論文
2-7-6

멀티미디어 실시간 태스크 집합의 스케쥴가능성 알고리즘 (A Schedulability Algorithm of Multimedia Real-Time Task Set)

송 기 현*
(Gi-Hyeon Song)

요 약

본 논문에서는 멀티미디어 데이터들로 구성된 부정확한 실시간 태스크 집합들을 여러 가지 인자들을 사용하여 생성하고 생성된 부정확한 실시간 태스크 집합의 스케쥴가능성을 이 태스크 집합의 실행 이전에 분석할 수 있는 알고리즘을 제안하였다. 또한 태스크 집합의 생성시 사용된 인자들의 값의 변화에 따른 태스크 집합의 스케쥴가능성도 살펴보았다.

실험결과에 의하면 태스크들의 필수적부분들의 실행요구시간이 커질수록 그리고, 임의의 시점에서 스케쥴될 수 있는 태스크들의 개수가 많아질수록 태스크집합의 스케쥴가능성은 희박해진다는 것이 입증되었다. 본 논문에서 제시된 스케쥴가능성 분석 알고리즘은 멀티미디어 데이터들의 QoS 서비스에서 유용하게 활용될 것으로 기대된다.

ABSTRACT

In this paper, An imprecise real-time task sets composed of multimedia datas are generated using several parameters and an algorithm which can analyse schedulability of generated imprecise real-time task sets before execution of this task sets is proposed. Also, The schedulability of task set depends on variation of parameter values which were used during the generation of the task set is studied.

As a result of experiment, It isproved that the schedulability of task set is more and more weak as large as execution requirement time of mandatory subtasks and, as many as the number of tasks which can be scheduled at some instant. The schedulabilty analysis algorithm which is presented on this paper is expected to use effectively on QoS service of multimedia datas.

1. 서론

1.1 실시간 처리 시스템의 개요

실시간 처리 시스템은 일반적인 범용 컴퓨터 시스템과는 달리 처리에 시간적 제약이 존재하는 시스템을 의미한다.

이러한 실시간 처리 시스템은 자동화 시스템이 보편화되면서 점차 그 수요가 증가되고 있는 실정이다. 또한 실시간 처리 시스템은 공장자동화, 항공기 운항

* 정회원 : 대전보건대학 경영정보과 조교수

논문접수 : 2001. 7. 12.
심사완료 : 2001. 7. 20.

제어, 공정 프로세스 제어, 레이더 추적기, 홍수 예방 시스템 등의 응용 분야에서 점차 보편화되어 가고 있으며 따라서 그 중요성이 날로 부각되어 가고 있는 추세이다[1].

실시간 처리 시스템에서는 이 시스템을 구성하고 있는 모든 태스크들이 그들 각 각의 수행을 그들 자신의 만기(deadline)이내에 완료해야 한다. 그러나 불행하게도 종종 동적 알고리즘들을 구현시키는 과정에서 각 태스크들의 수행시간들이 동적으로 변할 수 있고 통신네트워크상에서 태스크들의 수행이 지연될 수 있으므로 실시간 태스크들이 반드시 그들 고유의 만기 이내에 완료된다고 보장할 수는 없다[1].

이런 관계로, 실시간 태스크들을 반드시 수행해야 되는 부분(mandatory subtask)과 수행하지 않아도 이 태스크의 실행 결과에 심각한 영향을 끼치지 않는 부분(optional subtask)으로 나누어서 태스크들을 관리할 수 있는데 이러한 태스크들의 실행 결과는 일반적으로 정확한 결과가 될 수 없으므로 이러한 태스크들을 대상으로 스케쥴링하는 것을 부정확한 태스크 스케쥴링(imprecise task scheduling)이라고 한다[1].

또한, 제한된 시간 이내에 원하는 수준의 정확한 결과를 얻을 수없을 때 제한된 시간이 지나고 나서 얻을 수 있는정확한 결과는 실시간 시스템에서는 아무 필요도 없고 오히려 시스템에 파국적인 결과를 초래할 위험이 있기 때문에 제한된 시간까지만 실행해서 근사결과를 구하는 것을 부정확한 태스크 스케쥴링[4, 5, 6, 7, 8, 9, 10] 이라고도 한다.

그런데, 부정확한 태스크 시스템을 구성하는 각 태스크는 필수적 서브태스크(mandatory subtask)와 선택적 서브태스크(optional subtask)로 구성되는데 필수적 서브태스크는 반드시 실행되어야 한다. 그러나, 선택적서브태스크는 시스템에 과부하가 걸렸을 때는 일부가 실행되지 못 할 수 가 있는데 이렇게 실행되지 못한 부분의 실행시간이 이 실시간태스크의 오류(error)이다[1].

이러한 부정확한 태스크 스케쥴링의 예로서 "멀티미디어 데이터들의 QoS (Quality of Service) 서비스", "이미지 프로세싱(image processing)" 그리고 "레이더의 추적(radar tracking)"등을 들 수 있는데 멀티미디어 데이터들의 QoS 서비스[3] 에서는 사용자의 개입에 의하여 사용자가 임의의 멀티미디어 데이터가 기본적으로 충족되어야 한다고 생각되는 해상도나 초당 처리할 프레임 수를 필수적 서브 태스크로 설정함에 의하여 멀티미디어 데이터들에 대한 서비스의 질을 조정 할 수 있다. 이미지 프로세싱에서는이미지처리를 위한 제한 시간을 넘기면 컴퓨터로 이미지를 처리할 수 없기 때문에 제한 시간 이내에서 구할 수있는 근사이미지(fuzzy image)를 택한다. 또한, 레이더의 추적에서는 추적 시스템에 과부하가 걸려서 제한된 시간 이내에 상대방 항공기들의 위치를 정확하게 알아낼 수 없을 때 제한된 시간 이후에 구한 항공기들의 위치는 쓸모 없는 데이터이므로 제한된 시간 이내에 구한 근사적인 상대방 항공기들의 위치로서 상대방 항공기들의 위치를 정확하지는 못하지만 근사적으로 추정해 낼 때 부정확한 태스크 스케쥴링을 사용하고 있다[11].

1.2 멀티미디어 처리의 특성

멀티미디어의 처리의 특성은 기본적으로 실시간 처리(real-time processing)가 가능하여야 한다는 점이다. 실시간 처리란 주어진 작업이 정해진 만기시간(deadline) 내에 처리되도록 지원하는 처리 기법을 의미한다. 실시간 처리가 멀티미디어 처리에 있어 요구되는 이유는 다음과 같다. 예를 들어, 음향이나 동영상은 매체의 특성상 실시간이라는 제약 요건을 갖고 있다. 즉, 음향이 제대로 재생되기 위해서는 음향신호들이 정해진 시간 내에 처리되어야 하고, 동영상 역시 정해진 시간 단위 내에 해당 프레임의 처리가 이루어져야만 한다. 특히 텍스트나 정지화상과 같이 시간에 대한 제약을 고려할 필요가 없는 매체라고 하더라도, 음향이나 동영상과 같이 시간제약이 요구되는 매체와 서로 혼합되어 처리될 때에는 역시 간접적으로 시간에 대한 제약을 받게 된다.

1.3 연구배경 및 연구목적

한편, 실시간 처리 시스템은 그 특성상 잘못 실행되면 심한 경우 시스템 전체에 파국적인 영향을 끼칠 수도 있으며 심지어는 인간의 생명을 위협하는 예도 허다하다. 그래서 임의의 실시간 처리 시스템을 실행하기 전에 반드시 이 실시간 처리 시스템의 안전성을 사전에 파악해 볼 필요가 있다.

이와 같이 임의의 실시간 처리 시스템의 실행 이전에 이 실시간 처리 시스템의 안전성을 파악하는 문제는 이 실시간 처리 시스템을 구성하고 있는 실시간 태스크들을 이 실시간 태스크들의 특성에 맞는 방법으로 모의 스케쥴링하여 그 결과를 미리 살펴봄으로서 해결될 수 있다.

그래서 본 논문에서는 임의의 태스크가 필수적태스크와 선택적태스크로 구성된 부정확한 실시간 태스크들로 구성되어 있는 부정확한 실시간 태스크집합들을 여러 가지 인자들을 사용하여 생성하고 각 각의 생성된 실시간 태스크 집합들의 스케쥴가능성을 이 실시간 태스크 집합들의 실행 이전에 파악해 볼 수 있는 알고리즘을 제시하였다.

또한 부정확한 실시간 태스크 집합의 생성에 사용된 각 인자들의 값의 변화에 따른 부정확한 실시간 태스크 집합의 스케쥴가능성을 분석해 보았다.

본 논문의 구성은 다음과 같다. 먼저 제 2 장 에서는 본 연구내용과 관련되어 기존에 발표된 관련연구들에 관하여 기술하고 제 3 장 에서는 본 연구에서 사용된 부정확한 실시간 태스크집합의 시스템구성에 대하여 기술하고 제 4 장 에서는 부정확한 실시간 태스크 집합의 스케쥴가능성을 이 실시간 태스크 집합의 실행이전에 파악해 볼 수 있는 알고리즘을 제시하고 제 5 장 에서는 본 연구의 실험내용과 실험결과에 관하여 기술하고 마지막으로 제 6 장 에서는 결론을 기술하겠다.

2. 관련연구

실시간 처리 시스템을 구성하는 실시간 태스크들의 스케쥴가능성(schedulability)에 관한 연구는 주로 Rate Monotonic Scheduling(이하 RMS) 알고리즘과 Earliest Deadline Scheduling(이하 EDS) 알고리즘을 근간으로 하여 발전하여 왔다. RMS 알고리즘은 선점 가능한 스케쥴링 알고리즘이고, 각 태스크들은 독립적이라 가정되며, 각 태스크에게 정적(static)으로 우선순위를 부여하되, 주기가 짧은 태스크 일수록 높은 우선순위를 부여한다. EDS 알고리즘은 선점 가능한 알고리즘이고, 각 태스크들은 독립적이라 가정되며, 각 태스크에게 동적(dynamic)으로 우선순위를 부여하되, 매 스케쥴링 시점에서 볼 때 종료시한 까지의 시간구간이 가장 짧은 태스크에게 높은 우선순위를 부여한다. 따라서 각 태스크의 우선순위는 가변적이며 이를 결정하는 실행부담(overhead) 이 상당하다[2].

RMS 알고리즘의 경우에는 주어진 태스크 집합의 프로세서 이용률(utilization) U가

$n(2^{1/n}-1)$ (n은 태스크의 수) 보다 작거나 같으면 스케쥴링이 가능하고, EDS 알고리즘의 경우에는 $U \leq 1$ 이면 스케쥴링이 가능하다는 것이 증명된 바 있다. 그러나 RMS 알고리즘을 사용하는 경우 태스크들의 주기가 균일분포(uniform distribution)에 의해 주어질 때에, 스케쥴링이 가능한 평균 프로세서 이용률은 0.88에 이르는 것으로 알려져 있다[2].

RMS 알고리즘은 EDS 알고리즘에 비해 프로세서 이용률 이라는 측면에서는 다소 비효율적이지만 실제 환경에서 발생하는 제반 문제들에 대한 해결책이 EDS의 경우보다 잘 정립되어 있는 편이다. 이에 해당하는 것으로는 태스크의 종료시한이 주기보다 큰 경우의 스케쥴링 문제[12], 프로세서와 I/O의 통합 스케쥴링[13], 주기적인 태스크들과 비주기적인 태스크들의 통합 스케쥴링[14], 우선순위높임 프로토콜(priority ceiling protocol)을 이용한 태스크들간의 동기화(synchronization) 문제의 해결[15], 그리고 부정확한 실행(imprecise computation)이 허용된 경우의 스케쥴링[16] 등을 들 수 있다.

그러나 이러한 RMS 알고리즘과 EDS 알고리즘은 모두 단일 프로세서 환경을 가정하고 있으며 다중 프로세서 환경에서는 더 이상 최적알고리즘이 되지 못한다. 현재로는 다중프로세서 환경에서 작동될 수 있는 최적 스케쥴링 알고리즘은 개발된 것이 없고, 여러가지 조건이 가정된 단순화된 환경에서 제한적인 처리능력을 갖는 알고리즘이 제시되고 있는 정도이며, 대부분의 문제는 NP-complete이다[2].

한편 부정확한 실시간 태스크 시스템에서의 스케쥴가능성 문제는 기존의 연구결과가 미미하여 많은 연구가 필요한 연구과제이다. 따라서, 본 논문에서는 이러한 부정확한 실시간 태스크 시스템의 스케쥴가능성 여부를 판단할 수 있는 알고리즘을 제시하고 태스크집합의 생성시 사용되었던 여러 가지 인자들의 값의 변화에 따른 실시간 태스크 집합의 스케쥴가능성 여부를 분석해 보았다.

3. 시스템 구성

부정확한 실시간 태스크들의 스케줄가능성 여부를 판단하기 위하여 사용되는 실시간 태스크 T_i 의 모델을 다음과 같이 정의하였다.

T_i (r_i, d_i, m_i, o_i, p_i)
- r_i : 준비시간(ready time)
- d_i : 만기(deadline)
- m_i : 필수적태스크의 수행요구시간
- o_i : 선택적태스크의 수행요구시간
- p_i : 태스크의 수행요구시간(m_i + o_i)

여기서의 각 태스크 T_i는 필수적태스크 M_i와 선택적태스크 O_i로 구성되며 선점가능하며 비주기적이며 태스크들사이의 선후관계가 없는 부정확한 태스크로서 단일처리기상에서 수행된다고 가정하였다. 또한, 하나의 태스크집합을 300개의 태스크들로 구성하였다.

본 논문에서 제시하는 부정확한 실시간 태스크 모델의 예를 다음의 표에 나타내보았다.

<표 1> 부정확한 실시간 태스크 모델

<Table 1> A Imprecise Real-Time Task Model

	r_i	d_i	m_i	o_i	p_i
T_1	1	16	2	3	5
T_2	5	10	3	1	4
T_3	11	17	2	4	6

4. 부정확한 실시간 태스크 집합들의 스케줄가능성

4.1 부정확한 실시간 태스크 집합들의 스케줄가능성 분석 알고리즘

부정확한 실시간 태스크는 그 특성상 필수적 서브태스크와 선택적 서브태스크로 구성된다. 필수적 서브태스크는 이 태스크의 만기(deadline) 이전에 그의 수행을 반드시 완료해야 하는 서브태스크이다. 그러므로 임의의 부정확한 실시간 태스크 집합이 스케줄가능(schedulable) 하기 위해서는 이 실시간 태스크 집합을 구성하고 있는 모든 실시간 태스크들 각 각이 그들의 필수적 서브태스크의 수행을 자신의 만기 이전에 종료시켜야만 한다. 이런 관점에서 부정확한 실시간 태스크 시스템의 스케줄가능성 여부를 파악할 수 있는 알고리즘을 다음과 같이 제시해 보고자 한다.

다음의 알고리즘은 먼저 4번 Step 에서 실시간 태스크 집합 내의 모든 실시간 태스크들을 그 들 각 각의 만기(deadline)의 오름차순으로 순서배열한다.

다음으로 5번 Step에서 NIntervals() 라는 함수를 수행하는데 이 함수의 기능은 실시간 태스크 집합내의 모든 태스크들 각 각의 준비시간(ready time)들과 만기(deadline)들을 RiDi라는 배열에 저장하고 이 배열의 값들을 오름차순으로 정렬한 후 이 배열내의 연속된 두 배열요소 값들로 임의의 구간의 시작시간과 종료시간을 정의하고 이 구간의 종료시간에서 시작시간을 감하여 이 구간의 길이를 구하는 내용으로 되어 있다. 그러므로 주어진 실시간 태스크 집합 내의 태스크의 개수가 n개 라면 이 실시간 태스크 집합의 구간개수는 2n - 1 개가 된다.

```
1.   Sub CheckSchedulability()
2.     Dim j, t, idv, Task  As Integer
3.     Dim Check  As Boolean

4.     Call SortDeadline
5.     NumberIntervals = NIntervals()
6.     For idv = 1 To NumberIntervals
7.       Length(idv) = Intervals(idv).Length
8.       If Length(idv) = 0 Then
9.        GoTo 10
10.      End If

11.,     Call DE_TASK(idv)

12.      t = 1
13.      Do While (Length(idv) > 0 And TAK(t) > 0)
14.        Task = TAK(t)
15.        If TaskSystem(Task).Mi > 0 Then
16.          If Length(idv) >= TaskSystem(Task).Mi Then
17.             Length(idv) = Length(idv) - TaskSystem(Task).Mi
18.             TaskSystem(Task).Mi = 0
19.             t = t + 1
20.           Else
21.             TaskSystem(Task).Mi = TaskSystem(Task).Mi - Length(idv)
22.             Length(idv) = 0
23.           End If
24.         Else
25.           t = t + 1
26.         End If
27.       Loop
28.  10: Next idv

29.     Check = True
30.     For j = 1 To NumberTasks
31.       I f TaskSystem(j).Mi > 0 Then
32.         Check = False
33.         TaskSystem(j).SchState = False
34.        Exit For
35.       Else
36.         TaskSystem(j).SchState = True
37.       End If
38.    Next j
39.  End Sub
```

[그림 1] 스케줄가능성 알고리즘

[Fig. 1] A Schedulability Algorithm

다음으로 11번 Step에서 DE_TASK(idv) 함수를 수행하는데 이 함수의 기능은 구간 idv에서 스케줄가능한 태스크들을 찾는 것이다. 특정구간 I_i ($1 \le i \le 2n-1$)에서 임의의 태스크의 스케줄구간 $[r_i, d_i]$이 이 특정구간 I_i를 포함한다면 이 태스크는 구간 I_i에서 스케줄가능하다. 12번 Step에서 28번 Step 사이의 내용은 주어진 실시간 태스크 시스템의 첫 번째 구간에서 마지막 구간 까지 각 각의 구간 idv에서 이 구간 idv의 길이를 이 구간에서 스케줄가능한 실시간 태스크들의 필수적 서브태스크들에게 이 태스크들의 만기들

의 오름차순으로 배분해 주는 내용으로 되어 있다. 마지막으로, 29번 Step에서 38번 Step까지의 내용은 이전의 알고리즘을 수행한 후 이 태스크 집합에 속한 모든 태스크들의 필수적 서브태스크들의 수행요구시간들이 이 태스크 집합의 모든 구간들 내에서 할당되어 스케쥴 될 수 있는지 살펴보는 내용으로 구성되어 있으며 만약, 주어진 실시간 태스크 집합내에 속한 단 하나의 필수적 서브태스크라도 이 실시간 태스크 집합의 모든 구간들 내에서 할당되지 못하여 스케쥴되지 못한다면 부울변수 Check의 값이 "False"가 되어 이 태스크 집합 전체가 스케쥴되지 못한다.

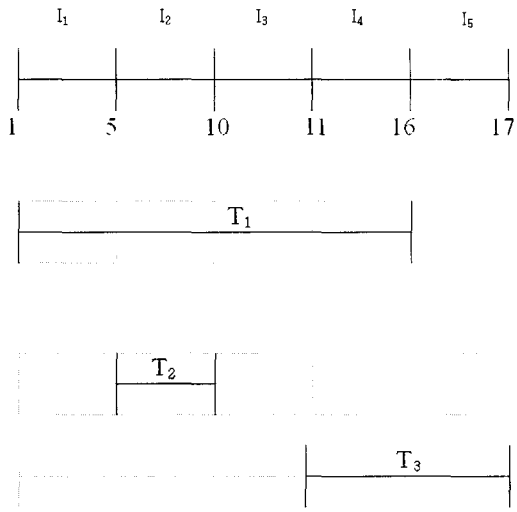
4.2 부정확한 실시간 태스크 집합들의 스케쥴가능성 분석 예제

한편, 이제부터는 [그림 1]의 스케쥴가능성 알고리즘을 <표 1>의 부정확한 실시간 태스크 모델을 대상으로 설명해보겠다.

제 3 장의 <표 1>에 있는 모든 태스크들의 준비시간들과 만기들을 오름차순으로 정렬하면 1 - 5 - 10 - 11 - 16 - 17 이 된다. 그러면, 이들로부터 다음과 같은 5개의 구간들을 구성할 수 있다.

I_1 = [1, 5], I_2 = [5, 10], I_3 = [10, 11], I_4 = [11, 16], I_5 = [16, 17]

특정구간 I_i ($1 \le i \le 5$)에서 임의의 태스크 T_i 의 스케쥴구간 $[r_i, d_i]$이 이 특정구간 I_i 를 포함한다면 이 태스크는 구간 I_i 에서 스케쥴가능하다. 각각의 구간 I_i ($1 \le i \le 5$)에서 스케쥴가능한 태스크들을 다음의 [그림 2]에서 살펴보자.



[그림 2] 임의의 구간에서 스케쥴가능한 태스크들
[Fig. 2] The Schedulable Tasks at Some Interval

위의 [그림 2]를 살펴보면 구간 I_1 에서 스케쥴가능한 태스크는 T_1이고 구간 I_2 에서 스케쥴가능한 태스크는 T_1 과 T_2 이며 구간 I_3 에서 스케쥴가능한 태스크는 T_1 이며 구간 I_4 에서 스케쥴가능한 태스크는 T_1 과 T_3 이며, 마지막구간 I_5에서 스케쥴가능한 태스크는 T_3 임을 알 수 있다. 그러면, 각 각의 구간에서 스케쥴가능한 태스크들의 필수적부분들이 만기들의 오름차순(이하 EDF전략) 으로 스케쥴 될 수 있는지 살펴볼 수 있다. 먼저, I_1에서는 m_1이 구간 I_1 의 크기 Length(1) = 4 보다 작아서 스케쥴 될 수 있으므로 m_1 을 2 만큼 스케쥴시킨다. 그러면 $m_1 = m_1 - 2 = 0$ 가 되어 m_1의 수행은 종료된다. I_2에서 스케쥴될 수 있는 태스크는 만기의 오름차순으로 T_2 - T_1 이 된다. 구간 I_2 에서 먼저 $m_2 = 3 <$ Length(2) = 5 이므로 m_2 가 3만큼 스케쥴 되어 m_2 의 수행도 종료된다. 다음으로 구간 I_3 에서는 T_1 이 스케쥴 될 수 있지만 m_1의 수행이 이미 구간 I_1 에서 종료되었으므로 다음 구간으로 넘어간다.

다음 구간 I_4에서 스케쥴가능한 태스크들은 만기들의 오름차순으로 T_1 - T_3 가 되는데 m_1의 수행이 이미 구간 I_1 에서 종료되었으므로 $m_3 = 2$만 스케쥴시키면 된다. 그런데 $m_3 = 2 <$ Length(4) = 5 이므로 $m_3 = 2$ 는 구간 I_4 에서 스케쥴가능하다. 그러므로 $m_3 = m_3 - 2 = 0$ 가 되어 m_3 의 수행도 이 구간 I_4

에서 종료된다.

마지막 구간 I_5 에서는 스케줄가능한 태스크가 오직 T_3 인데 m_3 의 수행이 이미 구간 I_4에서 종료되었으므로 더 이상 스케줄시킬 태스크가 없다. 이 시점에서 <표 1>에 있는 태스크 집합은 이 태스크집합을 구성하고 있는 모든 태스크들 T_1, T_2, 그리고 T_3 의 필수적 서브태스크 m_1, m_2, 그리고 m_3의 수행이 종료되므로 스케줄가능하다.

이와 같이 임의의 실시간 태스크 시스템의 수행 이전에 이 실시간 태스크 시스템을 구성하고 있는 태스크 집합을 미리 모의로 스케줄링시켜 봄으로서 이 실시간 태스크 시스템의 수행이 초래 할 수도 있는 엄청난 재앙을 미리 예측 할 수 있다는 사실은 실시간 시스템의 안정성과 직결되는 문제로서 대단히 중요한 의미를 갖게 된다.

5. 실험내용과 결과

부정확한 실시간 태스크 집합의 스케줄가능성을 분석하기 위하여 먼저 300개의 실시간 태스크들로 구성된 태스크 집합을 생성하였는데 이 태스크 집합을 생성하는데 있어서 다음과 같은 인자들을 사용하였다.

- λ : 태스크들의 도착율 (포아송분포(Poisson Distribution)를 따른다)
- p_i : 평균이 $1 / \mu$ 이며 지수적으로(exponentially) 분포하는 태스크 T_i 의 계산시간 ($p_i = d_i - r_i$, $p_i = m_i + o_i$)
- ρ : 일정시점에서 스케줄가능한 태스크들의 평균 개수 (λ / μ)
- p : 각 태스크의 계산요구시간에 대한 필수적부분 계산요구시간의 비율

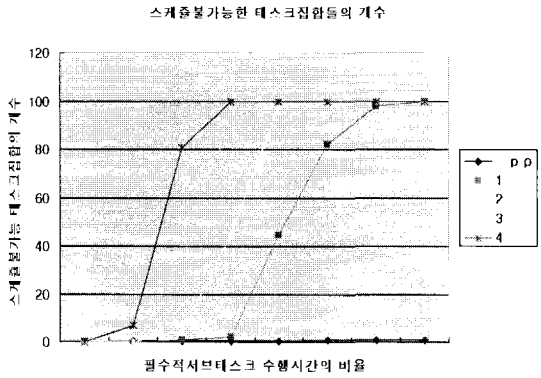
본 실험에서는 부정확한 실시간 태스크들의 필수적 부분들의 수행요구시간들이 0과 $p \times (d_i - r_i)$ 사이에서 베타분포(beta distribution)를 따른다고 가정하였다. 먼저, 태스크들의 필수적부분들이 베타분포를 따르도록 300개의 태스크들로 구성된 태스크집합들을 생성하는데 여기서, p는 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 그리고 0.9 중에서 임의로 선택되며 ρ는 1, 2, 3, 그리고 4중에서 임의로 선택된다. 그러면, p가 취할 수 있는 값이 8개이고 ρ가 취할 수 있는 값이 4개이므로 총 32개의 태스크집합들이 생성될 수 있다. 또한, 본 실험에서는 300개의 태스크들로 구성된 100개의 태스크 집합들을 생성하여 p와 ρ값의 변화에 따른 태스크 집합의 스케줄가능성을 분석해 보았다.

실험결과, 다음의 <표 2>와 [그림 3]에서 볼 수 있듯이 임의의 부정확한 실시간 태스크 집합을 구성하는 태스크들의 필수적부분들이 베타분포를 따를 때 p나 ρ의 값이 커질수록 이 태스크집합이 스케줄되지 못할 확률이 높아진다는 것을 알 수 있었다.

<표 2> 스케줄불가능한 태스크 집합들의 개수

<Table 2> The Number of Not-Schedulable Task Sets

ρ \ p	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1	0	0	1	2	45	82	93	100
2	0	0	5	45	93	100	100	100
3	0	0	34	93	100	100	100	100
4	0	7	81	100	100	100	100	100



[그림 3] 태스크생성 인자들의 값에 따른 스케줄불가능한 태스크집합의 개수

[Fig. 3] The Number of Not-Schedulable Task Sets depends on Variation of Task Generation Parameter Values

6. 결론

실시간 처리 시스템은 그 특성상 이 시스템을 구성하고 있는 태스크 집합내의 태스크들이 그들 각 각의 만기시간(deadline) 이내에 처리되지 못하면 시스템이나 심지어는 인간의 생명에 까지 파멸적인 영향을 끼칠 수 있는 매우 민감한 시스템으로서 주로 군사용 무기제어나 산업현장의 공정제어 시스템용으로 사용되어 왔다. 더욱이, 최근들어서 컴퓨터에서 처리하는 자료들이 과거의 텍스트나 정지화상과 같은 비실시간적인 데이타들로부터 동영상이나 음성 데이터와 같은 실시간적 처리를 요하는 멀티미디어 데이터들로 급속하게 전환되어 가는 추세에 따라서 이러한 실시간적 특성을 요하는 멀티미디어 데이터들을 실시간 시스템의 실행 이전에 모의 스케쥴링을 하여 이러한 실시간 태스크들의 스케쥴가능성 여부를 미리 파악하는 것이 이 실시간 시스템의 실행으로부터 야기될 지도 모르는 파국적인 결과를 미연에 방지 할 수 있다. 또한, 멀티미디어 데이터들의 QoS 서비스에서 각 각의 멀티미디어 데이터들의 서비스의 질을 사용자가 미리 결정하고 그 스케쥴가능성을 멀티미디어 데이터들의 실행이전에 모의 스케쥴링 해 볼 수 있는 알고리즘이 필요한다. 이러한 관점에서 본 논문에서는 300개의 태스크들로 구성된 100개의 부정확한 실시간 태스크 집합들을 여러 가지 생성인자들을 사용하여 생성하고 이렇게 생성된 부정확한 실시간 태스크 집합들의 스케쥴가능성을 이 실시간 태스크 집합들의 실행 이전에 모의 스케쥴링 해 볼 수 있는 알고리즘을 제시하였다. 또한 태스크 생성시 사용된 여러 개의 인자들의 값에 따른 이 태스크 집합의 스케쥴가능성을 분석해 보았다.

실험결과에 의하면 태스크들의 필수적부분들의 실행요구시간이 커질수록 그리고, 임의의 시점에서 스케쥴 될 수 있는 태스크들의 개수가 많아질수록 태스크 집합의 스케쥴가능성은 희박해진다는 것이 입증되었다. 그 이유는 이런 경우에 CPU 의 부담이 커지기 때문이라고 볼 수 있다.

그런데 본 연구논문에서 제시한 부정확한 실시간 태스크 시스템의 스케쥴가능성 분석 알고리즘은 임의의 태스크 집합을 구성하고 있는 각 각의 태스크들이 특정 준비시간(ready time)에 이미 도착되어 있다고 가정한 상태, 즉 오프라인(offline) 상태에서의 알고리즘인데 앞으로 임의의 태스크 집합을 구성하는 각 각의 태스크들이 도착하는 순간 순간 이 태스크의 스케쥴가능성 여부를 파악 할 수 있는 온라인용 스케쥴가능성 분석 알고리즘이 필요하리라 생각한다.

※ 참고문헌

[1] 송 기현, "부정확한 태스크의 최대가중치오류를 최소화시키는 온라인 스케쥴링", pp. 1-134, 아주대학교 박사학위 논문, 1999. 2.

[2] 김 인국, "흐름 공정 모델의 효율적인 실시간 스케쥴링", pp. 1-127, 아주대학교 박사학위 논문, 1995. 8.

[3] 황 대훈, 여 인국, "멀티미디어 시스템", pp. 1-390, 정익사, 1997. 2.

[4] P.D. Alexander, C.C.Lim, J.W.S.Liu, and W.Zhao, "Managing Transient Overload in An Imprecise Computation System", Proc. of IEEE Workshop on Imprecise and Approximate Computation, pp.1-4, Dec. 1992.

[5] R. Bettati, N.Bowen, and J.-Y. Chung, "Checkpointing Imprecise Computation", Proc. of IEEE workshop on Imprecise and Approximate Computation, pp. 45-49, Dec. 1992.

[6] J.W.S. Liu, W.K.Shih, K.J.Liu, R.Bettati, and J.Y.Chung, "Imprecise Computation", Proc. of the IEEE (Special Issue on Real-Time Systems), vol.82, no.1, pp.83-94, Jan. 1994.

[7] K.J.Lin and S.Natarajan, "Concord : A System of Imprecise Computations", Proceedings of the 1987 IEEE Compsac, pp.75-81, Tokyo, Japan, October 7-9, 1987.

[8] K.J.Lin, S.Natarajan, J.W.S.Lin, "Imprecise Results : Utilizing Partial Computations in Real-time Systems", Proceedings of the IEEE 8th Real-Time Systems Symposium, SanJose, California, December 1987.

[9] G.J.Gregory and K.J.Lin, "Building Real-Time Imprecise Computations in Ada", Proceedings of Tri-Ada, Baltimore, December 1990.

[10] Wei-Kuan Shih, "Scheduling in Real-Time Systems to ensure graceful degradation : the imprecise- computation and the deferred-deadline approaches",
Dept. of Computer Science, University of Illinois at Urbana-Champaign, Ph. D. dissertation, Oct. 1992.

[11] J.W.S Liu, K.J. Lin , W.K. Shih, A.C.Yu, J.Y. Chung , and W. Zhao , "Algorithms for Scheduling Imprecise Computations", Foundations of RealTime Computing Scheduling and Resource Management , 1991.

[12] J.P. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines'', Proceedings of the 11th IEEE Real-Time Systems Symposium, pp201-209, December 1990.

[13] J.P. Lehoczky and L. Sha, "Performance of Real-time bus scheduling algorithms", ACM Performance Evaluation Review, 14, 1986.

[14] J.P. Lehoczky, L. Sha and J.K. Strosnider, "Enhanced aperiodic responsiveness in hard real-time environments", Proceedings of the 8th IEEE Real-Time Systems Symposium, pp261-270, Dec. 1987.

[15] L. Sha, R. Rajkumar and J.P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization", IEEE Trans. on Computers, Vol 39, 1175-1185, 1990.

[16] J.Y. Chung and J.W.S. Liu, "Performance of Algorithms for Scheduling Jobs to Minimize Average Error", Proceedings of the 9th IEEE Real-Time Systems Symposium, Huntsville, Alabama, Dec. 1988.

송 기 현

1985년 2월 충남대학교 계산통계학과 졸업
1987년 2월 충남대학교 대학원 계산통계학과 졸업(이학석사)
1999년 2월 아주대학교 대학원 컴퓨터공학과(공학박사)
1990년 3월 대전보건대학 경영정보과에 전임강사
현재 대전보건대학 경영정보과 조교수
관심분야 : 멀티미디어 실시간 스케줄링, 웹 데이터베이스, 홈페이지저작