# 다중작업 분할처리를 위한 적응형 스케쥴링 기법

고정환* · 김영길**

*아주대학교

# The technique of adaptive scheduling for multi-tasking separation control

Jeong-hwan Go* · Young-Kil Kim**

*Ajou University

E-mail : kojh2000@ajou.ac.kr

## 요 약

프로그램의 복잡화와 대규모프로그램의 등장으로 다중작업을 분할하여 소규모 단위의 Task로 나누고 각각의 Task를 우선순위에 따라 스케쥴링을 수행해야하는 요구가 점점 확대되고 있다. 또한, 프로그램 개발환경의 다양화로 인하여 프로그램을 구현하다 보면 다양한 환경 조건에 맞춰서 개발하게 된다. 예를 들어 Embedded 환경인지 Windows 환경인지에 따라 다르고 운용체제의 사용에 따라서도 제약사항을 가져오는 경우가 많다. 이에 개발환경과 운용체제에 의존적이지 않도록 다중작업 분할처리를 수행할 수 있는 적응형 스케쥴링 기법을 소개한다. 본 논문에서는 적응형 스케쥴링 기법에 적용된 알고리즘에 대한 설명과 구현 후 적용한 사례를 기반으로 한 내용을 다룬다.

## ABSTRACT

Because of the substantial increase in program complexity and appearance of mega program, the needs to devide the program into small task with multiple partitions, and perform a scheduling based on the priority is required. And also, a program can be developed on specific environment according to the diversify of development environment. for instance, there are some restrictions upon O/S environment such as Embedded or Windows. therefore, the adaptive scheduling technique which perform multiple task partitioning process regardless environment or O/S is suggested. In this study, Adaptive scheduling technique algorithm and its application to be described.

키워드

Scheduling, Middleware, Task, OS, Queue

## Ⅰ. 서 론

프로그램의 복잡화와 각종 소프트웨어 언어 및 Tool 등의 발전으로 인하여 대규모의 프로그램이 점점 많아지고 있다. 이러한 이유로 실시간처리가 중요한 Embedded 시스템에 주로 적용되었던 스케쥴링 알고리즘이 대규모 프로그램의 효율성과 모듈화된 처리를 위하여 Windows 기반의 비실시간 프로그램등에도 많이 쓰여지고 있는 추세이다.

현재 여러가지 플랫폼에서 스케쥴링을 지원하는 OS 및 도구가 많지만 다양한 하드웨어의 특성과 사용되는 OS의 종류에 따라 호환성이 부족하기 때문에 하드웨어 또는 OS와 Application 간의 사이를 매끄럽게 연결하기 위하여 여러 형태의 소프트웨어적인 계층을 두기도 한다.

이러한 소프트웨어의 복잡한 구조 속에서 효과적으로 작업을 분할하고 스케쥴링하며 OS 환경 및 여러 가지 플랫폼에 영향을 받지 않고 프로그램의 계층인 Middleware 나 Application의 어디에서나 Task를 처리하기 위해 스케쥴링을 수행 수 있는 적응형 스케쥴링 기법을 소개한다. 적응형 스케쥴링 기법은 효과적인 Task처리를 위해 우선순위 기반의 스케쥴링 알고리즘을 적용하였으며 본 논문에서는 실제 모듈화하여 적용한 사례를 바탕으로 효과를 입증한다.

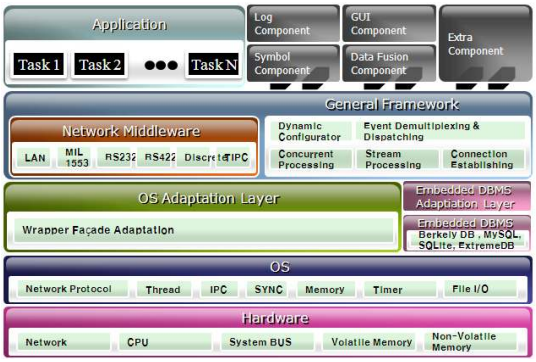


그림 1. 일반적인 프로그램의 계층구조

## II. 기존의 스케쥴링 방식

보통의 프로그램을 할때 RTOS를 사용하는 경우는 다음의 계층구조 형태로 소프트웨어가 구성된다.

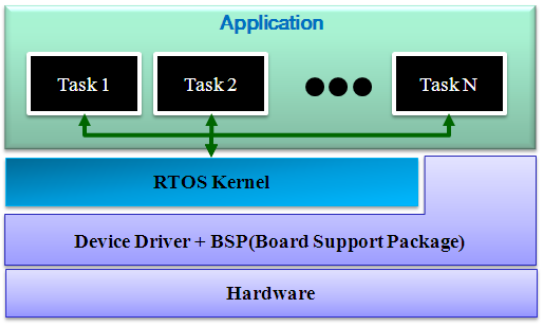


그림 2. RTOS를 사용한 소프트웨어 계층구조

위의 구조에서 보면 OS의 커널에서 제공하는 스케쥴러에 의해 사용자가 만든 Task(또는 Process, Thread)가 실행되며 OS와 스케쥴링 방식에 따라 또 다른 시스템에 동일한 Application을 적용하기 힘들뿐만 아니라 동일한 실행 동작을 예측하기 힘들다.

OS를 적용하지 않을 경우는 다음과 같이 하나의 Task로 모든 처리를 수행하는 형태로 프로그램하거나 별도의 스케쥴러를 개발하여 사용하는 경우가 많다.

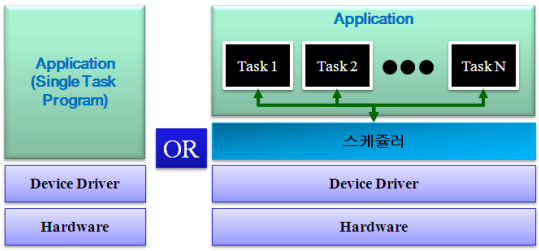


그림 3. OS 미적용 Embedded 시스템 계층구조

이에 OS의 스케쥴러와는 별개로 스케쥴링이 가능하고 시스템에 구애받지 않는 어떤 환경에도 이식이 용이한 모듈화된 구조의 스케쥴러가 필요하다.

## III. 적응형 스케쥴링 기법의 개념 및 구조

적응형 스케쥴링 기법은 기존에 OS를 사용하지 않는 Single Task 프로그램에서 스케쥴러의 역할을 수행하며 Multi-Tasking 프로그램에서도 OS 레벨에서의 Task 분할이외에 Application 레벨에서도 Task를 분할하여 별도의 스케쥴링을 수행할 수 있는 기법이다. 적응형 스케쥴링 기법은 프로그램 복잡도의 증가를 막고 Application단에서 별도의 스케쥴링을 구성할 수 있으며 독립적인 모듈로 구성 가능하여 OS나 플랫폼에 영향을 받지 않고 이식될 수 있다.

하나의 OS레벨의 Task 내에서 작은 단위의 Task를 분할하고 싶은 경우 다음의 예와 같이 Task 내부에 적응형 스케쥴링 기법을 적용하면 Task를 여러개의 더 작은 단위의 Task로 쪼개어 스케쥴링 할 수 있다.

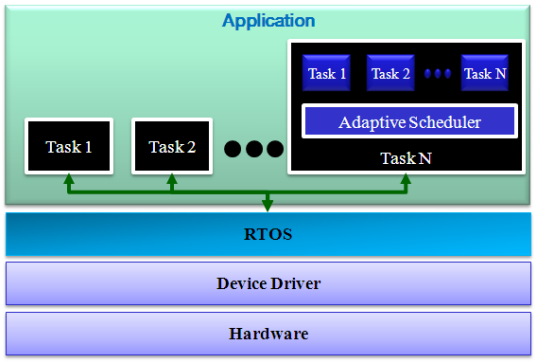


그림 4. 적응형 스케쥴러가 적응된 구조(예)

적응형 스케쥴러는 아래그림과 같이 Task의 스케쥴링을 위한 3가지의 스케쥴러와 Interface Controller, Message Handler, Queue Controller로 나뉜다.

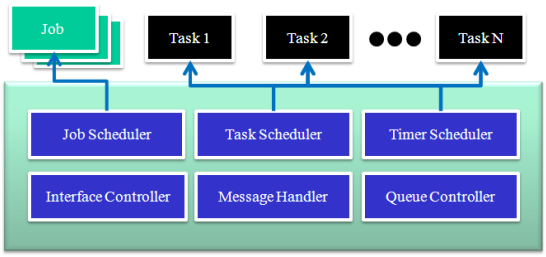


그림 5. 스케쥴러의 구성 및 Task 실행 개념

Job Scheduler, Task Scheduler, Timer Scheduler는 Task 및 Job의 우선순위를 관리하고 가장 높은 우선순위의 Task를 실행시키는 역할을 한다.

Interface Controller는 적응형 스케쥴러와 외부 모듈간의 인터페이스를 담당한다. OS를 적용하지 않는 Embedded 시스템에서는 사용자가 인터페이스를 정의하여 Middleware로 활용하며 Application이 실제 하드웨어 또는 디바이스드라이버와 인터페이스를 수행하도록 Control 할 수 있고, 다른 Middleware 및 모듈과도 연동할 수 있다. Interface Controller는 사용자가 필요에 따라 인터페이스를 정의하거나 새로이 프로그래밍 할 수 있다.

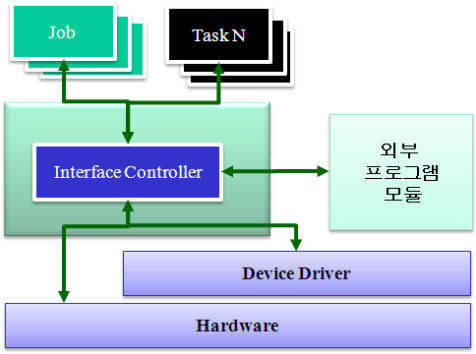


그림 6. Interface Controller의 동작 개념도

Message Handler는 외부로 부터의 메시지에 대하여 어느 Task에서 처리되는지를 설정하면 해당 Task로 메시지를 전달해 주는 역할을 수행하고 Task간의 메시지를 주고받을 때 원할한 동작을 지원한다.

## IV. 적용된 스케쥴링 알고리즘

적응형 스케쥴링 알고리즘은 Application 레벨의 원할한 다중 작업 분할처리를 위해 비선점형(Non-Preemptive) 스케쥴링 방식을 적용한다.

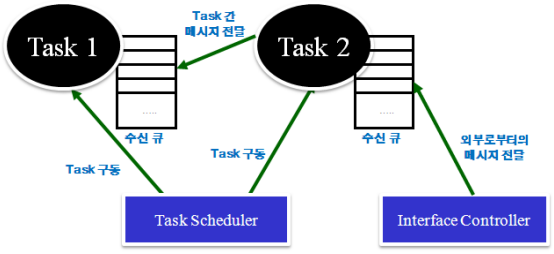


그림 7. Task 동작 개념도

적응형 스케쥴러에 쓰이는 Task는 Event나 메시지에 의해 수행되는 기능적으로 유사한 모듈들의 집합이며 생성시 메시지큐를 하나씩을 갖게 되며 메시지큐에 메시지가 수신되면 스케쥴러가 Task를 실행시킨다.

1. Job Scheduler

인터럽트 루틴상의 빠른 처리를 위한 방법으로 스케쥴링을 위한 우선순위 플래그가 존재하고 각각의 플래그 마다 우선순위를 가지며 각각 하나의 Job과 연결된다. 인터럽트가 발생하며 플래그가 세팅되며 스케쥴러는 해당 Job을 수행한다.

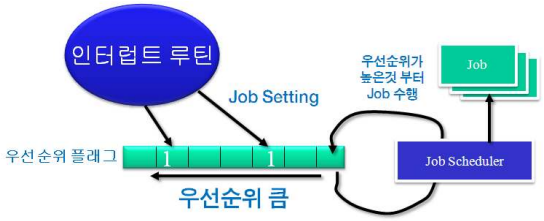


그림 8. Job Scheduler 동작 개념도

일반적으로 다른 스케줄러보다 우선순위가 높다.

2. Timer Scheduler

일정시간 후에 발생해야 할 모듈이 있을 때 프로그램에서 계속 기다리지 않고 원하는 일정시간 뒤에 메시지의 전달하는 기능을 수행하는 스케쥴러이다.

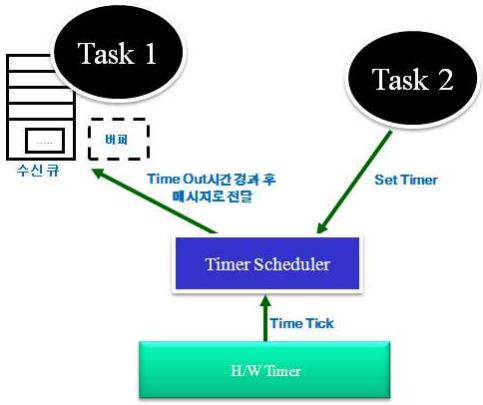


그림 9. Timer Scheduler 동작 개념도

3. Task Scheduler

Task Scheduler는 일반적으로 많이 사용되는 세가지 스케쥴링 방식을 지원한다. Round Robin, Fixed Priority, Changable Priority의 방식을 선택적으로 사용할 수 있다.

1) Round Robin 스케쥴링 방식

할당된 Task만큼 우선순위가 없이 순서대로

Task가 수행되는 방식으로 특별히 우선순위가 높은 Task가 없는 경우에 일반적으로 사용하며 특별히 빨리 수행되어야 하는 모듈이 있다면 이 경우 Job Scheduler에 등록하여 사용하면 된다.

2) Fixed Priority 스케쥴링 방식

Fixed Priority 스케쥴링 방식은 Task의 생성시 정의된 Priority 순위에 따라 스케쥴링이 일어나는 방식이다. Priority는 1-31중 하나의 값을 선택할 수 있다. Queue에 메시지가 저장 될 때 Flag는 Set 되며 Task가 수행되면 Clear된다.

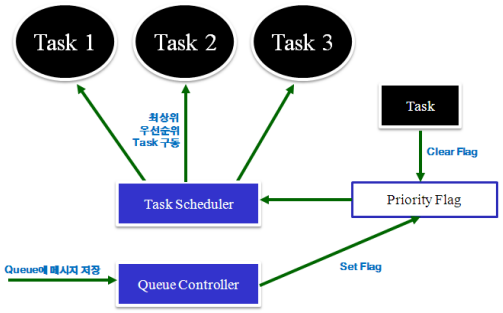


그림 10. Fixed-Priority 스케쥴링 개념도

3) Changable Priority 스케쥴링 방식

이 방식은 Task의 생성시 정의된 Priority 순위를 기본으로 스케쥴링이 일어나고 메시지를 보낼 때 우선순위를 바꾸거나 또는 프로그램상에서 우선순위를 바꿀 수 있는 스케쥴링 방식이다. Queue에 메시지가 저장될 때 Linked List로 구성된 우선순위 트리가 Re-Scheduling 되어 최상위 우선순위 Task를 정하며 스케쥴러는 선택된 최상위 Task만 구동한다.

## V. 적용사례

적응형 스케쥴링 기법은 앞서 언급했듯이 여러 가지 환경에 적용되어질 수 있다.

그 예로 그림처럼 대잠전 통제 콘솔에서 Windows기반의 프로그램에 적응형 스케쥴링 기법을 적용하여 개발하였으며 GUI와 인터페이스 단의 중간에서 Main Application을 구동하는 역할을 수행한다.

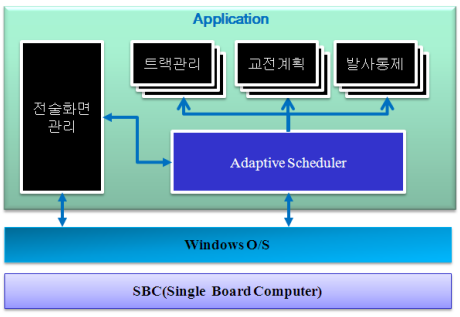


그림 11. 대잠전 통제 콘솔 적용 사례

또한 항공용 전자전 장비의 OFP에 적용되어 운용중에 있으며 이 경우 O/S를 사용하지 않은 Embedded 시스템에 O/S의 역할을 대신하여 Task의 스케쥴링을 수행한다. 미들웨어로서의 역할도 수행하여 SBC(single Board Computer) 환경과 PC 환경에서 Application이 동작 할 수 있게 Interface Controller가 구성되었다.

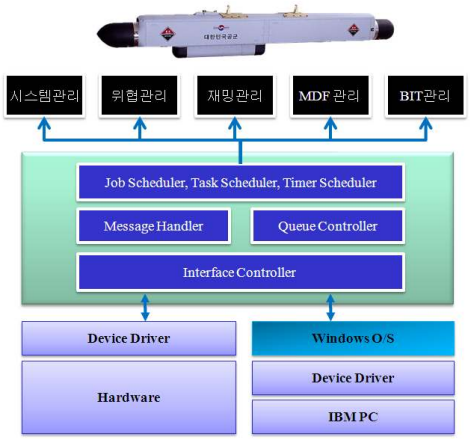


그림 12. 항공용 전자전 장비 적용 사례

## VI. 결 론

소프트웨어는 여러가지 계층구조를 갖고 있으며 그 속에서 각각의 정해진 Task를 수행하게 된다. 지금까지 Task의 스케쥴링은 OS의 커널에서 수행하는 것이 일반적이지만 적응형 스케쥴링 기법을 사용하면 프로그램내의 어디서나 스케쥴링이 가능한 형태로 구성할 수 있다. 실제 적용한 사례에서와 같이 적응형 스케쥴링 기법은 시스템에 구애받지 않고 Task를 스케쥴링 할 수 있으며 Interface Comtroller를 어떻게 구성하느냐에 따라 여러 가지 환경에 변화에 적응할 수 있다. 향후에도 여러체계에 적응형 스케쥴링 기법을 활용할 계획에 있으며 소프트웨어의 하나의 독립된 계층 또는 모듈로 활용될 수 있을 것이다.

## 참고문헌

[1] Jean J. Labrosse, "Embedded Systems Building Blocks", Logic Design Principles, R&D Books, 611p.

[2] Jean J. Labrosse, "microc/OS-II The Real-Time Kernel", R&D Books, 498p.

[3] WindRiver, "VxWorks Programmer's Guide" , www.windriver.com, 2002.

[4] 정덕영, Windows 구조와 원리 그리고 Codes, 가남사, 470p, 2003.