실시간 태스크 모델을 이용한 VBR비디오 서버에 대한 연구

o
오 명훈*, 정 민수* , 이 재동*
*경남대학교 컴퓨터 공학과

A Study on the VBR Video Server using the Real-Time Task Models

o
Myung-Hun Oh* , Min-Su Jung* , Jae-Dong Lee*
* Dept of Computer Engineering, Kyung-Nam University

요 약

멀티미디어를 실시간으로 서비스하는 비디오 서버의 설계 시, 네트워크 가용공간, 버퍼 공간 및 디스크 전송률 등이 고려되어야 하며, 특히 버퍼 공간과 디스크 전송률은 서로 많은 영향을 미치기 때문에 동시에 고려되어야 한다. 기존 연구에서는 CBR을 근간으로 하여 버퍼 공간과 디스크 전송률을 동시에 고려하는 기법이 제시되었지만, VOD 서버에서는 보통 MPEG데이터가 VBR형식으로 되어있기 때문에 CBR로 모델링하면 시스템의 가용성이 떨어지게 된다. 또한 수용제어를 위한 버퍼 예약기법에서는 VBR형태로 데이터를 모델링하는 기법들이 제시되었지만, 이 경우에도 디스크 전송률을 계산할 때는 CBR형식으로 가정하거나 아예 고려를 하지 않기 때문에 역시 시스템 자원을 효율적으로 사용할 수 없다. 본 논문에서는 VBR비디오 서버의 설계를 위하여 서버에서 처리되는 각 스트림들을 하드 실시간 태스크 모델로 모델링하고 실시간 태스크들을 효율적으로 처리할 수 있는 실시간 스케쥴링 알고리즘을 제안한다. 이 실시간 스케쥴링 알고리즘을 디스크 액세스의 승인제어, 디스크 스케쥴링, 버퍼 관리에 이용하므로서 연속성을 보장할 수 있는 효율적인 VBR연속 미디어 서버를 설계할 수 있다. 비디오 서버의 성능은 시뮬레이션을 이용하여 분석하였다.

1. 서론

VOD(Video On Demand)와 같은 멀티미디어(Multimedia)시스템은 시스템 자원을 많이 필요로 하는 동시에 실시간(real-time)이어야 한다는 제약조건을 가지고 있다. 즉, 오디오 또는 비디오들은 시간적 특성을 가진 데이터이다. 따라서, 연속성을 보장해야 된다. 특히, 서비스 되어야 할 스트림(stream)들을 디스크로부터 읽어들여 클라이언트(client)의 장비(Set-Top Box혹은 PC등)로 보내기 전에 잠시 버퍼에 저장되어야 한다. 따라서, 어느 순간 지원 가능한 사용자 혹은 스트림의 수는 디스크에서 데이터를 읽어들일 수 있는 능력(disk bandwidth)과 읽어들인 데이터를 잠시 저장하는데 필요한 버퍼의 총 크기 및 네트워크 가용능력 등에 의해 결정된다고 할 수 있다.

어느 클라이언트가 VOD시스템의 서비스를 받기를 원한다면 시스템에서는 그 클라이언트를 시스템 내로 들여 보내도 기존의 서비스 받던 클라이언트들에게 영향을 미치지 않고 정상적인 서비스를 계속할 수 있는 지를 판단하여 문제가 없는 경우 새로운 클라이언트에게도 서비스를 개시하게 된다.

본 논문에서는 VBR(Variable Bit Rate)특성을 고려하여 각 스트림들을 관리하는 방법을 제시하였다. 기존의 연구는 주로 CBR(Constant Bit Rate)을 기본으로 하는 기법이다[1,2]. CBR이란 각 스트림이 소비하는 데이터가 일정한 비율이므로 VOD서버가 일정한 비율로 각 클라이언트(스트림)에게 데이터를 제공해야 한다. 그러나, 현재 사용하고 있는 비디오들은 대개 MPEG데이터이므로 VBR특성을 가진다. 즉, 각 스트림이 소비하는 데이터의 비율이 시시각각 변한다. 따라서, VBR특성을 가진 스트림을 CBR로 가정하여 서비스 하면 효율적인 자원 사용이 불가능하다. VBR특성을 이용하여 효율적인 버퍼할당을 할 수 있는 방법이 제시되었으나, 이는 보통 디스크 전송율은 고려하지 않았다[3,4,5]. VBR특성을 버퍼뿐만 아니라 디스크 전송율에 고려한 연구[6]도 있으나, 이것은 연속성의 보장이 되지 않는다.

본 논문에서는 VBR비디오 서버의 설계를 위하여 서버에서 처리되는 각 스트림들을 하드 실시간 태스크 모델로 모델링하고 실시간 태스크들을 효율적으로 처리할 수 있는 실시간 스케쥴링 알고리즘을 제안한다. 이 실시간 스케쥴링 알고리즘을 디스크 액세스의 승인제어, 디스크 스케쥴링, 버퍼 관리에 이용하므로서 연속성을 보장할 수 있는 효율적인 VBR비디오 서버를 설계할 수 있다. 서버의 성능은 시뮬레이션을 이용하여 분석하였다.

2. 하드 실시간 태스크 모델

본장에서는 효율적인 VBR 연속미디어 서버를 설계하기 위하여 서버에서 처리되는 각 스트림들을 하드 실시간 태스크 모델(hard real-time task model)로 모델링 한다. 연속미디어 데이터들은 서버에서 사이클 단위로 처리되며, 서버에서 재공된 데이터들은 버퍼를 거쳐 클라이언트들에 의하여 소비된다. 이런 점들을 고려하여 다음과 같은 하드 실시간 태스크 모델을 만들 수 있다.

현재 서버에 의하여 n개의 태스크(스트림을 나타냄)가 처리되고 있다고 가정하고, 각 태스크들을 S_1, S_2, . . . , S_n 이라 두자. 각 태스크들은 연속된 여러 개의 사이클에 의하여 처리되며, 연속된 사이클의 수는 스케쥴링 알고리즘에 의존한다. 서버는 그림1와 같이 사이클 방식으로 각 태스크들을 처리한다. 각 사이클 내의 태스크의 수행순서는 바뀔 수 있다. 각 사이클은 길이[1]가 T이고, T동안 n개의 태스크들을 처리한다. 각 태스크가 수행될 때 수행시간에 비례하게 데이터가 생성되고, 생성된 데이터는 버퍼에 저장된다. 버퍼에 저장된 데이터는 그림2처럼 클라이언드 또는 통신서버에 의하여 소비되며, 소비되는 양은 VBR특성을 고려하여 각 사이클 마다 다른 값으로 정해져 있다고 가정한다.

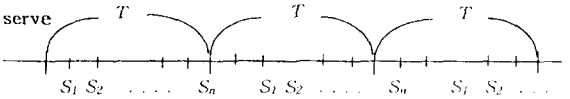


그림1. 스트림(태스크)들의 처리과정

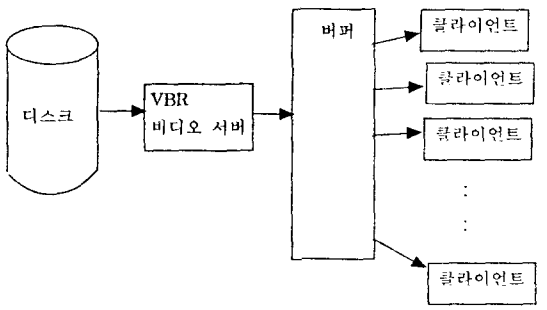


그림 2(a)

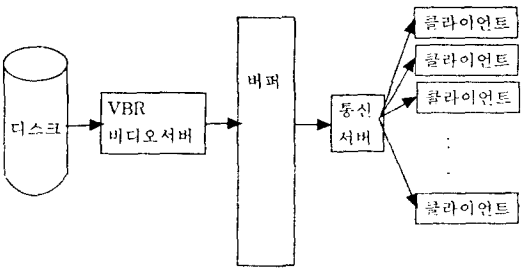


그림 2(b)

그림 2. VBR 비디오 서버

이 서버에는 크기가 B_{max}인 버퍼가 있다고 가정한다. 이 서버의 처리속도는 R이라고 하자. 즉, 한 태스크가 t초동안 수행되면 생성되는 데이터의 양은 $R*t$이다. 각 태스크들은 아래와 같은 특성을 가진다.

i) 각 태스크는 한 사이클내에서 비선점 (nonpreemptive)이다.

ii) 태스크 S_i의 $j(0 \le j \le l_i)$번째 사이클의 수행시간을 t_{ij}라 하고 이때, 생성되는 데이터의 양은 $R*t_{ij}$이다. 이 데이터는 버퍼에 저장된다. 태스크 S_i는

---

1) 디스크의 탐색시간과 회전지연시간을 고려하여 $T' = T + (\text{디스크 액세스를 위한 지연시간})$으로 두고, T'를 사이클 길이로 하여 각 사이클 동안 실제 수행될 수 있는 시간을 T로 생각 할 수 있다.

l_i+1 번의 연속된 사이클의 수행으로 처리가 끝난다.

iii) S_i의 $k(1 \le k \le L_i)$번째 사이클에서의 소비양은 C_{ik}이다. $\sum_{k=1}^{l_i} C_{ik} = C_i$, C_i는 S_i에 필요한 전체 데이터의 양이다.

iv) 하드 실시간 요구사항

모든 $s(1 \le s \le L_i)$에 대하여

$$\sum_{j=0}^{s-1} t_{ij} * R - \sum_{j=1}^{s} C_{ij} \ge 0 \qquad \text{(조건 1)}$$

즉, 태스크에 의하여 생성된 양이 소비되는 양보다 같거나 많아야 한다. S_i의 0번째 사이클에서 만들어진 데이터는 1번째 사이클에서 소비되는 양보다 같거나 많아야 한다. $k(0 \le k \le l_i)$번째 사이클까지 생성된 데이터의 양은 k번째 사이클까지 소비된 양 보다 같거나 많아야 한다.

v) $\sum_{j=0}^{l_i} t_{ij} = C_i / R$

vi) $C_i^{max} = \underset{1 \le k \le L_i}{Max} C_{ik}$

현재 사이클이 태스크 S_i의 j번째 실행 사이클일 때 $t(S_i) = t_{ij}$ 라 두고, $B(S_i) = \sum_{k=0}^{j} t_{ik} * R - \sum_{k=1}^{j} C_{ik}$ 라 두면 $t(S_i)$는 S_i가 현 사이클에서 수행에 필요한 시간이고 $B(S_i)$는 S_i를 위해 현 사이클까지 필요로 하는 버퍼의 양이다. 따라서, 모든 사이클에 대해 아래의 요구사항이 만족되어야 한다.

i) $t(S_1) + t(S_2) + ... + t(S_n) \le T$ (조건 2)

ii) $B(S_1) + B(S_2) + ... + B(S_n) \le B_{max}$ (조건 3)

각 스트림 S_i에 대하여 S_i가 수행되는 모든 사이클 동안 최대로 필요한 버퍼의 양을 B_i^{max}라 두면 모든 사이클에 대해 아래 조건이 만족되어야 한다.

iii) $B_1^{max} + B_2^{max} + B_3^{max} + ... + B_n^{max} \le B_{max}$ (조건 4)

(조건 4)가 만족되면 당연히 (조건 3)은 만족된다.

3. 하드 실시간 태스크 스케쥴링 알고리즘

연속미디어 서버의 설계를 위해서는 2.1절에서 제시한 모델에서 태스크들을 수행하기 위한 스케쥴링 알고리즘이 필요하다. 기존에 제시된 CBR을 기반으로 한 알고리즘과 VBR을 기반으로 한 알고리즘을 제시하고, 문제점을 분석한다. 이런 문제점들을 극복할 수 있는 새로운 알고리즘의 방향을 제시한다.

스케쥴링 알고리즘은 2장에서 언급한 (조건1), (조건2), (조건4)를 만족시키도록 스케쥴링 하여야 한다.

3.1 고정 수행시간 알고리즘(FCA : Fixed Exeution time Algorithm)

이 알고리즘은 연속미디어 서버가 CBR를 기반으로 수행하는 경우의 스케쥴링 알고리즘이다. 모든 태스크 S_i에 대하여 실행되는 모든 사이클에서 똑 같은 수행시간을 가지도록 스케쥴링한다. (조건1)을 만족시키기 위하여 $t_i = t_{ij} = C_i^{max} / R$, $1 \le j \le l_i$ 로 한다. 이때, 모든 사이클에 걸쳐 S_i에 필요한 최대 버퍼량 $B_i^{max} = \sum_{j=0}^{l_i} t_{ij} - \sum_{j=1}^{l_i} C_{ij}$ 이다. C_{ij}의 변화가 심하면 심할수록 B_i^{max}는 커진다. 다른 알고리즘에 비하여 B_i^{max}는 아주 큰 값이다. n개의 태스크 S_1, S_2, ... , S_n이 스케쥴된 상태에 새로운 태스크 S_{n+1}이 도착했을 때 S_{n+1}을 다음 사이클에 스케쥴시키기 위한 요구조건은 아래와 같다.

i) $\sum_{i=1}^{n+1} \le T$

ii) $\sum_{i=1}^{n+1} B_i^{max} \le B_{max}$

위에 언급한 바와 같이 스케쥴링 방식이 간단하고 매 사이클마다 태스트 수행시간이 고정되어 있어서, 태스크의 수행을 빨리시켜 비어 있는 서버의 수행시간을 기다리는 태스크들에게 줌으로써 서버의 이용율을 높힐 수 있다. 하지만, 과도한 버퍼요구 때문에 제한된 버퍼를 가진 서버에서는 성능이 떨어진다.

3.2 최소 버퍼 요구 알고리즘(MBA : Minimum Buffer Algorithm)

이 알고리즘은 순수한 VBR을 기반으로 한 연속미디어 서버를 위한 알고리즘이다. 각 사이클에서 소모할 데이터 양 만큼만 생산하므로써 버퍼의 요구양을 최소로 하는 알고리즘이다. 즉, $t_{ij} = C_{ij} / R$로 둠으로서, 다음 사이클에 필요한 데이터 양만큼 만들기 위해 각 사이클 마다의 수행시간을 변경시킨다. 이렇게 하므로써 버퍼가 $B_i^{max} = 2 * C_i^{max}$만 있으면 S_i를 처리하는데 문제가 없다. 그러나 각 태스크가 연속되게 수행되기 위해서 새로운 태스크를 스케쥴시키기 위해서는 모든 사이클의 t_{ij}를 C_i^{max} / R로 가정하여야 한다. 다시 말해서, 이 알고리즘을 사용하면 서버의 수행시간 중 쉬는(idle) 부분이 많아진다. 서버가 쉬고

있어도 다른 태스크를 스케쥴할 수 없다. 만약 쉬는 부분에 다른 태스크를 스케쥴하면 (조건1)을 만족시킬 수 없는 상황이 생길 수 있다. 또한 이런 태스크의 수행이 L_i 사이클에 걸쳐 수행되어야 한다. 즉, $l_i = L_i$ 이다. n개의 태스크가 존재할 때, S_{n+1}를 스케쥴 하기 위해서는 다음 조건이 만족되어야 한다.

i) $t_1^{max} + t_2^{max} + ... + t_n^{max} + t_{n+1}^{max} \le T$

ii) $B_1^{max} + B_2^{max} + ... + B_n^{max} + B_{n+1}^{max} \le B_{max}$

여기서, $t_i^{max} = C_i^{max} / R$, $1 \le i \le n+1$

$B_i^{max} = 2 * C_i^{max}$, $1 \le i \le n+1$

이 알고리즘은 최대 필요한 버퍼를 최소로 할 수 있지만, 연속성을 만족시키기 위하여 L_i사이클 동안 각 태스크를 수행시켜야 하며, 각 사이클에서 수행되는 시간을 최악의 경우로 가정하여 스케쥴되므로 실제로 쉬는 시간이 많다. 즉, 서버의 처리시간의 많은 낭비를 초래한다. 또한 각 사이클에서 필요로 하는 C_{ij} $(1 \le j \le L_i)$의 정보를 저장하고 있으면서 매 사이클마다 t_{ij}를 C_{ij}를 참조하여 변경시켜야 한다.

3.3 혼합 알고리즘 (Mixed algorithm)

3.1의 알고리즘은 버퍼가 과도하게 필요한 반면, 3.2의 알고리즘은 서버의 수행시간의 손실이 과도하고 매 사이클의 소비량을 저장하고 있어야 되며, 매 사이클마다 이를 참조하여 태스크의 수행시간을 수정해야 한다. 본 장에서는 이 두가지의 단점을 보완하는 2개의 알고리즘을 혼합한 형태인 알고리즘을 제시한다.

본 알고리즘에서는 각 태스크 S_i의 수행시간 함수 $t_i(x)$를 미리 결정하여 이 함수를 사용하여 스케쥴링 한다. x번째 사이클에서의 수행시간을 $t_i(x)$, $0 \le x \le l_i$로 두면, 각 태스크 S_i가 (조건1)를 만족시키고 필요한 버퍼양을 최소로 하기 위하여 다음과 같은 조건이 만족되어야 한다.

r1) $\sum_{k=0}^{s} t_i(k) - \sum_{k=1}^{s+1} C_{ik} \ge 0$, $1 \le s \le L_i - 1$

r2) $t_i(x)$는 감소함수 이어야 한다.

r3) $\max_{s} (\sum_{k=0}^{s} t_i(k) - \sum_{k=1}^{s+1} C_{ik})$, $1 \le s \le L_i - 1$을 최소로 하는 $t_i(x)$를 선정해야 한다.

r1)은 연속성을 위한 조건이고, r2)는 각 사이클에서 서버의 수행시간의 쉬는 부분을 줄이기 위한 조건이며, r3)는 필요한 버퍼의 양을 감소시키기 위한 조건이다.

이때, S_{n+1}을 스케쥴 하기 위해서는 아래의 조건이 만족되어야 한다.

i) $t_1(cycle(S_1)) + t_2(cycle(S_2)) + ... + t_n(cycle(S_n)) + t_{n+1} \le T$

여기서 $cycle(S_i)$는 태스크 S_i의 현재 사이클번호.

ii) $B_1^{max} + B_2^{max} + B_3^{max} + ... + B_n^{max} + B_{n+1}^{max} \le B_{max}$

B_i^{max}는 앞 r3)에서 구한 $\max_{s} (\sum_{k=0}^{s} t_i(k) - \sum_{k=1}^{s+1} C_{ik})$ 값이다.

3.3.1 일차 함수 알고리즘(LFA : Linear Function Algorithm)

위 r1), r2), r3)를 만족시키기 위한 $t_i(x)$함수를 찾는 것은 어려운 일이다. 본 논문에서는 $t_i(x)$를 일차 함수로 두고 열거(enumeration)방법으로 추정하였다. $t_i(x) = ax + b$ 로 두고, $t_i(x)$가 감소 함수이기 위하여 $a<0$, b는 $x=0$일때의 값으므로 $b=C_i/R$로 선정한다.

열거 방법으로 r1)과 r3)를 만족시키는 a를 선정할 수 있다.

4 시뮬레이션

4.1 시뮬레이션 모델

알고리즘의 성능을 분석하기 위하여 아래의 파라메타로 시뮬레이션하여 필요한 버퍼의 양과 동시에 처리 가능한 스트림의 수를 구하여 분석 하였다.

- 사이클의 길이 T는 1초로 가정.
- 각 스트림의 소비율(Consumption rate)은 최소 20Kbyte/sec에서 최대 128Kbyte/sec 사이의 균일 분포(Uniform Distribution)로 가정.
- 최대 디스크 판독률은 2000Kbyte/sec로 가정.
- 각 스트림의 서비스 길이는 60, 600, 3600, 6000초에 대해서 각각 시뮬레이션.
- 서버의 버퍼 크기는 16, 32, 64, 128, 256, 512, 1024Mbyte에 대하여 각각 시뮬레이션 하였다. 모든 시뮬레이션은 100개의 스트림에 대해서 5번 반복한 결과의 평균값이다.

4.2 시뮬레이션 결과

4.2.1 최대 버퍼량

그림 3은 각 스트림을 위해 필요한 최대 버퍼양을 나타내고 있다. 각 스트림을 위해 필요한 최대 버퍼양은 VBR로 가정한 알고리즘(MBA)에서는 3장에서 언급하였듯이 $2 * C_i^{max}$만 필요한 반면, CBR로 가정한 알고리즘(FCA)에서는 많은 버퍼를 요구한다. 본 논문

에서 제시한 알고리즘(LFA)는 두 알고리즘이 필요로 한 양의 중간 정도 필요하다.

4.2.2 동시 수행가능한 스트림의 수

그림 4는 세 알고리즘에서 처리할 수 있는 평균 스트림의 수를 나타내고 있다. VBR로 가정한 알고리즘(MBA)은 스트림의 요구길이, 서버의 버퍼양에 관계없이 일정한 수의 스트림만 서비스할 수 있다.

반면 CBR을 가정한 알고리즘(FCA)와 LFA는 스트림의 요구길이와 서버의 버퍼양에 따라 변한다. 스트림의 요구길이가 짧든가, 또는 서버의 버퍼가 크면 본 논문에서 제시한 알고리즘(LFA)가 다른 알고리즘에 비해 훨씬 많은 스트림을 동시에 처리할 수 있다.

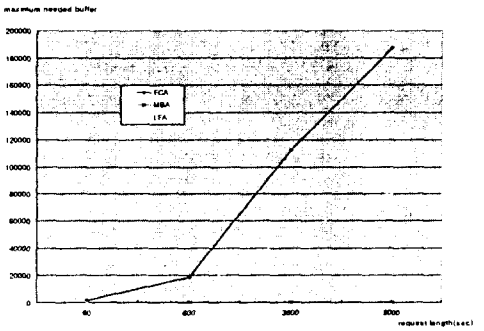
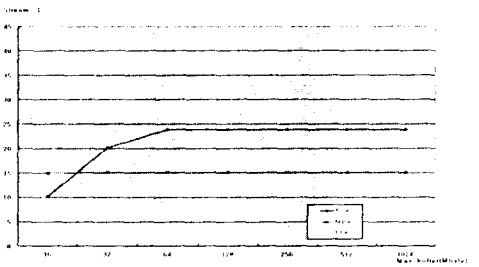
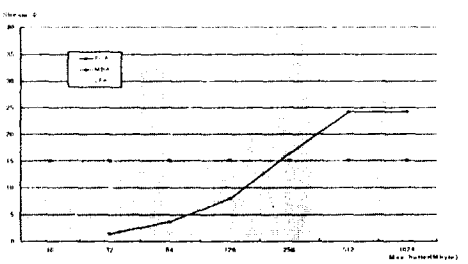


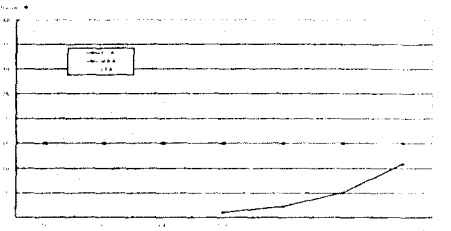
그림 3. 스트림의 필요한 최대 버퍼양



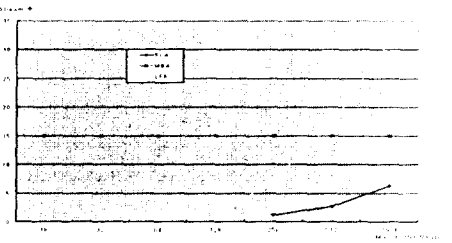
(a) request length = 60 초



(b) request length = 600 초



(c) request length = 3600



(d) request length = 6000

그림 4. 동시에 처리 가능한 평균 스트림의 수

5. 결론 및 향후 연구 과제

본 논문에서는 VBR비디오 서버의 설계를 위하여 서버에서 처리되는 각 스트림들을 하드 실시간 태스크 모델로 모델링하고 실시간 태스크들을 효율적으로 처리할 수 있는 실시간 스케쥴링 알고리즘을 제안하였다. 이 실시간 스케쥴링 알고리즘을 디스크 액세스의 승인제어, 디스크 스케쥴링, 버퍼 관리에 이용하므로서 연속성을 보장할 수 있는 효율적인 VBR연속 미디어 서버를 설계할 수 있다. 서버 성능은 시뮬레이션을 이용하여 분석하였다. 분석 결과 스트림의 요구 길이가 짧든가 서버의 버퍼가 크면 훨씬 많은 스트림들을 동시처리할 수 있었다.

향후 연구 과제는 다음과 같다.

- 이산모델로 변환하는 작업.
- VBR비디오 특성을 잘 반영하는 함수를 찾는 문제.
- 이 모델을 이용한 대화형 서비스 처리를 위한 방법을 찾는 작업.

참 고 문 헌

[1] H.J.Chen and T.D.C. Little, "Storage Allocation

Policies for Time-Dependent Multimedia Data,"IEEE Trans. on Knowledge and Data Engineering, Vol. 8, No. 5, pp. 855-864,1996.

[2] Ghandeharizadeh, S.H. Kim,and C. Shahabi,"On Configuring a Single Disk Continuous Media Server," in Proc. of ACM SIGMETRICS /PERFORMANCE Conf. on Measurement and Modeling of Computer Systems, pp.164-171, 1996.

[3] J. Gemmel and S. Christodoulakis,"Principles of Delay-Sensitive Multimedia Data Storage and Retrieval," ACM Trans. of Information Systems, Vol. 10, No. 1, pp. 51-90, 1992.

[4] K.L. Wu and P.S.Yu, "Consumption-Based Buffer Management for Maximizing System Throughput of a Mutimedia System", Proc. of IEEE International Conference on Multimedia Computing and Systems, pp.164-171, 1996.

[5] Y.S.Ryu and K.Koh, "A Dynamic Buffer Management Technique for Minimizing the Necessary Buffer Space in a Continuous Media Server", Proc. of IEEE ICMCS, pp.181-185,1996

[6] 이경오, 염헌영,"디스크 전송율과 버퍼 크기를 동시에 고려한 효율적 수용제어 기법", 정보과학회 논문지, 제 24권, 8호, pp. 731-740, 1997.