멀티미디어 태스크와 경성 실시간 태스크를 위한 동적 스케쥴링 기법

김진환*, 김남윤**
한성대학교 *컴퓨터공학부, **정보공학부

Dynamic Scheduling Methods for Mutlimedia Tasks and Hard Real-time Tasks

Jinhwan Kim*, Namyun Kim**
*School of Computer Engineering, **School of Information Engineering, Hansung University

요 약

본 논문에서는 분산 실시간 멀티미디어 시스템에서 경성 실시간 태스크들과 멀티미디어 태스크들을 효율적으로 통합할 수 있는 동적 스케쥴링 기법이 제시된다. 경성 실시간 태스크가 최악의 경우에 대한 실행 시간을 기반으로 하는 반면 멀티미디어 태스크는 평균 실행 시간을 기반으로 스케쥴링된다. 동일한 시스템에 존재하는 두가지 태스크들에 대하여 CPU 대역폭을 분할하고 해당 대역폭의 비율을 동적으로 조정하는 스케쥴링 기법을 제시함으로써 한 부류의 태스크들의 수와 도착 비율이 변동될 때 발생하는 과부하 문제를 해결할 수 있다. 경성 실시간 태스크가 서버의 주기내에서 실행될 수 있는 시간이 제한되는 반면 멀티미디어 태스크에 설정된 대역폭은 동적으로 변할 수 있다. 제시된 기법은 경성 실시간 태스크들의 실시간성을 모두 보장하는 한편 멀티미디어 태스크들의 평균 지연 시간을 최소화할 수 있다.

1. 서론

공장 자동화, 방위 시스템 등 경성 실시간(hard real-time) 시스템 응용 분야에서 음성 및 영상 등의 멀티미디어 정보가 이용되고 있다[1]. 이러한 정보는 경성 실시간적 제어 정보보다는 덜 중요하나 지연 시간과 지터(jitter) 허용 등 연성 실시간적(soft real-time) 특성을 가지게 된다. 멀티미디어 정보와 경성 실시간 제어 정보를 모두 처리해야 하는 시스템의 경우 연성 실시간 태스크들과 경성 실시간 태스크들이 공존하게 된다[2].

제어 기능이 필요한 분산 실시간 멀티미디어 시스템에서는 멀티미디어 데이터를 처리하는 연성 실시간 프로세스들과 시스템의 이상 현상을 제한된 시간 내에 해결하고자 하는 경성 실시간 프로세스들이 공존하게 되므로 서로 다른 두 형태의 태스크들을 효율적으로 스케쥴링할 수 있는 기법이 필요하다[3]. 현재 일정한 주기를 갖는 경성 실시간 태스크들은 대부분 최악의 실행 시간(worst case execution time)을 기반으로 하는 특성이 있고 기존의 실시간 스케쥴링 기법인 최조종료시한(EDF; Earliest Deadline First) 기법이나[4, 5] RM(Rate Monotonic) 기법에[4, 6] 기반하여 스케쥴링되고 있다. 그러나 연성 실시간 특성을 갖는 멀티미디어 태스크들은 처리하고자 하는 데이터의 규모에 따라 실행시간이 가변적이기 때문에 최악의 실행시간을 기반으로하는 실시간 스케쥴링 기법을 적용할 경우 CPU 대역폭의 낭비가 심해져서 자원의 활용도가 낮아지는 문제가 발생하게 된다[3]. 경성 실시간 태스크들과는 달리 연성 실시간 태스크들은 종료시한이 경과되더라도 시스템에 미치는 영향이 심각하지 않은 것으로 간주되기 때문에 최악의 실행시간보다는 평균 실행 시간을 기반으로 하는 스케쥴링 기법들이 제시된 바 있다[3, 7, 8].

경성 실시간 태스크들과 멀티미디어 태스크들을 스케쥴링하는 CBS(Constant Bandwidth Server) 기법[3]에서는 모든 경성 실시간 태스크들의 종료시한이 보장되는 반면 멀티미디어 태스크들의 평균 실행 시간의 변동이 클 경우 종료시한이 경과되는 시간이 증가될 수 있다.

본 논문에서는 경성 실시간 태스크들의 종료시한을 모두 보장하면서 멀티미디어 태스크들이 종료시한 이후에 종료되는 경과 시간을 가능한 최소화하도록 CPU 대역폭을 효율적으로 사용하는 동적인 스케쥴링 기법이 제시되었다.

2. 신축적 스케쥴링 기법

2.1 스케쥴링 정책

연성 실시간성 특성을 갖는 멀티미디어 태스크들과 경성 실시간 태스크들은 모두 주기를 가지고 있는 것으로 가정한다. 제시된 스케쥴링 기법에서는 경성 실시간 또는 멀티미디어 태스크들 중 가장 주기가 작은 태스크의 주기를 서버의 주기로 설정한다. 임의의 경성 실시간 태스크 H_i는 주기 TH_i와 최악의 경우에 대한 실행 시간 CH_i로 구성되는 반면 임의의 멀티미디어 태스크 M_j는 주기 TM_j와 평균 실행 시간인 CM_j로 구성되며 다음과 같이 표현된다.

경성 실시간 태스크 $H_i = (CH_i, TH_i)$

멀티미디어 태스크 $M_j = (CM_j, TM_j)$

일정한 주기마다 수행되는 각 멀티미디어 태스크는 특정한 동영상 스트림의 재생 기능을 수행할 수 있다. 태스크들 중 가장 주기가 작은 것을 택하여 두 태스크들을 스케쥴링하는 서버의 주기로 설정하며 EDF 스케쥴링 원칙에[5] 따라 각 태스크의 실행 시간을 주기로 나눈 값들의 합이 1보다 작거나 같을 때만 태스크들이 스케쥴링될 수 있는 것으로 간주한다[13].

$$\sum_{i=1}^{n} \frac{CH_i}{TH_i} + \sum_{j=1}^{m} \frac{CM_j}{TM_j} \le 1 \quad (1)$$

결정된 서버의 주기를 T_s라 할 때 주기 내에서 경성 실시간 태스크들을 위한 실행 시간은 E_H로 멀티미디어 태스크들을 위한 실행 시간은 E_M으로 정의하며 다음과 같이 결정된다.

$$E_H = \sum_{i=1}^{n} (CH_i \frac{T_s}{TH_i}) \quad (2)$$

$$E_M = \sum_{j=1}^{m} (CM_j \frac{T_s}{TM_j}) \quad (3)$$

경성 실시간 태스크 H_i가 최소주기 서버내에서 차지하는 실행 시간은 자신의 실행 시간 CH_i를 T_s/TH_i 비율과 곱한 값으로 결정된다. 즉 최소 주기보다 주기가 큰 경성 실시간 태스크들은 서버가 두 번 이상 수행됨으로써 실행될 수 있다. 멀티미디어 태스크 M_j도 마찬가지 방법으로 자신의 평균 실행 시간 CM_j를 T_s/TM_j와 곱한 값이 최소 주기내에서의 실행 시간으로 결정된다. 그리고 E_H와 E_M의 합 E_s는 최소주기 서버내에서의 실행 시간이 된다.

2.2 스케쥴링 알고리즘

제시된 스케쥴링 알고리즘은 그림 1과 같이 기술된다. 최소주기 서버가 설정된 후 매 주기마다 서버는 경성 실시간 태스크들의 대기 행렬인 Queue_H를 먼저 탐색하며 태스크 H_i의 주기 내 할당 시간인 Allot_time(H_i)이 0보다 크면서 우선순위가 가장 높은 태스크 H_s를 선택하여 실행시킨다.

```
Search_Queue_H:
  for Hi in Queue_H
   if Allot_time(Hi) > 0
     and Prior(Hi) > High_prior_H
          Hs = Hi;
   if Hs is not Null and EH > 0
    begin
      Execute(Allot_time(Hs));
      EH = EH - Allot_time(Hs);
      CHs = CHs - Allot_time(Hs);
      Allot_time(Hs) = 0;
      if CHs = 0  Kill(Hs);
      else Queue(Hs) in Queue_H;
    end
   goto Search_Queue_H;
   for Mi in Queue_M
    if Prior(Mi) > High_prior_M
        Ms = Mi;
    if Ms is not null and EM > 0
     begin
       Execute(Ms);
       if Hnew arrives
       Preempt(Ms) and Queue(Hnew) in Queue_H;
       EM = EM - Execute_time(Ms);
       CMs = CMs - Execute_time(Ms);
       if CMs = 0  Kill(Ms);
       else Queue(Ms) in Queue_M;
     end
   goto Search_Queue_H;
```

그림 1. 최소주기 서버 알고리즘

H_s의 우선순위는 알고리즘에서 Prior(H_s)로 정의되며 지금까지의 우선순위가 가장 높았던 경성 실시간 태스크의 우선순위 High_prior_H와 비교된다. H_s는 자신의 할당 시간인 Allot_time(H_s) 이하의 시간만큼 CPU를 사용하며 이 과정은 그림 1에서 Execute(Allot_time(H_s))로 표현된다. 이후 H_s의 실행

시간 CH_s가 0이 된 경우 종료되며(Kill(H_s)) 과정으로 기술됨) 0보다 큰 경우는 다음 번 실행을 위하여 다시 Queue_H에 유입된다(Queue(H_s) in Queue_H 과정으로 표현됨).

이때 서버는 Queue_H를 다시 탐색하며(goto Search_Queue_H 과정으로 기술됨) 조건에 부합하는 경성 실시간 태스크들이 없는 경우에 멀티미디어 태스크들의 대기 행렬인 Queue_M을 탐색한다. 서버는 Queue_M에 존재하는 태스크들 중 우선순위가 가장 높은 것을 High_prior_M으로 정의하면서 모든 태스크들을 비교하여 종료시한이 가장 이른 태스크 M_s를 선정하여 실행하게 된다(Execute(M_s))로 기술함). 이때 태스크 M_s의 실제 실행 시간이 평균 실행시간 CM_s보다 커지는 경우에도 계속 CPU를 사용할 수 있으나 서버의 주기내에서 E_M을 초과할 수는 없다.

M_s는 실행 도중에 새로운 경성 실시간 태스크 H_{new}가 시스템에 도착하는 경우 선점되며(Preempt(M_s)로 기술됨) 이제까지의 실행 시간(M_s가 실행된 시간을 Execute_time(M_s)로 기술함)을 E_M과 CM_s에서 각각 감하게 된다. H_{new}가 없었던 경우에는 M_s가 정상적으로 실행된 시간을 E_M과 CM_s에서 각각 감하게 된다. CM_s가 0이 되는 경우에는 해당 멀티미디어 태스크 M_s의 실행이 종료되며 시스템에서 제거되고 CM_s가 0보다 큰 경우에는 다음 번 서버의 주기 때 실행되도록 Queue_M에 다시 유입된다.

3. 성능 평가

3.1 시뮬레이션 모델

제시된 동적 스케쥴링 기법은 주기가 상이한 5 개의 경성 실시간 태스크들과 5 개의 멀티미디어 태스크들을 대상으로 성능이 분석되었다. 먼저 경성 실시간 태스크들의 주기와 최악의 경우에 대한 실행 시간(단위 ms)은 다음과 같이 정의된다.

H_1=(30, 3), H_2 = (50, 5), H_3 = (70, 7),
H_4 = (90, 9), H_5 = (110, 11)

멀티미디어 태스크들의 주기와 평균 실행 시간은 다음과 같다.

M_1 = (40, 4), M_2 = (60, 6), M_3= (80, 8),
M_4 = (100, 10), M_5 = (120, 12)

멀티미디어 태스크들의 실제 실행 시간은 1과 2*평균 실행 시간 -1 사이에 존재하는 임의의 값이 설정되도록 균등 분포 함수가 이용되었다. 경성 실시간 태스크들과 멀티미디어 태스크들은 수식 (1)의 결과에 따라 CPU 자원에 대한 활용도는 1.0 이하가 되며 모든 태스크들이 스케쥴링될 수 있음을 의미한다. 최소 주기가 30 ms인 H_1의 주기를 서버의 주기로 설정할 때 경성 실시간 태스크들과 멀티미디어 태스크들은 수식 (2)와 수식 (3)의 결과에 의하여 각각 15 ms만큼의 실행 시간 E_H와 E_M을 각각 가지게 된다.

$$E_H = \sum_{i=1}^{5} CH_i \times \frac{T_s}{TH_i} = 15 \text{ ms}$$

$$E_M = \sum_{j=1}^{5} CM_j \times \frac{T_s}{TM_j} = 15 \text{ ms}$$

본 실험에서 CPU에 대한 태스크들의 문맥 교환(*context switching*)시 이에 수반되는 오버헤드는 고려하지 않았다. 제시된 최소주기 서버 스케쥴링 방법은 고정 대역폭 스케쥴링(CBS; Constant Bandwidth Scheduling)[3] 기법과 성능이 비교 분석되었다. CBS 기법은 멀티미디어 태스크의 평균 실행 시간과 주기를 기반으로 모든 멀티미디어 태스크마다 별도의 서버가 설정되며 평균 시간을 초과하는 CPU 사용 시간이 요청된 경우 서버의 다음 번 주기때 이를 수용하는 스케쥴링 방법이다.

3.2 성능 분석

제시된 스케쥴링 기법(이하 DS 기법으로 기술함)의 목표는 경성 실시간 태스크들의 종료시한을 모두 보장하면서 멀티미디어 태스크들의 종료시한이 경과되는 시간을 최소화시키는 것이다. 본 실험에서는 경성 실시간 태스크 5개와 멀티미디어 태스크 5개를 동일한 시스템에서 스케쥴링함으로써 멀티미디어 태스크들의 종료시한에 대한 지연 시간을 측정하였다. DS 기법과 CBS 기법 모두 경성 실시간 태스크들의 종료시한을 정확히 보장하는 실험 결과가 확인되었다.

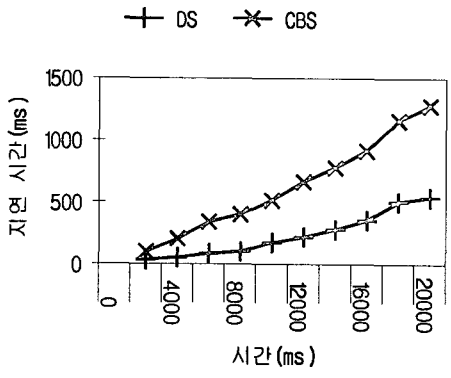

그림 2. 종료시한이 경과된 지연 시간

그림 2는 멀티미디어 태스크들에 대해서 매 2000 ms 마다 종료시한이 경과된 시간의 합을 해당 멀티미디어 태스크들의 수로 나눈 평균 지연 시간을 측정한 결과이다. 제시된 DS 기법은 CBS 기법보다 종료시한을 초과하는 평균 지연 시간이 보다 작은 것으로 확인되었다. 즉 평균 실행 시간을 초과하는 멀티미디어 태스크를 최소주기 서버내의 멀티미디어 태스크들에 설정된 실행 시간 E_M의 범위내에서 가능하면 계속적으로 실행하는 것이 종료시한이 경과되는 시간을 감소시킬 수 있음을 의미한다. CBS 기법의 경우 항상 주기내에 설정된 평균 시간만큼만 해당 태스크를 스케쥴링하기 때문에 DS 기법보다 종료시한이 경과되는 지연 시간이 커지게 된다.

표 1. 디코딩된 프레임의 수

단위 시간: ms

기법 \ 시간	2000	4000	6000	8000	10000	12000
DS	141	284	422	567	711	852
CBS	131	236	403	545	674	816

표 1은 매 2000 ms 시간마다 멀티미디어 태스크들에 의해 실제로 디코딩된 프레임의 수를 나타내고 있다. 제시된 DS 기법이 CBS 기법보다 동일한 시간 간격 동안 더 많은 수의 프레임들을 디코딩하는 것으로 파악되었다. 서버의 주기마다 멀티미디어 태스크들이 사용할 수 있는 실행 시간을 최대한 활용한 결과로 분석된다.

4. 결론

멀티미디어 정보와 데이터를 제어 목적으로 활용할 수 있는 분산 실시간 멀티미디어 시스템에서 경성 실시간 태스크들과 연성 실시간적 멀티미디어 태스크들이 공존하게 된다. 두 종류의 실시간 태스크들을 효율적으로 스케쥴링할 수 있는 동적 기법이 본 논문에서 제시되었다. 태스크들의 주기중 가장 작은 것이 서버의 주기로 설정되며 주기내에서 경성 실시간 태스크들과 멀티미디어 태스크들이 CPU를 사용할 수 있는 시간이 각각 설정된다. 특히 경성 실시간 태스크들은 서버의 주기마다 일정 시간만큼만 실행되는 반면 멀티미디어 태스크들은 예상했던 평균 시간보다 CPU를 더 많이 사용하려는 경우에도 멀티미디어 태스크들에 대한 전체 실행 시간 범위내에서는 해당 태스크가 계속적으로 실행될 수 있도록 동적으로 스케쥴링되고 있다. 결과적으로 경성 실시간 태스크들은 모두 종료시한내에 종료될 수 있으며 멀티미디어 태스크들은 종료시한이후 실행이 완료되는 지연 시간이 최소화될 수 있다. 실제 프레임을 구성하는 데이터의 크기가 예상보다 큰 경우에도 가능하면 화면에 출력되는 시간이 최소화되도록 함으로써 본 논문의 스케쥴링 기법은 멀티미디어 태스크의 실시간성이 향상되는 분산 제어 환경에 적용될 것으로 기대된다.

[참고문헌]

[1]H. Kaneko, and et al., "Integrated Scheduling of Multimedia and Hard Real-time Tasks," In proc. of IEEE Real-Time Systems Symposium, Dec. 1996.

[2]O. Gonzalez and et al., "Incorporation of Multimedia Capabilities in Distributed Real-time Applications," Workshop on Databases: Active and Real-Time, Nov. 1996.

[3]L. Abeni and et al., "Integrating Multimedia Applications in Hard Real-time Systems," In proc. of IEEE Real-Time Systems Symposium, Dec. 1998.

[4]C. L. Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment," Journal of the ACM, Vol. 20, No. 1, 1973.

[5]K. Jeffay, "Scheduling Sporadic Tasks with Shared Resources in Hard Real-time Systems," In proc. of IEEE Real-Time Systems Symposium, Dec. 1992.

[6]L. Sha and et al., "Priority Inheritance Protocols: an Approach to Real-time Synchronization," IEEE Transactions on Computers, Vol. 39, No. 9, 1990.

[7]C. W. Mercer and et al., Processor Capacity Reserves for Multimedia Operating Systems, Technical Report CMU-CS-93-157, Carnegie Mellon University, May 1993.

[8]L. Abeni and G. Buttazzo, "Adaptive Bandwidth Reservation for Multimedia Computing," In proc. of IEEE Conf. on Real-Time Computing Systems and Applications, Dec. 1999.