SenOS : 동적 센서 노드 재구성을 위한 상태 기반 운영 체제 구조

홍성수* 김태형** 한승현*O 박선희***

*서울대학교 전기·컴퓨터공학부 **한양대학교 컴퓨터공학과 ***서울대학교 자동화시스템공동연구소
{sshong,shhan^O}@redwood.snu.ac.kr tkim@csehanyang.ac.kr shpark@redwood.snu.ac.kr

SenOS: State-driven Operating System Architecture for Dynamic Sensor Node Reconfigurability

Seongsoo Hong*, Tae-Hyung Kim**, Seung-Hyun Han*O, Sunheui Park***

*School of Electrical Engineering and Computer Science, Seoul National University
**Department of Computer Science and Engineering, Hanyang University
Automation and Systems Research Institute (ASRI), Seoul National University

요 약

무선 센서 네트웍은 동작 환경과 구조가 매우 특이하여 개발자들은 센서 네트웍의 노드를 디자인할 때 많은 제약 조건들과 요구 조건들을 고려해야 한다. 먼저 무선 센서 네트웍 상의 각 센서 노드에는 극도로 제한된 하드웨어 자원 조건 하에서도 무선 통신 기능뿐만 아니라 동시에 여러 이벤트를 재빠르게 처리할 수 있는 기능이 포함되어야 한다. 또한 환경과 응용 프로그램의 변화에 잘 대처하기 위해 런타임(run-time)에 각 센서 노드들을 동적으로 재구성할 수 있는 기능이 제공되어야 한다. 이러한 디자인 요구 조건들과 제약 조건들은 얼핏 서로 상반된 것처럼 보이는데, 무선 센서 노드들을 위한 실행 환경을 디자인할 때는 이러한 조건들을 모두 만족시킬 수 있는 운영 체제가 반드시 필요하다. 본 논문에서 우리는 무선 센서 노드들을 위한 매우 효율적이고 효과적인 유한 상태 머신(finite state machine) 기반의 운영체제, SenOS를 제안한다. 또한 새로운 운영 체제인 SenOS가 극도의 제한적인 자원에서도 동시성과 반응성, 재구성성의 요구 조건을 모두 만족시키면서 동작할 수 있다는 것을 보일 것이다.

1. 서 론

무선 센서 네트웍은 유비쿼터스 컴퓨팅에서 하나의 핵심 기술로 떠오르고 있는 분야이다. 작은 범위에서 동작하는 RF 통신 매체를 통해 연결된 무선 지능 센서들은 유비쿼터스 컴퓨팅 환경에서 사람과 컴퓨터 사이에 중개인으로써의 역할을 수행한다. 기존의 컴퓨팅 플랫폼과 달리 무선 센서 노드들은 다음과 같은 세 가지 특징을 가진다. (1) 컴퓨팅 파워와 메모리, 배터리 등의 모든 자원이 극도로 제한적이다. (2) 센서 네트웍을 수만개의 작은 자율적인 노드들로 구성된 분산 컴퓨팅 플랫폼으로 보는 데이터 중심형(data-centric) 프로그래밍 스타일이 일반적이다. (3) 재사용을 고려하지 않는 일회용 컴퓨팅 플랫폼이다. 네트웍 센서 노드들의 이러한 특징 때문에 극도의 저전력 경량 장치에서 동작할 수 있으면서 동시에 환경과 응용 프로그램의 변화에 대처하기 위해 동적인 재구성을 지원할 수 있는 독특한 구조의 운영 체제가 필요하다. 또한 이런 운영 체제는 동시에 여러 개의 비동기적인 이벤트를 다룰 수 있어야 하며 미들웨어를 통한 분산형, 데이터 중심형 프로그래밍 모델을 지원해야 한다. 사실 이렇게 상반된 것처럼 보이는 요구 조건을 모두 만족시키는 운영 체제를 디자인하고 구현하는 것은 매우 어려운 일이다.

하지만 다행히 센서 네트웍 노드의 응용 프로그램들은 연속된 일련의 동작들을 수행하거나 프로그램의 모드에 따라 입력 이벤트를 다르게 처리하는 경향이 있기 때문에 상태 머신(State Machine) 기반의 컴퓨팅 모델에 적합하다. 상태 머신 기반 컴퓨팅 모델은 다음과 같은 많은 장점을 가지고 있다. (1) 프로그래머가 쉽게 디자인 모델을 개발할 수 있으며 일반적으로 널리 쓰이는 코드 생성 툴들을 이용하여 디자인 모델을 자동으로 실행 코드로 변환할 수 있다. (2) 동시에 여러 입력 이벤트들을 다룰 수 있으며 반응이 매우 빠르다. (3) 프로그램이 선점되는 과정에서 저장되고 복구되는 상태 정보가

명확하기 때문에 런타임 시스템은 효율적으로 프로그램을 정지시키거나 복구시킬 수 있다. 결과적으로 상태 머신 기반의 프로그램은 실제로 구현되었을 때 매우 간결하면서도 효율적인 실행 코드로 변환될 수 있다.

본 논문에서 우리는 센서 네트웍 노드에 적합한 이상적인 운영 체제를 디자인하기 위해 유한 상태 머신 기반의 실행 모델을 이용한다. 그리고 결과물로써 SenOS라 이름 붙인 새로운 운영 체제를 제안한다. SenOS는 동시성과 반응성, 재구성성의 요구 조건을 모두 만족시키면서 동시에 코드의 크기와 전력 소모라는 제한 조건을 만족시키도록 디자인되었다. 본 논문의 나머지 부분에서는 SenOS의 구조에 대해 기술하고 다양한 요구 조건들이 어떻게 SenOS에 반영되었는지에 대해 살펴 본다.

2. 유한 상태 머신 기반의 프로그램 실행 환경

유한 상태 머신은 불연속적인 입력과 출력을 가지는 시스템의 수학적 모델이다. 상태 머신은 유한 개의 내부 상태들 중 하나의 상태를 가질 수 있으며 이는 초기 상태로부터 입력된 이벤트들에 의해 결정된다. 보다 자세히 살펴 보면 유한 상태 머신은 (1) 유한 개의 상태 집합, (2) 유한 개의 입력 집합, (3) 유한 개의 출력 집합, (4) 현재 상태와 입력으로부터 다음 상태를 구해내는 상태 전이 함수, (5) 현재 상태와 입력으로부터 현재 출력을 구해내는 출력 함수, (6) 초기 상태로 구성된다. 이러한 구성은 출력이 단지 현재 상태의 함수가 되는 무어 머신(Moore machine)이 아닌, 출력이 현재 상태와 입력의 함수가 되는 밀리 머신(Mealy machine)의 경우를 묘사한 것이다.

유한 상태 머신에서 유효한 입력 이벤트는 머신을 현재 상태에서 다음 상태로 옮기는 상태 전이와 출력을 발생시킨다. 이러한 상태 전이는 즉각적으로 일어나며 전이된 상태와 연관된 출력 함수가 발동된다. 이런 실행 매커니즘을 이용하여, 유한 상태 머신은 연속된 일련의 동작들을 수행하거나 머신의 상태에 따라 입력 이벤트를 다르게 처리한다.

유한 상태 머신을 구현하기 위해서는 다음과 같은 네 가지 구성 요소가 필요하다. (1) 이벤트 큐로부터 입력을 받아들이는 상태 정렬기(state sequencer)와 (2) FIFO 방식으로 동작하며 입력을 저장하는 이벤트 큐, (3) 출력 함수들을 담고 있는 콜백(callback) 함수 라이브러리, (4) 각 유효한 상태 전이와 연관된 콜백 함수들을 정의하고 있는 상태 전이 테이블이다. 각 콜백 함수는 순간적인 상태 전이 체제를 유지하기 위해 " run-to-completion" 조건을 만족시켜야 한다.

많은 임베디드 실시간 시스템들이 유한 상태 머신으로 구현하기 적당하며 이는 센서 네트웍 응용 프로그램들의 경우에도 마찬가지이다. 시중에는 개발자들이 쉽게 상태 머신 기반 시스템 모델을 디자인할 수 있도록 도와주고 개발한 디자인을 자동으로 실행 코드로 합성해주는 CASE 툴들이 몇 개 있다. UML-RT는 이러한 툴 중 하나이며 임베디드 산업에서 널리 쓰이고 있다[4]. SenOS 응용 프로그래머들은 이러한 툴들을 이용하여 보다 쉽게 센서 네트웍 노드에서 동작하는 응용 프로그램을 개발할 수 있다.

3. 상태 기반 운영 체제 구조

SenOS 커널의 구조는 유한 상태 머신 모델에 기반을 두고 있다. SenOS 커널은 다음과 같이 세 가지 부분으로 구성된다. (1) 상태 정렬기와 이벤트 큐로 구성된 핵심 커널부, (2) 상태 전이 테이블, (3) 콜백 라이브러리. 다음 그림 1은 SenOS 커널의 구성을 잘 보여주고 있다.

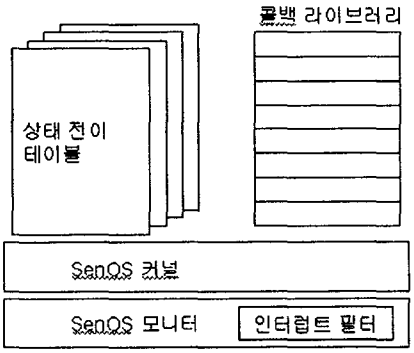

그림 1. 상태 기반 운영 체제 구조

커널은 무한히 이벤트 큐에 이벤트가 도착하기를 기다린다. 만약 하나 이상의 입력 이벤트가 큐에 들어와 있다면 커널은 큐의 최상단에 있는 입력 이벤트를 빼내어 입력이 유효할 경우 현재 상태를 전이시키고 관련된 출력 함수를 호출한다. 이렇게 동작하기 위해 커널은 상태 머신의 현재 상태 정보를 유지하고 있어야 하며 콜백

함수가 선점되지 않고 실행을 마칠 수 있도록 뮤텍스(mutex)로 함수를 보호해야 한다.

콜백 라이브러리는 미리 정의된 빌트인 함수들의 집합으로, 응용 프로그래머가 센서 노드의 기능을 결정할 때 사용한다. 상태 전이 테이블이 런타임에 리로드되거나 수정될 수 있는 반면에 커널과 콜백 함수 라이브러리는 정적으로 빌드되어 센서 노드의 플래쉬롬에 저장된다. SenOS는 여러 개의 상태 전이 테이블을 동시에 유지함으로써 여러 개의 응용 프로그램을 지원할 수 있으며 상태 전이 테이블을 교체하는 방법으로 여러 개의 응용 프로그램을 동시에 수행 시킬 수 있다. 이 때 하나의 상태 전이 테이블은 한 개의 응용 프로그램을 의미한다. 응용 프로그램이 선점될 때 커널은 선점되는 응용 프로그램의 현재 상태 정보를 저장하고 선점하는 응용 프로그램의 상태 정보를 복구시킨다. 그리고 상태 전이 테이블을 변경함으로써 프로그램 교체를 완료한다.

SenOS에는 또한 동적으로 응용 프로그램을 로딩해주는 런타임 모니터(monitor)가 포함되어 있다. 그림 1에서도 볼 수 있듯이, 사실상 모든 외부 인터럽트에 대해 호출되는 일반적인 인터럽트 서비스 핸들러인 인터럽트 필터가 SenOS의 런타임 모니터에 내장되어 있다. SenOS가 인터럽트를 통해 통신 어뎁터로부터 응용 프로그램 리로드 메시지를 받았을 때, 모니터는 커널을 안전한 상태에 두고 정지시킨 후 새로운 상태 전이 테이블을 리로드한다. 이 경우 모니터는 커널이 상태 전이 상태에 있지 않는 한 언제라도 커널을 인터럽트 시킬 수 있다. 상태 전이가 뮤텍스에 의해 보호되기 때문에 유한 상태 머신은 이러한 인터럽트에도 불구하고 안전성을 유지할 수 있다.

이상을 통해 우리는 SenOS가 동시성, 반응성, 런타임 재구성성 등의 요구조건을 확실히 만족시킬 수 있다는 것을 알 수 있다.

4. 결 론

우리는 지금까지 센서 네트웍 노드를 위한 상태 기반 운영 체제인 SenOS에 대해 알아보았다. SenOS는 핵심 커널부와 런타임 모니터, 교체 가능한 상태 전이 테이블들, 그리고 콜백 라이브러리들로 구성된다. 프로그래머들은 손쉽게 SenOS 응용 프로그램을 디자인할 수 있으며 SenOS 디자인 툴을 이용하여 자동으로 실행 코드로 변환하고 모니터를 이용하여 런타임에 응용 프로그램의 실행 코드를 동적으로 로딩시킬 수 있다. SenOS는 많은 장점들을 제공하는데, 우선 상태 머신 모델 기반이라 구현이 간결하고 효율적이라는 장점이 있다. 또한 교체 가능한 상태 전이 테이블과 콜백 라이브러리를 이용하여 매우 효과적인 동적 노드 재구성을 지원한다는 장점이 있다. 이러한 장점으로 인해 SenOS는 센서 네트웍 노드에 가장 적합한 운영 체제가 될 수 있을 것이다. 현재 구현 중에 있으며 결과가 매우 기대된다.

참고문헌

[1] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, System Architecture Di-rections for Networked Sensors. International Conference on Architectural Support for Programming Languages and Operating Systems (2000).

[2] P. Levis and D. Culler, Mate: A Tiny Virtual Machine for Sensor Networks. International Conference on Architectural Support for Programming Languages and Operating Systems (2002).

[3] J. Hopcroft and J. Ullman, Introduction to Automata Theory, Languages, and Computa-tion. Addison Wesley (1979).

[4] IBM (former Rational Software), http://www.rational.com.