μTMO 모델 기반의 실시간 센서 네트워크 운영체제의 설계

이재안° 최병규 허신
한양대학교 컴퓨터 공학과
{jalee°, bkchoi, shinheu}@cse.hanyang.ac.kr

Design of Real-Time Operating System for Sensor Network based on μTMO Model

Jae-An Lee° B.K. Choi Shin Heu
Dept. of Computer Science & Engineering, Hanyang University

요 약

무선 센서 네트워크는 유비쿼터스 컴퓨팅에서 생활 환경과 컴퓨터 사이의 중계자 역할을 하는 매우 중요한 연구 분야이다. 매우 제약적인 자원 환경에서 동작하여야 하는 센서 노드의 동작 환경적 특성 때문에 제한된 자원을 효율적으로 관리할 수 있는 센서 노드용 운영체제가 요구된다. 센서 노드는 제약적인 자원을 가지고 있지만 데이터 수집, 데이터 프로세싱, 다른 노드로부터 수신된 데이터의 전달 등 여러 가지 작업들이 동시에 발생된다. 기존의 범용 센서네트워크 운영체제에서는 극도로 제한된 자원을 최대한 효율적으로 사용할 수 있는 방법에 대하여 주로 연구해 왔다.

무선 센서 네트워크의 응용 범위가 점차 넓어지고 있다. 방사능 감지와 같이 실시간성을 요구하는 응용 분야들이 생겨나기 시작하면서 센서 네트워크에서도 실시간성의 필요성이 대두되게 되었다. 실시간 센서 네트워크 연구 분야에서 실시간 통신 프로토콜의 연구 결과가 발표되고 있지만, 실시간 운영체제의 지원 없이 완전한 실시간성을 보장하기 힘들다. 하지만 센서 노드용 실시간 운영체제에 대한 연구는 아직까지 활발히 진행되지 않고 있다. 본 논문에서는 정시성을 보장하는 분산 객체 모델인 TMO를 센서네트워크의 제한된 자원 환경에 알맞도록 경량화 시킨 μTMO 모델을 제시하고, 센서 노드용 운영체제에 μTMO 모델을 적용하여 실시간성 지원에 따른 오버헤드를 감소시킨 실시간 센서 네트워크 운영체제의 구조를 제안한다.

1. 서 론

무선 센서 네트워크는 산재되어 있는 무선 센서 노드가 데이터를 수집하고 이를 가공하며, 노드들 간의 협업을 통하여 만들어진 무선 네트워크를 통하여 데이터를 전송함으로써 사용자가 이를 활용할 수 있도록 해주는 기술이다. 센서 네트워크를 구성하는 센서 노드는 극도로 제한된 자원을 가지고 있기 때문에 효율적인 자원의 활용이 매우 중요한 이슈가 된다.[1]

일반적인 센서네트워크 응용 환경에서는 극도로 제한된 자원을 효율적으로 운영하기 위해 효율적으로 동작하는 운영체제가 필요하다. 이러한 범용 센서 네트워크용 운영체제는 운영체제 자체의 크기가 매우 작아야 하며, 메모리 관리가 효율적이어야 한다. TinyOS[2]나 MANTIS[4]와 같은 범용 센서네트워크 운영체제는 자원의 효율적인 사용에 초점을 두고 연구 되었다.

센서 네트워크의 응용 분야가 점차 넓어지면서 군사 분야에서 적을 탐지하거나 핵발전소 또는 핵 위험지역의 방사능 측정과 같이 실시간성을 요구하는 응용 분야가 나타나게 되었다. 방사능과 같은 유독성 물질을 감시하는 센서 네트워크 응용의 경우, 센서 노드가 수집한 데이터를 제한된 시간 이내에 전달하지 못할 경우 지연된 데이터의 의미가 없어질 수 있다.[5] 무선 센서네트워크 연구 분야에서 이러한 실시간성 요구를 만족시키기 위해 SPEED[7]나 RAP[8]와 같은 무선 센서 네트워크용 실시간 통신 프로토콜들이 연구 되고 있지만 아직까지 무선 센서 네트워크용 실시간 운영체제에 대한 연구는 거의 이루어 지지 않고 있다.

연구가 진행 된 실시간 통신 프로토콜 역시 센서네트워크에서 가장 널리 사용되는 TinyOS와 같은 범용 센서 네트워크 운영체제 또는 시뮬레이션을 통해 연구가 이루어지고 있다. 기존 센서네트워크용 운영체제는 자원의 효율적인 사용 측면에 초점을 맞추고 개발되었기 때문에 실시간성을 지원하기에는 적합하지 않은 구조를 가지고 있다. 실시간 요구사항을 만족시키기 위해서는 실시간 프로토콜로만 해결 가능한 것이 아니라 센서 노드를 운영하는 운영체제 역시 작업에 대한 데드라인을 보장해 주는 실시간성을 지원해야 한다.

일반적으로 실시간 운영체제는 범용 운영체제에 비하여 스케줄링 단계에서 계산이 추가되며, 실시간 지원을 위한 부가적인 기능들의 추가에 의하여 오버헤드가 발생된다. 센서 노드는 제한적인 자원을 가지고 동작하여야 하기 때문에 센서 노드에서 동작하는 실시간 운영체제를 개발하기 위해서는 실시간 지원에 따른 오버헤드를 최대한 줄여야 한다.

TMO 모델은 정시보장, 분산 환경의 특징을 갖는 분산 실시간 객체 모델이다. 본 논문에서는 실시간 분산 객체 모델인 TMO를 센서 네트워크 응용에 알맞은 형태로 변형한 μTMO(MicroTMO) 모델을 제시하고, μTMO 모델을 이용하는 실시간 센서 네트워크 운영체제의 설계를 제안하고자 한다.

2. 관련연구

2.1 TMO

TMO 모델은 Time-treggered Message-treggered Object의 약자로써 U.C. Irvine의 Kane Kim 등에 의하여 개발된 정시보장 컴퓨팅 패러다임(Timeliness guaranteed computing paradigm)을 지향하는 분산 객체 모델이다. 여러 개의 TMO 네트워크로 설계된 시스템은 분산 환경에서 시간 조건에 의하여 수행된다.[11]

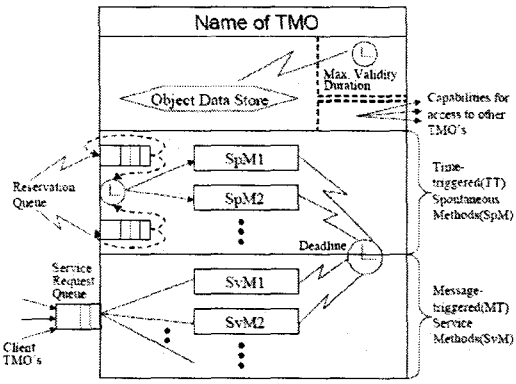

그림1. TMO 모델의 구조

그림1은 TMO의 모델 구조를 보여주고 있다. TMO는 일반적인 객체 멤버 형태 이외에 새로운 형태인 SpM과 SvM 두가지 멤버 스레드를 가질 수 있다.

* 시간 구동 메소드인 SpM(Spontaneous Method)은 주어진 시간 조건에 의해 자율적으로 구동된다. SpM에는 시작과 종료시간, 주기, 데드라인으로 구성된 시간 제약이 주어지며 커널이나 TMO 엔진에서 이러한 시간 조건에 따라 실시간 스케줄링을 제공한다.
* SvM(Service Method)은 IPC 이벤트 메시지 수신에 의해 구동이 되며 실행이 시작되면 주어진 데드라인에 의해 스케줄링 된다. SvM을 구동시키는 IPC 메시지는 분산 환경에서도 변경 없이 그대로 적용 가능 한 네트워크에 투명한 IPC이다.
* ODS는 객체 자료 저장소(Object Data Store)로써 SpM과 SvM 메소드들이 공유하는 데이터를 가지고 있으며, 최대 유효기간(Maximum Validity Duration)을 초과하게 되면 그 데이터는 유효성을 잃는다.

경성 실시간 응용의 경우 TMO 모델은 SpM을 통하여 설계 시 실시간 보장 개념을 제공한다.

2.2 센서네트워크 운영체제

센서네트워크를 운영체제 중 가장 널리 사용되는 것은 버클리 대학의 TinyOS이다. TinyOS는 nesC 언어를 이용한 컴포넌트 기반의 구조를 가지고 있으며, 이벤트 기반(Event Driven) 동작으로 많은 스택 메모리를 요구하는 멀티 쓰레드 방식의 단점을 보완하고 있다. TinyOS에서는 컴포넌트 단위로 시스템을 모듈화 하여 구성하고 있다.

MANTIS는 콜로라도 대학에서 만든 센서 노드용 운영체제로써 초소형 멀티스레드 구조를 채택한 운영체제 이다. 기존의 클래식 운영체제 구조와 같은 계층화된 멀티 스레드 운영체제의 형태를 가지고 있으며, C언어를 이용하여 프로그래밍 한다. 프로그래머들은 기존의 멀티스레드 응용프로그램 방식을 그대로 적용 가능하기 때문에 새로운 개념을 습득할 필요 없이 쉽게 프로그래밍 가능한 장점이 있다.

PEEROS[9]는 EU의 EYES 프로젝트를 위해 개발된 하드웨어 독립 방식을 채택한 운영체제이다. EDF를 기반으로 하여 경량화 시킨 EDFI[10] 스케줄링 기법을 적용하여 실시간 스케줄링을 제공한다. 간단한 명령 구현을 위한 커멘드 쉘 (Command Shell)을 탑재하고 있어 사용자가 시리얼 통신을 이용하여 간단한 명령할 수 있는 특징이 있다.

한국의 ETRI에서는 Nano-Qplus[3]라는 센서용 운영체제를 개발하였다. Nano-Qplus는 MANTIS와 같은 형태의 멀티 스레드를 기반으로 하는 초경량 센서 운영체제이다. 제한된 메모리의 효율적 운영을 위해 멀티 스레드 간 스택을 공유하며 멀티 스레드 스케줄러를 제공한다. 개발 시에 필요한 모듈만 선택하여 사용할 수 있도록 설계 하여 응용에 따라 매우 작은 크기로 만들 수 있다.

3. μTMO 모델 기반의 실시간 운영체제

3.1 센서 네트워크 응용에 TMO 모델 적용 가능성

TMO 모델을 이용하여 무선 센서 네트워크 응용에 적용시켜 보면 그림2와 같은 형태로 표현할 수 있다.

그림2는 방사능 유출을 탐지하는 센서 네트워크 응용을 나타낸 것이다. 방사능 정보를 수집하여 전달하는 것은 반드시 정해진 시간 이내에 처리가 완료 되어야 하는 실시간성이 요구되는 작업이며, 다른 정보에 비해 수집된 정보 변화에 민감한 반응이 필요하므로 정보 수집 주기가 짧아야 한다. 본 예제에서는 50ms 마다 방사능 물질 정보를 수집하고, 20ms의 데드라인을 주었다. 주변의 부수적인 정보인 온도, 습도는 방사능에 비하여 상대적으로 덜 중요한 정보이므로 500ms 마다 정보를 수집하고 200ms의 데드라인을 가진다. 매 700ms 마다 수집된 온도와 습도, 방사능 정보를 Base Station으로 전송한다. 다른 노드로부터 수신된 메시지는 라우팅 경로를 통하여 전달하고, 방사능 수치가 급격히 변하는 경우 긴급 메시지를 전송한다.

본 예제에서와 같이 TMO 모델을 센서 응용에 적용 시 데드라인 개념을 적용 가능할 뿐 아니라 센서 노드의 동작 특성을 SpM과 SvM을 이용하여 높은 추상화 수준에서 표현이 가능하다.

무선 센서 노드의 동작 특성을 살펴보면, 정보 수집을 위한 주기적인 작업들과 특정 이벤트 발생 시 생겨나는 비주기적인 작업으로 나눌 수 있다. 기존의 멀티 스레드 기반 센서 네트워크 운영체제에서 주기적인 작업은 실행 코드 중간에 시스템 API인 sleep()과 같은 대기 함수를

이용하여 구현하였다. 이렇게 코드 중간에 sleep()과 같은 코드가 복잡한 순환문이나 분기문에 적용되는 경우 코드의 동작을 이해하는데 어려움이 있었으며, 타이밍 문제에 의한 디버깅이 어려운 단점이 있다. TMO 모델을 적용하는 경우 주기적인 작업을 SpM에 기술하여 사용자가 이해하기 쉬운 높은 추상화 수준을 제공한다. 비주기적인 이벤트성 작업은 SvM을 통하여 지원한다.

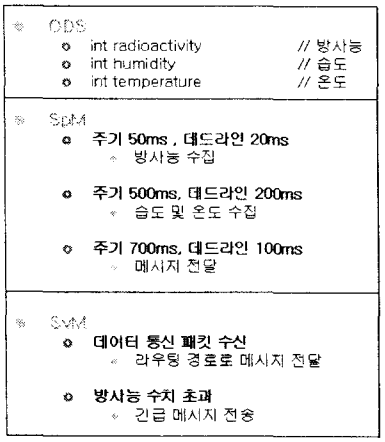

그림2. TMO 모델을 적용한 센서 응용

3.2 TMO 모델 적용 시 발생 가능한 문제점

TMO는 실시간 분산 객체 패러다임으로써 객체지향 프로그래밍 및 다수의 분산된 TMO들이 실시간으로 협업하여 동작하는 방식으로 구성되어 있다. 하지만 이러한 모델을 센서네트워크에 그대로 적용하는 경우 문제점이 발생 할 수 있다.

객체지향 프로그래밍 패러다임을 그대로 이용하는 경우 일반적으로 C 언어 기반의 순차적 프로그래밍에 비하여 크기가 커지며 실행 시 동적으로 결정되는 부분들에 의하여 실행이 느려질 수 있다. 일반적인 컴퓨팅 환경에서는 크게 문제가 되지 않는 부분이지만, 센서 노드의 경우 대부분 8bit 기반의 8Mhz 정도의 동작 속도를 가지는 MCU를 주로 사용하므로 객체지향 방식으로 작성할 경우 센서 노드에서는 속도 저하가 일어 날 수 있으며, 메모리 사용량이 늘어난다. 보통의 센서 노드에서는 4KB 정도의 메모리를 제공하기 때문에 메모리 사용량이 많은 객체지향 방식 프로그래밍은 센서 노드에서 사용하기에는 부담이 될 수 있다.

TMO 모델은 분산 객체 개념을 기반으로 동작하기 때문에 여러 TMO 객체 사이에 IPC나 로컬 네트워크를 통한 통신이 빈번하게 일어난다. 유선 환경에서는 로컬 네트워크에 브로드케스팅이나 멀티케스팅을 통한 분산 환경 구축이 큰 문제를 일으키지 않지만, 아주 작은 대역폭의 무선 네트워크를 사용하는 센서 네트워크에서는 문제가 발생할 가능성이 있다. TMO 모델 사이의 메시지가 센서 노드의 전체 네트워크로 전송되면 무선 대역폭을 차지하여 다른 정보의 송수신을 방해하는 것 뿐 아니라, 잦은 무선 송수신은 센서 노드의 중요한 자원인 전력을 많이 소모한다. 따라서 무선 통신에 따른 전력 소모가 많은 분산 모델을 센서 네트워크에 그대로 적용하기에는 무리가 따른다.

3.3절에서는 위에서 언급한 문제점을 해결하고자 분산 객체 프로그래밍 패러다임인 TMO 모델을 센서네트워크에 그대로 적용하지 않고, 센서 노드에 적합하도록 경량화 시킨 μTMO 모델을 제안한다.

3.3 센서 네트워크에 알맞도록 TMO 모델의 경량화

3.2절에서 설명한 TMO 모델의 센서 네트워크 적용 시 발생 가능한 문제점을 해결하기 위하여 TMO 모델을 센서 네트워크에 알맞도록 경량화한 μTMO(microTMO) 모델을 제안한다.

계산 능력 및 사용 가능한 메모리의 크기가 제한적인 MCU를 사용하기 때문에 객체지향 언어인 C++를 사용하는 대신 C언어를 통하여 TMO 모델을 지원할 수 있도록 하였다. SpM과 SvM 함수의 경우 TMO 객체의 멤버 함수이었던 것에 반하여 μTMO 모델에서는 SpM의 List와 SvM의 List에서 실제 동작을 하는 함수와 함수 포인터를 이용하여 맵핑 시키는 구조를 취함으로써 C언어를 이용하여 TMO 구조를 구축할 수 있도록 하였다. 이를 위해 시스템 API에서 시스템에 SpM과 SvM을 등록 수 있는 시스템 콜을 제공하여야 한다. 이러한 등록 함수는 운영체제가 시작될 때 한번만 수행되면 된다. 센서 노드용으로 널리 사용되는 MCU인 Atmel사의 Atmega 8bit RISC의 경우 gcc 기반의 효율적인 C언어 컴파일러를 기본적으로 제공하고 있다.

기존 TMO 모델에서는 IPC를 통해 분산 환경에서 오는 클라이언트의 이벤트 메시지를 처리 한다. 센서 네트워크에서 이러한 분산 처리 방식을 그대로 사용할 경우 무선 통신을 통하여 브로드캐스트 또는 멀티캐스트로 메시지를 보내야 하는데, 이는 통신 대역폭이 작은 센서 네트워크의 특성상 너무 많은 대역폭을 사용하게 되며, 빈번한 무선 전송으로 인해 에너지 소모가 많아진다.

센서 노드의 협업 동작 특성을 TMO 모델과 비교하여 살펴 보자. TMO 모델은 다수의 분산된 TMO 객체들이 IPC 또는 로컬 네트워크를 이용한 협업에 의하여 응용 프로그램이 동작하게 된다. 센서 응용의 경우 그림2에서 볼 수 있듯이 하나의 노드에 하나의 TMO 모델만으로도 동작 표현이 가능하다. 또한 노드의 자원 제약 때문에 복수개의 TMO 모델을 단일 노드에서 실행하는 것은 큰 부담이 된다. 하나의 노드에 하나의 TMO 모델이 수행되므로 IPC에 의한 메시지 송수신이 필요 없으며, 스레드 사이의 동기화는 운영체제에서 제공하는 동기화 메커니즘을 사용하여 처리할 수 있다. 센서 네트워크에서 다른 노드와의 협업은 주로 경로 계산을 위한 라우팅 정보 전달이나 데이터 전달과 같은 간단한 형태로 일어난다. TMO 모델처럼 IPC나 로컬 네트워크를 통해 계속 메시지를 전달하기 보다는 협업이 필요한 경우에 대한 몇 가지 약속된 패킷을 만들어서 전달하는 일반적인 센서 네트워크의 협업 방식으로 처리가 가능하다.

그림3은 센서 네트워크에 알맞도록 변형된 μTMO 모델

의 구조를 보여준다. TMO 객체의 멤버 함수로 구성되어 있던 SpM을 맵핑할 함수의 포인터와 주기, 데드라인으로 구성된 리스트로 관리 하고, SvM을 함수 포인터와 이벤트 종류, 데드라인을 가지는 리스트로 관리 한다.

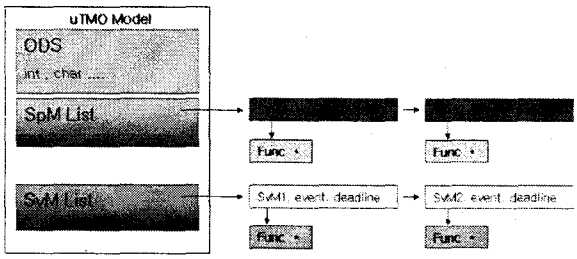

그림 3. μTMO 모델 구조

주기 시간에 따라 동작하는 SpM 리스트의 관리는 TMO 모델과 마찬가지로 WTMT (Watchdog TMO Management Task)에서 타이머에 의하여 관리가 되며, 지정된 주기가 되면 WTMT에 의하여 깨어나서 운영체제의 실시간 스케줄러에 의하여 관리를 받게 된다.

μTMO 모델을 센서 응용프로그램에 적용하면 그림2에서 볼 수 있듯이 디자인 단계에서 실시간성에 대한 제약을 명확하게 줄 수 있으며, 간단한 동작을 하는 함수를 시간 조건이나 특정 이벤트 메시지들과 맵핑시키는 방식을 선택하여 실시간 프로그래밍에 대한 특별한 학습 없이 기존 멀티스레드 기반의 프로그래밍 기법을 통하여 쉽게 실시간성 지원이 가능하다.

3.4 μTMO 실시간 운영체제의 구조

본 논문에서 제안하는 μTMO 모델을 적용하는 실시간 센서네트워크 운영체제의 구조는 그림4와 같다.

최하위 계층에 MCU나 무선 통신을 위한 모듈, 센서 모듈과 같은 하드웨어 계층이 존재하며, 이 하드웨어를 편리하게 사용하기 위한 디바이스 드라이버의 역할을 하는 하드웨어 추상화 계층(HAL)이 존재한다.

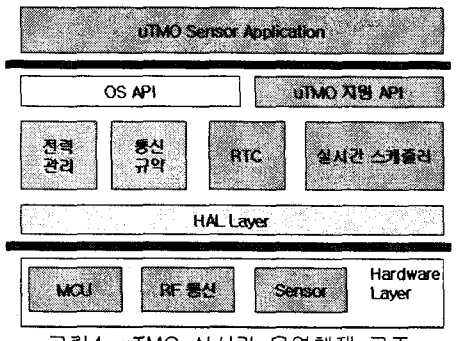

그림4. μTMO 실시간 운영체제 구조

HAL 계층 상위에는 실시간성 지원을 위한 실시간 스케줄러와 실시간 지원을 위한 고해상도 클럭인 RTC가 있다. μTMO 모델을 지원하기 위하여 추가되는 부분은 실시간 스케줄러와 RTC가 있으며, μTMO 모델의 SvM과 SpM을 시스템에 등록하기 위한 μTMO 지원 API가 필요하다.

실시간 스케줄러는 계산 능력이 떨어지는 MCU의 효율적인 사용을 위하여 CPU 활용률이 높은 EDF[6] 스케줄러를 이용하거나 EDF를 센서 네트워크에 알맞도록 경량화 시킨 EDFI[10]와 같은 스케줄러를 이용할 수 있다.

고해상도 클럭인 RTC에 의하여 깨어나는 WTMT가 SpM의 주기적인 동작을 관리하게 된다. 이외의 구조는 멀티 쓰레드 기반의 센서 네트워크 운영체제인 MANTIS나 Nano-Qplus의 구조와 유사하다. μTMO 모델은 SpM과 SvM에 맵핑된 함수가 깨어날 때 스레드에 등록한 함수를 실시간 스케줄러에 의해서 스케줄링 되도록 하는 구조로 설계되었기 때문에 기존 클래식 운영체제와 유사한 형태를 가지는 멀티 스레드 기반의 센서 네트워크 운영체제에 적용시킬 수 있다.

4. 결론 및 향후과제

센서 네트워크의 응용분야가 점차 넓어지면서 실시간성에 대한 요구가 생겨나기 시작하였다. 하지만 기존의 센서 네트워크용 운영체제는 자원의 효율적인 운영측면에 중점을 두고 개발되었기 때문에 이러한 실시간 요구사항을 만족시키기 어렵다. 실시간성을 지원하기 위한 커널은 기존의 커널에 비하여 스케줄링 우선순위 계산에 따른 계산량이 늘어나고 많은 자원을 소모하게 된다. 센서 노드는 자원이 매우 제한되어 있는 동작 환경이므로 센서 노드에서 실시간성을 지원하기 위한 운영체제에서는 실시간 지원에 따른 오버헤드를 최대한 줄여야 한다.

본 논문에서는 센서 네트워크 환경에서 실시간의 필요성에 대해 간략히 알아보았고, TMO 모델을 실시간 센서 응용프로그램에 적용해 보았다. TMO 모델 적용 시 예상되는 문제점을 진단하고, 이를 해결하기 위하여 센서 네트워크에 알맞게 경량화 시킨 μTMO 모델을 제안하였으며, μTMO 모델을 이용하여 실시간성을 지원하는 운영체제의 구조를 제안하였다.

향후 과제로, 본 논문에서 제시한 μTMO를 이용하는 실시간 운영체제 모델을 실제 센서 노드용 운영체제에 적용시켜 무선 센서 노드 상에서 활용 가능한 실시간 운영체제를 개발하는 연구가 진행되어야 한다. 또한, 실시간 운영체제에서 효율적인 스케줄링은 매우 중요한 요소이다. μTMO 모델의 특징을 이용한 효율적이고 빠른 실시간 스케줄링에 대한 연구가 이루어져야 한다.

5. 참고문헌

[1] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, Erdal Cayirci, "A Survey on Sensor Networks", IEEE Communications Magazine, pp102-114, August 2002

[2] TinyOS, www.tinyos.net

[3] Nano-Qplus, http://qplus.or.kr

[4] S. Bhatti et al., 'MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms," ACM/Kluwer Mobile Networks and Applications (MONET), Special Issue on Wireless Sensor Networks, vol. 10, no. 4, August 2005

[5] John A. Stankovic, 'Research challenges for wireless sensor networks', ACM SIGBED vol1, Issue 2, pp9-12, Jul 2004

[6] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," Journal of the ACM, vol. 20, no. 1, pp. 46-1, 1973.

[7] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Ad Hoc Sensor Networks", IEEE ICDCS, May 2003.

[8] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He, "RAP: A Real-Time Communication Architecture for Large-Scale Wireless SensorNetworks", RTAS, June 2002.

[9] J. Mulder, "PEEROS - Preemptive EYES Real-Time Operating System", Master's thesis, Faculty of EEMCS, University of Twente, 2003.

[10]Jansen, P.G., Mullender, S.J., Havinga, P.J.M., Scholten, J. "Lightweight EDF scheduling with deadline inheritance", Centre for Telematics and Information Technology, University of Twente Technical report (TR-CTIT-03-23), 2003

[11]K.H. Kim and Kopetz, 'A Real-Time Object Model RTO.k and an Experimental Investigation of Its Potentials," Proc. 18th IEEE Computer Software and Applications Conference, pp.392-402, November 1994.