임베디드 운영체제의 스케줄링 기술 동향 및 선정에 관한 연구

민재홍* · 조평동* · 함진호*

*한국전자통신연구원

A study on the trend and selection for scheduling technology of embedded operating system

Jae-Hong Min* · Pyung-Dong Cho*· Jin-Ho Hahm*

*ETRI

E-mail : jhmin@etri.re.kr

요 약

임베디드 운영체제 기술은 향후 도래할 유비쿼터스 환경을 이루게 할 핵심적 소프트웨어 기술로써 그 파급효과가 매우 큰 기반 기술이다. 따라서 최근 몇 년간 많은 발전과 더불어 큰 변화를 거치고 있다. 그 중에서도 실시간 지원 기능은 임베디드 운영체제 기술과 연관되어 많은 연구가 이루어지고 있다. 본 논문에서는 실시간 임베디드 시스템의 특성을 분석하고 이러한 특성을 지원하기 위한 핵심기술인 스케줄잉 기법의 기술동향은 분석하고, 구현하고자 하는 임베디드 시스템의 특성을 고려한 스케줄링 기법의 선정방안을 제시하고자 한다.

ABSTRACT

Embedded operation system is core software technology which will be able to implement the ubiquitous environment. Also, it is basic technology which has a great ripple effect. Therefore, it is advanced and greatly changed for recent several years. Especially, a lot of studies have been done on real-time supporting technology relating to embedded operating system. In this paper, I analyze the characteristics of real-time embedded system and the trend of scheduling technique in order to support them. As a result, I suggest selection technique for selecting scheduling algorithm through considering the characteristics of embedded system.

키워드

임베디드 시스템, 임베디드 소프트웨어, 임베디드 운영체제, 스케줄링 기법

Ⅰ. 서 론

임베디드 시스템은 특정한 제품 속에 내장되어, 이용자가 직접 볼 수 없는 정보처리시스템이다. 예들 들면, 통신 장비, 교통 시스템, 제조 장비 및 전자제품에 포함된 정보처리시스템이 임베디드 시스템이다[1]. 즉 임베디드 시스템은 미리 정해진 특정 기능을 수행하기 위한 하드웨어와 소프트웨어가 조합된 전자 제어 시스템을 뜻한다. 임베디드 소프트웨어는 마이크로프로세서에 내장되어 다양한 기능을 제공하며, 운영체제, 미들웨어, 응용프로그램 등으로 구성된다. 특히 이용자의 요구가 다양해지고 마이크로프로세서의 성능 향상과 함께 범용 PC에서 사용되는 운영체제 와는 달리 특정한 프로세스 처리에 강한 운영체제를 요구하게 되었다. 이러한 요구 중 가장 큰 제약조건이 실시간 처리를 요구하는 것이다[2]. 이와 같은 실시간처리는 운영체제 내부의 스케줄러에 의해서 수행됨으로 임베디드 운영체제의 스케줄링 알고리즘의 선정은 매우 중요하다.

따라서 본 논문에서는 임베디드 시스템의 특정한 기능을 수행하는데 중요한 역할을 하는 스케줄링 알고리즘에 대한 동향을 살펴보고, 특정 임베디드 시스템에 적합한 알고리즘을 선정하는 방안을 제시하고자 한다. 따라서 본 논문에서는 스케줄링 알고리즘을 분석하고 특정한 임베디드 시

스템에 적합한 알고리즘 선정방안을 제시하고자 한다. 2장에서는 스케줄링 동향을, 3장에서는 스케줄링 알고리즘을, 4장에서는 스케줄링 알고리즘 선정 방안을 모색하고, 마지막으로 결론 및 향후 연구 방향에 대해 언급 한다.

Ⅱ. 스케줄링(Scheduling)

임베디드 운영체제는 보통 태스크(Task)라 불리는 프로그램 수행 단위를 가지게 된다. 임베디드 시스템에서는 복수 개의 태스크가 동시에 수행되는 환경을 가지게 되며 이때 운영체제 내부의 스케줄러에 의해서 다음번에 수행되어야 할 태스크를 선정하게 된다[2]. 이와 같이 가장 중요한 자원인 CPU 스케줄링은 운영체제의 기본기능이고, 운영체제 설계의 핵심이 된다.

■스케줄러(Scheduler)

프로세서(processor)가 유휴 상태가 될 때마다, 운영체제는 준비 완료 큐에 있는 프로세스들 중에서 하나를 선택해 실행해야 한다. 선택 절차는 스케줄러는 실행 준비가 되어 있는 메모리내의 프로세스들 중에서 선택하여 프로세서를 할당한다[3].

■선점 스케줄링(Preemptive Scheduling)

비선점 스케줄링 하에서는, 하나의 프로세스가 프로세서를 할당받았을 때 자신에게 할당된 시간동안 다른 작업에 의해 간섭받지 않고 끝까지 프로세서를 소유한다. 반면에 선점 스케줄링에서는 하나의 프로세스가 이미 프로세서를 점유하고 실행 중인 프로세스로부터 프로세서(processor)를 선점하여 실행한다[6]. 선점 스케줄링은 공유 자료에 대한 접근을 조정할 새로운 매커니즘이 필요하다[3].

■디스패처(Dispatcher)

프로세서 스케줄링 기능에 포함된 또 하나의 요소는 디스패처(dispatcher)이다. 디스패처는 프로세서의 제어를 스케줄러가 선택한 프로세스에게 주는 모듈이다. 디스패처는 다음 기능을 포함한다.

· 문맥을 교환하는 일
· 사용자 모드로 전환하는 일
· 프로그램을 다시 시작하기 위해 사용자 프로그램의 적절한 위치로 이동(JUMP)하는 일

디스패처가 하나의 프로세스를 정지하고 다른 프로세스의 수행을 시작하는 데까지 소요되는 시간을 디스패치 지연(dispatch latency)이라고 한다[3].

Ⅲ. 스케줄링 알고리즘(Scheduling Algorithms)

임베디드 OS의 프로세서 스케줄링은 준비완료 큐에 있는 어느 프로세스에 프로세서를 할당할 것인가를 결정하는 문제를 다룬다. 여러 가지 다른 스케줄링 알고리즘이 있다. 이 장에서는 이들 알고리즘에 대하여 알아본다[3].

■선입선처리(First-Come First-Serviced)스케줄링

가장 간단한 CPU 스케줄링 알고리즘은 선입선처리 스케줄링 알고리즘이다. 이 방법에서는 프로세서를 먼저 요청한 프로세스가 프로세서를 먼저 할당받는다. 선입선처리 스케줄링 알고리즘은 비선점형이며, 시분할 시스템 적용이 곤란하다[3].

■최단 작업 우선(Shortest-Job-First) 스케줄링

프로세서가 이용 가능해지면, 프로세스들 중에서 실행시간 추정치가 가장 작은 프로세스부터 처리하는 방식이다[6]. 이 알고리즘의 실제 어려움은 다음 요청의 프로세스의 실행시간을 파악하는 것이다. 따라서 장기 스케줄링에서 자주 사용되고, 단기 CPU 스케줄링 수준에서는 구현할 수 없다. 선점형 또는 비선점형일 수 있다[3].

■우선순위(priority) 스케줄링

가장 높은 우선순위를 가진 프로세스를 먼저 처리하는 방식이다. 우선순위가 같은 프로세스들은 선입 선처리(FCFS)순서로 스케줄된다. 우선순위 스케줄링은 선점형이거나 또는 비선점형이 될 수 있다[6].

■라운드 로빈(Round Robin) 스케줄링

라운드 로빈(RR) 스케줄링 알고리즘은 시분할 시스템을 위해 특별히 설계되었다. 라운드 로빈 스케줄링은 준비완료 큐에 있는 첫 번째 프로세스에게 지정된 시간만 프로세서를 할당한다. 지정된 시간 이후에는 프로세스를 준비완료 큐의 꼬리로 이동한다. RR 스케줄링 알고리즘은 선점형이다[6].

■다단계 큐(Multilevel Queue)스케줄링

다단계 큐 스케줄링 알고리즘은 준비완료 큐를 다수의 별도의 큐로 나누고, 각 큐는 자신의 스케줄링 알고리즘을 갖고 있다. 예를 들면, 포그라운드 큐는 RR알고리즘에 의해 스케줄될 수 있고, 반면에 백그라운드 큐는 선입 선출 알고리즘에 의해 스케줄될 수 있다. 그리고 포그라운드 큐는 백그라운드 큐보다 절대적인 우선순위를 가질 수 있다.

■다단계 피드백 큐(Multilevel Feedback Queue) 스케줄링

다단계 피드백 큐 스케줄링 알고리즘에서는 프로세스의 프로세서 사용 성격에 따라서 큐들 사이를 이동한다. 즉, 입/출력 중심의 프로세스와 대화형 프로세스들은 높은 우선순위의 큐로 이동할 수 있다[3].

■ 다중 처리기(Multi-Processor) 스케줄링

최근 임베디드 시스템에 사용되는 다중 처리기 시스템-온-칩(Multi-processor System-on-Chip)은

하나 이상의 프로세서, 메모리, 입출력 장치와 주변장치를 하나의 칩에 구현한다[4]. 따라서 여러 개의 프로세서 사용 가능하여, 스케줄링 문제는 그에 상응하여 더욱 복잡해진다. 프로세서들이 그 기능 면에서 동일한 동질의(homogeneous)시스템일 경우 큐에 있는 임의의 프로세스를 수행하기 위해 임의의 사용가능한 처리기를 사용할 수 있다. 이 경우 두 가지 스케줄링 방식 중에서 하나가 사용된다[3].

- 대칭적 다중처리(symmetric multiprocessing, SMP) : 각 처리기가 공동의 준비완료 큐를 조사해 실행할 프로세스를 선택한다. 그리고 각 처리기가 각자 스케줄링한다.
- 비대칭적 다중처리(asymmetric multiprocessing) : 주 서버(master server)라는 단일 처리기가 모든 스케줄링 결정과 I/O처리 그리고 다른 시스템의 활동을 취급하게 한다. 다른 처리기들은 다만 사용자 코드만을 수행한다. [3].

■실시간(Real-Time) 스케줄링

실시간 스케줄링은 외부의 사건에 시간적 제약성을 가지고 반응하며, 기능적 정확성(functional correctness) 뿐만 아니라 시간적 정확성(time correctness)이 만족되어야 한다. 시간의 제약 정도에 따라 경성 스케줄링(hard scheduling)과 연성 스케줄링(soft scheduling)으로 구분할 수 있다. 경성 스케줄링 하에서는 마감시간을 어길 경우 시스템의 품질에 치명적인 타격을 입히고, 연성 스케줄링에서는 마감시간을 어기는 것이 품질의 저하를 가져오더라도 시스템의 동작이 여전히 의미를 가진다[5].

실시간 임베디드 운영체제의 스케줄링 정책에서 고려해야 할 사항은 다음과 같다[2].

- 주어진 시간 내에 가능한 한 많은 작업들을 실행함으로써 처리능력(throughput)을 최대화
- 빠른 대화식 요청(interactive request)에 의하여 응답시간을 최소화
- 전체 작업이 시스템으로 빨리 들어가고 나가게 함으로써 경과시간(turnaround time)을 최소화
- 가능한 한 빨리 준비상태 큐로부터 작업들을 나오게 함으로써 대기시간을 최소화

Ⅳ. 알고리즘 평가

임베디드 시스템 설계는 응용분야의 요구사항 명세서로부터 시작되고, 전형적으로 하드웨어와 소프트웨어 설계가 동시에 고려된다. 따라서 운영체제의 핵심인 스케줄링 알고리즘 선정도 하드웨어 및 소프트웨어 공동 설계 시 주요하게 고려되어야 하고, 최종적으로 설계에 대한 평가 및 검증 과정에서 검토되어야 한다[1]. 알고리즘 선정 과정은 다음과 같다.

1. 실행 시간 예측(prediction of execution times)

태스크(task)의 스케줄링은 작업 수행 시간에 대한 정보를 요구한다. 특히 시간 제약을 충족하는 것이 보장된 실시간 시스템에서 필요하다. 최악 경우의 실행시간(the worst-case execution time)은 대부분의 스케줄링 알고리즘의 기반이 된다. 최악 경우 실행 시간은 태스크들의 수행시간 중에서 상위 경계를 의미한다. 이러한 상한을 계산하는 것은 일반적으로 비 결정적이다. 이것은 하나의 프로그램이 종료 여부가 확실하지 않기 때문이다. 예를 들면 반복과 while 루푸가 없고 반복 계수가 명확하면 최악 경우 실행 시간의 계산이 가능하다[1]. 또한 어떤 경우에는 평균 태스크 수행 시간에 대한 정보도 필요하다.

2. 스케줄링 기준(Scheduling Criteria)

스케줄링 알고리즘을 비교하기 위한 여러 기준이 제시되었다. 비교하는 데 사용되는 특성에 따라서 최선의 알고리즘을 결정하는 데 큰 차이가 발생한다. 사용되는 기준은 <표1>과 같다[3].

<표1> 스케줄링 알고리즘 비교 기준

기 준	내 용
CPU 이용률 (utilization)	- 이용률은 0에서 100퍼 센트까지 이른다. 실제 시스템에서는 40퍼센트 (부하가 적은 시스템의 경우)에서 90퍼센트(부하가 큰 시스템의 경우)까지의 범위를 가짐
처 리 량 (throughput)	- 단위 시간당 완료된 프로세스의 개수
총처리 시간 (turnaround time)	특정한 프로세스를 실행하는 데 소요된 시간으로 메모리에 들어가기 위해 기다리며 소비한 시간, 준비완료 큐에서 대기한 시간, CPU에서 실행한 시간, 그리고 입출력 시간을 합한 시간
대기 시간 (waiting time)	준비완료 큐에서 대기하면서 보낸 시간의 합
응답 시간 (response time)	하나의 요구를 제출한 후 첫 번째 응답이 나올 때까지의 시간

프로세서 이용률과 처리량을 최대화하고 총처리 시간, 대기 시간, 응답 시간을 최소화하는 것이 바람직하다. 대부분의 경우, 평균 측정 시간을 최적화하여야 한다. 그러나 어떤 경우에는 평균보다는 최소값 또는 최대값을 최적화하는 것이 바람직할 수도 있다. 예를 들어, 모든 사용자들이 좋은 서비스를 얻도록 보장하기 위해, 우리는 최대 응답 시간을 최소화하여고 할 수도 있다. 대화식 시스템(시분할 시스템 같은)의 경우에는 평균 응답 시간을 최소화하기 보다는 응답 시간의 변동폭을 최소화하는 것이 중요하다고 제시하고 있다[3].

3. 알고리즘의 평가 방법

앞 장에서 살펴본 바와 같이 다양한 알고리즘이

있으며 각각 자신의 매개 변수를 갖고 있다. 그 결과, 임베디드 운영체제의 스케줄링 알고리즘을 선택하는 것은 매우 어려운 일이다. 첫 번째 문제는 알고리즘을 선택하는 데 사용하는 기준을 정의하는 것이다. 앞장에서 본 바와 같이 기준은 종종 CPU이용률, 응답시간 또는 처리량에 의해 정의된다. 알고리즘을 선택하기 위해, 먼저 이들 값들의 상대적인 중요성을 반드시 정의해야 한다. 또한 기준은 다음과 같은 다수의 기준을 포함할 수 있다[3].

- 최대 응답시간이 1초라는 제약조건 하에서 CPU 이용률을 극대화.
- 총 처리시간이 전체 실행 시간에(평균으로)선형으로 비례하도록 처리율을 극대화

일단 선택 기준이 정의되면, 고려 중인 여러 가지 스케줄링 알고리즘을 다음과 같은 방법으로 평가한다.

■ 결정론적 모델링(Deterministic Modeling)

분석적 평가의 한 형태로 사전에 정의된 특정한 작업 부하를 받아들여 그 작업 부하에 대한 각 알고리즘의 성능을 정의한다. 이 모델은 단순하고 신속하다. 그러나 일반적으로 결정론적 모델링은 너무 특정적이고 정확한 지식을 많이 필요로 하기 때문에 사용하기 곤란한 점이 있다[3].

■ 큐잉 모델링(Queueing Modeling)

프로세서와 입/출력 사용 분포를 측정하여 사용 시간의 확률을 기술하는 수학적인 공식(보통 지수적(exponential)이며 평균에 의해 기술)과 프로세스들이 시스템에 도착하는 시간의 분포를 구한다. 이들 두 분포로부터 대부분의 알고리즘에 대한 평균 처리율, 이용률, 대기 시간을 계산하는 것이 가능하다. 큐잉분석은 스케줄링 알고리즘을 비교하는 데 유용하게 사용할 수 있으나, 취급할 수 있는 알고리즘과 분포의 부류가 상당히 제한되어 있다. 또한 정확하지 않을 수 있는 다수의 독립된 가정을 하는 것이 일반적으로 필요하다[3].

■ 모의 실험(Simulation)

모의실험을 시행하기 위한 자료들은 난수 발생기(random number generator)로 생성하고, 분포는 수학적[균등(uniform), 지수(exponential), 포아송(poisson)] 또는, 경험적으로 정의될 수 있다. 그러나 빈도 수 분포는 단지 각 사건들이 얼마나 많이 발생하는 가를 의미하며, 생성 순서에 대해서는 아무 정보도 제공하지 못한다. 이 문제를 해결하기 위해, 실제 시스템을 관찰해 실제 사건들의 순서를 기록하여 추적 테이프를 생성한다[3].

■ 구현(implementation)

모의실험도 정확성에 한계가 있다. 스케줄링 알고리즘을 완전히 정확히 평가하는 유일한 방법은 실제 코드로 작성해 운영 체제에 넣고 실행해 보는 것이다. 이 방식은 알고리즘 코드 및 운영체제 구현 등 많은 비용을 요구한다[3].

알고리즘을 평가하는 다른 어려움은 알고리즘이 사용되는 환경이 변화하고 있는 것이다. 환경은 새로운 프로그램이 도입 및 스케줄러의 성능 결과에 의해서도 변화할 것이다. 가장 융통성 있는 스케줄링 알고리즘은 시스템 관리자 또는, 사용자에 의해 알고리즘이 특정 운용이나 응용 집합에 맞도록 조정될 수 있는 것이다[3].

Ⅴ. 결 론

본 고는 현재 국내에서 관심의 커지고 있는 임베디드 시스템 운영체제의 핵심인 스케줄링 기술 동향을 분석하고, 이를 기반으로 임베디드 시스템 설계 시 스케줄링 알고리즘 선정을 위한 방법을 제시하고자 하였다. 따라서 일반적인 운영체제 스케줄링 알고리즘 및 선정방법을 임베디드 시스템의 특성에 접목하는 시도를 하였으나, 임베디드 시스템에 대한 구현사례 분석 부족으로 전반적인 방향을 제시하는 한계가 있었다.

그리고 임베디드 시스템의 다양한 응용 분야에 맞는 임베디드 운영체제 스케줄링 선정 방법론에 대한 연구가 이루어 져야 한다. 또한 급속하게 발전하는 하드웨어 환경 하에서 이를 고려한 임베디드 운영체제 설계의 일환으로 스케줄링 선정 방법론에 대한 지속적인 연구가 필요하다.

참고문헌

[1]Peter Marwedel, "Embedded System Design", Springer, 2006

[2]김창환, "임베디드 소프트웨어 추진동향", 주간기술동향, 2008.6.18.

[3]조유근외, "응용 운영체제 개념", 홍릉과학출판사, 2003

[4]Katalin Popovici, "Embedded Software Design and Programming of Multiprocessor System-on-Chip: Simulink and System C Case Studies", Springer,2010

[5]e-Campus, "재미있는 임베디드 이야기",삼성 SDS멀티캠퍼스 , 2011.1.

[6]김평수, "임베디드 시스템 기초", 한국과학기술사이버연수원, 2010.11.