiny Embedded System Debugger" ICACT 2006, Feb 2006.

[27] Philip Levis, Nelson Lee, Matt Welsh, and David Culler, "TOSSIM : Accurate and Scalable Simulation of Eitire TinyOS Applications", SysSys 2003

[28] Fredrik Osterlind, Adam Dunkels, Koakim Eriksson, Nicals Finne, and Theimo Voiqt, "Cross-level Sensor Network Simulation with COOJA"



송 준 근

2005년 아주대학교 학사
2005년 ~ 현재 한국전자통신연구원 연구생, University of Science & Technology 석사
관심분야 : WPAN, 센서 네트워크, 무선 MAC 프로토콜

마 평 수

1985년 서울대학교 학사
1992년 City Univ of Newyork 석사
1995년 Wright State Univ 박사
1996년 ~ 현재 한국전자통신연구원, 책임연구원, 팀장
관심분야 : 센서노드용 초소형 OS, 센서 네트워크, 임베디드 SW

박 승 민

1983년 홍익대학교 대학원 전자공학과(석사)
1983년 ~ 1984년 (주)LG전자
1984년 ~ 현재 한국전자통신연구원 임베디드SW연구단
임베디드SW플랫폼연구그룹장/책임연구원