전자무역의 RTS 효율성에 관한 연구

정 분 도*

A Study of Real-Time System(RTS) Efficiency in e-Trade

Boon-do Jeong*

요 약

전자무역에서 실시간 시스템은 매우 중요한 역할을 담당한다. 각각의 태스크들은 제한된 시간이 주어져 있고, 제한된 시간 내에 문서 처리를 수행하지 못 할 경우 큰 피해를 입을 수 있으므로 반드시 약속된 규정이 지켜져야 한다. 전자무역에서 스케줄링 가능성 기법들은 주로 주기적인 태스크를 사용하는데 이 방법 외의 다른 태스크 시간 조건과 비주기적인 태스크 시간조건을 이용하여 보다 안정적인 사전예측 스케줄링 가능성 알고리즘의 연구가 필요하다. 본 논문은 개별 태스크 이용율을 사용하여 예측가능성을 높이기 위한 알고리즘을 제안하며, 기존의 태스크 전체 이용율과 제안한 알고리즘을 이용한 스케줄링 가능성 조건을 제시 하였다.

ABSTRACT

In e-Trade, Real-Time System(RTS) plays a very important role. Each task is set with limited time, and appointed regulations must be followed because it can be greatly damaged if it cannot be executed in limited time. In e-Trade, the scheduling possibility techniques generally use periodical tasks; however, it is necessary to study more stable prediction scheduling possibility algorithm by using other task timing conditions and non-periodical task scheduling tasks. This study proposed an algorithm to increase the prediction possibility using individual task utilization rate, and presented scheduling possibility conditions using existing whole task utilization rate and the proposed algorithm.

키워드

e-Trade, RTS(Real-Time System), RM(Rate Monotonic), EDF(Earlist Deadline First)

Ⅰ. 서 론

전자무역에서 Real-Time System(이하 RTS라 한다.)의 태스크는 마감시간을 가지고 있어서 실행이 마감시간 이전에 종료되어야만 비로소 가치를 가지는 경우가 많다. RTS의 동작은 논리적 정확성뿐만 아니라, 시간적 정확성에도 좌우되는 시스템이라 할 수 있다[1]. 특히 경성RTS의 경우 태스크가 마감시간 내에 실행을 종료하지 못하는 경우 시스템 고장의 원인이 되기도 한다. RTS는 스케줄링이 실시간 태스크의 마감시간에 맞추어 실행될 수 있도록 태스크를 프로세서에게 할당하는 중요한 요소로 성능을 좌우한다[2]. 일반적인 RTS에서 자원 제어와 마감시간의 문제에 대한 해결법을 제안하는데 있어서 요구되는 속성은 예측성과 한계성을 들 수 있는데, 예측성은 자원 할당에 관련된 결정이 시스템 수행 이전에 예측되어야 함을 의미하며 한계성은 태스크의 실행시간을 자원 접근에 관련해서 실행시간이 마감시간 내에 수행할 수 있는지에 대한 내용을 뜻한다. 이 실행시간과 마감시간은 스케줄 가

* 조선대학교 경상대학 경제무역학부 초빙객원교수

접수일자 : 2005. 8. 5

능성을 결정할 수 있도록 수행 전에 계산되어야 한다. 이러한 속성들을 만족시키기 위한 방법은 결정적인 수행 동작들로 제한되어야 하고 이러한 요건을 만족시키지 못하면 태스크 집합의 스케줄 가능성은 어려워지며 최악의 경우 태스크의 마감시간의 보장은 실현될 수 없다[3].

실시간 스케줄링 방법은 시분할 시스템에서 사용되는 스케줄링 방법과는 달리 태스크가 마감시간 내에 종료될 수 있음을 보장해주어야 한다. 또한, 실시간 스케줄링 방법은 새로운 태스크의 실행을 허가하기 전에 새로운 태스크 집합의 스케줄 가능성을 분석함으로써 시스템 전체의 안정성을 유지할 수 있어야 한다[4].

따라서 본 논문에서는 태스크 한계성에 대한 스케줄링 가능성을 예측함으로써 스케줄링 수행 이전에 태스크 마감시간의 문제점들을 예측 할 수 있게 하고 태스크 집합 성격에 가장 맞는 스케줄링 알고리즘에 관한 연구를 수행하였다.

Ⅱ. 관련연구

2.1 RTS

전자무역에서 시스템의 입력·출력은 전자거래에 수반하는 응답으로 대응될 수 있으며 임의의 거래와 응답 사이에는 시간적인 지연이 존재하는데 이를 응답시간이라 한다. 외부의 사건에 의하여 임의의 연산이 시작된 이후 그 연산의 결과가 주어진 연산 시간이나 외부의 특정 기준 시간에 의존하는 시스템으로도 정의되고 있다[5].

일반적으로 실시간의 실행시간 제약조건에 따라 경성(Hard Real-Time), 준경성(Firm Real-Time), 연성(Soft Real-Time)으로 분류된다. 경성RTS는 외부 이벤트에 대해 명시된 시간 내에 응답을 하지 못했을 경우 완전한 실패로 여겨지는 시스템으로 한번이라도 마감시간을 초과한다면 수용할 수 없는 시스템을 의미한다. 준경성RTS는 경성과 연성의 중간 형태로 마감시간을 넘겨 수행을 마치는 것은 무의미한 경우를 의미하며 손실이 치명적이지 않는 경우를 말한다. 연성RTS는 외부 이벤트에 대한 응답의 시간 제약이 중요하지만 마감시간을 초과하더라도 중대한 문제를 유발하지 않는 시스템이다.[6]. RTS에서의 응답시간에는 절대적인 기준이 없고 자신의 기능적 역할을 충분히 수행함과 동시에 내·외부의 다양한 사건에 대하여 예측 가능한 시간 범위 내에 응답할 수 있는 시스템이다[7].

전자무역에서 무역자동화시스템, e-SCM 분야에서 각종 제어 시스템을 중심으로 형성되어온 RTS는 점차 확대되어 인터넷 셋-탑 박스, 라우터, 카드 단말기, 초고속 통신스위치, 프린터, PDA, CDMA, 스케너, 복사기, 팩스, 인터넷 전화기, 단말기, 등과 같은 각종 정보기기 등에서 널리 사용되고 있다.

2.2 실시간 스케줄러

전자무역에서 실시간 태스크는 시간 속성으로 그 태스크를 정의할 수 있다. 시간 속성은 해당 태스크의 시간 제약 및 시간 행위를 의미한다. 태스크의 도착 시간은 그 태스크가 프로세서에 의해 언제라도 실행 가능하게 준비되어진 시점이다. 태스크의 실행시간은 해당 태스크의 실행을 완료하기 위해 요구되어지는 최악의 경우 할당 시간을 의미한다. 태스크의 만기는 그 태스크의 실행이 완료해야 하는 요구된 시점이다. 전자무역의 태스크 만기는 그 태스크의 실행이 완료해야 하는 요구된 시점이다. 태스크의 허용 가능한 최대 응답 시간을 그 태스크의 상대만기라 하고 태스크의 도착 시간에 상대 만기를 더한 값을 절대 만기라 한다. 일반적으로 어떤 태스크의 시간 행위에 가해진 제약을 그 태스크의 시간제약이라 한다.

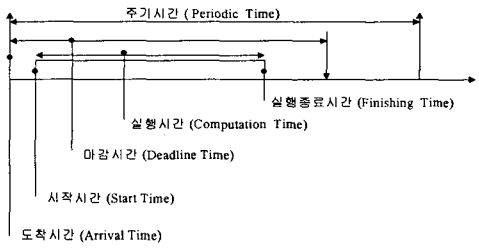

그림 1. 주기시간의 구조
Fig. 1 Structure of Periodic Time

시간 제약 조건을 만족시키지 못하면 곧 심각한 시스템 장애 또는 인명 피해로 이어지는데 그러한 태스크를 경성 태스크(또는 엄격한 시간 제약을 가진 태스크)라 한다. 반대로 가능한 만기 보다 빠른 응답을 기대하고 늦은 결과를 원치 않지만 그럼에도 불구하고 늦은 결과라도 사용 가치가 있을 때 그러한 태스크를 연성 태스크라 한다. 연성 태스크의 응답 시간은 빠르면 빠를수록 좋다. 실시

간 태스크는 주기적 태스크와 비주기적 태스크가 있다. 주기적 태스크는 주기적으로 발생하는 이벤트에 대해 동일한 작업을 반복하여 수행한다. 태스크 주기는 연속된 작업들의 도착시간 간격들의 최소 길이이다. 주기적 태스크는 시간 제약 요소 외에 주기를 시간 제약으로 가진다. 반면, 사전에 예측할 수 없는 외부 이벤트에 의해 생성된 작업 오버헤드는 비주기적 태스크에 의해 수행된다. 비주기적 태스크는 일련의 연속된 비주기적 작업들로 구성된다. 이러한 태스크의 연속적인 작업들의 도착 시간 간격은 상당히 큰 폭으로 변화하며 경우에 따라 임의의 작은 값일 수도 있다[8].

2.3 스케줄링 기법

스케줄러는 태스크들 또는 다른 어느 특정영역의 그룹 중에 태스크 우선도, 태스크 마감시간, 서비스 객체, 시간 분할응답, 공평성 위에 스케줄링 정책을 기초로 하고 있으며 모든 시스템에서 의무적이다[9]. 시분할 스케줄링에는 FIFO(선점형 우선순위에 근거한 스케줄링), 라운드 로빈(RR)등 이며 실시간 스케줄링은 Rate monotonic(RM), earliest deadline first(EDF)등과 같은 스케줄링 전략을 제공한다.

2.4 비실시간 스케줄링

(1) FIFO (First In First Out)

FIFO 스케줄링는 태스크 작업을 처음으로 처리하면 처음으로 작업이 종료하는 전략을 제공한다. 모든 태스크는 우선순위에 근거한 선점형이며 선입선출 전략을 제공한다[10]. 모든 태스크는 우선순위에 의거하면 스케줄된다. 낮은 우선순위 태스크가 실행하고 있을 때 만일 더 높은 우선순위를 가진 태스크가 허가된다면 실행하는 태스크는 선점된다. 만약 같은 우선순위를 가진 1개 이상의 태스크가 동시에 실행하려고 준비된다면, 그것들은 issue의 명령에 의하여 이루어진다. 이 상황에서, 모든 태스크는 실시간 태스크라고 간주된다. 블록된 후에 실행할 준비가 된 태스크는 그 우선 순위 준비 큐(FIFO 전략) 끝의 항상 맨 마지막에 삽입된다. 만일 가장 높은 우선순위의 태스크가 엄격하게 실행할 준비가 되어 있게 되면 실행하는 태스크는 다만 선점된다는 것을 뜻한다. 선점된 태스크는 우선 순위 큐의 선두에 놓인다. 이것은 선점하는 태스크가 완료할 때나 또는 블록 될 때, 첫 번째로 선택되게 된다.

(2) RR(Round-Robin)

우선순위에 근거를 두고 있는 선점형의 라운드-로빈 전략을 제공한다. 선택된 태스크가 고정된 시간의 양이 주어진 것을 제외하고는 FIFO 전략과 비슷하다. 같은 우선순위를 가진 태스크들은 라운드-로빈 방법에 의하여 스케줄 된다. 만일 태스크들이 아직도 만기의 양에서 실행하고 있는 중이라면 그것은 우선순위 준비 큐의 끝에 위치하게 되거나 같은 우선순위의 태스크에 의해 선점될 지도 모른다. 태스크가 블록 된 후, 준비 상태가 될 때 그 태스크의 양은 리셋 된다. 그러나 선점 후에 태스크가 다시 선택된다면 리셋 되지 않는다. 시간 양은 모든 우선 순위 수준들과 모든 태스크 들에 대해서 같은 것이다.

2.5 실시간 스케줄링

(1) 정적 스케줄링 알고리즘

우선 태스크가 상대적인 마감시간과 주기시간이 같음(Di = Pi)을 가정한다. RM은 각 태스크의 주기의 역수인 빈도율(Rate)이 높을수록, 즉 주기가 짧을수록 더 높은 우선순위를 부여하는 방식이다. 각 태스크의 주기가 주어지면 태스크들간의 우선순위는 정적으로 결정될 수 있으며 이 우선순위는 시스템이 수행되는 동안 고정된다[11]. 스케줄작업 수행이전에 태스크의 우선순위가 정해져 있고 새로운 태스크 작업을 위해 스케줄 우선순위를 할당할 수 없는 상태가 정적 스케줄링 알고리즘이다. 단일 프로세서 환경에서 비율 단조 스케줄링 알고리즘을 사용하는 경우 태스크의 집합의 스케줄 가능성은 주어진 주기 태스크들의 집합이 프로세서 이용율(Utilization, U)이 n(21/n-1) (n의 값은 태스크의 수) 보다 작거나 같으면 스케줄링이 가능하다고 확인되며, 이에 의해 확인되지 못하는 경우는 동시에 시작된 각 태스크의 첫 번째 작업은 수행시간을 구한 값이 마감시간보다 작은가의 여부에 따라 결정된다.

단일 프로세서 상에서 고정 우선순위 기반형 스케줄링(Preemptive scheduling based on fixed priority)방식은 다음과 같은 가정을 하고 있다[12]. ① 태스크들은 주기적이고, 주기와 종료 시간은 같으며, 실행중에 스스로 중지되지 않는다. ② 태스크들은 선점될 수 있고, 문맥 교환과 태스크 스케줄링에 드는 비용은 무시한다. ③ 모든 태스크들은 서로 독립적이다. 선행 제약은 존재하지 않는다.

RM 태스크 스케줄링 주기를 Pi로, 수행시간은 Ci로 표기하면 태스크의 프로세서 이용율(U)은 Ci/Pi으로 표시될 수 있고,

$$U=\frac{C1}{P1}+\frac{C2}{P2}+\frac{C3}{P3}+\frac{Ci}{\Pi}+\cdots+\frac{Cn}{Pn} \quad (1)$$

로 표시되며, 다음과 같은 조건이 성립되면 독립적인 주기 태스크들의 집합은 각 태스크의 마감시간을 만족시킬 수 있다.

$$U \le U(n) = n(2^{1/n}-1) \quad (2)$$

이 식에서 $n(2^{1/n}-1)$ 값은 태스크의 개수 n이 증가함에 따라 점점 감소하여 69.3%에 수렴한다. RM 스케줄링 기법은 69.3%보다 작은 태스크 집합이 스케줄 가능함을 의미한다. 그러나 식(2)에 의한 스케줄 가능 분석 방법은 정확한 것이 아니다. 이 조건식을 만족시키지 못하는 태스크 집합이라 할지라도 RM 스케줄링 기법에 의해서 타당한 스케줄이 생성될 수 있다.

Task	Utilization
1	100%
2	81.3%
3	77.9%
4	75.6%
5	74.3%
...	...
n	69.3%

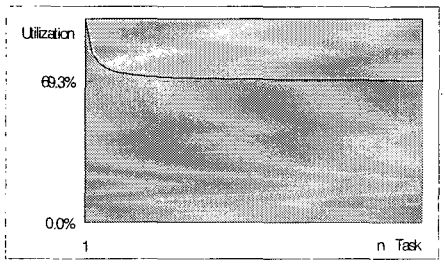

그림 2. RM 스케줄링 조건
Fig. 2 Condition of RM Scheduling

(2) 동적 스케줄링 알고리즘

마감시간 우선(EDF) 스케줄링 알고리즘은 동적 우선 스케줄링 정책에서 가장 흔히 사용되고 있는 스케줄링 방법으로 Liu와 Layland에 의해서 제안되었다. 태스크의 우선순위는 마감시간에 따라서 할당된다. 즉, 마감시간이 짧을수록 높은 우선 순위가 할당되므로, 임의의 순간에 실행되는 태스크는 실행이 완료되지 않은 태스크들 중에서 마감시간이 가까운 것을 선택한다. 정적 스케줄링 알고리즘과 달리 태스크의 우선순위가 시간에 따라 변하게 된다[13]. 마감시간 스케줄링 알고리즘에서 다음의 조건이 만족되면 주기적으로 발생하는 독립적인 n개의 태스크들은 스케줄 가능하다고 할 수 있다. 이 때, 태스크의 주기는 Pi로 수행 시간은 Ci로 표기한다.

$$U=\frac{C1}{P1}+\frac{C2}{P2}+\frac{C3}{P3}+\frac{Ci}{\Pi}+\cdots+\frac{Cn}{Pn} \le 1 \quad (3)$$

(3) 정적 · 동적 알고리즘의 장단점

정적알고리즘은 태스크 수행 이전에 태스크의 특성에 대한 집합을 모두 알고 있다는 가정 하에서 오프라인시에 스케줄링을 하는 방식으로 실제 실행시의 시스템 부하가 적은 장점을 지니고 있다. 그렇지만 시스템 환경 변화에 대처할 수 없는 것은 단점이다. 정적알고리즘은 고정된 수를 가지고 있으면서 시스템 환경 및 처리 요구가 잘 정의된 시스템에서 주로 목표를 이루기 위해서 이용된다. 첫째, 경성 태스크의 종료시간을 만족시키면서 높은 이용율을 유지. 둘째, 경성태스크의 종료시간을 만족하면서 연성태스크의 빠른 응답시간 유지. 셋째, 낮은 스케줄링 오버헤드이다. 최적인 정적스케줄링알고리즘은 다른 정적스케줄링알고리즘이 태스크의 제약 조건을 만족하는 스케줄링을 할 때마다, 태스크의 수행 이전에 태스크의 특성에 대한 집합을 모두 알고 있다고 가정하면 태스크 집합에 대해서 스케줄링이 가능해야 한다. 최적인 정적스케줄링알고리즘으로 스케줄 될 수 없는 태스크 집합에 대해서는 다른 어떤 정적 알고리즘도 스케줄 할 수 없다. 동적인 스케줄링 알고리즘은 모든 태스크 집합에 대해서 그 특성을 알지는 못하지만 현재 활성화인 태스크에 대해서는 모든 정보를 가지고 있다. 그러나 새로운 태스크가 미래에 도착할 수 있기 때문에 시간에 따라 스케줄이 변하게 된다. 동적스케줄링알고리즘은 수행 중에 스케줄링을 하기 때문에 외부 환경의 변화에 대처할 수 있는 적응력이 뛰어나다는 장점이 있다. 그리고 동적으로 스케줄링을 하기 때문에 스케줄링 가능성의 검사시 정적인 스케줄링 알고리즘에서 스케줄 가능하다고 말할 수 없는 태스크의 집합에 대해서도 스케줄링 가능성을 검사할 수 있다. 반면에 실행시의 오버헤드가 큰 것이 단점이다. 이로 인해 태스크의 종료시간을 초과할 가능성이 있고 실제로 한 태

스크의 종료시간이 초과되면 연쇄적으로 다른 태스크의 종료시간마저 초과되어 버리는 도미노 현상이 일어날 가능성을 가진다. EDF(Earlist Dealline First)알고리즘 기법의 동적 우선 순위 기반 스케줄링 방법은 스케줄링 성능면에서 정적 우선 순위 기반 스케줄링 방법보다 우수하다. 그러나, 시스템에 대한 예측성 측면에서는 정적 우선 순위 기반 스케줄링 방법보다 못하다. RM알고리즘 기법에 의해서 스케줄 되는 시스템은 높은 우선 순위의 태스크가 언제 실행이 시작되어 언제 종료될 지에 대한 정확한 예측이 어느 정도 가능하다. 이는 태스크의 우선 순위가 정적으로 고정되어 있기 때문이다. 그러나, EDF 알고리즘 기법에 의하면 중요도가 높은 태스크라 할지라도, 그 태스크의 어떤 작업은 낮은 우선 순위를 동적으로 부여받을 수 있기 때문에 태스크의 실행 형태에 대한 예측을 정확하게 하기 어렵다는 단점을 가진다.

Ⅲ. 스케줄링 가능성 알고리즘

3.1 스케줄링 가능성(Schedulability)

전자무역에서는 태스크 작업 이용율을 높이는 방향으로 스케줄링을 수행할 뿐 태스크의 마감시간에는 구애받지 않는다. 새로운 태스크가 실행을 요구해 왔을 때 그 실행을 거부하지 않으므로 RTS에서는 적합하지 않다. 태스크로 인해 시스템이 과부하 상태일 경우 그 태스크와 새로운 태스크뿐만 아니라 기존의 태스크들 역시 마감시간을 지키지 못할 수 있기 때문이다. 결과적으로 RTS에서는 시분할 스케줄링 방법과는 다른 스케줄링 방법이 필요하며 또한 새로운 태스크의 실행을 허가하는 것이 시스템 전체의 안정성을 손상시키는지 검사하기 위한 방법이 필요하다. 각 태스크가 만기 전에 실행을 완료하였을 경우, 해당 스케줄이 가능하다면 해당 태스크들의 집합은 스케줄링 가능성 알고리즘에 의해 스케줄 가능하다 라고 말할 수 있다. 스케줄링 가능성에서 예측성이란 시스템에 정의된 고장이나 작업 부하 조건에 태스크 종료시간의 만족을 보장하는 것을 의미한다. 스케줄링 가능성 검사는 우선순위 구동방식의 알고리즘을 유용하게 사용할 수 있는 기법이다. 시스템에 적합한 스케줄링 기법을 선택하고 태스크의 주기시간, 실행시간, 마감시간 등을 결정할 때 유용하게 판단 할 수 있다. 이러한 이유로 시스템에 안정적인 스케줄링을 할 수 있도록 다중 프로세서 시스템에서 우선 순위 구동 알고리즘이 많이 사용되고 있다. 스케줄이 정적으로 결정되어 있기 때문에 이러한 스케줄링 방법을 오프라인 스케줄링 이라고 한다. 실시간 스케줄링 방식은 시분할 스케줄링 방식에서와 마찬가지로 선점 가능한 방식과 불가능한 방식으로 구분되며 대부분 스케줄링 방법은 선점형 방식이다.

3.2 기존 알고리즘

태스크는 실행시간, 마감시간, 주기시간 파라미터 값이 미리 설정되어 있고 예측 스케줄링 공식을 이용하여 안정적인 스케줄링이 가능한지를 사전에 예측할 수 있게 하는 방법이다.

표 1. 태스크 조건
Table 1 Condition of Task

기호	내 용	비 고
n	태스크의 수	오름차순으로 우선 순위 설정
U	전체 태스크의 이용율 값	$=\sum_{i=1}^{n}\frac{C}{P}$ $=\sum_{i=1}^{n}\frac{C+(P-D)}{P}$,
Util[i]	개별 태스크의 이용율 값	$=\frac{C+(P-D)}{P}$
C	태스크의 실행시간	사전에 실행시간 값을 설정
D	태스크의 마감시간	사전에 마감시간 값을 설정
P	태스크의 주기시간	사전에 주기시간 값을 설정

스케줄링 가능성 예측 이용율 공식은 다음과 같다. 정적 스케줄링 이용율 공식 $U=\sum_{i=1}^{n}\frac{C}{P}$, 동적 스케줄링 이용율 공식 $U=\sum_{i=1}^{n}\frac{C+(P-D)}{P}$ 이며 <그림 3>에서 U는 스케줄링 이용율 값을 가진다.

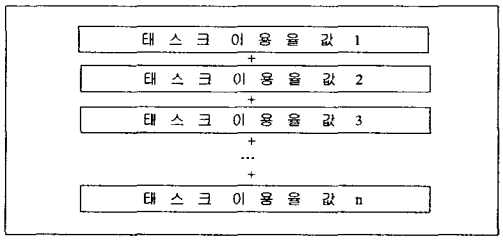

그림 3. 태스크 이용률 구조
Fig. 3 Structure of Task Availability Ratio

(1) 주기시간과 마감시간 비교처리

모든 태스크의 주기시간과 마감시간을 서로 비교하여, 동일하면 RMS변수의 값을 "TRUE"라고 설정하며, 태스크의 주기시간과 마감시간이 하나 이상 다를 때는 판단조건에서 정적 우선 순위 알고리즘인 RM 스케줄링 할 수 없고, RMS 변수의 값은 "FALSE"라고 설정을 한다.

```
……/* 태스크 집합에 대해 주기와 종료시간을 검사하여 동일한가를 판단*/
RMS = TRUE; for( i=0 ; i < 태스크 수 ; i++) {if( 주기시간(P) == 마감시간(D) ) looping; else { RMS = FALSE; brack; }}……
```

- 주기시간(P) : 5, 마감시간(D) 5이면 RMS 변수 값은 "TRUE"
- 주기시간(P) : 5, 마감시간(D) 3이면 RMS 변수 값은 "FALSE"

(2) 정적 · 동적 이용율 공식 처리

```
…… /* 태스크 집합에 대한 스케줄 가능성 검사 */
if(RMS) { U = U + 실행시간(C) / 주기시간(P);
} else { U=U+( 실행시간(C) + ( 주기시간(P) - 마감시간(D) ))
/ 주기시간(P);……
```

마감시간과 주기시간 동일 할 때 (전체 태스크 이용율 값)

$$\sum_{i=1}^{n} \frac{C}{P} \rightarrow \boxed{U}$$

마감시간과 주기시간이 다를 때 (전체 태스크 이용율)

$$\sum_{i=1}^{n} \frac{C+(P-D)}{P} \rightarrow \boxed{U}$$

RMS 변수 값이 "TRUE"라고 가정을 하면 마감시간과 주기시간이 동일하므로 정적인 스케줄링 이용율 공식에 U 변수 값에 태스크 이용율 값을 누적한다. RMS 변수 값이 "FALSE"이면 마감시간과 주기시간이 다른 경우이므로 U 변수 값에 동적 이용율 공식을 사용하여 이용율 값을 누적한다.

- RMS 변수 값이 "TRUE"이고 주기시간 5, 마감시간 5, 실행시간 2 이면 U = 0.4
- RMS 변수 값이 "FALSE"이고 주기시간 5, 마감시간 4, 실행시간 2 이면 U = 0.6

(3) 스케줄 가능조건에서 정적 · 동적 분기 처리

```
……
/* 태스크 집합에 대한 스케줄 가능성 검사*/
U = schedulable_check(NTASKS);
if( U ≥ 0 and U ≤ 태스크 수 * ( (2 1/태스크수) - 1)) // RM 스케줄링 가능 조건
  SCHED_OK = TRUE; else if( U ≥ 0 and U ≤ 1) // EDF 스케줄링 가능조건
  SCHED_OK = TRUE;
  else { SCHED_OK = FALSE; break; }
/* 스케줄링 불가능 */ ……
```

RM 스케줄링 가능조건과 전체 이용율 값인 U를 비교하여 참 일 경우 SCHED_OK 변수의 값이 "TRUE"를 설정해 둔다. 거짓일 경우는 EDF 스케줄링 가능조건과 전체 이용율 값인 U를 비교하여 참 일 때 SCHED_OK 변수의 값이 "TRUE"를 설정해 둔다. 이때 거짓일 경우에는 RM과 EDF 스케줄 가능 조건이 둘다 충족하지 못하므로 스케줄링을 수행할 수 없게 된다. <2개의 태스크가 있다고 가정하며 스케줄가능 조건(RM:0.813,EDF:1)과 이용율 값 0.74를 비교하면 RM, EDF 스케줄 가능하므로 SCHED_OK변수 값은 "TRUE" 설정한다>.

(4) 선택 알고리즘 스케줄링 분기 처리

```
… … if(SCHED_OK and RMS) /*
RMS로 태스크 스케줄링 판별 */
RM 스케줄링 수행; else
EDF 스케줄링 수행; … …
```

선택 스케줄링 구분은 SCHED_OK 변수 값이 "TRUE"이면 정적 또는 동적 알고리즘의 조건에 충족됨을 알 수 있다. RMS 변수 값에서 "TRUE"인 경우 마감시간과 주기시간이 동일하다. 정적인 스케줄링 조건은 RMS 변수값 "TRUE"에만 RM 스케줄링을 선택한다. 만일 RM 스케줄링 조건에 초과한 경우에 EDF 스케줄링을 수행한다. RMS 변수 값이 "FALSE"일때에도 동적 스케줄링을 수행한다.

(5) 전체 태스크 이용율 알고리즘

```
double schedulable_check(NTASKS)
 {/* 태스크 집합에 대해 주기와 종료시간을 검사하여 동일한가를 판단
*/for(i=0;i<NTASKS;i++)  if(Parameters[i].period==Parameters[i].deadline) looping;
------- ①
   else { RMS = FALSE; brack; }
 /* 태스크 집합에 대한 스케줄 가능성 검사 */
 if(RMS) { /* RM인 경우 스케줄링 가능성 검사 */ ---------- ②
  U = U + Parameters[i].compute / Parameters[i].period;
 } else { /* EDF인 경우 스케줄링 가능성 검사 */
 U= U + (Parameters[i].compute+(Parameters[i].period-Parameters[i].deadline))
 / Parameters[i].period;
 }return (U);}
```

① 태스크의 실행시간, 마감시간, 주기시간 등 파라미터 값에 의해서 마감시간과 주기시간이 같은지를 비교하

여 RMS 변수값은 "TRUE", "FALSE" (RM 알고리즘과 EDF알고리즘 이용율 공식)둘중 하나를 설정한다.

② RMS 변수값을 "TRUE"이면 $U = \sum_{i=1}^{n} \frac{C}{P}$과 같이 계산하고, "FALSE"일 경우 $U = \sum_{i=1}^{n} \frac{C+(P-D)}{P}$ 이용율 집합을 각각 틀리게 계산하게 된다.

③ 고정 우선 순위 기반 스케줄링 방식으로 가장 널리 사용되고 있는 방법은 RM(Rate Monotonic)스케줄링 기법으로 Liu와 Layland에 의해 제안된 스케줄링 가능성 분석 조건에 의해 가능한지 불가능한지를 판단한다.

④ 동적 우선 순위 기반 스케줄링 방식으로 가정 널리 사용되고 있는 EDF(Earlist Deadline First) 스케줄링 기법은 Liu와 Layland에 의해서 제시되었다. EDF기법은 전체 프로세서 이용을 Fixed Priority Scheduling Utilization ($\sum_{i=1}^{n} \frac{C}{P} \leq 1$)를 이용하고 있다. 만일 불가능하다면 이 파라미터의 값은 스케줄링 불가능하다고 판별하게 되어져 있다.

⑤ RMS 변수값과 SCHED_OK 변수값은 서로 참 값 인지를 비교하여 스케줄링을 동적 스케줄링 방식(EDF) 과 정적인 스케줄링 방식(RM)으로 선택하도록 분기하는 역할을 해준다.

⑥ RM 스케줄링은 태스크 주기의 역수인 빈도율(Rate) 이 높을수록, 즉 주기가 짧을수록 더 높은 우선 순위를 부여하는 방식이다.

⑦ EDF 스케줄링은 새로운 작업이 도착할 때마다 현재 실행중인 작업의 마감시간을 서로 비교하여 새로운 작업의 우선순위를 결정한다. 이때 마감시간이 빠를수록 더 높은 우선 순위를 부여받게 된다.

(6) 기존 선택 알고리즘

```
double schedulable_check(NTASKS)
    {/* 태스크 집합에 대해 주기와 종료시간을 검사하여 동일한가를 판단
*/for(i=0;i<NTASKS;i++)
    if(Parameters[i].period==Parameters[i].deadline) looping; ------- ①
      else { RMS = FALSE; brack; }
    /* 태스크 집합에 대한 스케줄 가능성 검사 */
     if(RMS) { /* RM인 경우 스케줄링 가능성 검사 */ ---------- ②
      U = U + Parameters[i].compute / Parameters[i].period;
     } else { /* EDF인 경우 스케줄링 가능성 검사 */
    U= U + (Parameters[i].compute+(Parameters[i].period-Parameters[i].deadline))
    / Parameters[i].period; }return (U);}
Scheduler_Select()
{ /* 태스크 집합에 대한 스케줄 가능성 검사 */
 U = schedulable_check(NTASKS);
 if( U >= 0 && U <= NTASKS * ( POW(1/NTASKS))-1) ) --- ③ SCHED_OK = TRUE; /*
  RM, EDF 스케줄 가능 */
 else if( U >= 0 && U <= 1) ---------- ④ SCHED_OK = TRUE; /* EDF 스케
  줄 가능 */
  else { SCHED_OK = FALSE; break;
 /* 스케줄링 불가능 */
 if(SCHED_OK && RMS) /* RMS로 태스크 스케줄링 판별 */ --- ⑤ RM 스케줄링
  수행; ---------- ⑥ else EDF 스케줄링 수행;
  ---------- ⑦ return 0; }
```

3.3 제안 알고리즘

(1) 주기시간/마감시간 이용율 값 비교처리

```
.....
RMS='Y'; for( i=0 ; i < 태스크 수 ; i++) {
 if( 주기시간(P) != 마감시간(D) )
RMS='N';
 Util[i] = ( 실행시간(C) + ( 주기시간(P) - 마감시간(D) ))
/ 주기시간(P) .....
```

단 하나 이상의 태스크가 마감시간과 주기시간이 다른 경우 RMS변수 값에 'N'이라고 설정하고, 마감시간과 주기시간이 같다면 RMS변수 값이 'Y'값이 설정되어 있다. 스케줄링 가능성 조건 처리는 다음과 같다.,

주기시간(P) : 5, 마감시간(D) 5이면 RMS 변수 값은 'Y'
주기시간(P) : 5, 마감시간(D) 4이면 RMS 변수 값은 'N'
주기시간 5, 마감시간 5, 실행시간 2 이면 Util[1] = 0.4
주기시간 5, 마감시간 4, 실행시간 2 이면 Util[1] = 0.6

(2) 적합한 스케줄링 판정 처리

```
.....
if( RMS == 'Y' and RM_count == 태스크 수)
  RM schedule; // RM 스케줄링 실행
else if( EDF_count == NTASKS )
  EDF schedule; // EDF 스케줄링 실행
else return -1; // RM, EDF 스케줄링 불가능
.....
```

RM 스케줄링 가능검사에서, RMS 변수 값이 'Y'이면서 RM_count 수와 전체 태스크 수가 동일하면 RM 스케줄링 수행이 된다. RM 스케줄링 가능검사에서 스케줄링 가능성 조건이 충족되지 않으면서 EDF 스케줄링 가능검사에서 EDF_count 수와 전체 태스크 수가 같으면 EDF 스케줄링을 수행, RM, EDF 카운터 수가 서로 다르면 RM과 EDF의 스케줄링 수행 불가판정을 하게 된다.

(3) 개별 태스크 이용율 알고리즘

```
void schedulable_check(void){int i; RMS='Y';
for(i=0;i<NTASKS;i++)
{if(Parameters[i].period!=Parameters[i].deadline)---- ①
RMS='N'; Util[i =(Parameters[i].compute+(Parameters[i].
period-Parameters[i].deadline))/ Parameters[i].period; --- ②
```

태스크 수만큼 수행, 이용율 공식에 의해서 각기 저장된 고유의 변수 값을 점차적으로 누적하면서 태스크 이용

율이 스케줄 가능조건에 있는 정적 스케줄링 RM은 이용율 조건이 total >= 0에서 total <= n * (21/n - 1) 공식을 대입하여 가능하면 RM_count변수 값 하나를 증가시킨다. 마찬가지로, 동적 스케줄링 (EDF) 가능조건을 보면 total >= 0 에서 total <= 1까지 태스크가 수행가능 하다고 판단하여 가능하다고 하면 EDF_count 변수 값을 하나 증가시킨다. 기존에 제안한 알고리즘은 단지 이용율 집합 값의 전체를 가지고 정적 및 동적 스케줄링이 수행가능 한지를 판별하였으나 본 연구는 RM과 EDF 스케줄링 이용율 조건으로 각각의 태스크 문제를 알아 보고 사전에 마감시간과 주기시간이 초과하는 한계성을 예측 해 보았다.

V. 결 론

본 논문은 전자무역에서 활용되고 있는 RTS 태스크 구조 및 정적·동적 스케줄링 기법과 기존 스케줄링 조건, 제안한 스케줄링 가능성 알고리즘에 대하여 연구하였다. 전자무역에서 RTS의 기존 태스크 전체 이용율 알고리즘을 이용하여 주기가 설정된 값을 가지고 스케줄링 가능성 검사 판정시 전체 e-트랜스포메이션 이용율에서는 전체 집합의 이용율 값에 대해서 판단하는 것은 오직 실행 가능성과 실행 불가능성의 방법 두 가지만 존재함을 알 수 있다. 본 논문에서 제한한 알고리즘은 개별 이용율이며 개별 집합의 이용율 값을 두어 실행가능조건을 판별시 사전에 몇 번째 태스크가 문제를 발생하는지 개별 태스크 이용율 값을 사용하여 한계성에 대해서 예측 가능하다. 제안된 알고리즘이 수행하고 마감시간과 주기시간에 따라 스케줄링을 판단하는 조건에 의해서 재구성된 선택 알고리즘은 정적 알고리즘과 동적 알고리즘으로 선택될 수 있도록 설정되어져 있다. 본 논문에서 알고리즘에 대한 평가는 스케줄링을 수행하기 이전에 마감시간, 실행시간, 주기시간의 태스크 값을 가지고 스케줄링 가능성 검사의 한계성을 예측하여 RTS에서 사전에 안정적인 무역자동화 시스템을 유지하는 것이다. 본 논문은 단순 이용율 공식을 이용하여 사전에 스케줄링 가능성 판별 조건에만 이용된 점이며 실시간 스케줄링 실험에서는 예상치 못한 결과 값을 도출할 가능성이 있다. 스케줄링 가능성 기법은 주로 주기적인 태스크를 사용하였지만 이 방법 외의 다른 태스크 시간 조건과 비주기적인 태스크 시간 조건을 이용하여 보다 안정적인 사전 예측 스케줄링 가능성 알고리즘의 연구가 향후 필요하다.

참고문헌

[1] An Fredette and Rcleaveland, "A Generalized to Real-Time Schedulability Analysis", 10th workshop on real-time operating system and software, pp. 35-54, 2000.

[2] K.M. Zuberi, K. G. Shin, "EMERALDS: A Microkernel for Embedded Real-Time System", in Proc, Real-Time Technology Applications Symposium, pp. 241-249, June 1996.

[3] Krithi Ramamritham, John A. Stankovic, "Scheduling Algorithms and Operating System Supports for Real-Time Systems," Proceedings of the IEEE. Vol. 82, No.1, January pp. 55-67, 1999.

[4] Manabe, Y. and Aoyagi, S. "A Feasibility Decision Algorithm for Rate Monotonic and Deadline Monotonic Scheduling", RT systems, v.14, no.2, pp.171-182, 1998.

[5] Krithi Ramamritham, John A. Stankovic, "ibid".

[6] B. A. Blake K. Schwan, "Experimental evaluation of a real-time scheduler for a multiprocessor system," IEEE Trans, Software Eng., vol 17, no.1, Jan. 1998.

[7] An Fredette and Rcleaveland, "ibid".

[8] L. Sha, R. Rajkumar, and J. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. IEEE Transactions on Software Engineering, 39: pp.1175-1185, September 1999.

[9] N. Audsley et al., "Hard Real-Time Scheduling : The Deadline Monotonic Approach," Proc. of IEEE Workshop on Real-Time Operating Systems, pp.251-282, 1997.

[10] Peterson, Silberschatz, "Operating System Concepts" Prentice-Hall. 2002.

[11] J. Lehoczky, L. Sha, and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior", In Proceedings of the 10the Real-Time System Symposium, pp.166-171, 1999.

[12] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", Journal of the ACM, vol. 20, no.1, pp.46-61, 1993.

[13] Chetto. H and Chetto. M, "Some Results of the Earliest Deadline scheduling Algorithm", IEEE Transactions on Software Engineering vol.15, no.10 , pp.1261-1269, Oct. 2001.

저자소개

정 분 도(Boon-Do Jeong)

조선대학교 지역사회발전연구원
전임연구원 역임
조선대학교 경영경제연구소
전임연구원 역임

현재, 조선대학교 경제무역학부 초빙객원교수(경영학박사/ 전자무역 전공)
※ 관심분야 : 정보통신 행정 및 정책