WBAN 환경의 센서노드에서 최소 수행시간을 보장하기 위한 파티셔닝 기법

김대훈○ 조진성
경희대학교 컴퓨터공학과
gogohhcom@khu.ac.kr chojs@khu.ac.kr

A Partitioning Scheme to Guarantee Minimum Running Time in Sensor Node of WBAN Environment

Daehun Kim○ Jinsung Cho
Department of Computer Engineering
KyungHee University

요 약

최근 임베디드 장비의 사용이 보편화되고, 무선통신기술이 발전됨에 따라 센서 장비의 소형화, 저전력이 가능하게 되었다. 이러한 센서 장비의 성능이 향상됨에 따라, 다수의 소프트웨어가 하나의 하드웨어에서 동작할 수 있게 되었으며, 다수의 소프트웨어들의 독립성을 보장하고 안전성을 제공하기 위해 파티셔닝이라는 개념이 등장하였다. 파티셔닝은 특정 소프트웨어 및 프로그램을 시간과 공간을 기준으로 파티션으로 나누는 것을 의미하며, 이를 각각 시간 분할 (temporal partitioning)과 공간 분할 (spatial partitioning)이라 한다. 한편, 최근 활발히 연구되고 있는 WBAN (Wireless Body Area Network)은 초저전력을 요구하므로 매우 제한적인 하드웨어 성능을 요구한다. 이러한 WBAN 환경에서는 기존의 파티셔닝 기법을 적용하기에 알맞지 않다. 따라서, 본 논문에서는 제한된 하드웨어 성능을 기반으로 한 WBAN 환경의 센서노드에서 수행시간을 보장하기 위한 파티셔닝 기법을 설계하고 제안한다.

1. 서 론

최근 항공기, 에어컨, TV, 자동차 등 임베디드 장비가 늘어남에 따라 이를 제어하기 위한 소프트웨어 역시 급속도로 증가하였으며, 다수의 소프트웨어들은 하나의 하드웨어에서 수행되는 일이 많아졌다. 이렇듯 다수의 소프트웨어가 하나의 하드웨어에서 통합적으로 수행되는 경우에는 각 소프트웨어들의 신뢰성을 보장해야 하며, 이를 위해서 소프트웨어간의 독립성에 대한 중요성이 강조되었다. 예를 들어 항공기의 경우, 방향을 결정하는 프로그램, 엔진 동작횟수를 결정하는 프로그램 등 여러 프로그램이 항공기라는 하나의 장비에서 통합되어 수행된다. 이 프로그램들은 모두 정상적으로 수행이 되어야, 항공기가 정상적으로 비행을 할 수 있게 된다. 그러나 엔진 동작횟수를 결정하는 프로그램의 우선순위가 높고 작업량이 많아짐에 따라 방향을 결정하는 프로그램에 영향을 주게 된다면, 이는 전체적인 시스템 자체에 위협을 가할 수 있게 된다. 이러한 문제점을 해결하기 위해, 각각의 일을 담당하는 소프트웨어들의 실시간 요구사항과 독립성을 보장할 수 있는 파티셔닝 개념이 제안되었다.

파티셔닝 기법은 고성능의 하드웨어를 가정하여, 단일 하드웨어에서 동작하는 다수의 소프트웨어 및 프로그램을 파티션으로 나누어서 시간분할과 공간분할을 수행함으로써, 서로 다른 프로그램들이 파티션을 나누어 물리적 자원과 프로세스 자원의 독립성을 제공한다.[1]

한편, 무선 기기의 발달은 의료용 서비스를 제공해 주는 목적으로 u-Healthcare, u-Lifecare에 대한 연구의 기폭제가 되고 있다. 이러한 의료용 서비스를 제공하는 것을 목적으로 IEEE 802.15.6은 WBAN (Wireless Body Area Network)의 표준화을 2007년부터 시작하여 2012년 2월에 완료하였다. WBAN을 구성하는 센서 장비는 매우 제한적인 하드웨어 성능을 가지고 있으며, 다양한 응용 제공을 요구한다. 하지만 WBAN을 구성하는 센서노드는 하드웨어의 성능이 뛰어나지 않고, CPU에서 메모리 프로텍션 기능도 지원하지 않아, 지금까지의 파티셔닝 기법을 적용하기 어렵다. 따라서, 본 논문에서는 WBAN을 구성하는 센서장비와 같이 하드웨어 성능이 제한된 상황에서 각 소프트웨어의 최소 수행시간을 보장하기 위한 파티셔닝을 설계한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 파티셔닝 기법에 대한 배경지식 및 관련연구를 소개한다. 3장에서는 WBAN 환경의 센서노드에 적용할 수 있는 파티셔닝 기법에 대한 설계를 소개한다. 이어지는 4장에서는 본 논문의 결론과 향후 계획에 대하여 기술한다.

2. 배경지식 및 관련연구

"이 논문은 미래창조과학부 및 정보통신산업진흥원의 대학 IT 연구센터 지원사업(NIPA-2013-(H0301-13-2001)) 및 교육과학기술부 및 한국과학재단의 중견연구자 사업(No. 2011-0015744)의 지원으로 수행된 연구결과임"

2.1 배경지식

2.1.1 파티셔닝

파티셔닝이란 프로그램들을 파티션으로 나누어 독립적인 공간을 제공하는 개념이다. 파티셔닝은 공간분할, 시간분할 두 가지의 개념으로 나눌 수 있다. 공간분할이란 한 파티션의 물리적 메모리가 다른 파티션에 영향을 미치지 않는 것을 의미하고, 시간분할이란 한 파티션의 프로세스 자원이 다른 파티션에 영향을 미치지 않는 것을 의미한다. 이와 같은 개념은 ARINC 653 표준 [2]을 사용하고 있는 항공기, 자동차와 같은 임베디드 시스템에 사용되고 있다.

2.1.2 ARINC 653

항공 전자 시스템은 상이한 임무를 수행하는 다양한 전자 장치들로 이루어지며, 전자 장치들은 점차 통합 구조 시스템(IMA, Integrated Modular Avionics)으로 구성되고 있다. 통합 구조 시스템은 전자 장치의 다양한 종류와 무거운 중량을 이유로 단일 컴퓨터 환경에서 구성된다. 이와 같은 통합구조에서 갖는 응용프로그램들의 특성을 고려하여 시·공간적으로 분리된 파티션으로 구분하는 ARINC 653과 같은 표준이 등장하였고, 가상화 기반의 파티셔닝을 적용한 다양한 연구가 진행 되었다[2-4].

2.1.3 KHIX

경희대학교에서 개발한 OS인 KHIX는 확장 및 재구성이 가능하고 POSIX 기반의 API를 제공하는 의료용 센서 OS다[5]. 센서네트워크의 임베디드 시스템은 CPU성능, 메모리, 소비 전력 등 제약사항이 많기 때문에 시스템마다 목적에 맞는 구성을 필요로 한다. 그림 1은 KHIX의 구성도를 보여주고 있다. 각 기능들은 컴포넌트화 하여 확장에 용이하도록 하였고, HAL(Hardware Abstract Layer)을 통해 다른 프로세서에 이식하기 용이한 구조로 되어 있다. 또한 POSIX 기반의 API를 제공하여 응용 프로그램 작성에도 용이하도록 하였다.

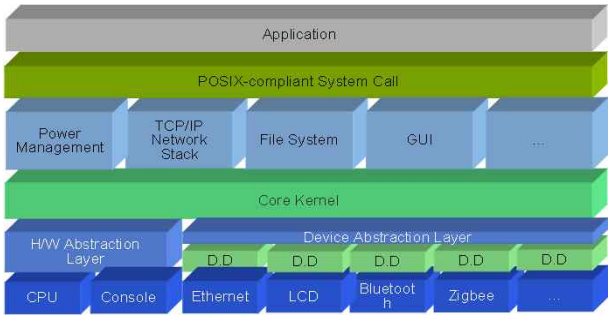

그림 1 KHIX OS 구성도

2.2 관련연구

파티셔닝 기법은 하나의 하드웨어에서 수행되는 여러 소프트웨어의 독립성이 중요해짐에 따라, 많은 연구가 진행되었다.

차량용 인포테인먼트 시스템은 자동차 안에서 즐길 수 있는 엔터테인먼트 및 정보 시스템을 총칭한다. 이 시스템은 최근 소프트웨어에 대한 요구사항이 급격하게 증가함에 따라, 다양한 소프트웨어 플랫폼 개발이 활발히 진행되고 있다. 그 대표적인 예로 비가상화 파티션 두 개와 가상화 파티션 한 개로 나누어 시스템을 설계 및 구현한 방안이 제안되었다.[6] 제안된 방안은 서로 다른 플랫폼이 각 파티션에서 수행시킴으로써 성능평가를 수행하였으며, 그 결과 한 파티션의 과부하가 다른 파티션에 영향을 주지 않는다는 것을 증명하였다.

COS는 DECOS 프로젝트[7] 의 CORE OS이다. COS는 시간분할과 공간분할을 구현하여 완성하였다. 시간분할의 경우 두 레벨의 스케쥴링을 통해 구현하였다. 첫 번째 레벨의 스케쥴러는 여러 개의 타임 슬롯으로 스케쥴 사이클을 나누고, 이를 각 파티션마다 할당하였다. 이는 파티션을 분할할 때, 주어진 가중치에 따라 고정으로 스케쥴링 작업이 정해진다. 두 번째 레벨의 스케쥴러는 각 파티션의 이벤트나 이에 해당하는 이벤트 핸들러가 사용된다. 이는 각 파티션의 작업량에 따라, 동적으로 스케쥴링 작업이 진행된다. 공간분할의 경우는 CPU가 메모리에 접근하는 것을 관리하는 MMU(Memory Management Unit)라는 하드웨어를 사용하여 메모리 프로텍션을 이루어 냈다. 이 두 가지 파티셔닝 기법을 통하여 모든 자원을 분할하여 사용할 수 있는 기법을 제안하고 구현하였다.

하지만 기존의 연구들은 메모리 프로텍션 기능을 지원하는 등, 하드웨어의 성능이 뛰어난 상황을 가정한 파티셔닝 기법으로, 하드웨어 성능의 제약이 있는 WBAN 환경에 적용할 수 없다. 따라서 본 논문에서는 제한적인 하드웨어 성능을 기반으로 한 WBAN 환경의 센서노드에서 2.1.3절에 언급한 의료용 센서 OS인 KHIX를 기반으로 각 어플리케이션들의 최소 수행시간을 보장하는 파티셔닝을 설계한다.

3. 파티셔닝 기법의 설계

3.1 시간분할 (Temporal Partitioning)

시간분할은 CPU의 사용량(프로세스 자원)을 각 파티션별로 나누어 서로에게 영향을 주지 않게 해야 한다. 본 논문에서는 파티션들의 독립적인 스케쥴링 기법을 제안한다.

그림 2는 시간분할이 적용되지 않은 WBAN 환경에서 하나의 센서노드에 여러 어플리케이션이 수행되는 KHIX 시스템을 도시한다. 각 어플리케이션은 여러 쓰레드 (thread)들을 보유하고 있고, 각각의 가중치가 존재한다. 커널에서는 스케쥴러를 통해 각 가중치에 따라 해당 쓰레드들을 실행시킨다. 시간분할이 적용되지 않은 시스템에서 특정 쓰레드의 가중치가 가장 높고, 매우 많은 작업량이 주어지게 되면 CPU 사용량을 독점할 수 있다. 이는 다른 쓰레드들의 수행에 영향을 주게 되어 최소 수행시간을 만족시키지 못하게 되므로, 독립성을 제공하지 못하게 된다. 이를 해결하기 위해, 본 논문에서는 다수의 쓰레드들을 파티션으로 나누어

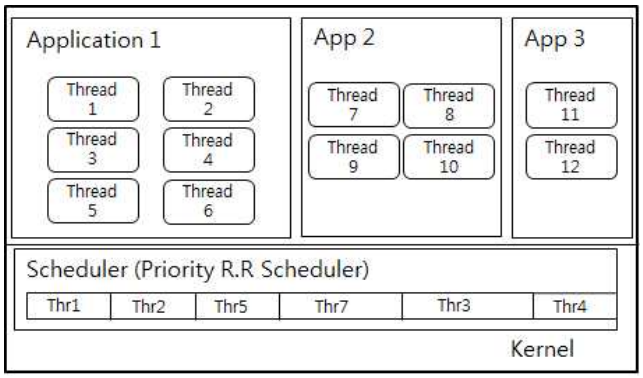

그림 2 WBAN환경의 KHIX 시스템

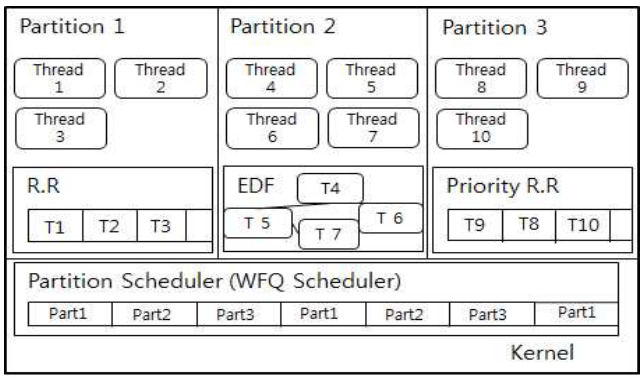

그림 3 파티셔닝을 적용한 시스템

각 파티션별로 독립적인 스케쥴러를 수행하고, 이들을 통합적으로 관리하는 파티션 스케쥴러를 사용하여 시간분할을 수행하는 방안을 제안한다.

그림 3은 파티션 스케쥴러를 이용한 시간분할을 적용한 시스템의 구조도 이다. 각 파티션은 여러 쓰레드들의 집합으로 이루어지고, 각 파티션마다 다른 스케쥴러를 보유하게 된다. 각 파티션은 분할 시 고정 가중치를 정해주고, 커널에서는 해당 가중치에 맞게 타임슬롯을 분할하여 스케쥴링을 설정한다. 이를 통하여 CPU의 독점을 막을 수 있게 되고, 각 파티션마다 정해진 가중치에 맞게 시간을 보장받음으로써 최소 수행시간을 만족시킬 수 있게 되므로 각 파티션들의 독립성이 보장된다.

3.2 파티션

하나의 파티션은 여러 쓰레드들의 집합으로 이루어진다. 각각의 파티션을 분할할 때, 파티션에 속한 쓰레드들의 요구사항을 판단하여 스케쥴링 기법을 정하게 된다. 예를 들어, 그림 3에서 도시하고 있는 분할된 파티션은 각각 공평한 CPU 자원 분할이 필요한 경우는 라운드로빈 (round robin), 쓰레드들의 우선순위를 중요시할 경우에는 우선순위 라운드로빈 (priority round robin), hard-realtime의 특성을 가지는 쓰레드들의 최소 수행시간 요구사항을 보장할 때에는 EDF (Earliest Deadline First)을 사용한다. 이처럼 분할된 각 파티션의 요구사항에 맞는 스케쥴러를 선택하여 사용하게 되며, 이를 통해 서로 다른 요구사항을 가지는 쓰레드들의 요구사항을 모두 만족시킬 수 있다.

3.3 파티션 스케쥴러

각 파티션들은 파티션 내부의 쓰레드들의 실행순서를 정해주는 스케쥴러를 보유하고 있다. 이 모든 파티션들을 파티션의 가중치에 맞게 실행순서를 분배해주는 것이 파티션 스케쥴러이다.

파티션 스케쥴러는 적은 작업량을 갖는 파티션이 상대적으로 많은 작업량을 갖는 파티션에 의해 영향을 받지 않도록 파티션 별로 queue를 두어 파티션의 작업량을 조절하는 WFQ (Weighted Fair Queue) 스케쥴러를 사용하게 된다. 수행시간을 공평하게 분배해준다는 개념은 라운드로빈 스케쥴러와 동일하지만, 각 파티션의 가중치에 따라 스케쥴러의 타임슬롯을 나누어 분배하기 때문에 상황에 맞는 스케쥴링을 할 수 있다. 이로 인해 한 쓰레드 혹은 파티션이 CPU를 독점하는 것을 막을 수 있게 되고, 각 파티션들의 독립성 및 최소 수행시간을 보장해준다.

4. 결론 및 계획

본 논문에서는 하드웨어 성능이 좋지 않은 WBAN환경의 센서노드에서 각 파티션들의 독립성을 제공하고 수행시간을 보장하기 위해 스케쥴링을 이용한 파티셔닝 기법을 설계하였다. 향후 계획으로는 설계한 파티셔닝 기법을 경희대학교에서 개발한 POSIX 기반의 의료용 센서 OS인 KHIX에 직접 구현하고 검증할 것이다.

참 고 문 헌

[1] Bernhard Leiner, Martin Schlager, Roman Obermaisser, Bernhard Huber “A Comparison of Partitioning Operating Systems for Integrated Systems” in SAFECOMP, 2007

[2] Steven H.VanderLeest “ARINC 653 HYPERVISOR” in DASC, 2010

[3] Sanghyun Han, Hyun-Wook Jin “Kernel-Level ARINC 653 Partitioning for Linux” in SAC 2012

[4] Sang-Hyun Han, Hyun-Wook Jin “Virtualization-based ARINC 653 Partitioning for Avionics Software” in KCC 2011

[5] YongGyu Baek, Jinsung Cho “KHIX : A Scalable and Reconfigurable Embedded System Operating” in KCC 2007

[6] Sang-Hyun Han, Jong-Soo Seok, Hyun-Wook Jin “A Partitioning Scheduling Scheme to Support Efficient Mixed Partitioning” in KIISE, 2013

[7] Joao Craveiro, Jose Rufino, Frank Singhoff “Architecture, Mechanisms and Scheduling Analysis Tool for Multicore Time-and Space-Partitioned Systems” in SIGBED, 2011