Journal of Korea Multimedia Society Vol. 7, No. 7, July 2004(pp. 886-895)

경성 실시간 태스크와 MPEG 비디오 스트림 기반 멀티미디어 태스크를 위한 CPU 대역폭의 동적 할당 기법

김 진 환[+]

요 약

본 논문에서는 멀티미디어 태스크와 경성 실시간 태스크가 공존하는 시스템에서 두 태스크들을 효율적으로 통합하고 스케쥴링하기 위한 CPU 대역폭의 동적 할당 기법이 제시된다. 연성 실시간적인 멀티미디어 태스크는 평균 실행 시간을 기반으로 처리되는 반면 경성 실시간 태스크는 최악의 경우에 대한 실행 시간을 기반으로 실행이 보장된다. 서버를 기반으로 하는 본 논문의 할당 기법은 CPU 대역폭을 두 태스크들에 대하여 분할한 후 특히 MPEG 비디오 스트림을 위한 멀티미디어 태스크들에 대하여는 할당된 대역폭을 다시 동적으로 조정한다. 제시된 기법의 목적은 시스템에 존재하는 경성 실시간 태스크들의 시간적 제약성을 모두 보장하면서 멀티미디어 태스크들의 종료시한이후 실행이 지연되는 시간을 최소화하는 것이다. 본 논문에서는 시뮬레이션 실험을 통하여 제시된 기법에서 멀티미디어 태스크들의 지연 시간이 다른 기법에 비하여 작아지는 결과를 보여주고 있다.

Dynamic Allocation Method of CPU Bandwidth for Hard Real-Time Task and Multimedia Task Based on MPEG Video Stream

Jinhwan Kim[+]

ABSTRACT

In this paper, we propose the dynamic allocation scheme of the CPU bandwidth to efficiently integrate and schedule these tasks in the same system, where multimedia tasks and hard real-time tasks can coexist simultaneously. Hard real-time tasks are guaranteed based on worst case execution times, whereas multimedia tasks modeled as soft real-time tasks are served based on mean parameters. This paper describes a server-based allocation scheme for assigning the CPU resource to two types of tasks. Especially for MPEG video streams, we show how to dynamically control the fraction of the CPU bandwidth allocated to each multimedia task. The primary purpose of the proposed method is to minimize the mean tardiness of multimedia tasks while satisfying the timing constraints of hard real-time tasks present in the system. We showed through simulations that the tardiness experienced by multimedia tasks under the proposed allocation scheme is much smaller than that experienced by using other scheme.

Key words: MPEG, Hard Real-Time(경성 실시간), Bandwidth(대역폭), Scheduling(스케쥴링), Deadline (종료시한)

※ 교신저자(Corresponding Author) : 김진환, 주소 : 서울특별시 성북구 삼선동 3가 389(136-792), 전화 : 02)760-4340, FAX : 02)760-4488, E-mail : kimjh@icc.hansung.ac.kr
접수일 : 2003년 4월 15일, 완료일 : 2003년 11월 4일
[+] 정회원, 한성대학교 컴퓨터공학부
※ 본 연구는 2004년도 한성대학교 교내연구비 지원 과제임.

1. 서 론

실시간 제어 응용 분야에서 멀티미디어 정보를 제어 환경에 포함시키는 경우 제어 프로세스의 상태를 보다 직관적이며 가시적으로 표현할 수 있고 프로세

스의 품질이나 안전도를 더욱 향상시킬 수 있다[1, 2]. 또한 관리와 감시 측면에서도 정확한 의사 결정을 내릴 수 있게 된다. 최근 공장 자동화와 군사적 방위 시스템 등의 경성 실시간(hard real-time) 시스템 응용 분야에서 음성 및 영상 등의 멀티미디어 정보가 이용되고 있다[3]. 이러한 정보는 종료시한(deadline)을 반드시 준수해야 하는 경성 실시간적 제어 정보보다는 덜 중요하나 지연 시간과 지터(jitter) 허용 등 연성 실시간적(soft real-time) 특성을 가지게 된다. 실제 멀티미디어 정보를 처리하는 태스크는 연성 실시간 태스크로 모델링되고 있다[4]. 경성 실시간 시스템에서 멀티미디어 정보를 이용하는 경우 멀티미디어 태스크와 경성 실시간 태스크들이 시스템에 공존하게 되며 이들을 효율적으로 통합하여 스케쥴링할 수 있는 기법이 필요하게 된다.

동일한 시스템에서 경성 실시간 태스크와 멀티미디어 태스크를 통합하여 스케쥴링하는 기법들은 이미 여러 논문에서 발표된 바 있다[3-6]. 일정한 주기를 갖는 경성 실시간 태스크들은 최악의 실행 시간(WCET; Worst Case Execution Time)을 기반으로 스케쥴링되며 연성 실시간 특성을 갖는 멀티미디어 태스크들은 대부분 평균 실행 시간을 기반으로 스케쥴링되고 있다. 경성 실시간 태스크들과는 달리 연성 실시간 태스크들은 종료시한이 경과되더라도 시스템에 미치는 영향이 심각하지 않은 것으로 간주되기 때문에 최악의 실행시간(WCET)보다는 평균 실행 시간을 기반으로 스케쥴링되는 것이다. 기존의 실시간 스케쥴링 기법인 최조종료시한(EDF; Earliest Deadline First) 기법이나[7,8] RM(Rate Monotonic) 기법은[5,7,9] 사실상 경성 실시간 태스크들을 위한 기법이므로 각 태스크의 WCET가 종료시한내에 실행되는 것을 보장하고 있다. 그러나 처리하고자 하는 데이터의 규모에 따라 실행시간이 가변적인 멀티미디어 태스크는 WCET를 기반으로 하는 경성 실시간 태스크의 스케쥴링 기법을 적용할 경우 CPU 대역폭의 낭비가 심해지며 결과적으로 자원의 활용도가 낮아지는 문제가 발생하게 된다[4]. 특히 MPEG 비디오 스트림을 처리하는 멀티미디어 태스크의 경우 프레임의 크기에 따라 실제 디코딩 시간이 달라지기 때문에 WCET를 설정하는 것 자체도 어려운 문제이며 WCET보다는 평균 실행 시간을 고려하여 스케쥴링하는 것이 바람직하다. 멀티미디어 태스크의 평균 실행 시간을 기반으로 스케쥴링하는 경우에도 대부분의 MPEG 디코더는 각 프레임의 처리 특성과 크기를 반영하는 우선순위를 설정하지 않고 있다. 본 논문에서는 MPEG 비디오 스트림을 구성하는 프레임 특성에 따라 우선순위를 설정하고 우선순위에 따라 프레임을 디코딩하는 방법이 제시된다.

특히 멀티미디어 태스크와 경성 실시간 태스크를 통합하여 스케쥴링하는 기존의 연구 결과에서도 MPEG 비디오 스트림에 대한 프레임의 특성이 제대로 반영되지 않고 있다. 본 논문에서는 MPEG 비디오 응용을 위한 멀티미디어 태스크와 경성 실시간 태스크들을 효율적으로 통합하고 스케쥴링할 수 있는 CPU 대역폭 할당 기법이 제시된다. 스케쥴링 기능을 전담하는 서버는 CPU 대역폭을 경성 실시간 태스크들이 사용하는 대역폭과 멀티미디어 태스크들이 사용하는 대역폭으로 분할한다. 이후 WCET를 기반으로 하는 경성 실시간 태스크와 평균 실행 시간을 기반으로 하는 멀티미디어 태스크를 위하여 일정한 CPU 대역폭이 태스크별로 할당된다. 경성 실시간 태스크의 경우에는 서버의 주기마다 할당된 CPU 대역폭의 사용이 보장된다. 그러나 본 논문에서는 MPEG 비디오 스트림의 다양한 프레임들을 효과적으로 디코딩하기 위해서 우선순위 정책이 사용되며 멀티미디어 태스크에 할당된 CPU 대역폭이 동적으로 조정된다.

제시된 CPU 대역폭 할당 기법을 이용하여 MPEG 비디오 스트림을 위한 멀티미디어 태스크들을 스케쥴링하는 경우 지연 시간(각 태스크의 종료시한 이후 태스크가 실제 종료될 때까지 소요된 시간)을 최소화할 수 있으며 경성 실시간 태스크들의 종료시한도 모두 보장할 수 있다. 멀티미디어 응용에서 연성 종료시한(soft deadline)이 경과될 경우 치명적인 시스템의 결함이 발생하는 것은 아니나 QoS가 저하된다. 본 논문에서는 멀티미디어 태스크의 지연시간을 최소화함으로써 MPEG 비디오 스트림의 QoS가 저하되는 것을 방지하고자 하였다. 제시된 기법의 성능은 시뮬레이션을 이용한 실험 결과에서 분석된다. 본 논문의 구성은 다음과 같다. 2 장에서는 실시간 태스크 모델과 CPU 대역폭 할당 기법이 기술되며 3 장에서는 경성 실시간 태스크와 멀티미디어 태스크를 스케쥴링하는 서버의 알고리즘과 예제가 설명된다. 4 장에서는 제시된 기법의 성능이 시뮬레이션 결과를 토대로 분석되며 5 장에서는 결론이 기술된다.

2. 실시간 태스크를 위한 CPU 대역폭 할당

2.1 대역폭 예약

본 논문에서 실시간 태스크는 WCET와 일정한 주기를 갖는 경성 실시간 태스크 그리고 평균 실행시간과 일정한 주기를 갖는 MPEG 비디오 스트림 기반 멀티미디어 태스크로 구성된다. 임의의 경성 실시간 태스크 H_i의 WCET와 주기는 WCET(H_i)와 Period(H_i)로 표기되며 MPEG 기반 멀티미디어 태스크 M_j의 평균 실행 시간과 주기는 Mean(M_j)와 Period(M_j)로 각각 표기된다. 스케줄링 기능을 수행하는 멀티미디어 서버의 주기는 경성 실시간 태스크와 멀티미디어 태스크들 중 가장 주기가 작은 것으로 결정된다. 따라서 멀티미디어 서버도 일정한 주기를 갖는 경성 실시간 태스크로 수행되며 이 서버는 T_s로 표기된다. 멀티미디어 서버 T_s의 주기는 Period(T_s)로 표기되며 T_s의 실행 시간은 주기보다 작거나 같은 시간으로 설정된다. T_s의 실행시간은 사실상 CPU 대역폭을 의미하며 이 대역폭 B_T는 경성 실시간 태스크들을 위한 대역폭 B_H와 멀티미디어 태스크들을 위한 대역폭 B_M으로 분할된다.

$$B_T = B_H + B_M \quad \text{(수식 1)}$$

경성 실시간 태스크 H_i가 서버 T_s의 주기내에서 실행되는 시간 Budget(H_i)는 WCET(H_i)를 Period(T_s)/Period(H_i) 비율과 곱한 값으로 결정된다.

$$\text{Budget}(H_i) = \text{WCET}(H_i)\frac{\text{Period}(T_s)}{\text{Period}(H_i)} \quad \text{(수식 2)}$$

즉 H_i는 서버 T_s의 주기 내에서 WCET(H_i) 동안 실행되는 것이 아니라 Budget(H_i) 동안만 실행 시간이 보장되는 것이다. 따라서 서로 다른 n개의 경성 실시간 태스크들을 위한 대역폭 B_H는 수식 3에 의하여 계산된다.

$$B_H = \sum_{i=1}^{n}\text{Budget}(H_i) = \sum_{i=1}^{n}(\text{WCET}(H_i)\frac{\text{Period}(T_s)}{\text{Period}(H_i)}) \quad \text{(수식 3)}$$

MPEG 기반 멀티미디어 태스크 M_j가 T_s내에서 실행되는 시간 Budget(M_j)는 수식 2와 동일한 방법으로 계산된다.

$$\text{Budget}(M_j) = \text{Mean}(M_j)\frac{\text{Period}(T_s)}{\text{Period}(M_j)} \quad \text{(수식 4)}$$

태스크 M_j는 서버 T_s의 주기 내에서 Budget(M_j) 시간만큼 대역폭이 예약되는 것이며 m개의 멀티미디어 태스크들을 위한 대역폭 B_M은 다음과 같이 결정된다.

$$B_M = \sum_{j=1}^{m}\text{Budget}(M_j) = \sum_{j=1}^{m}(\text{Mean}(M_j)\frac{\text{Period}(T_s)}{\text{Period}(M_j)}) \quad \text{(수식 5)}$$

경성 실시간 태스크는 최악의 경우에 대한 실행 시간을 가정하기 때문에 일정 회수 이상의 서버가 수행되면 반드시 종료시한내에 태스크의 실행 종료가 보장되는 반면 멀티미디어 태스크는 평균 실행 시간을 가정하기 때문에 실제로 요구된 실행 시간이 평균 시간보다 큰 경우 설정된 종료시한이 경과될 수 있다.

본 논문에서는 이론상 CPU 자원의 활용도가 69%인(평균적인 경우를 분석할 시 88%까지 증가될 수 있음[7,9]) RM 스케줄링 기법보다 활용도를 100%까지 증가시킬 수 있는 EDF 스케줄링 기법[7,8]을 채택함으로써 자원의 활용도를 극대화하고자 하였다. EDF 스케줄링 원칙에[7] 따라 각 태스크의 실행 시간을 주기로 나눈 값(이하 자원 활용률로 정의함)들의 합이 1.0보다 작거나 같을 때만 태스크들이 스케줄링될 수 있는 것으로 간주한다[10].

$$\sum_{i=1}^{n}\frac{\text{WCET}(H_i)}{\text{Period}(H_i)} + \sum_{j=1}^{m}\frac{\text{Mean}(M_j)}{\text{Period}(M_j)} = U_H + U_M \le 1.0 \quad \text{(수식 6)}$$

수식 6에서 경성 실시간 태스크들에 대한 자원 활용률과 멀티미디어 태스크들에 대한 자원 활용률은 U_H와 U_M으로 각각 표기된다. 스케줄링 도중 새로운 경성 실시간 태스크 또는 멀티미디어 태스크가 시스템에 유입되는 경우는 수식 6에 따라 승인 여부가 결정된다. 새로운 경성 실시간 태스크를 반드시 처리해야 하는 특별한 경우에는 U_M을 감소시킨 만큼 U_H를 증가시킴으로써 새로운 경성 실시간 태스크를 스케줄링할 수 있다. 그러나 기존 멀티미디어 태스크들에 대한 처리 시간이 지연되는 현상을 감수해야 하며 이에 대한 구체적인 스케줄링 방법은 본 논문에서 기술하지 않기로 한다.

2.2 MPEG 기반 멀티미디어 태스크

MPEG 비디오 스트림을 위한 압축 알고리즘은 이

미지의 시간과 공간의 중복성을 최대한 이용하여 동영상 데이터를 압축하며 세계적인 표준으로 활용되고 있다. 압축이 수행된 MPEG-1 또는 MPEG-2 비디오 스트림은 계층 구조상 일정 개수의 픽처 또는 프레임들로 구성된 GOP(group of picture) 층이 형성된다[11]. 이 GOP 층은 I, P, B, D 프레임 등의 픽처들이 저장되며 임의로 접근할 수 있는 단위가 된다. 실제로 MPEG-1과 MPEG-2 비디오는 한 GOP를 구성하는 프레임 수에 의해 서비스 품질(QoS; quality of service) 수준이 결정된다. 이는 MPEG 표준의 압축 변수인 N과 M에 의하여 달라지며 N은 한 GOP 내의 프레임 수를 M은 I 또는 P 프레임이 나타나는 주기를 의미한다. 예를 들어 N=15, M=3일 경우 GOP 내에서 프레임들의 배열은 IBBPBBPBBPBB PBB 순서가 된다.

WCET가 아닌 평균 실행 시간을 기반으로 스케쥴링되는 멀티미디어 태스크 M_j의 특성을 감안할 경우 MPEG 비디오 스트림은 프레임의 크기가 종류별로 상이하기 때문에 태스크마다 동일한 평균 실행 시간을 설정하는 것이 사실상 어렵다. 프레임의 크기가 가장 큰 I 프레임에 대한 평균 실행 시간을 Mean(M_j)로 설정할 경우 I 프레임보다 작은 P 프레임과 B 프레임을 처리할 때 CPU 대역폭이 낭비되는 결과가 발생한다. 또한 I 프레임보다는 작고 B 프레임보다는 큰 P 프레임의 평균 실행 시간을 Mean(M_j)로 설정할 경우 B 프레임을 처리할 때는 대역폭의 낭비가 발생되는 반면 I 프레임을 처리할 때는 대역폭이 부족한 현상이 발생하게 된다. 따라서 본 논문에서는 특정 프레임에 대한 평균 실행 시간을 직접 반영하는 대신 한 GOP에 대한 평균 실행 시간을 설정한 후 이 시간을 GOP를 구성하는 프레임의 수(예, N=15)로 나눈 값을 태스크 M_j의 평균 실행 시간인 Mean(M_j)로 설정함으로써 평균 실행 시간의 변동성을 다소 감소시키고자 하였다. 한 GOP를 구성하는 프레임 수 N이 15인 경우 15개의 태스크들이 동일한 평균 실행 시간을 갖게 된다. 이러한 방법으로 설정된 평균 실행 시간은 GOP내의 프레임 크기와 수에 따라 변동성이 있으나 I 프레임과 P 프레임을 디코딩하기 위한 평균 실행 시간보다는 작게 되며 B 프레임을 디코딩하기 위한 평균 실행 시간보다는 큰 값을 갖는 것으로 분석되었다.

2.3 우선 순위 설정

MPEG 동영상 압축 기법의 특성상 GOP내의 기준 프레임인 I 프레임을 먼저 디코딩하지 않고서는 P 프레임이나 B 프레임들을 디코딩할 수 없다[12]. 결국 GOP내의 모든 B 프레임들을 디코딩하기 위해서는 I 프레임과 P 프레임을 먼저 디코딩해야 한다. 본 논문에서는 이러한 특성을 위하여 각 프레임을 처리하는 멀티미디어 태스크마다 우선 순위를 설정한다. 즉 I 프레임을 디코딩하는 태스크는 우선순위를 1로 설정하며 P 프레임을 처리하는 태스크와 B 프레임을 처리하는 태스크의 우선순위는 2와 3으로 각각 설정한다. 우선순위 번호가 작을수록 논리적인 우선순위가 높게 설정됨을 의미하며 멀티미디어 태스크들은 우선순위별로 구성된 큐(queue)에서 각각 대기하게 된다. 서버가 멀티미디어 태스크를 스케쥴링하게 될 경우 우선순위 순서에 따라 우선순위가 가장 높은 큐부터 태스크를 찾게 되며 이때 동일한 우선순위의 태스크들이 여러 개 있는 경우에는 EDF 방법을 적용하여 종료시한이 가장 빠른 태스크를 선택하게 된다. 따라서 I 프레임을 디코딩하는 태스크가 P 프레임을 디코딩하는 태스크보다 먼저 스케쥴링되며 B 프레임을 디코딩하는 태스크보다는 P 프레임을 디코딩하는 태스크가 먼저 스케쥴링될 수 있는 기회를 갖게 된다. 본 논문에서는 D 프레임을 제외한 I, P, B 프레임을 디코딩하는 세 부류의 멀티미디어 태스크들이 구성된다. 서버가 선택한 임의의 멀티미디어 태스크가 실행되는 동안 우선순위가 더 높은 다른 멀티미디어 태스크에 의하여 선점(preemption)되는 경우는 제시된 기법에서 허용되지 않는다.

경성 실시간 태스크들은 가장 높은 우선순위인 0이 설정되며 별도의 큐에서 대기하게 된다. 서버는 큐에서 종료시한이 가장 빠른 경성 실시간 태스크를 선택하게 된다. 선택된 경성 실시간 태스크 H_i는 서버의 주기내에서 할당된 시간 즉 Budget(H_i)보다 작거나 같은시간 동안만 실행이 보장된다. 그리고 H_i가 실행되는 동안 다른 태스크에 의하여 선점되는 현상은 발생하지 않는다. 본 논문에서는 경성 실시간 태스크들 간은 물론 멀티미디어 태스크들 간에도 선점 현상을 허용하지 않는다. 그러나 멀티미디어 태스크들이 실행되는 도중 경성 실시간 태스크에 의하여 선점되는 경우는 허용된다. 반드시 종료시한을 준수해야 하는 경성 실시간 태스크의 실행을 보장하기 위하여 우선순위를 가장 높게 설정한 것이며 종료시한이 경과된 이후에도 실행될 수 있는 멀티미디어 태스크들에 대하여는 이보다 낮은 우선순위를 설정하여 선점 현상이 허용되도록 하였다. 따라서 경성

실시간 태스크가 우선순위가 더 낮은 멀티미디어 태스크로 인하여 실행이 지연되거나 종료시한이 경과되는 경우는 없다.

3. 멀티미디어 서버의 태스크 스케줄링

3.1 스케줄링 기법

경성 실시간 태스크들과 멀티미디어 태스크들 중 가장 작은 주기를 갖는 멀티미디어 서버 T_s는 매 주기마다 그림 1과 2에서 기술된 알고리즘에 따라 태스크들을 스케줄링하게 된다. 편의상 행 번호가 부여된 그림 1과 2에서는 다음과 같이 정의된 용어와 변수들이 사용된다.

Queue-i: 우선순위가 i(0 이상 3 이하의 정수)인 태스크들의 큐

Selects(H_i 또는 M_j): 해당 큐에서 경성 실시간 태스크 H_i 또는 멀티미디어 태스크 M_j를 선택함.

Deadline(H_i 또는 M_j): H_i 또는 M_j의 종료시한

ED(k): 우선순위가 k인 태스크들 중 가장 빠른 종료시한

Priority(M_j): M_j의 우선순위

HP: 멀티미디어 태스크들 중 가장 높은 우선순위

Executes(H_i 또는 M_j): H_i 또는 M_j가 실행됨.

Exec_time(H_i 또는 M_j): H_i 또는 M_j가 실행된 시간

Queues(H_{new} 또는 M_j): H_{new}또는 M_j를 해당 큐에 대기시킴.

멀티미디어 서버는 우선순위가 가장 높은 경성 실시간 태스크들을 먼저 스케줄링하게 되며 이 과정은

```
1.  while ( B_H > 0 )
2.    begin
3.      for i = 1 to n in Queue-0
4.        if Budget(H_i) > 0 and Deadline(H_i) = ED(0)
5.            Selects(H_i);
6.      if H_i exists
7.        begin
8.          Executes(H_i);
9.          B_H = B_H - Exec_time(H_i);
10.         Budget(H_i) = Budget(H_i) - Exec_time(H_i);
11.       end
12.   end
```

그림 1. 경성 실시간 태스크를 위한 스케줄링 알고리즘

```
1.  while ( B_M > 0 )
2.    begin
3.      for k = 1 to 3 in Queue-k
4.        for  j = 1 to m in Queue-k
5.          if Priority(M_j) = HP and Deadline(M_j) = ED(k)
6.              Selects(M_j);
7.      if M_j exitsts
8.        begin
9.          Executes(M_j);
10.         if H_new arrives
11.           begin
12.             Queues(M_j) in Queue-Priority(M_j);
13.             Queues(H_new) in Queue-0
14.           end
15.         B_M = B_M - Exec_time(M_j);
16.       end
17.   end
```

그림 2. 멀티미디어 태스크를 위한 스케줄링 알고리즘

그림 1의 1행부터 12행까지 while 순환문 내에 기술된다. 서버 내에 경성 실시간 태스크들을 위한 대역폭 B_H가 0보다 큰 경우 Queue-0에 있는 n 개의 태스크들 중 Budget(H_i)가 0보다 크고 가장 종료시한이 빠른 태스크 H_i를 선택한다(4행과 5행에서 기술됨). 서버의 주기내에 할당된 대역폭을 모두 소진한 태스크는 Budget(H_i)가 0이 되므로 종료시한이 빠르더라도 선택되지 않으며 이 태스크는 다음 서버의 주기에서 새로 대역폭을 할당받은 후 스케줄링될 수 있다. 만일 Queue-0에서 4행의 두 조건을 만족하는 경성 실시간 태스크 H_i가 존재하는 경우에는 실행이 시작된다. 이때 H_i의 실제 실행 시간 Exec_time(H_i)은 Budget(H_i)보다 작거나 같을 수 있다. 9행과 10행에서 대역폭 B_H와 Budget(H_i)은 H_i가 실행된 Exec_time(H_i) 시간만큼 각각 감소된다. 이후 서버는 여전히 B_H가 0보다 큰 경우 다른 경성 실시간 태스크를 찾게 된다. 그러나 6행에서 조건에 부합하는 경성 실시간 태스크 H_i가 존재하지 않을 경우 서버는 멀티미디어 태스크들의 큐를 검색하기 위하여 그림 2의 1행부터 17행까지 기술된 while 순환문을 수행하게 된다.

멀티미디어 태스크들을 위한 대역폭 B_M이 0보다 큰 경우 우선순위가 가장 높은 큐부터 종료시한이 가장 빠른 태스크 M_j를 선택한다. 그림 2의 3행부터 5행까지의 과정에서 우선순위가 1인 태스크들이 존재하면 그 중 종료시한이 가장 빠른 태스크가 선택되

며 3행과 4행의 for 순환문을 모두 벗어나게 된다. 그러나 우선순위가 1인 태스크들이 존재하지 않는 경우에는 우선순위가 2인 태스크들을 대상으로 종료 시한이 가장 빠른 태스크를 선택하게 되며 우선순위가 2인 태스크들이 없는 경우에는 마지막으로 우선순위가 3인 태스크들을 탐색하게 된다. 적절한 M_j가 선택된 경우 서버는 실행을 시작하게 된다(7행부터 기술됨). 이때 서버는 M_j를 평균 실행 시간 Mean(M_j) 동안만 실행하는 것이 아니고 해당 프레임을 디코딩하는 시간만큼 필요한 대역폭을 사용하게 한다. I 프레임이나 P 프레임을 디코딩하는 경우에는 Mean(M_j)보다 많은 대역폭이 필요하므로 서버는 전체 멀티미디어 태스크들을 위한 대역폭 B_M 범위내에서 다른 멀티미디어 태스크에 할당된 대역폭의 사용을 허용한다. 따라서 M_j는 서버의 주기에서 최대 B_M만큼의 CPU 대역폭을 사용할 수 있으며 이를 사용하고도 해당 프레임의 디코딩이 완료되지 않은 경우에는 다음 주기의 서버에서 다시 실행될 기회를 갖게 된다. 본 논문에서는 처리하는 프레임 종류에 따라 태스크의 우선순위가 설정되며 우선순위가 높은 태스크일수록 많은 대역폭을 사용하여 해당 프레임의 디코딩 시간을 감소시킴은 물론 태스크가 종료시한이 경과된 이후 실행이 완료되는 시간(이하 멀티미디어 태스크의 지연 시간으로 정의함)을 최대한 감소시키고자 하였다.

그러나 M_j가 실행중인 동안 새로운 경성 실시간 태스크 H_{new}(H_1과 H_n 사이의 태스크로 가정함)가 시스템에 도착하면 서버는 일단 M_j를 해당 우선순위의 큐에 대기시키고 H_{new}도 경성 실시간 태스크 큐인 Queue-0에 대기시킨다. 12행과 13행은 멀티미디어 태스크 M_j가 우선순위가 더 높은 경성 실시간 태스크에 의하여 선점되는 것을 의미한다. 선점이 된 경우나 해당 프레임의 디코딩이 완료된 경우 그리고 B_M 대역폭을 모두 사용한 경우 서버는 이제까지 실행된 시간 Exec_time(M_j)를 B_M에서 감소시킨다(10행에서 기술됨). 이후 H_{new}와 같이 새로운 경성 실시간 태스크가 존재하고 B_H가 0보다 큰 경우에는 그림 1의 과정을 다시 수행하게 된다. 그러나 스케쥴링할 수 있는 경성 실시간 태스크가 없고 아직 B_M이 0보다 큰 경우에는 다른 멀티미디어 태스크를 큐에서 탐색하기 위하여 그림 2의 while 순환문 내용을 다시 수행하게 된다.

3.2 알고리즘 적용 사례

두 개의 경성 실시간 태스크 H_1과 H_2의 WCET와 주기 그리고 두 개의 멀티미디어 태스크 M_1과 M_2의 평균 실행시간과 주기는 다음과 같이 설정되며 단위 시간은 ms(1/1000초)로 가정한다.

WCET(H_1) = 6, Period(H_1) = 30
WCET(H_2) = 15, Period(H_2) = 50
Mean(M_1) = 12, Period(M_1) = 40
Mean(M_2) = 12, Period(M_2) = 60

M_1과 M_2는 GOP를 구성하는 프레임의 수 N이 15이며 프레임의 순서는 IBBPBBPBBPBBPBB로 가정한다. 그리고 15개의 프레임을 디코딩하는 평균 시간을 180ms로 가정하여 Mean(M_1)과 Mean(M_2)는 모두 12(= 180/15)ms로 설정된 것이다. 네 태스크들 중 주기가 가장 작은 H_1의 주기가 서버 T_s의 주기로 설정되며 주기내에 할당되는 각 태스크의 대역폭은 다음과 같이 계산된다.

$$\text{Budget}(H_1) = \text{WCET}(H_1)\frac{\text{Period}(T_s)}{\text{Period}(H_1)} = 6\times\frac{30}{30} = 6$$

$$\text{Budget}(H_2) = \text{WCET}(H_2)\frac{\text{Period}(T_s)}{\text{Period}(H_2)} = 15\times\frac{30}{50} = 9$$

(수식 2 참조)

$$\text{Budget}(M_1) = \text{Mean}(M_1)\frac{\text{Period}(T_s)}{\text{Period}(M_1)} = 12\times\frac{30}{40} = 9$$

$$\text{Budget}(M_2) = \text{Mean}(M_2)\frac{\text{Period}(T_s)}{\text{Period}(M_2)} = 12\times\frac{30}{60} = 6$$

(수식 4 참조)

경성 실시간 태스크에 대한 대역폭 B_H와 멀티미디어 태스크에 대한 대역폭 B_M은 다음과 같이 결정된다.

$$B_H = \text{Budget}(H_1) + \text{Budget}(H_2) = 6+9 = 15$$

(수식 3 참조)

$$B_M = \text{Budget}(M_1) + \text{Budget}(M_2) = 9+6 = 15$$

(수식 5 참조)

서버의 대역폭 B_T는 B_H와 B_M의 합이며 이 예제의 경우 서버의 주기와 대역폭이 같다고 가정한다. 즉 태스크들 간의 문맥 교환(context switching) 오버헤드는 없는 것을 가정한 결과이다. 그리고 수식 6에서 정의된 자원 활용율 U_H와 U_M은 다음과 같이 각각 0.5이며 합이 1.0이하이므로 EDF 스케쥴링이 가능하다[10].

$$U_H = \frac{WCET(H_1)}{Period(H_1)} + \frac{WCET(H_2)}{Period(H_2)} = \frac{6}{30} + \frac{15}{50} = 0.2 + 0.3 = 0.5$$

$$U_M = \frac{Mean(M_1)}{Period(M_1)} + \frac{Mean(M_2)}{Period(M_2} = \frac{12}{40} + \frac{12}{60} = 0.3 + 0.2 = 0.5$$

그림 1과 2에서 기술된 멀티미디어 서버의 스케쥴링 알고리즘에 따라 H_1과 H_2 그리고 M_1과 M_2가 스케쥴링되는 순서가 그림 3에서 기술된다. H_1, M_1, H_2, M_2 들이 시스템에 처음 도착한 시간은 각각 5, 9, 13, 17ms로 가정한다. 이후에는 각 태스크의 주기에 따라 새로운 태스크들이 도착하게 된다. 시간 5ms(이하 단위 시간은 생략함)에서 서버는 최초로 H_1을 Budget(H_1)(= 6)만큼 실행하게 된다. 태스크의 실행 중인 시간은 그림 3에서 상향 대각선으로 표현된다. 시간 9에서 도착한 M_1은 실제 실행시간이 Mean(M_1)(= 12)보다 1이 큰 13이며 우선순위가 2인 것으로 가정한다. 서버는 H_1의 실행이 완료되는 시간 11까지 M_1을 해당 큐에서 대기시키며(그림 3에서 대기 시간은 W 문자로 표현됨) 11이 되면 M_1을 실행하게 된다. 시간 11에서 Budget(H_1)은 0이 되며 B_H는 9(= 15 - Exec_time(H_1) = 15 - 6)가 된다. 시간 13에서 도착한 H_2는 우선순위가 낮은 M_1을 즉시 선점하고 Budget(H_2)(= 9) 동안 CPU 대역폭을 사용하게 된다. 시간 13에서 B_M은 13(= 15 Exec_time(M_1) = 15 - 2)이 된다. 시간 17에서 도착한 M_2는 실제 실행 시간이 Mean(M_2)보다 작은 8이며 우선순위는 3으로 가정한다. 시간 22에서 Budget(H_2)와 B_H는 모두 0(= 9 - Exec_time(H_2) = 9 - 9)이 된다. 서버는 아직 실행이 완료되지 않은 H_2를 Queue-0에서 대기시킨 후 멀티미디어 태스크 M_1과 M_2 중 M_1의 우선순위가 더 높기 때문에 M_1을 실행하게 되며 M_2는 큐 Queue-3에서 계속 대기하게 된다. 시간 33에서 M_1의 실행이 완료되면 서버는 남은 B_M(= 13 Exec_time(M_1) = 13 - 11 = 2)동안 M_2를 실행한다.

시간 35가 되면 새로운 H_1이 도착하고 각 태스크에 대한 CPU 대역폭이 새로 초기화된다. 따라서 서버는 M_2의 실행을 중단하고 경성 실시간 태스크인 H_1과 H_2 중 종료시한이 더 빠른 H_2를 실행하게 된다. 시간 35에서 도착한 H_1의 종료시한은 65(= 35+30)이며 시간 13에서 도착한 H_2의 종료시한은 63(= 13+50)이다. 이때 H_2의 남은 실행시간은 6(= 15-9)이므로 서버는 시간 41에서 Budget(H_2)를 3(= 9-6)으로 B_H는 9(=15-6)로 각각 설정한다. 그리고 서버는 시간 35에 도착한 후 41까지 큐에서 대기 상태로 있었던 H_1을 시간 47까지 실행한다. 시간 47에서는 Budget(H_1)은 0이 되며 B_H는 3(= 9-6)이 된다. 이 시점에서 스케쥴링할 수 있는 경성 실시간 태스크가 없기 때문에 서버는 M_2를 실행하게 되며 시간 49에서 도착한 새로운 M_1은 M_2의 실행이 완료될 때까지 큐에서 대기하게 된다. 이후의 스케쥴링 과정은 유사한 과정이 반복되므로 생략한다. 이 예제에서 경성 실시간 태스크 H_1과 H_2는 항상 종료시한내에 실행 완료가 보장되고 있다.

4. 성능 평가

4.1 시뮬레이션 환경

본 논문에서 제시된 CPU 대역폭 할당 기법의 목적은 모든 경성 실시간 태스크들의 종료시한을 반드시 보장하면서 MPEG 비디오 응용을 위한 멀티미디어 태스크들이 종료시한을 경과한 후 실행이 종료되는 시간 즉 지연 시간을 최소화하는 것이다. 디코딩되는 프레임의 종류에 따라 멀티미디어 태스크의 우선순위가 설정되는 본 논문의 대역폭 할당 기법은 PBA(Priority-based Bandwidth Allocation) 기법으로 기술한다. 제시된 할당 기법은 고정 대역폭 스케쥴링(CBS; Constant Bandwidth Scheduling)[5]을 기반으로 하는 NPBA(Non-Priority-based Bandwidth Allocation) 방법과 성능이 비교 분석되었다.

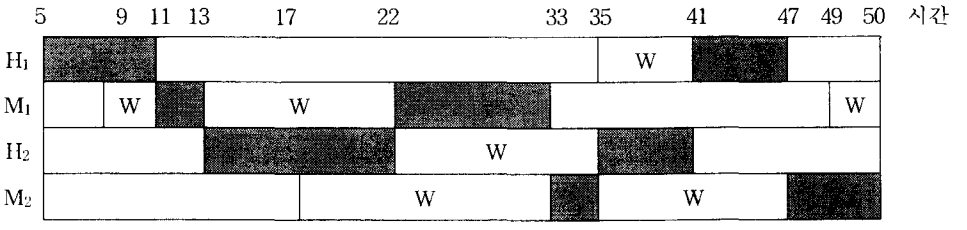


그림 3. H_1, M_1, H_2, M_2 태스크들의 스케쥴링 과정과 순서

본래 CBS 기법은 멀티미디어 태스크의 평균 실행 시간과 주기를 기반으로 모든 멀티미디어 태스크마다 별도의 서버가 설정되며 평균 시간을 초과하는 CPU 대역폭이 요청된 경우 다음 주기의 서버에서 이를 수용하는 스케쥴링 방법이다. 즉 멀티미디어 태스크의 실행 시간이 예상 평균 시간보다 증가한 경우 다른 멀티미디어 태스크의 실행 시간을 이용할 수 없으며 MPEG 비디오 프레임의 특성이 거의 반영되지 않고 있다. CBS 기법을 다소 변형한 NPBA 기법도 PBA 기법과 동일하게 태스크들 중 가장 작은 주기를 서버의 주기로 설정하며 경성 실시간 태스크들과 멀티미디어 태스크들에 대한 대역폭이 각각 설정된다. 그러나 NPBA 기법에서는 멀티미디어 태스크의 우선순위가 이용되지 않으며 모든 멀티미디어 태스크는 서버의 주기마다 할당된 대역폭만을 사용하게 된다.

실험 결과 PBA 기법과 NPBA 기법 모두 경성 실시간 태스크들의 종료시한이 항상 보장되는 것으로 분석되었기 때문에 멀티미디어 태스크들에 대한 성능 평가 결과만 기술하기로 한다. 3.2 알고리즘 적용 사례에서 기술된 것처럼 M_1과 M_2의 GOP는 모두 15개의 프레임으로 구성되며 GOP에 대한 평균 디코딩 시간이 180ms인 동일한 MPEG 비디오를 디코딩하는 것으로 가정한다. 각 프레임의 평균 디코딩 시간은 다음과 같이 설정된다.

I 프레임의 평균 디코딩 시간: 55.380ms
P 프레임의 평균 디코딩 시간: 13.845ms
B 프레임의 평균 디코딩 시간: 6.924ms

MPEG-1 비디오 스트림의 경우 프레임 간의 크기 비율은 스트림의 종류에 따라 다르다[13]. 움직임이 적절한 영화의 경우 I, P, B 프레임 간의 평균 크기 비율이 약 8:2:1인 점을 감안하여 각 프레임의 평균 실행 시간을 위와 같이 설정하였다. 시뮬레이션에서 각 프레임의 실제 실행 시간은 프레임의 크기 변동성에 따라 일정 범위내에서 ±50% 범위와 ±80% 내에서 변동성이 유지되도록 임의의 난수 생성기가 사용되었다.

4.2 결과 분석

3.2 예제에서 기술된 네가지 태스크 H_1, M_1, H_2, M_2들에 대한 시뮬레이션을 수행하여 멀티미디어 태스크들이 매 2000ms마다 실제 디코딩이 완료된 태스크들의 수가 누적된 결과가 표 1과 표 2에 나타나 있다. 표 1은 각 프레임을 디코딩하는 실행시간의 변동성이 평균 시간의 ±50% 범위 내에서 유지된 결과이며 표 2는 실행시간의 변동성이 평균 시간의 ±80% 범위 내에서 유지된 결과이다. 표 1과 표 2에서 Tasks 항목은 시스템에 의하여 생성된 M_1과 M_2 태스크들의 누적된 숫자를 의미하는 것이며 PBA 기법과 NPBA 기법에서 괄호안의 숫자는 종료시한 이후에 디코딩이 완료된 태스크들의 수를 의미한다. 표 1과 표 2에서 나타난 바와 같이 PBA 기법은 NPBA 기법보다 디코딩이 완료된 태스크들 수 자체가 다소 많은 것으로 측정되었다. 그러나 종료시한이 경과된 후 디코딩이 완료된 태스크들의 수는 점차 시간이 지남에 따라 PBA 기법보다 NPBA 기법이 현저히 증가되는 결과를 보이고 있다.

표 1과 표 2에서 괄호 안에 표기된 수가 클수록 멀티미디어 태스크들의 지연시간이 결과적으로 증가하는 것을 의미하게 된다. 결과적으로 멀티미디어

표 1. 디코딩이 완료된 멀티미디어 태스크들의 수(실행시간 변동성 ±50%)

기법 \ 시간	2000ms	4000ms	6000ms	8000ms	10000ms	12000ms
Tasks	84	167	250	334	417	500
PBA	76(51)	159(121)	250(195)	329(248)	406(318)	499(389)
NPBA	73(50)	153(121)	243(209)	321(286)	399(365)	494(444)

표 2. 디코딩이 완료된 멀티미디어 태스크들의 수(실행시간 변동성 ±80%)

기법 \ 시간	2000ms	4000ms	6000ms	8000ms	10000ms	12000ms
Tasks	84	167	250	334	417	500
PBA	70(51)	167(122)	243(167)	328(232)	412(307)	499(357)
NPBA	72(57)	163(138)	243(193)	325(263)	410(345)	495(412)

태스크들에 대한 대역폭 B_M을 효과적으로 이용함으로써 PBA 기법은 우선순위를 사용하지 않는 NPBA 기법보다 종료시한이후 실행이 종료되는 태스크들의 수를 감소시킴으로써 실시간적 성능이 우수한 것으로 분석된다.

멀티미디어 태스크 M_1의 경우 주기가 40ms이므로 초당 재생되는 프레임 수는 25가 되며 M_2의 경우 주기가 60ms이므로 초당 재생되는 프레임 수는 약 16.7이 된다. 실험 결과 M_1의 1초 동안 디코딩된 평균 프레임 수가 PBA 기법에서는 24.93으로 나타났으며 NPBA 기법에서는 24.67인 것으로 분석되었다. 즉 NPBA 기법보다 PBA 기법에서 1초 동안 재생되는 프레임 수가 다소 높은 것으로 나타났다. M_2의 경우에도 PBA 기법에서 재생된 프레임 수는 16.65이며 NPBA 기법에서 재생된 프레임 수는 16.48로 근소한 차이이긴 하나 높은 것으로 분석되었다. 즉 두 기법에 대한 시간적 해상도 차이가 존재하며 이러한 결과는 멀티미디어 태스크의 수를 증가시킬수록 초당 재생되는 프레임 수의 차이가 점차 커지게 되는 실험 결과가 확인되었다.

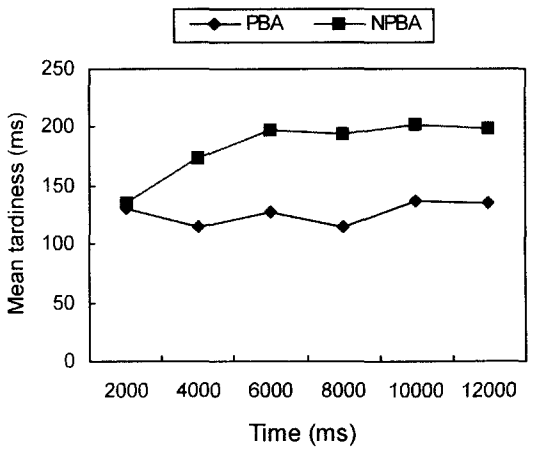


그림 4. 평균 지연 시간 비교

매 2000ms마다 누적된 태스크들의 지연 시간(실제 실행이 완료된 시간과 종료시한의 차)을 표 1에서 기술된 괄호안의 숫자로 나눈 값은 멀티미디어 태스크의 평균 지연 시간으로 정의하며 이 결과는 그림 4에 나타나 있다. 서버의 주기내에서 대역폭 B_M을 우선순위에 따라 최대한 활용하는 PBA 기법은 NPBA 기법에 비하여 평균 지연 시간이 훨씬 작은 것으로 분석되었다. NPBA 기법은 태스크마다 서버의 주기내에서 할당된 대역폭만을 사용하므로 I 프레임이나 P 프레임과 같이 평균 실행 시간보다 큰 프레임을 디코딩할 경우 소요되는 시간이 많아지게 되며 이는 결국 태스크의 종료시한을 상당히 경과한 후 디코딩이 완료되는 결과를 유발한다. 이는 다른 태스크들도 디코딩을 시작할 수 있는 시간 자체가 늦어지는 현상을 만들게 되어 PBA 기법에 비하여 평균 지연 시간이 시간이 지남에 따라 점점 커지게 되는 것으로 분석되었다. 실행시간의 변동성이 ±80% 범위일 때의 결과(표 2 참조)에 대한 평균 지연 시간 비교도 그림 4와 큰 차이가 없는 것으로 분석됨에 따라 이에 대한 그림을 생략한다.

12000ms까지 두 기법에 의하여 각 프레임이 디코딩 과정동안 소요된 평균 실제 디코딩 시간이 표 3에 나타나 있다. 실제 디코딩 시간은 각 태스크가 해당 프레임에 대한 디코딩을 일단 시작한 이후 완료될 때까지의 시간을 의미하기 때문에 도중에 큐에서 대기 상태로 머물렀던 시간도 모두 포함이 된다. 표 3의 결과는 실행시간의 변동성이 ±50% 범위에서 적용된 것이다. PBA 기법은 NPBA 기법에 비하여 I 프레임과 P 프레임의 평균 실제 디코딩 시간이 훨씬 작은 것으로 나타났으며 B 프레임의 경우에는 두 기법의 평균 실제 디코딩 시간이 거의 동일한 것으로 측정되었다. 대기 상태가 없는 각 프레임의 순수 디코딩 시간(I 프레임: 55.380ms, P 프레임: 13.845ms, B 프레임: 6.924ms)과 비교하면 두 기법 모두 대기 상태로 인하여 실제 디코딩 시간이 2~4배 정도 증가되었음을 알 수 있다.

표 3. 각 프레임의 평균 디코딩 시간

단위: ms

기법 \ 프레임	I	P	B
PBA	106.55	28.40	26.51
NPBA	212.28	50.98	26.17

5. 결 론

경성 실시간 태스크와 MPEG 비디오 응용을 위한 멀티미디어 태스크들이 동일한 시스템에 존재할 경우 이 태스크들을 효율적으로 스케쥴링할 수 있는 CPU 대역폭 할당 기법이 본 논문에서 제시되었다. 태스크들 중 가장 작은 주기를 서버의 주기로 정하며

서버의 대역폭은 경성 실시간 태스크와 멀티미디어 태스크를 위한 대역폭으로 분할된다. 경성 실시간 태스크는 최악의 경우에 대한 실행 시간을 기반으로 자신의 주기와 서버의 주기에 비례한 만큼 서버의 매 주기마다 일정한 대역폭을 할당받기 때문에 항상 종료시한 내에 실행 완료가 보장된다. 한편 멀티미디어 태스크는 연성 실시간적 특성을 가지기 때문에 종료시한 이후에 실행이 완료되어도 시스템에 치명적인 영향을 미치지 않는다.

그러나 본 논문에서는 멀티미디어 태스크가 종료시한 이후에 실행이 완료되는 시간을 최소화하여 서비스 품질을 향상시키고자 하였다. MPEG 동영상 기법에서 사용되는 각 프레임의 특성을 고려하기 위하여 우선순위를 멀티미디어 태스크에 설정하였으며 멀티미디어 태스크를 위한 대역폭 범위 내에서 우선순위가 높을수록 CPU 대역폭을 많이 사용할 수 있도록 대역폭을 동적으로 조정하였다. 제시된 기법은 우선순위를 사용하지 않는 기법에 비하여 동일한 시간 동안 더 많은 수의 프레임들을 디코딩할 수 있으며 태스크의 평균 지연 시간과 프레임에 대한 평균 디코딩 시간을 감소시킴으로써 실시간적 성능을 향상시킬 수 있는 것으로 분석되었다.

참 고 문 헌

[1] O. Gonzales and et al., Incorporation of multimedia capabilities in distributed real-time applications, *Workshop on Databases: Active and Real-time*, 1996.

[2] L. Palopoli and et al, Real-time control system analysis: an integrated approach, *Proc. of IEEE Real-Time Systems Symposium*, Dec. 2000.

[3] Kaneko and et al., Integrated scheduling of multimedia and hard real-time tasks, *Proc. of IEEE Real-Time Systems Symposium*, Dec. 1996.

[4] L. Abeni and et al., Integrating multimedia applications in hard real-time systems, *Proc. of IEEE Real-Time Systems Symposium*, Dec. 1998.

[5] L. Abeni and G. Buttasso, Adaptive bandwidth reservation for multimedia computing, *Proc. of IEEE Conf. on RTCSA*, Dec. 1999.

[6] M. Caccamo and et al., Handling execution overruns in hard real-time control systems, *IEEE Transactions on Computers, Vol. 51, No. 7*, pp. 835-849, 2002.

[7] 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.

[8] K. Jeffay, "Scheduling sporadic tasks with shared resources in hard real-time systems," *Proc. of IEEE Real-Time Systems Symposium*, Dec. 1992.

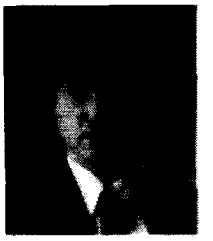
[9] L. Sha and et al., "Priority inheritance protocols: an approach to real-time synchronization," *IEEE Transactions on Computers, Vol. 39, No. 9*, 1990.

[10] I. Stoica, H. Abdel-Wahab and K. Jeffay, "On the duality between resource reservation and proportional share resource allocation", *Proc. of Mutltimedia Computing and Networking*, Feb. 1997.

[11] Mitchell, J. L., et al., MPEG video compression standard, Chapman & Hall, 1996.

[12] M. Ditze and P. Altenbernd, A method for real-time scheduling and admission control of MPEG-2 streams, *Proc. of 7th Australasian Conference on Parallel and Real-Time Systems (PART2000)*, Nov. 2000.

[13] John Watkinson, MPEG Handbook, Butterworth-Heinemann, Oct. 2001.



김 진 환

1986년 서울대학교 컴퓨터공학과 졸업(학사)
1988년 서울대학교 컴퓨터공학과 졸업(석사)
1994년 서울대학교 컴퓨터공학과 졸업(박사)
1994년~1996년 서울대학교 컴퓨터신기술공동연구소 특별연구원
1995년~현재 한성대학교 컴퓨터공학부 부교수
관심분야 : 멀티미디어 시스템, 분산 실시간 시스템