감시정찰 센서네트워크를 위한 초소형 내장소프트웨어

이우용[a], 김진우[a], 김석환[a], 엄두섭[b], 권미영[c]

[a] *고려대학교 전자전기공학과*
서울시 성북구 안암동 5-1, 136-713
Tel: +82-2-3290-3802, Fax: +82-2-3290-3802, E-mail: {waryong, jjin300, sukka}@final.korea.ac.kr

[b] *고려대학교 전자전기공학과*
서울시 성북구 안암동 5-1, 136-713
Tel: +82-2-3290-3248, Fax: +82-2-3290-3802, E-mail: eomds@korea.ac.kr

[c] *국방과학연구소 2기술연구본부 2부*
서울시 송파구 거여동 25번지, 138-110
Tel: +82-2-3400-2625, Fax: +82-2-3400-2625, E-mail: kmyadd@paran.com

Abstract

감시정찰 센서네트워크의 모든 센서노드 및 싱크 노드들은 한정된 자원과 저사양의 하드웨어로 동작하며, 각 침입탐지 센서들이 수집한 상황 데이터를 신뢰성 있게 전송할 수 있어야 한다. 본 초소형 내장소프트웨어는 이러한 감시정찰 센서네트워크의 특성에 맞게 설계되어 센서 및 싱크노드에 탑재될 수 있는 소프트웨어로서, 센서 OS 커널, 센서미들웨어, 보안커널로 구성된다. 센서 OS는 Multithread 기반으로 실시간, 비실시간 태스크를 위한 각기 다른 스케줄링 방식을 제공하며 지연된 인터럽트 처리 기능, 주기적 태스킹 기능과 효율적 에너지 관리 기능을 제공하여 센서 네트워크에 특화된 어플리케이션 개발을 용이하게끔 한다. 또한 센서미들웨어는 OS 커널과 어플리케이션 사이에 존재하여 위치인식, 시간동기, 네트워크 관리, 원격 업데이트 기능 등 어플리케이션에서 공통적으로 요구하는 필수 기능들을 제공한다.

Keywords:

감시정찰; 센서네트워크; 초소형 내장소프트웨어

서론

센서네트워크란 수백만 개의 매우 작은 센서를 우리가 살고 있는 곳곳에 설치해 상호간에 무선으로 데이터를 주고 받으며 인간과 상호작용 할 수 있도록 하는 기술이다. 이런 무선 센서네트워크는 다양한 상황에서 인간과 환경의 상호 작용에 필요한 정보 수집과 처리에서 큰 변혁을 일으킬 것으로 예상되며, 인류의 삶을 다시 한번 근본적으로 변화시킬 수 있는 새로운 기술로 주목 받고 있다. 또한 다양한 센서(온도, 습도, 물체의 움직임, 빛의 밝기, 압력, 토질, 잡음 레벨, 물체의 유무, 이동 방향 등)들을 부착하여 군사, 의료, 홈 네트워크, 환경 감시, 공장 관리, 재난 감시 등 여러 응용에 적용할 수 있다.

최근에 이러한 응용 중에서 특히 감시정찰 센서네트워크의 경우, 군사적 또는 민수적 측면에서 그 개발의 필요성이 크게 대두되고 있다. 앞으로의 미래 전쟁은 다양한 센서로부터 들어온 표적, 항적, 위치, 피아식별 등의 디지털 기술 정보를 가공하여 곧바로 타격하는 sensor-to-shooter 개념의 네트워크 중심전 양상을 지니게 되는데, 이 경우 기존 플랫폼 중심 무기체계에서 감지하지 못하는 각종 전술 정보를 실시간으로 수집할 수 있는 센서기반의 감시체계의 개발은 반드시 필요하다. 또한 민수 분야에서도 산불 방지 시스템이나 교량, 도로 등 인간의 접근이 쉽지 않은 건축물에 대한 각종 원격 탐지 시스템에 감시정찰 센서네트워크 기술은 없어서는 안 될 중요한 기술이다.

센서네트워크의 핵심 기술 요소로는 하드웨어 플랫폼과 센서노드에 탑재되는 소프트웨어 기술을 들 수 있다. 일반적으로 센서네트워크용 내장형 소프트웨어는 크게 어플리케이션 프로그램 부분과 센서 네트워크용 OS 부분으로 나눌 수 있으며, 센서 네트워크용 OS는 실시간 초소형 센서 OS 커널, 네트워크 프로토콜 스택, 센서노드 미들웨어, 보안커널 및 타겟 시스템에서 동작하기 위한 통합 개발환경으로 구성된다. 이러한 내장형 소프트웨어는 컴퓨팅 능력과 자원이 극도로 제한된 센서네트워크 환경에서 동작하기 때문에, 극소형의 크기를 가져야 하며, 기존의 소프트웨어보다 더 많은 제약사항을 갖는다. 본 논

문에서는 OS 커널과 센서노드 미들웨어를 중심으로 하여, 군사 및 민수용 감시정찰 센서네트워크에서 요구하는 다양한 서비스 품질을 만족시킬 수 있는 내장형 소프트웨어의 구조 및 기능에 대해 살펴본다.

내장형 소프트웨어 연구 동향

운영체제 연구 동향

최근 센서네트워크 운영체제에 대한 많은 연구들이 이루어지고 있으며, 스케줄링 방식에 따라 크게 event-driven 방식과 multithread 방식으로 구분할 수 있다. Event-driven 방식을 사용하는 대표적인 운영체제에는 TinyOS[1]와 SOS[2]가 있으며, 반면 multithread 방식을 사용하는 대표적인 운영체제로는 Mantis[4], NanoOS[3] 등을 들 수 있다.

Event-driven 방식은 이벤트 발생 순서에 따라 해당 태스크를 처리하는 방식으로 매우 간단하여 제한된 자원을 가지는 센서노드에서 구현 및 구동되기 쉽다는 장점을 가진다. 하지만 긴급하게 처리하여 전송해야 하는 태스크가 현재 수행중인 태스크를 선점하지 못하여 전체 지연 시간을 증가시킬 수 있으며, 이 때문에 OS로 인하여 전체 센서네트워크의 성능을 저하시킬 수 있는 단점을 가진다. 이와 같은 단점은 센서네트워크에서 처리해야 하는 일에 대한 정확한 이해를 바탕으로, 이벤트 발생에 따른 태스크의 처리 순서와 각 태스크에서 처리해야 할 일들을 잘 정의할 수 있다면 극복이 가능하다고 할 수 있으나 일반 사용자에게 이를 기대하기는 쉽지 않다고 할 수 있다.[1,2]

반면 multithread 방식은 높은 우선순위를 가지는 태스크가 낮은 우선순위를 가지는 태스크를 선점할 수 있어, event-driven 방식에 비해 더욱 높은 실시간성을 요구하는 응용 프로그램을 효과적으로 지원할 수 있다. 그러나 문맥전환, thread 간 동기화 및 통신, 자원관리 등 추가적인 기능을 필요로 하여 event-driven 방식과 비교하여 복잡하고 무겁다는 단점을 가진다.[3,4]

실시간 운영체제

일반적으로 센서네트워크에서의 실시간성이라 함은 하나의 노드에 국한되어 논하기 보다는 전체 센서네트워크의 관점에서, 사용자가 요구하는 특정 시간 내에 임의의 센서노드에서 발생한 이벤트가 센서네트워킹을 통하여 성공적으로 싱크노드 혹은 싱크노드에 연결된 서버까지 전달될 수 있어야 함을 의미한다. 센서네트워크에서 사용자의 실시간 요구사항을 만족시키는 방법에는 다음과 같이 두 가지가 있다. 첫째는 사용자가 정량적으로 정한 시간을 엄격히 만족시키는 hard real-time 보장 방법이며, 둘째는 정량적으로 사용자가 정한 시간을 엄격히 만족시키는 대신, 센서네트워크를 구성하는 센서노드들이 가능한 한 이벤트 처리시간 및 노드지연 시간을 줄일 수 있도록 운영체제에서 지원하는 soft real-time 보장 방법이다.

Hard real-time 보장 방법은 앞서 언급한 것처럼, 매우 다양한 요소들이 이벤트 처리시간 및 노드지연에 영향을 미치기 때문에 이들 요소들에 제약을 가하지 않으면 보장이 현실적으로 어렵다. 만일 영향을 미치는 요소들에 제약을 가한다면 운영체제의 범용성은 떨어질 수 밖에 없다. 더 큰 문제는 전체 센서네트워크 차원에서의 hard real-time 보장은 운영체제 자체보다는 사용하는 센서네트워크 프로토콜에 보다 종속적이라는 점이다. 즉, 특정 센서네트워크 프로토콜을 사용하면 hard real-time 보장을 할 수 있으나, 반대로 다른 센서네트워크 프로토콜을 사용하면 hard real-time 보장을 할 수 없게 된다.[5,6]

이와 같이 hard real-time을 보장하는 운영체제를 구현하는 데에는 현실적으로 많은 어려움이 있으며, 설사 구현이 가능하다고 하여도 상당한 제약조건이 따르거나 다른 기능과의 trade-off가 필요하다. 이는 센서네트워크에서 한 센서노드에게 요구되는 기능은 상당 부분 로컬 태스킹만으로 구현되는 것이 아니라, 이웃하는 센서노드들과의 협력과 네트워킹을 필요로 하는 태스크에 의하여 구현되기 때문이다. 감시정찰 센서네트워크용으로 개발되는 본 OS는 효율적인 태스크 스케줄링, 인터럽트 처리, 자원관리 등을 통하여 이벤트 처리시간을 줄일 뿐만 아니라 각 홉마다 네트워킹 관련 처리 시간을 최소로 하여 패킷의 전달 지연을 줄일 수 있도록 하는 soft real-time 보장만을 지원한다.

미들웨어 연구 동향

센서네트워크에서의 미들웨어는 센서 운영체제와 응용 프로그램 사이에 존재하여, 사용자 또는 상위 응용에게 인터페이스 형태의 다양한 기능과 하위 레벨의 추상화를 제공한다. 또한 일반적인 네트워크와는 다른 특징을 갖는 센서네트워크의 특성을 고려하여 응용 프로그램에 적합한 제어 및 관리 기술 등을 제공할 수 있어야 한다. 본 장에서는 센서노드 미들웨어 컴포넌트 중 위치인식, 시각동기 기술에 관한 동향을 기술한다.

위치인식

센서네트워크에서 센서노드의 위치를 아는 것은 매우 중요하며, 이를 통해 다양한 응용이 가능해진다. 위치 정보가 포함되지 않은 센싱 정보는 아무런 의미 없는 정보이며, 위치 정보가 결합되어야만 탐지 및 식별을 위한 target localization, target tracking 등을 할 수 있게 된다. 또한 네트워크에서의 효율적인 라우팅 알고리즘과 안정된 토폴로지 관리 알고리즘들은 대부분 노드의 위치 정보를 기반으로 한다.

위치인식 기법은 입력 데이터를 기준으로, 측정된 거리를 기반으로 하는 range-based 기법과 단순히 노

드의 연결 정보를 이용하는 range-free 기법으로 분류할 수 있다. Range-free 기법을 이용하는 대표적인 알고리즘으로는 DV-Hop[7]이 있다. DV-Hop에서는 각 노드가 레퍼런스 노드까지의 홉 수와 홉 간 평균 거리 정보를 이용하여 거리를 계산하고, 이 정보를 바탕으로 삼각측량을 통하여 자신의 위치를 결정하게 된다. DV-Hop을 포함한 range-free 기법의 경우, 연산이 간단하고 거리 측정 오차의 영향을 덜 받긴 하나 위치인식 오차는 다소 크다는 단점이 있다.

한편, range-based 기법에서는 위치인식을 위해 RSSI(수신 신호 세기) 또는 신호의 ToA(Time of Arrival)을 기반으로 한 거리 정보를 사용한다. 대표적인 것으로는 DV-Distance[7]가 있으며 DV-Hop과 다른 점은 홉 수 대신에 신호의 세기를 기반으로 측정한 거리 정보를 사용한다는 점이다. RSSI 또는 ToA는 전송 채널과 주위 환경에 따라 민감하게 변하여서 이를 거리로 환산하는 과정에서 어느 정도 오차가 발생하지만, range-based 기법이 range-free 기법에 비해 위치인식 정확도 면에서 약 2배 정도 우수하다.

시각동기

센서네트워크에서 각 노드들이 감지한 이벤트의 발생 순서 구분, 침입체의 이동 속도 계산 등에 있어서 시각 정보는 필수적이다. 또한, 제한된 에너지의 효율적인 사용을 위해 각 노드는 주기적으로 sleep 모드와 active 모드로의 전환을 반복하게 되는데, 이때 노드 간의 정확한 데이터 송수신을 위하여 시각 동기화는 기본적으로 이루어져있어야 한다.

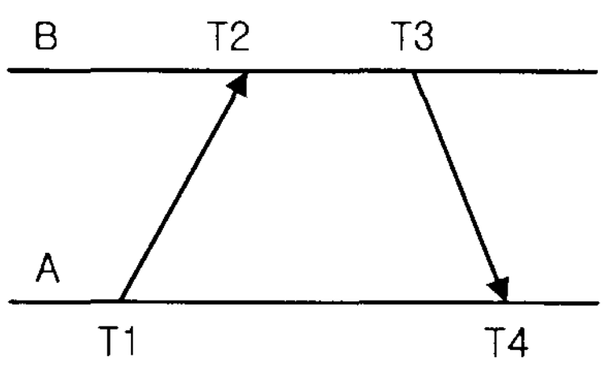

그림 1 – Time Stamp 메시지 교환

센서네트워크의 시각동기를 하는 데 있어서는 그림 1처럼 time stamp 메시지 교환을 통한 클럭차 보정이 기본적인 메커니즘으로 사용된다. 노드 A는 노드 B의 시각에 맞추기 위해, T1의 시각에 시각동기 요청 패킷을 노드 B에게 보낸다. 노드 B는 T2의 시각에 패킷을 받게 되고, T3의 시각에 ACK 패킷을 노드 A에게 전송한다. 이 ACK 패킷에는 T1, T2, T3의 값이 포함되어 있다. 노드 B는 T4의 시각에 ACK 패킷을 받으면 T1, T2, T3, T4를 이용하여 식 (1)과 같이 클럭차(Δ)를 계산하고 자신의 클럭을 보정한다.

$$\Delta = \frac{(T2 - T1) - (T4 - T3)}{2} \quad (1)$$

대표적인 시각동기 기법으로는 RBS[8]와 TPSN[9]이 있다. RBS는 receiver-receiver 간의 동기화 방식을 취하고 있으며, 레퍼런스 노드가 이웃 노드들에게 비컨 신호를 브로드캐스트하면 이를 수신한 노드들이 각각 수신한 시각 정보를 교환함으로써 서로 간의 시각을 동기화시킨다. 이 방식의 이점은 패킷 전달 과정에서의 지연 시간의 불확실성을 상당히 줄여 시각동기 에러를 감소시킨다는 점이다. 한편, TPSN은 sender-receiver 동기화 방식을 취하고 있으며 레벨 탐색 단계와 동기화 단계로 나누어 동작한다. 레벨 탐색 단계에서는 루트노드를 기준으로 하여 각 노드가 레벨을 할당받아 계층적 구조를 형성하고, 동기화 단계에서는 하위 레벨의 노드가 상위 레벨의 노드와 time stamp 메시지를 교환하여 클럭차를 계산하고 자신의 클럭을 보정하게 된다. 일반적으로 sender-receiver 방식은 receiver-receiver 방식에 비해 패킷 전달 과정에서 지연 시간의 불확실성이 크다. 이러한 불확실성을 줄이기 위해 TPSN에서는 패킷 송수신시 MAC 계층에서 시각 소인을 생성한다.

감시정찰용 내장소프트웨어

감시정찰을 위한 센서네트워크에서 모든 센서노드 및 싱크노드들은 한정된 자원과 저사양의 하드웨어로 동작한다. 또한 각 노드에 부착되어 있는 침입탐지 센서들이 수집한 상황 데이터를 신뢰성 있게 실시간으로 싱크노드 또는 서버에 전송할 수 있어야 한다. 본 장에서는 이러한 감시정찰 센서네트워크의 특성에 맞게 설계된 내장소프트웨어의 기능 및 구조에 대해 설명한다.

실시간 센서 OS 커널의 구조

본 OS는 기존의 multithread 방식에 deadline 개념이 도입된 선점형 스케줄링을 추가하여 기존 방식에 비하여 실시간 응용프로그램을 보다 효과적으로 지원하고, 다른 OS에 비해 더욱 정교한 전력관리 기능을 제공하여 센서노드의 생명주기를 연장하며, 효율적인 메모리관리 기능을 제공하여 자원제약이 있는 상황에서도 전체 센서네트워크가 정상적인 동작을 할 수 있도록 지원한다. 본 OS의 구조는 그림 2와 같다.

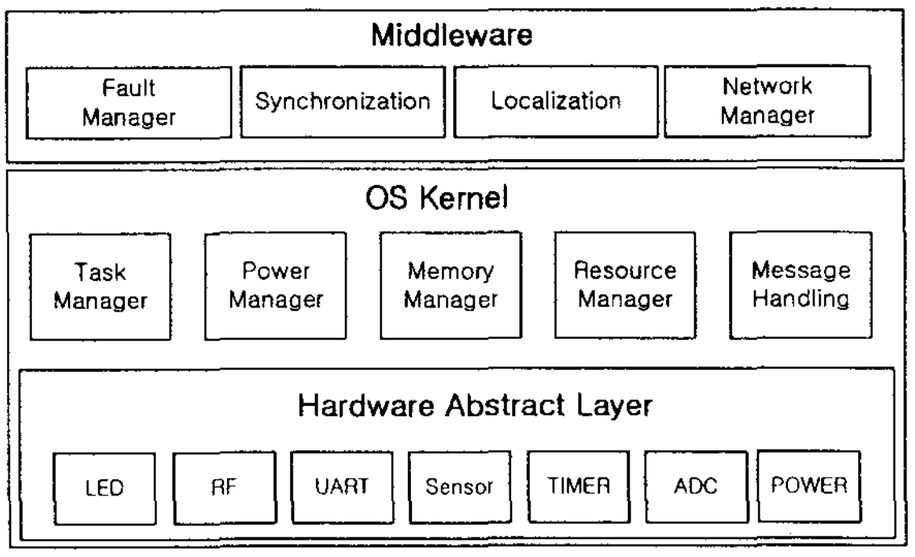

그림 2 – 내장소프트웨어 구조

태스크 매니저

본 OS가 지원하는 동시 실행 태스크의 수는 최대 10개이며, 그 중 실시간 처리를 요하는 태스크들은 네트워크관련 태스크로 동기화, 이벤트형 패킷 전달 등으로 많고 대부분 비실시간 태스크가 많을 것으로 예상된다. 때문에 실시간 태스크는 time sensitive task queue에서 deadline으로 우선권에 차등을 두어 관리하며, 나머지 비실시간 태스크는 time insensitive task와 system task queue를 통해 FIFO로 관리한다. 또한, 다수의 태스크가 동시에 수행되므로, 태스크간 통신 및 동기화 기능을 제공하여야 한다. 본 OS의 경우, 태스크간 통신은 공유 메시지 큐를 이용한 message passing 기법을 사용하였으며, 동기화를 위해 counting semaphore를 제공한다.

본 OS는 태스크에게 sleep(t)을 제공하여 설정 시간(t ms) 후에 sleep 상태에서 깨어날 수 있도록 한다. 개발 OS는 설정한 슬립구간을 정확하게 구현하기 위해 슬립구간 설정을 위한 별도의 타이머가 존재한다. 때문에 정확한 주기를 구현할 수 있으며, 감시형 센서 네트워크 어플리케이션 개발에 효과적인 환경을 제공할 수 있다.

또한 본 OS는 보다 나은 실시간성을 제공하기 위하여, 인터럽트 발생시 현재 수행중인 태스크와 실행우선권을 비교하여 현재 수행중인 태스크의 실행이 우선시되면 인터럽트 처리를 나중으로 미루도록 설계되어, 인터럽트로 인한 수행중인 태스크의 실행시간 지연을 최소화 하였다.

전력관리 매니저

저전력 소비를 위하여, 본 OS 역시 기존의 센서네트워크용 OS와 동일하게 실행할 태스크가 없을 경우에 노드 전체가 슬립 모드로 전환하도록 하였으며, MCU와 RF 모듈의 동작모드를 사용자가 제어할 수 있도록 다양한 인터페이스를 제공한다. 또한, 한 태스크가 RF 모듈을 껐을 경우, 문맥전환 이후 다른 태스크는 RF 모듈을 사용할 수 없게 된다. 다시 사용하기 위해서는 RF 모듈을 켜야 하며, 이에 따른 실행시간의 지연이 발생하게 된다. 이러한 현상을 방지하기 위하여, 본 OS는 각각의 태스크가 사용하는 디바이스의 전력 레벨을 가장 높은 전력 레벨로 동일하게 유지하도록 하였다.

디바이스 드라이버

OS는 사용자가 하드웨어의 세부적인 지식이 없더라도 노드의 하드웨어들을 제어할 수 있도록 인터페이스를 제공해야 한다. 이를 위해 본 OS는 LED, ADC, UART, SPI, I2C, RF 모듈, 타이머, 전원 등을 제어하도록 하는 디바이스 드라이버를 제공한다.

메모리 매니저

본 OS는 기존의 OS들과 마찬가지로 힙 메모리를 동적으로 할당해주는 기능을 제공하고 있으며, 할당한 힙 메모리의 오버플로우를 감지하는 기능을 제공하고 있다.

센서미들웨어 기능

감시정찰 응용을 위해 본 센서미들웨어는 위치인식, 시각동기 기술뿐만 아니라 네트워크 관리, 원격 업데이트 기능을 제공한다.

위치인식

감시정찰 응용이 요구하는 위치인식 정밀도는 수 m 정도이다. 이러한 위치인식 수준을 내기 위해서는 range-based 기법을 사용하여야 하며, 모든 노드 간 연결 정보를 이용하여 중앙집중 형태로 위치 계산이 이루어져야 한다. 이러한 집중형 기법의 경우 많은 통신량이 발생한다는 단점이 있으나, 싱크노드 또는 서버를 통해 복잡한 연산을 수행함으로써 보다 정확한 위치를 구할 수 있게 된다. 또한 응용 특성상 노드가 이동할 가능성은 극히 적고, 노드가 뿌려지고 난 후 초기 단계에서 위치인식이 한 번 수행되면 충분하므로 위치인식에 따른 통신 오버헤드는 크게 고려하지 않아도 된다.

위치인식은 데이터를 모으는 단계와 위치정보를 계산 및 보정하는 단계로 이루어진다. 데이터를 모으는 단계에서는 각 노드가 이웃노드를 찾고 이웃노드와 서로 비컨 신호를 주고받아 신호의 세기(RSSI)를 통해 각 이웃노드간 거리 정보를 유지한다. 이 1-홉 정보는 클러스터를 통해 싱크노드에 전달되고 싱크노드에 연결된 서버에서는 shortest path 알고리즘을 수행하여 1-홉 노드 간 거리 정보뿐만 아니라 n 개의 노드에 대해 모든 노드 간 거리 정보를 포함한 n*n 행렬을 생성시킨다. 이후, 위치정보를 계산 및 보정하는 단계에서는 이 행렬을 이용하여 위치계산 알고리즘을 적용시키면 2차원 혹은 3차원의 상대좌표를 가진 맵을 그릴 수 있으며, 3개 이상의 앵커노드 좌표가 정해지면 좌표를 보정하여 절대좌표를 가진 맵을 얻을 수 있게 된다. 여기서 위치계산 알고리즘으로는 통신과 연산에 따른 오버헤드, 연산 시간, 요구되는 정확도 등을 고려하여 MDS[10], MLE 등을 적용할 수 있다.

시각동기

앞에서 언급한 바와 같이 시각동기는 이벤트의 발생 순서 구분, 침입체의 이동 속도 계산, 효율적인 RF 통신 등을 위해 필요하며, 이를 위해 감시정찰 센서네트워크에서의 시각동기는 수백 ms 이하 수준으로 이루어져야 한다. 감시정찰 및 추적 대상이 사람, 탱크, 궤도차 등으로 그리 빠르게 이동하지 않는 물체이므로 그리 높은 수준의 시각동기를 유지해야 할 필요는 없다. 정밀한 수준의 동기를 위해서는 그만큼 시각동기 이벤트가 빈번이 수행되어야 하고 이는 통신 오버헤드를 증가시키기 때문이다.

시각동기 이벤트는 서버에 의해 검출되어 절대적인 클럭을 갖고 있는 노드들(GPS를 탑재한 클러스터 헤드 또는 싱크노드)에게 전달되며, 이들 노드가 루트노드가 되어 네트워크의 시각동기화가 수행된다. 시각동기 이벤트의 발생 주기는 식 (2)에 의해 지정

될 수 있다. 이 식에서 τ는 동기주기로 단위가 초(sec)이며, α는 요구되는 동기 정확도로서 단위는 마찬가지로 초(sec)이다. ρ는 센서노드의 최대 클럭 표류율이며 단위는 ppm이다.

$$\tau \leq \frac{\alpha \cdot 10^6}{\rho} \quad (2)$$

시각동기화 명령이 클러스터 헤드 또는 싱크노드에 전달되면 RBS나 FTSP, 또는 TPSN 알고리즘이 적용되어 시각동기화가 수행될 수 있다. 시각동기 알고리즘 역시 통신 오버헤드, 전체 동기화가 이루어지는 데 있어서의 시간 지연 등을 고려하여 선택되어야 하며, 이러한 수치를 최소화할 수 있는 효율적 시각동기 알고리즘에 대한 계속적인 연구가 필요하다.

원격 업데이트

감시정찰 센서네트워크에서 각 센서노드는 한번 뿌려지면 사람의 물리적 개입을 받지 않고 정해진 목적을 수행하도록 동작하여야 한다. 이는 센서네트워크 특성상 감시정찰을 위해 센서노드가 뿌려진 장소는 사람의 접근이 용이하지 않은 장소일 수도 있으며, 설치된 센서노드의 개수는 수백에서 많게는 수천 개에 달하기에 설치된 각 노드에 사람이 물리적으로 개입한다는 것은 현실적으로 불가능하기 때문이다. 이런 점에서 만약 이미 설치된 센서노드에, 새로운 기능이 추가되거나 또는 버그를 수정한 새로운 커널 및 응용 S/W 프로그램이 업로드 되어야 한다면 이는 무선 네트워크를 통해 원격 업데이트 되어야 할 것이다. 따라서 본 센서미들웨어는 장치가 실행되는 동안에도 커널 및 응용 S/W를 원격으로 메모리에 로딩하는 기술을 지원하도록 한다.

센서네트워크에서의 대표적인 원격 업데이트 방식에는 XNP[11], MNP[12], MOAP[13] 등이 있으며 이들은 전체 코드 이미지 전송을 통하여 업데이트를 수행한다. 그러나 제한된 에너지로 구동하는 센서노드 특성상 외부 메모리에 접근하거나 대량의 패킷을 전송하는 것은 큰 오버헤드가 아닐 수 없다. 따라서 원격 업데이트를 수행하는 과정에서 전달하는 패킷의 양이 적을수록, 그리고 외부 메모리에 대한 접근 횟수가 적을수록 에너지 소모를 줄일 수 있다. 이는 기존 프로그램의 이미지와 새로운 프로그램 이미지와의 차이(difference)에 해당하는 부분을 패킷화하여 전달하는 incremental 기법을 통해서 달성할 수 있으며, 대표적인 것으로는 [14]과 [15]이 있다. 그러나 이러한 incremental 기법은 코드 형식이 프로세서에 종속적이기 쉬우며 또한 코드에서의 각 함수 별로 여분의 메모리 공간을 두어야 한다는 단점이 있다. 따라서 원격 업데이트의 발생 빈도와 원격 업데이트 시의 에너지 소모 등을 고려하여 적절한 업데이트 방식을 취해야 한다.

OS 커널 성능

본 OS의 코드 크기는 9Kbytes 이며, 데이터 사이즈는 1074bytes이다. 그에 대한 내용은 표 1에 나와 있다. 표 2에는 현재까지 개발된 센서 노드 OS와의 특징을 비교하였다.

표 1-감시정찰 센서 OS 코드 크기

	코드 크기 (KB)	데이터 크기 (Bytes)
Task Manager	3	321
Power Manager	1	56
Resource Manager	2	431
RF Manager	3	266
Total	9	1074

표 2- 센서네트워크 OS 별 기능 비교

특징	TinyOS	SOS	MANTIS	NanoOS	본 OS
저전력 모드	○	○		○	○
선점형 multithread 스케줄링			○	○	○
실시간 태스크 지원					○
동적 재프로그래밍	○	○			○

결론

감시정찰용 센서네트워크는 앞으로 군 또는 민수 분야에 다양하게 적용될 수 있는 기술로 각광받고 있다. 본 논문에서는 감시정찰 센서네트워크를 위한 고신뢰 초소형 내장소프트웨어의 구조 및 기능에 대해 살펴보았고, 또한 이를 구현하면서 느낀 디자인 이슈 등에 대해서도 제시하였다. 센서 OS는 선점형 multithread 스케줄링을 사용하여 실시간 응용프로그램을 보다 효과적으로 지원하고, 더욱 정교한 전력 관리 기능을 제공하여 센서노드의 수명을 연장시키며, 효율적인 메모리관리 기능을 제공하여 자원 제약의 영향을 덜 받도록 한다. 또한 센서미들웨어는 위치인식, 시각동기, 원격 업데이트 기능 등을 제공하여 감시정찰 응용에서 공통적으로 요구하는 사항을 충분히 지원할 수 있도록 한다.

References

[1] Levis, P., Madden, S., Gay, D., Polastre, J., Szewczyk, R., Woo, A., Brewer, E., and Culler, D. (2004). "The emergence of networking abstractions and techniques in tinyos," *Proceedings of the First Symposium on Networked Systems Design and Implementation*, pp. 1-14.

[2] Han, C., Kumar, R., Shea, R., Kohler, E., and Srivastava, M. (2005). "A Dynamic Operating System for Sensor Networks," *Proceedings of the 3rd International Conference on Mobile Systems, Application, and Services*.

[3] Shin, Y., Lee, K., Choi H., and Park S. (2005). "A Design and Implementation of a Multi-hop Wireless Sensor Network based on Nano-Qplus Platform," *Proceedings of the 20th International Technical Conference on Circuits/Systems, Computers and Communication*.

[4] Bhatti, S., Carlson, J., Dai, H., Deng, J., Rose, J., Sheth, A., Shucker, B., Gruenwald, C., Torgerson, A., and Han, R. (2005). "Mantis os: An embedded multithreaded operating system for wireless micro sensor platforms," *ACM Kluwer Mobile Networks and Applications (MONET) Journal, Special Issue on Wireless Sensor Networks*.

[5] Laplante, P. A. (2004). *Real-Time Systems Design and Analysis*, Third Edition. Wiley.

[6] Li, Q., and Yao, C. (2003) *Real-Time Concepts for Embedded Systems*, CMP Books.

[7] Niculescu, D., and Nath, B. (2003). "DV based positioning in ad hoc networks," *Journal of Telecommunication Systems*, Vol. 22, pp. 267-280.

[8] Elson, J., Girod, L., Estrin, D. (2002). "Fine-grained network time synchronization using reference broadcasts," *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation.*

[9] Ganeriwal, S., Kumar, R., Srivastava, M. B. (2003). "Timing-sync protocol for sensor networks," *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems.*

[10] Shang, Y., Ruml, W., and Zhang, Y. (2003). "Localization from mere connectiviey," *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 201-212.

[11] Jeong, J., Kim, S., and Broad. A. (2003). "Network Reprogramming," TinyOs Document.

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

[13] Stathopoulos, T., Heidemann, J., and Estrin, D. (2003). "A Remote Code Update Mechanism for Wireless Sensor Networks," CENS Technical Report.

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

[15] Jeong, J., and Culler, D. (2004). "Incremental Network Programming for Wireless Sensors," *Proceedings of the 1st IEEE International Conference on Sensor and Ad Hoc Communications and Networks.*