Journal of Institute of Control, Robotics and Systems (2012) 18(4):302-307
http://dx.doi.org/10.5302/J.ICROS.2012.18.4.302 ISSN:1976-5622 eISSN:2233-4335

위상 정렬과 여유 시간 기반 주기 및 실시간 비주기 태스크 스케줄링 알고리즘

Periodic and Real-Time Aperiodic Task Scheduling Algorithm based on Topological Sort and Residual Time

김 시 완, 박 홍 성*
(Si Wan Kim[1] and Hong Seong Park[1])
[1]Kangwon National University

Abstract: Real-time systems perform periodic tasks and real-time aperiodic tasks such as alarm processing. Especially the periodic tasks included in control systems such as robots have precedence relationships among them. This paper proposes a new scheduling algorithm based on topological sort and residual time. The precedence relationships among periodic tasks are translated to the priorities of the tasks using topological sort algorithm. During the execution of the system the proposed scheduling algorithm decides on whether or not a newly arrived real-time aperiodic task is accepted based on residual time whenever the aperiodic task such as alarm is arrived. The proposed algorithm is validated using examples.

Keywords: real-time scheduling, precedence, topological sort, residual time, periodic task, aperiodic task

I. 서론

일반적으로 실시간 시스템은 실시간 비주기 태스크와 주기 태스크, 비실시간 태스크들로 이루어 진다. 주기 태스크는 특정 주기에 따라 수행하는 태스크로, 데이터를 생성하거나 다른 태스크로부터 데이터를 받아 처리한 후 전송하는 태스크들과 전송된 데이터를 처리만 하는 태스크로 나눌 수 있다. 로봇 제어 시스템을 예로써 생각하면, 로봇의 각 센서로부터 생성된 데이터를 읽어오는 태스크, 해당 센서 데이터들을 사용하여 제어 데이터를 계산하는 태스크와 전송된 제어 데이터를 액츄에이터로 출력하는 태스크가 상호 작용하여 로봇을 제어하게 된다. 즉, 이러한 태스크들은 동작에 있어서 선행관계가 존재하기 때문에, 가능한 한 주기 안에 모두 수행이 되어야 하고 동시에 정해진 선행 제약에 따라 데이터를 생성하고 사용하여야 한다.

일반적으로 주기 태스크와 관련된 실시간 시스템의 스케줄링은 NP-Hard 문제로 알려져 있기 때문에 휴리스틱(heuristic) 방법을 사용하여 최적보다는 차선의(suboptimal) 해결 책을 얻는 기법들이 연구되어 왔다[1,2,14,15]. 알람 같은 불특정 시간에 발생하는 이벤트들을 처리하기 위한 실시간 비주기 태스크와 주기 태스크를 함께 처리하는 스케줄링 관련 연구도 수행되었다[3,4,8,12,13].

실시간 스케줄링에 관한 기존 연구를 살펴보면 스케줄링이 이루어 지는 시점에 따라 온라인(on-line) 스케줄링과 오프라인(off-line) 스케줄링으로 나눠진다[5]. 온라인 스케줄링은 유연성이라는 장점을 가지고 있지만 태스크간 선행 제약 등의 문제를 해결할 수 없는 문제점을 지니고 있다. 오프라인 스케줄링 방법은 시스템의 수행 전에 주기 태스크들의 기동 시간이나 우선순위들을 미리 할당하는 방법으로 불특정 시간에 들어오는 비주기 실시간 태스크의 스케줄링에는 적합하지 않다.

오프라인 스케줄링의 대표적인 방법인 Rate monotonic [6]과 온라인 스케줄링의 대표적인 방법인 Earliest Deadline First [7,10] 방법은 모든 주기 태스크가 독립적으로 수행 된다는 가정하에 이루어지는 스케줄링 방법으로 선행 관계에 있는 주기 태스크들을 스케줄링 하기에는 적당하지 않다. 반면 주기 태스크들의 초기 기동 시간(offset-time)을 조정하는 스케줄링 방법[2,8,12,13]은 오프라인 스케쥴링 방법으로 주기 태스크들간의 선행 관계를 만족하지만 스케줄링 시 초기 기동 시간을 계산해야 하는 오버헤드가 발생하고 불특정 시간에 도착하는 비주기 실시간 태스크에 대하여 고려하지 않았다. 비주기 실시간 태스크와 주기 태스크 모두를 고려한 EDF 기반의 방법[3]과 TBS (Total Bandwidth Server) [4] 기반의 온라인스케줄링 방법들은 주기 태스크의 선행 관계를 고려하지 않고 있기 때문에 센서 입력 태스크로부터 제어기 태스크, 액츄에이터 출력 태스크까지 한 주기 내에 실행될 수 있지만, 특정 주기에서 센서로부터 입력한 데이터가 액츄에이터까지 출력할 때까지 해당 주기 내에 순서대로 수행된다는 것을 보장하지 않는다. 즉, 선행 관계를 고려하지 않고 액츄에이터 출력 태스크, 제어기 태스크, 센서 입력 태스크 순으로 주기내에서 수행하면, 한 센서로부터 입력한 데이터의 제어 결과는 3 주기 이후에 출력되기 때문에 [3]과 [4]의 스케쥴링 방법은 알맞지 않다. Slack Steeling [9] 방식은 온라인 스케줄링 방법으로 모든 슬랙을 미리 계산 하여 테이블을 생성하는 방식이다. 주기 태스크의 수가 많아지면 테이블이 사용하는 메모리의 크기 또한 커진다는 단점이 있다.

본 논문은 주기 태스크들의 주기와 위상 정렬 방법

* 책임저자(Corresponding Author)
논문접수: 2011. 10. 12., 수정: 2011. 11. 24., 채택확정: 2012. 3. 8.
김시완, 박홍성: 강원대학교 전자통신공학과
(malshan@control.kangwon.ac.kr/hspark@control.kangwon.ac.kr)

(topological sort) [11]을 통하여 오프라인 상에서 우선 순위를 활용하여 주기 태스크들의 선행 관계를 만족하는 오프라인 스케줄링 방법과 오프라인으로 스케줄링된 주기 태스크들을 기반으로 주기 태스크들의 수행 후 여유시간(residual time)을 사용하여 비주기 실시간 태스크의 수락 여부를 결정하고 비주기 실시간 태스크가 수행가능하면 우선 순위를 할당하는 온라인 스케줄링 방법을 제안한다. 이때 같은 주기의 주기 태스크를 하나의 집합을 사용하여 표현 하고 여유 시간을 구하기 위해 주기 태스크 집합의 파라메터(주기, 수행 시간, 발생 빈도)를 동적으로 갱신하여 여유 시간을 구하기 때문에 테이블 생성의 메모리 낭비를 줄이고 주기 태스크 집합을 사용하여 계산 량을 줄였다. 또한 주기 태스크의 임계시간을 사용하여 비주기 실시간 태스크의 빠른 응답 시간을 만족 시켰다.

본 논문의 구성은 II 장에서 시스템 모델링에 관해 기술하고, III 장에서는 선행 관계의 주기 태스크를 위상 정렬 방법을 활용한 오프라인 스케줄링 방법을 설명하고 예제를 통하여 검증한다. IV 장에서는 스케줄링된 주기 태스크를 기반으로 초월 주기(hyper period)상에서 주기 태스크의 수행 후 남은 여유 시간을 사용하여 비주기 실시간 태스크의 온라인 스케줄링 방법을 설명하고 예제를 사용하여 검증한다. 마지막 장에서 결론을 맺는다.

II. 시스템 모델링

시스템 상의 i 번째 주기 태스크 τ_i는 $\{e_i, T_i, p_i, E_i, N_i^p\}$ 의 다섯 가지 요소로 그 특성이 표현 된다. 여기서 e_i, T_i, p_i는 주기 태스크 τ_i의 수행시간, 주기, 우선 순위를 나타내고 E_i는 τ_i에서 생산된 데이터를 소비하는 태스크로 연결되는 간선(Edge) 혹은 소비자 태스크의 집합, N_i^p 는 τ_i가 사용하는 데이터를 생산하는 생산자 태스크의 수 이며 진입 차수(input degree)라고 한다. 주기 태스크의 우선순위는 제안하는 알고리즘을 통해 구해지며 가장 높은 우선순위는 1이고 오름차순으로 우선순위가 낮아진다.

태스크간 선행 관계는 방향성을 가지는 비순환 그래프(DAG: Direct Acyclic Graph)이며 $G(V, E)$로 표현할 수 있다. 이때 V는 시스템 상의 모든 정점(vertex) 혹은 태스크(τ_i)의 집합이고 E는 시스템 상의 모든 간선(edge)의 집합 혹은 생산자 태스크 τ_i τ_i의 소비자 태스크의 집합인 E_i의 집합으로 표시된다. 간선의 집합은 생산자 태스크와 소비자 태스크의 쌍으로 표시된 집합이다. 그림 1과 같이 주기 태스크들의 선행 관계는 화살표를 통하여 나타내고 각 주기 태스크들은 모두 다수의 생산자 태스크와 다수의 소비자 태스크를 가질 수 있다. 생산자와 소비자 관계에 따라 모든 소비자 태스크들은 생산자 태스크가 수행되기 전에는 수행이 불가능하다고 가정한다. 이 가정은 생산자 태스크가 같은 주기 내에 생성한 결과를 소비자 태스크가 사용한다는 관점에서 매우 합리적인 가정으로 동기화 조건이라 한다. 그림 1의 예에서 소비자 태스크인 τ_4는 생산자 태스크 τ_2와 τ_3이 모두 수행 완료되어야 수행이 가능하다. 이러한 생산자와 소비자 관계에서 주기 태스크의 동기화 조건을 만족하기 위해서 각 주기 태스크의 주기는 하모닉스 관계를 가지도록 할당한다[2,8]. 여기서 하모닉스 관계란 생산자와 소비자 태스크의 주기를 정수 배의 관계를 가지게 함으로써 소비자 태스크가 자신의 주기 안에서 생산되는 생산자의 데이터를 사용할 수 있도록 하는 것이다.

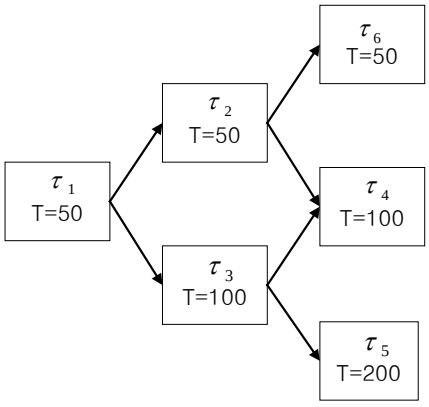

그림 1. 주기 태스크 그래프의 예.
Fig. 1. Example of period task graph.

표 1. 그림 1의 주기 태스크 구성 요소 예.
Table 1. Period task element example of Fig. 1.

Task	e_i	T_i	E_i	N_i^P	p_i
τ_1	10	50	τ_2,τ_3	0	-
τ_2	10	50	τ_4,τ_6	1	-
τ_3	10	100	τ_4,τ_5	1	-
τ_4	10	100	Ø	2	-
τ_5	20	200	Ø	1	-
τ_6	5	50	Ø	1	-

불특정 시간 t에 발생하는 실시간 비주기 태스크는 $A(t)$로 표시하며 수행 시간과 우선순위의 쌍으로 이루어 진다. 그들은 각각 $A_e(t)$와 $A_p(t)$로 표시한다. 분석을 단순화 하기 위해 실시간 비주기 (이후 RTA: Real-Time Aperiodic이라 한다.) 태스크 에 대한 가정은 다음과 같다.

- RTA 태스크는 동시에 발생하지 않는다.
- RTA 태스크는 같은 우선순위를 갖는다.

또한 주기 태스크와 RTA태스크는 선점(preemption)방식으로 동작한다고 가정한다.

III. 주기 태스크 스케줄링

1. 위상 정렬 기반 우선순위 설정 알고리즘

그래프 $G(V, E)$에 존재하는 각 정점들의 선행 순서를 위배하지 않으면서 모든 정점을 나열하는 것을 그래프의 위상 정렬(topological sort)방법 이라고 한다[10]. 위상정렬을 사용하기 위해서 본 논문에서는 주기 태스크의 그래프 구조를 인접 리스트(adjacency list)를 사용하여 다시 표현하였다. 그림 2는 그림 1의 주기 태스크 그래프를 인접 리스트를 사용하여 표현한 예이다. 각 주기 태스크 τ_i는 자신의 소비자 태스크를 인접 태스크로 저장하고 있으며 이 인접 태스크는 주기 태스크 τ_i에서 나오는 간선 E_i와 같은 의미를 가진다. 그림 1의 주기 태스크 τ_1에 τ_2와 τ_3를 연결하기 때문에 $E_1 = \{\tau_2, \tau_3\}$ 이고 그림 2와 같이 태스크 τ_1에 연결된 인접리스트로 표현된다.

주기 태스크 그래프의 위상정렬 알고리즘은 기본적으로 위상정렬 알고리즘을 사용하지만 진입차수 N_i^p 가 0인 주기

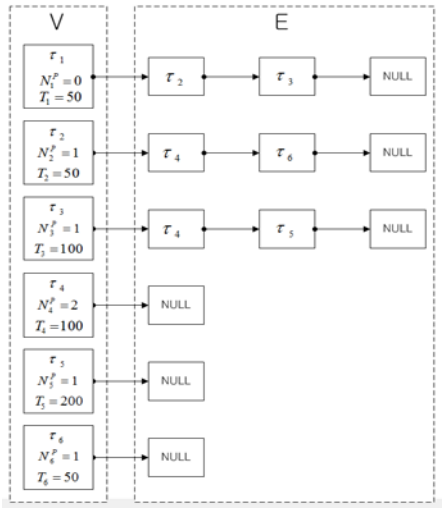

그림 2. 그림 1의 태스크 그래프 재구성 인접리스트.
Fig. 2. Task graph restructure adjacency list of Fig. 1.

```
/* input : G(V,E) 태스크 그래프 인접 리스트(Adjacency list) */
/* output : 우선순위 리스트(FIFS) */
/* Ls : N_j^P 가 0인 태스크 리스트 */
/* Lp : 우선순위대로 태스크가 저장되는 리스트(FIFS)*/

If (1)을 만족하지 않음
        exit algorithm
end

Foreach ∀ τ_i ∈ V do
    If (N_i^P = 0)
        Ls 에 T_i 가 작은 순으로 τ_i 를 정렬 삽입
    end
end

While V ≠ φ do
    Ls 의 첫 번째 τ_k 선택
    Foreach ∀ τ_j ∈ E_k do //τ_j 는 τ_k 의 인접 태스크
        τ_j 의 N_j^P 감소
        If (N_j^P = 0)
            Ls 에 T_j 가 작은 순으로 τ_j 를 정렬 삽입
        end
    end
    Lp 에 τ_k 삽입
    Ls 에서 τ_k 삭제
    V 에서 τ_k 삭제
end
```

그림 3. 위상 정렬 기반 우선순위 할당 알고리즘.
Fig. 3. Priority assignment algorithm based on topological sort.

태스크의 선택에 있어서 해당 주기 태스크의 주기를 비교하는 방식이 추가 되었다. 또한 모든 주기 태스크의 주기가 하모닉스 관계를 따르고 있으므로 (1)에 표시된 이용률 U가 1을 넘지 않는 가정하에 스케줄링 가능하다.

$$U = \sum_{i=1}^{n} \frac{e_i}{T_i} \le 1 \qquad (1)$$

그림 3은 주기 태스크 그래프로부터 각 주기 태스크에 우선순위를 할당하는 알고리즘이다. 그림 3에서 Ls는 N_i^p 가0인 태스크들이 주기가 작은 순으로 정렬되어 있는 리스트이다. Lp는 주기 태스크의 리스트로써 Lp에 저장된 순서대로 우선순위가 부여되고 구조는 선입선출(first in first out) 큐의 형태이다.

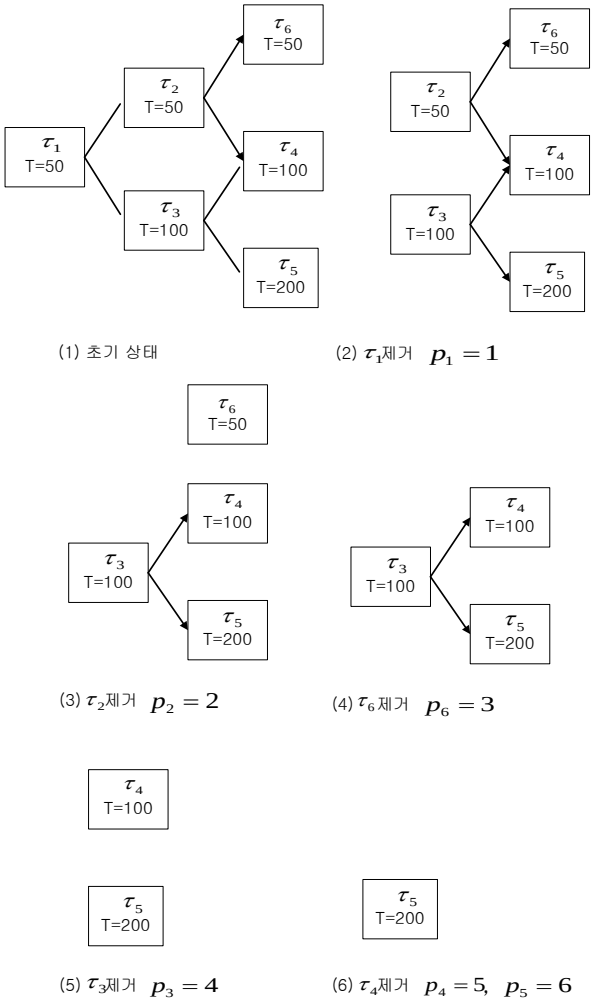

그림 4. 위상정렬 기반 우선순위 할당 예.
Fig. 4. Example of priority assignment based on topological sort.

표 2. 우선순위 할당 결과 예.
Table 2. Result of priority assignment.

Task	e_i	T_i	p_i
τ_1	10	50	1
τ_2	10	50	2
τ_3	10	100	4
τ_4	10	100	5
τ_5	20	200	6
τ_6	5	50	3

2. 알고리즘 검증

그림 1과 표 1의 주기 태스크를 사용하여 그림 3의 위상정렬 기반 우선순위 할당 알고리즘을 수행하면 그림 4와 같은 순서대로 주기 태스크가 그래프 $G(V,E)$에서 삭제되고, 삭제되는 순서대로 우선순위가 부여된다. 표 2는 할당된 우선순위를 보여주고 있다.

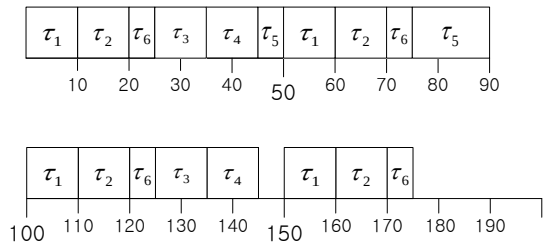

그림 5. 주기 태스크 타이밍 그래프.
Fig. 5. Period task timing graph.

그림 5의 타이밍 그래프는 우선순위 할당 후 우선순위에 따라 주기 태스크를 수행한 결과이다. 그림 5에서 볼 수 있듯이 모든 주기 태스크가 자신의 주기를 위배하지 않고 수행되고 또한 초기 기동 시간의 할당을 하지 않고 선행 관계를 만족하는 결과를 볼 수 있다.

IV. RTA 태스크 스케줄링

주기 태스크의 초월 주기(Hyper Period)를 구한 후 초월 주기 안에서 주기 태스크가 수행 되고 남은 여유 시간(residual time)을 사용하여 RTA 태스크의 수락 및 제어를 한다. 이때 RTA 태스크의 응답시간을 최소화 하기 위해서 임계 시간(critical time)이 임박한 주기 태스크가 없을 때까지 RTA 태스크의 우선순위를 최상위로 할당하게 된다.

초월주기는 시스템에 속한 모든 주기 태스크의 최소 공배수이며 (2)와 같이 H로 표현한다.

$$H = LCM\{T_1, T_2, ..., T_n\} \tag{2}$$

우선순위 할당 알고리즘에 의해 할당된 우선순위를 보면 주기가 작은 순서대로 우선순위가 할당된 것을 알 수 있다. 또한 같은 주기의 주기 태스크들은 우선순위가 다르게 할당되지만 우선순위가 연속하여 배열되어 있다. 본 논문에서는 온라인 스케줄링 수행 전에 계산시간을 절약하기 위해 같은 주기의 주기 태스크를 하나의 집합(set)으로 정의하고 S_j로 표현한다. S_j는 $\{T_j^s, N_j, e_j^s, P_j, C_j, \overline{C}_j\}$ 의 6개의 요소를 가진다.

T_j^s 는 집합 S_j에 속한 주기 태스크의 주기이다. N_j은 초월주기 H안에서 S_j가 발생 하는 횟수로 (3)을 통해서 구할 수 있다. N_j는 S_j의 주기가 끝나는 시점에서 1감소한다. (t_s는 초월주기의 시작 시간)

$$N_j(t) = \frac{H}{T_j^s} - \left\lfloor \frac{t - t_s}{T_j^s} \right\rfloor \tag{3}$$

e_j^s 는 S_j에 속한 태스크의 수행 시간의 합으로 (4)로 표현된다.

$$e_j^s = \sum_{\forall \tau_i \in S_j} e_i \tag{4}$$

P_j는 S_j에 속한 태스크들이 현재까지 수행된 시간을 나타내고 초기 값 혹은 S_j주기 시작 시 값은 0이며 (5)로 표현된다. ($e_i(t)$ 는 S_j의 주기 시작 시간에서 t까지의 수행 시간)

$$P_j(t) = \sum_{\forall \tau_i \in S_j} e_i(t) \tag{5}$$

시간 t에 도착한 RTA 태스크의 수락 여부를 결정하기 위해 시간 t에 주기 태스크의 수행 후 남는 여유 시간을 구해야 한다. 여유 시간은 $R(t)$로 표현하고 $R(t)$는 (6)과 같다.

$$R(t) = H - t - \sum_{\forall S_j} \{(e_j^s \times N_j) - P_j\} \tag{6}$$

시간 t에 도착한 RTA 태스크 $A(t)$에 대하여 여유시간 $R(t)$가 RTA 태스크의 수행시간 $A_e(t)$ 보다 크면 수행되며, 이것은 (7)로 표현된다.

$$R(t) \geq A_e(t) \tag{7}$$

C_j는 임계시간 값으로 주기 태스크가 자신의 주기를 만족하며 수행 될 수 있는 경계의 오프셋(Offset)시간 이다. 이 임계시간을 넘기면 주기 태스크는 자신의 주기를 벗어나게 된다. C_j 값은 EDL알고리즘의 유휴 시간 테이블 생성과 같은 방법으로 구해진다[9]. $\overline{C}_j$ 은 S_j가 한 주기를 마쳤을 때 또는 자신보다 우선 순위가 높은 주기태스크 혹은 RTA 태스크에 의해서 선점(preemption)이 발생 했을 때 다시 구하게 되는 임계 시간으로, RTA 태스크가 도착하지 않았거나 선점이 일어나지 않으면 C_j 값과 동일한 값을 유지한다. (8)를 통해 $\overline{C}_j$ 를 다시 갱신 한다

$$\overline{C}_j(t) = t + C_j \tag{8}$$

$\overline{C}_j$ 갱신 후 자신의 자신의 $\overline{C}_j$ 보다 빠른 임계시간을 가지는 S_k의 $\overline{C}_k$ 값도 (9)를 사용하여 다시 갱신해 주어야 한다. 그 이유는 빠른 임계시간을 가지고 있다는 것은 스케줄링 동안에 우선순위가 높은 태스크에 의해 선점을 당하는 주기 태스크를 의미 하므로 선점을 하는 주기 태스크 집합의 임계시간이 늦춰진 만큼 선점을 당하는 주기 태스크 집합의 임계시간 역시 여유가 생겨 늦춰 주어야 한다. ($\overline{C}_k < \overline{C}_j$ 인 S_k에 대하여)

$$\overline{C}_k(t) = t + C_k \tag{9}$$

```
/* 실시간 비주기 태스크 A(t) 도착 시*/

If (6)과 (7)을 사용하여 스케줄 가능성 검사 통과
    Ap(t)를 최상위로 할당
    (8)을 사용하여 임계시간 재할당
    If C̄k < C̄j
      (9)를 사용하여 임계시간 재할당
    end
    우선순위에 따른 주기 및 실시간 비주기 태스크 스케줄링
    If 임계시간이 임박한 주기 태스크 발생
        Ap(t)를 최하로 할당
        If 임계시간이 임박한 주기 태스크의 수행 종료
            Ap(t)를 최상으로 할당
        end
    end
Else
    Reject A(t)
end
```

그림 6. RTA 태스크 스케줄링 알고리즘.
Fig. 6. RTA task scheduling algorithm.

RTA 태스크 도착 시 스케줄링의 알고리즘을 정리하면 그림 6과 같다. RTA 태스크의 도착이 없는 시점에서는 주기 태스크의 우선순위에 따른 수행이 이루어지고 RTA 태스크가 도착하면 (7)을 사용하여 RTA 태스크의 스케줄링 가능성(schedulability)을 검사한다. 검사를 통과하지 못하면 수락을 거부(reject)한 후 주기 태스크의 스케줄링을 계속 수행한다. 만약 RTA 태스크가 (7)의 검사를 통과하면 RTA 태스크의 우선순위를 최상으로 할당 하고 (8)과 (9)를 사용하여 주기 태스크들의 임계시간을 할당 한 후 우선순위에 따라 주기 및 RTA 태스크를 스케줄링 한다. 이때 임계시간에 임박한 주기 태스크가 있을 시 RTA 태스크의 우선순위를 최하위로 재할당하여 임계시간이 임박한 주기 태스크의 수행이 끝날 때까지 RTA 태스크의 수행을 지연 시킨다. 임계 시간이 임박한 주기 태스크가 수행 완료되면 다시 RTA 태스크의 우선순위를 최상으로 할당하여 RTA 태스크를 수행 시킨다.

표 2의 태스크를 사용하여 시간 110에서의 태스크의 집합 $G_1=\{\tau_1,\tau_2,\tau_6\}$, $G_2=\{\tau_3,\tau_4\}$, $G_3=\{\tau_5\}$ 를 구하여 정리하면 표 3과 같다.

만약 시간 t=110에 수행 시간이 30인 RTA 태스크 $A(110)$이 도착 했을 때 식(6)을 사용하여 $R(110)$를 구하면 30이고 $A_e(110)$와 $R(110)$가 같으므로 $A(110)$의 스케줄링이 가능하다. 그림 6의 알고리즘에 따라서 S_1의 $\overline{C}_1$ 값을 표 4과 같이 135로 갱신하고 온라인 스케줄링을 실시한다.

그림 7의 타이밍 그래프를 보면 시간 135에서 RTA 태스크 $A(110)$은 임계시간이 임박한 주기 태스크 집합 S_1의 τ_3와 τ_4의 주기를 보장 하기 위하여 우선순위가 최하위로 할당 되어 수행 이 지연되고 S_1의 τ_3와 τ_4가 모두 수행되고 난 후 다시 최상위 우선 순위를 할당 받아 남은 시간을 수행하게 된다. 이러한 주기 태스크의 임계 시간과 RTA 태스크의 동적 우선순위 할당을 통해 주기 태스크와 RTA 태스크의 스케줄링을 모두 만족하는 결과를 볼 수 있다.

표 3. t =110에서의 태스크 집합 테이블 예.
Table 3. Example of task list table at t =110.

Task Set ID (s)	T^s	N	e^s	P	C	$\overline{C}$
1	50	2	25	10	25	125
2	100	1	20	0	55	155
3	200	0	20	0	-	-

표 4. RTA 태스크 도착 후 태스크 집합 테이블 예.
Table 4. Example of task list table after RTA task arrival.

Task Set ID (s)	T^s	N	e^s	P	C	$\overline{C}$
1	50	2	25	10	25	**135**
2	100	1	20	0	55	155
3	200	0	20	0	-	-

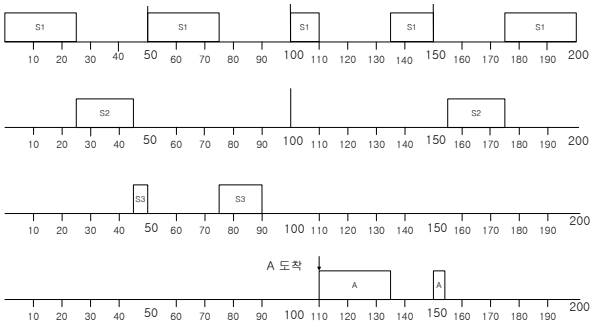

그림 7. RTA 태스크 스케줄링 결과.
Fig. 7. Result of RTA task scheduling.

Rate monotonic [6]과 Earliest Deadline First [7,10] 방식은 선행 관계를 고려하지 않았기 때문에 이전 주기에 계산된 결과가 다음 태스크로 입력되는 등의 문제가 발생하고, [8,12,13]의 방식은 주기 태스크 관점에서는 같은 결과가 나오지만 실시간 비주기 태스크들을 온라인상으로 스케줄링하지 못한다. [4]는 주기 태스크와 실시간 비주기 태스크를 스케줄링하지만 선행관계를 고려하지 못하여 [6]과 [7]의 문제에서 발생한 같은 문제가 존재한다. [9]는 미리 스케줄링된 주기 태스크를 사용하여 스케줄링 하는 방법으로 주기 태스크의 선행 관계를 만족할 수 있게 만들 수 있지만, Slack을 미리 계산하여 테이블을 생성하는데 드는 공간적 비용이 주기 태스크의 수와 동일하다는 점에서 비효율적일 수 있다. 본 논문에서 모델로 제시하는 시스템을 예로 들면 최소한 6개의 테이블이 필요하다. 반면 본 논문에서 제시하는 방법을 사용하면 같은 주기의 주기 태스크들을 하나의 집합으로 사용하였기 때문에 3개의 테이블만 필요하기 때문에 효율적이다.

V. 결론

본 논문에서는 주기 태스크의 선행 관계를 만족시키기 위하여 위상 정렬을 통하여 우선 순위를 할당하였고 할당된 고정 우선순위를 갖는 주기 태스크를 기반으로 온라인 상에서 불특정 시간에 도착하는 RTA 태스크의 스케줄링 방법에 대하여 제안하였다. 태스크의 기동 시간을 파라메터로 할당하여 스케줄링 하는 방법과 달리 우선순위만을 할당함으로써 태스크의 오버헤드를 줄이고 기동시간을 사용하지 않는 스케줄러에서도 사용 가능하며 기동 시간이 없는 불특정 시간에 도착하는 RTA 태스크 역시 스케줄링 가능함을 보였다. 또한 온라인 스케줄링 시 같은 주기를 가지는 주기 태스크를 하나의 리스트로 관리하여 스케줄링 시간을 줄였으며 주기 태스크의 임계시간을 만족하는 범위 안에서 RTA 태스크의 수행 시간을 최대한 빠르게 함으로써 향후에 발생할 RTA 태스크들의 수행을 가능하도록 하였다.

참고문헌

[1] J. Y.-T Leung and J. Whitehead, “On complexity of fixed-priority scheduling of periodic real-time tasks,” *Performance Evaluation*, vol. 2, no. 4, pp. 237-250, Dec. 1982.

[2] R. Gerber and S. Hong, “Guaranteeing real-time requirements with resource-based calibration of periodic processes,” *IEEE Tr. on Software Engineering*, vol. 21, no. 7, pp. 579-592, Jul. 1995.

[3] D. Isovic and G. Fohler, “Efficient scheduling of sporadic, periodic, and periodic tasks with complex constraints,” *Proc. of the IEEE 21st Real-Time Systems Symposium*, Orlando, Florida, pp. 23-27, Nov. 2000.

[4] G. Buttazzo and E. Bini, “Optimal dimensioning of a constant bandwidth server,” *Proc. of the IEEE Real-Time Systems Symposium*, Rio de Janeiro, pp. 169-177, Dec. 2006.

[5] D. I. Katcher, H. Arakawa, and J. K. Strosnider, “Engineering

and analysis of fixed priority schedulers," *IEEE Transactions on Software Engineering*, vol. 19, no. 9, pp. 920-934, Sep. 1993.

[6] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and average case behavior," *Proc. of the 10th Real Time Systems Symposium*, Washington DC, USA, pp. 166-171, Dec.1989.

[7] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of ACM*, vol. 20, no.1, pp. 46-61, 1973.

[8] H. Y. Kim and H. S. Park, "Optimal period and priority assignment using task & message-based scheduling in distributed control systems," *Journal of Control, Automation, and Systems Engineering (in Korean)*, vol. 8, no. 6, pp. 506-513, Jun. 2002.

[9] R. I. Davids, K. W. Tindell, and A. Burns, "Scheduling slack time in fixed-priority preemptive systems," *Proc. of the Real-Time Systems Symposium*, Raleigh Durham, USA, pp. 222-231, Dec. 1993.

[10] M. Spuri and G. C. Buttazzo, "Efficient aperiodic service under earliest deadline scheduling," *Proc. of the IEEE Real-Time Systems Symposium*, San Juan , Puerto Rico, pp.2-11, Dec. 1994.

[11] C. Moon, Y. Yun, and C. Leem, "Evolutionary algorithm based on topological sort for precedence constrained sequencing," *Proc. of 2007 IEEE Congress on Computational Intelligence*, Singapore, pp. 1325-1332, Sep. 2007.

[12] H. Y. Kim and H. S. Park, "Scheduling of sporadic and periodic tasks and messages with End-to-End constraints," *Journal of Control, Automation, and Systems Engineering (in Korean)*, vol. 11, no. 2, pp. 175-185, Feb. 2005.

[13] H. Y. Kim and H. S. Park, "Optimal period and priority assignment using task & messagebased scheduling in distributed control systems," *Journal of Control, Automation, and Systems Engineering (in Korean)*, vol. 8, no. 6, pp. 506-513, Jun. 2002.

[14] G. Park, D. Kum, B. Son, and S. Lee, "scheduling design and simulation of software components for EPS system based on AUTOSAR," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 16, no. 6, pp. 539-545, June 2010.

[15] D. Lee and H. Ahn, "Real-time characteristics analysis ans improvement for OPRoS component scheduler on windows NT operating system," *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 17, no. 1, pp. 38-46, Jan. 2011.

김 시 완

2010년 강원대학교 전자통신공학과 졸업. 2010년~현재 동 대학원 석사 과정. 관심분야는 스케줄링, 로봇 제어, 테스팅 자동화, 소프트웨어 공학.

박 홍 성

1983년 서울대학교 제어계측공학과 학사 졸업. 1986년 동 대학원 석사. 1992년 동 대학원 박사. 1992년~현재 강원대학교 전자통신공학과 교수. 관심분야는 로봇 S/W 플랫폼, 무선데이터통신, 실시간 통신.