주문형 비디오 시스템을 위한 빠른 동적 방송 기법
(A Fast Dynamic Broadcasting Scheme For Video-On-Demand Systems)

권 혁 민 †
(HyeokMin Kwon)

요 약 주문형 비디오 시스템에서 방송에 기초한 스케쥴링 기법은 방대한 규모의 클라이언트에게 인기 비디오를 전파하기 위한 매우 효율적인 기술로 알려져 있다. 방송 스케쥴링 기법의 주된 동기는 그들은 확장성이 매우 좋으며 아주 적당한 수준의 대역폭을 요구한다는 것이다. 본 논문은 FDBS로 명명된 새로운 동적 방송 스케쥴링 기법을 제안하고 이 기법의 정확성을 증명한다. 그리고 시뮬레이션 방법을 통하여 FDBS의 성능을 평가한다. 성능 평가 결과에 의하면 FDBS는 매우 적당한 수준의 대역폭을 요구하면서도 UD, CBHD, 그리고 NPB 기법보다 평균 응답시간에 있어서 더 우수한 성능을 보인다.
키워드 : 주문형 비디오, 비디오 방송 기법, 비디오 스케쥴링 기법

Abstract In video-on-demand(VOD) systems, a broadcast-based scheduling mechanism is known to be a very efficient technique for disseminating popular videos to very large client populations. The main motivations of broadcasting scheduling mechanisms are that they scale up extremely well and they have very modest bandwidth requirements. This paper proposes a new dynamic broadcasting scheduling mechanism, named FDBS(fast dynamic broadcasting scheme), and proves its correctness. This paper also evaluates the performance of FDBS on the basis of a simulation approach. The simulation results indicate that FDBS shows a superior performance over UD, CBHD, and NPB in terms of the average response time with very reasonable bandwidth requirements.
Key words : video-on-demand, video broadcasting protocol, video scheduling mechanism

1. 서 론

컴퓨터 및 통신 기술, 그리고 비디오 압축 기술의 비약적인 발전으로 인하여 주문형 비디오(video-on-demand: VOD) 시스템이 큰 관심을 끌고 있으며, 네트워크를 통한 실시간 비디오 서비스가 어느 정도 가능하게 되었다. VOD 시스템에서 비디오 서버에는 많은 비디오들이 저장되어 있고, 사용자는 자신이 원하는 비디오를 서버에 명시적으로 요청하여 수신하거나, 또는 서버가 일방적으로 방송하는 비디오를 수신하여 시청하게 된다. VOD 환경에서 사용자는 시간과 장소에 구애받지 않고 자신이 원하는 비디오를 시청할 수 있다는 장점이 있다. 그러나 비디오 데이타는 비록 압축을 하더라도 그 데이타의 양이 매우 크며, 사용자의 연속적인 비디오 시청을 위하여 데이타가 시간에 맞게 제공되어야 하는 특징을 가지고 있다. 따라서 고품질 및 저비용의 VOD 서비스를 실현하기 위해서는 매우 높은 통신 대역폭을 필요로 한다.

VOD 시스템의 가장 큰 제약은 비디오 서버의 통신 대역폭 한계로 인하여 서버가 동시에 전송할 수 있는 비디오의 수가 제한된다는 것이다[1]. VOD 환경에서는 이 한계를 보통 논리적 채널의 수로 표현하는데, 각 논리적 채널은 비디오 데이타의 소비율(consumption rate)과 같은 속도로 데이타를 전송한다. 만일 비디오 서버가 가용 가능한 비디오 채널을 모두 사용하고 있다면, 새로운 사용자의 요구는 즉시 만족될 수 없다. 비디오는 그 재생시간이 매우 길기 때문에 이 경우에 사용자의 대기시간은 예측하기가 쉽지가 않을 뿐만 아니라 매우 길어질 수도 있다. 따라서 한정된 비디오 채널들을 다수의 사용자들이 효율적으로 공유하기 위한 방법에 관한 많은 연구들이 수행되어 통신비용 측면에서 경제적이면서도 적은 서비스 지연시간을 갖는 많은 비디오 스케쥴링 기법들이 제안되었다[1-16].

이 기법들은 비디오 채널을 공유하기 위한 방식에 따라 일괄처리(batching)[2,3], 패칭(patching)[4] 및 피기

† 정 회 원 : 세명대학교 소프트웨어학과 교수
hmkwon@semyung.ac.kr
논문접수 : 2004년 11월 10일
심사완료 : 2005년 7월 16일

백(piggyback)[5,6], 그리고 방송(broadcast) 기법으로 [7-16] 분류할 수 있다. 일괄처리 기법은 어느 정도의 시간동안 비디오 요청을 모아 두었다가 동일 비디오에 대한 다수의 요청들을 하나의 멀티캐스트(multicast) 스트림으로 서비스한다. 그러나 일괄처리 기법에서 각 채널은 비디오 스트림 전체를 처음부터 끝까지 다 전송해야 하므로 서버가 동시에 전송할 수 있는 비디오의 수가 그다지 크지 않다는 단점이 있다.

피기백 기법과 패칭 기법은 어떤 조건이 만족되면 비디오 스트림 전체를 전송하는 것이 아니라 그 일부분만을 전송한다. 만일 동일 비디오가 두 개의 채널을 통하여 전송되고 있다면, 앞선 스트림은 낮은 속도로 전송하여 낮은 속도로 재생시키고 뒤진 스트림은 빠른 속도로 전송하여 빠르게 재생시킨다면, 시간이 경과하게 되면 두 스트림은 같은 프레임을 전송하게 될 것이다. 이 시점에서 두 스트림은 하나로 병합되어 하나의 채널로 전송될 수 있을 것이다. 피기백 기법은 이와 같은 방법으로 여러 스트림들을 하나의 스트림으로 병합하여 채널의 공유도를 향상시킨다. 그러나 피기백 기법에서 스트림간의 재생율의 차이를 사용자가 인지하지 못할 정도로만 조정해야 하므로 병합될 수 있는 스트림의 수가 극히 제한될 뿐만 아니라 뒤진 스트림이 앞선 스트림을 따라 잡는데는 많은 시간이 필요하며 구현이 어렵다는 단점이 있다.

패칭 기법은 비디오 스트림을 비디오 전체를 다 포함하는 정규 스트림과 비디오의 일부분만으로 구성되는 패칭 스트림으로 구분한다. 패칭 기법은 이미 전송되고 있는 정규 스트림이 존재할 때, 사용자의 동일한 비디오 요청에 대해서는 가장 최근의 정규 스트림의 시작시간과 현재 사용자 요청에 대한 시작시간과의 차이에 해당되는 앞 부분만의 스트림으로 구성되는 패칭 스트림을 새로운 채널을 할당하여 전송한다. 그리고 그 이후의 나머지 스트림은 정규 스트림을 공유한다. 일괄처리, 피기백, 그리고 패칭 기법은 서로 대안적으로 사용될 수도 있지만, 일괄처리 기법에 적절한 피기백이나 패칭 기법을 접목하여 비디오 채널의 공유도를 높일 수도 있다. 그러나 이 기법들은 비디오 요청율이 어느 이상으로 증가하면 필요로 하는 채널의 수가 급격하게 늘어난다는 단점이 있다.

이 문제에 대처하기 위하여 방송(broadcast) 방식에 기초한 스케쥴링 기법들이 제안되었다[7-15]. 이 기법들은 비디오 스트림을 일련의 다수의 세그먼트(segment)들로 분할하고 이들을 다수의 채널을 통하여 방송하는데, 비디오의 연속적인 시청이 가능하도록 각 세그먼트들의 방송 스케쥴을 구성한다. 이 비디오 방송 스케쥴링 기법들은 서버가 미리 정해진 스케쥴대로 비디오를 주기적으로 방송하는 정적 기법과 사용자의 요청에 응답하여 필요에 따라 방송 스케쥴을 구성하는 동적 기법으로 분류할 수 있다. 정적 방송 기법은 비디오 요청율이 아무리 높더라도 항상 일정한 수의 채널을 필요로 하며 일정한 응답시간의 성능을 보인다는 장점이 있지만, 사용자가 비디오를 시청하지 않더라도 항상 다수의 채널을 통하여 비디오를 방송해야 한다. 따라서 정적 방송 기법은 인기 비디오를 스케쥴링하는데 매우 바람직하지만, 비인기 비디오에 이 기법을 적용하면 심각한 채널의 낭비를 가져오게 된다.

동적 방송 기법은 사용자의 요청에 따라 동적으로 방송 스케쥴을 구성하여 비디오 세그먼트들을 방송함으로써 채널의 낭비를 방지할 수 있다. 따라서 동적 방송 기법은 비디오 요청율이 낮은 환경에서 정적 방송 기법에 비하여 매우 낮은 통신 대역폭을 필요로 한다. 그리고 비디오 요청율이 높아지더라도 어떤 일정수준 이상의 통신 대역폭을 요구하지는 않는다. 그렇지만, 기존에 개발된 동적 기법들은[7-10] 사용자가 비디오 요청하고 이를 시청하기까지의 평균 응답시간의 성능에 있어서는 정적 기법에 비해 동일하거나, 오히려 더 나쁜 성능을 나타낸다. 이는 비디오 요청율이 낮은 환경에서도 마찬가지이다. 본 논문은 이와 같은 단점을 개선하여, 적당한 수준을 대역폭을 요구하면서도 우수한 평균 응답시간의 성능을 보일 수 있는 FDBS(fast dynamic broadcasting scheme)로 명명된 새로운 동적 방송 스케쥴링 기법을 제안한다. 그리고 FDBS의 정확성을 증명하고, 시뮬레이션을 통하여 이 기법의 성능을 평가한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존에 개발된 방송 스케쥴링 기법에 대하여 기술하고, 3장에서는 새로이 제안되는 기법을 기술한다. 그리고 4장에서는 성능평가 모델을 기술하고 성능 결과를 분석한다. 그리고 5장에서 결론을 맺는다.

2. 관련연구: 비디오 방송 스케쥴링 기법

본 논문은 설명의 편의성을 위해 한 비디오를 방송하기 위하여 필요로 하는 평균 채널의 수를 AVG_CH_NO로 표현한다. 그리고 사용자가 비디오를 요청하고 나서 실제 그 서비스가 시작되기까지 걸리는 시간을 응답시간 또는 대기시간으로 표현한다. 비디오 방송 스케쥴링 기법은 정적 기법과 동적 기법으로 분류할 수 있는데, SB(staggered broadcasting)[11], FB(fast broadcasting)[12], PB(pagoda broadcasting)[13], 그리고 NPB(new pagoda broadcasting)[14] 기법은 정적 방송 기법으로 분류되는 대표적인 기법들이다. 이 중에서 가장 간단한 것은 동일 비디오를 일정한 시차를 두고 다수 개의 채널을 통하여 지속적으로 방송하는 SB

기법이다. 이 기법은 비록 사용자의 셀톱 박스가 간단하다는 장점이 있지만 적당한 수준의 응답시간을 시간을 얻기 위해서는 많은 채널을 필요로 한다는 단점이 있다.

SB 기법 이후에 효율적인 정적 방송 기법들이 많이 제안되었는데[12-16], 이들은 각 비디오 스트림을 일련의 다수의 세그먼트(segment)로 분할하고 이들을 다수의 채널을 통하여 방송한다. 이 기법들은 다수의 채널로부터 동시에 비디오 세그먼트를 수신할 수 있는 능력과 수신한 세그먼트를 재생순서에 맞게 재구성하여 재생할 수 있는 능력, 그리고 수신한 데이타를 재생하기까지 자신의 버퍼 또는 저장장치에 저장할 수 있는 진보된 사용자 셀톱 박스를 필요로 한다. 비디오 스트림을 분할하는 방법에는 세그먼트 크기를 일정하지 않게 분할하는 방법과 이를 동일 크기로 분할하는 방법이 있다. 전자의 방법으로 비디오 스트림을 분할하는 기법으로는 Sky-scraper와[15] Pyramid 기법이[16] 있는데, 이들은 동일한 응답시간을 보장하기 위해서는 다음에 논의할 동일 크기로 비디오를 분할하는 기법들보다 더 높은 통신 대역폭을 필요로 한다고 알려져 있다[14].

FB(fast broadcasting)[12], PB(pagoda broadcasting)[13], 그리고 NPB(new pagoda broadcasting)[14] 기법은 동일 크기의 세그먼트로 비디오 스트림을 분할한다. 한 비디오를 위해 k 개의 채널을 할당한다면, FB 기법은 비디오를 2^k-1 개의 세그먼트로 분할하고 이들을 재생 순서대로 S_1에서 S_{2^k-1}로 명기한다. 그리고 각 j번째(1부터 시작) 채널은 $S_{2^{j-1}}$~S_{2^j-1}를 주기적으로 방송한다. 예를 들어 한 비디오를 위해 3 개의 채널을 사용하면 비디오 스트림은 7 개의 세그먼트로 분할되고 다음과 같이 각 세그먼트를 스케쥴링한다.

표 1 FB 기법의 세그먼트 스케쥴링

채널 1	S_1	S_1	S_1	S_1
채널 2	S_2	S_3	S_2	S_3
채널 3	S_4	S_5	S_6	S_7

사용자가 비디오 시청을 위해서는 다음 S_1이 방송될 때까지만 기다리면 된다. 그리고 나면, 비디오 시청을 하면서 나머지 모든 세그먼트들이 제 때에 수신될 수 있도록 스케쥴링되어 있기 때문에 비디오의 연속적인 시청이 가능하다. 이 경우에 비디오의 재생시간이 120분이면 사용자의 최대 대기시간은 17분 정도가 된다.

PB와 NPB 기법은 FB 기법보다 더욱 세심한 스케쥴링 정책을 사용하여 동일한 채널에 더 많은 수의 세그먼트를 할당할 수 있다. 이들은 어떤 특정 알고리즘에 의해 스케쥴링하는 것이 아니라, 휴리스틱(heuristic) 방법을 사용하여 매우 복잡하게 세그먼트를 채널에 할당한다. PB 기법은 한 비디오를 위해 2k 개의 채널을 사용할 경우에는 비디오를 $4(5^{k-1})-1$ 개의 세그먼트로 분할하고, 2k+1 개의 채널을 사용하면 $2(5^k)-1$ 개로 분할한다. NPB는 PB 기법보다도 더 많은 수의 세그먼트를 할당할 수 있다. FB, PB, 그리고 NPB 기법은 동일한 수의 채널에 할당할 수 있는 세그먼트의 수가 다르기 때문에 평균 응답시간의 성능에서 차이가 있다. 이 기법들을 비교하기 위해 한 비디오를 위해서 사용 가능한 채널 수 별 세그먼트 수 및 평균 응답시간의 관계를 정리하면 다음과 같다. 비디오 재생 시간은 120분 기준이다.

표 2 FB, PB, NPB 기법의 비교

기법 / 채널 수	세그먼트 수(개)			평균응답시간(초)		
	FB	PB	NPB	FB	PB	NPB
3	7	9	9	514	400	400
4	15	19	26	240	189	138
5	31	49	66	116	73	54
6	63	99	172	57	36	21
7	127	249	442	28	14	8

이 기법들은 한 비디오를 위해서 7 개의 채널을 사용하면, 사용자는 평균적으로 30초 이내에 해당 비디오를 시청할 수 있다. SB 기법은 이 성능을 내기 위해서는 120 개의 채널을 사용해야 한다는 점과 비교하면, 비디오 채널을 매우 효율적으로 사용함을 알 수 있다. 그렇지만 이 정적 방송 기법들은 사용자가 비디오를 시청하지 않더라도 항상 비디오를 방송해야 하므로 통신 대역폭의 낭비를 유발할 가능성이 있다.

이 단점을 극복하기 위하여 제안된 동적 방송 기법의 대표적인 것으로는 UD(universal distribution)[7], CBHD(channel-based heuristic distribution)[8], DHB(dynamic heuristic broadcasting)[9], DSB(dynamic sky-scraper broadcasting)[10] 기법들이 있다. UD는 FB 기법에 기초하여 다음과 같이 동적으로 세그먼트를 채널에 할당한다. 각 채널의 한 슬롯 구간에는 하나의 세그먼트 전송이 가능하다.

표 3 UD 기법의 세그먼트 스케쥴링

슬롯번호	0	1	2	3	4	5	6	7	8
채널 1	-	S_1	-	-	S_1	S_1	-	-	-
채널 2	-	-	S_2	S_3	-	S_2	S_3	-	-
채널 3	-	-	-	-	S_4	S_5	S_6	S_7	S_4

슬롯 0에서 새로운 비디오 요청이 도착하면, UD 기법은 슬롯 1에서부터 이탤릭체로 표현된 세그먼트들을 차례로 스케쥴링하는데, S_1 이후의 세그먼트들은 공유도

를 높이기 위하여 연속적인 시청에 문제가 없는한 가급적 방송을 지연시킨다. 시간이 경과하여 슬롯 3에서 새로운 요청이 도달하면, 슬롯 4에 S_1을 할당하고 채널 2의 슬롯 5, 6에 각각 S_2와 S_3를 할당하고 나머지 세그먼트들은 이미 스케쥴링된 것들을 공유한다. 그리고 슬롯 4의 새로운 요청을 위해서는 슬롯 5에 S_1을 할당하고 슬롯 8에 S_4만을 할당하면 된다. UD 기법은 사용자의 요청에 따라 필요한 세그먼트들을 방송한다. 따라서 비디오 요청율이 시간당 60 이하를 유지할 경우에는 정적 기법보다 더 우수한 AVG_CH_NO의 성능을 보인다[7]. 그러나 그 요청율이 200 이상이 되면 거의 모든 채널들이 포화 상태가 되어 FB 기법과 거의 동일한 성능을 낸다.

[8]에서 제안된 CBHD 기법은 한 비디오를 위해 k 개의 채널을 사용하면, 비디오를 (2^k-1)m 개의 세그먼트로 분할한다. 이 기법에서 각 채널 i(i=1~k)는 $S_{(2^{i-1}-1)m+1}$~$S_{(2^i-1)m}$ 까지의 세그먼트들의 방송을 담당한다. 비디오 요청이 슬롯 s에서 도달했다면, CBHD 기법은 각 세그먼트 S_j를 다음과 같이 채널 슬롯에 할당한다. S_j를 방송하는 채널의 슬롯 $(s+1)$~$(s+j+m-1)$의 구간에서 이미 스케쥴링된 S_j가 존재할 경우에는 이를 공유하면 된다. 만일 공유 가능한 S_j가 존재하지 않는다면, $(s+1)$~$(s+j+m-1)$의 슬롯들 중에서 어떤 세그먼트도 할당되어 있지 않은 가장 맨 뒤의 빈 슬롯을 찾아 S_j를 할당한다. CBHD 기법에서 사용자는 비디오 요청을 하고 나서 반드시 최소한 $(m-1)$ 슬롯이 지나가기를 기다린 후 시청을 시작해야 연속적인 비디오 시청이 보장된다.

CBHD는 UD 기법뿐만 아니라 정적 기법인 PB와 NPB보다도 대부분의 환경에서 더 우수한 AVG_CH_NO의 성능을 보인다고 알려져 있다[8]. 그러나 이 이점은 대부분 사용자의 평균 응답시간의 성능을 희생시키면서 얻는 대가이다. 또한 CBHD 기법에서는 S_1을 수신하더라도 사용자가 바로 비디오 시청이 가능한 것이 아니기 때문에 비디오 시청 시작시간을 적절하게 판단할 수 있어야 한다. [9]에서 연구된 DHB 기법은 CBHD보다 채널의 공유도가 더 우수하다. 그러나 이 기법은 각 세그먼트가 스케쥴링되는 채널이 고정된 것이 아니기 때문에 스케쥴링이 어렵다는 단점이 있다. 그리고 세그먼트 수 대비 최대로 필요로 하는 채널 수가 고정된 것이 아니기 때문에 CBHD나 UD보다도 순간적으로 필요로 하는 최대 채널 수는 더 크다는 단점이 있다. [10]에서 제안된 DSB 기법은 사용자 셋톱 박스가 동시에 수신해야 하는 스트림의 수가 [7-9]의 연구에서 개발된 다른 동적 기법들보다 작다는 장점이 있다. 그러나 DSB 기법은 다른 동적 기법들보다는 더 높은 통신 대역폭을 필요로 한다고 알려져 있다[7].

3. 새로운 동적 방송 스케쥴링 기법

이 장은 본 논문이 제안하는 새로운 동적 방송 스케쥴링 기법에 대하여 기술한다. 동적 방송 기법은 사용자의 요청에 따라 동적으로 방송 스케쥴을 구성하여 비디오 세그먼트들을 방송함으로써 채널의 낭비를 방지할 수 있다. 그렇지만, 기존에 개발된 동적 기법들은[7-9] 정적 기법에 비해 동일하거나, 오히려 더 나쁜 평균 응답시간의 성능을 보인다. 본 논문은 이와 같은 단점을 개선하여 적당한 수준을 통신 대역폭을 요구하면서도 우수한 평균 응답시간의 성능을 보일 수 있는 FDBS (fast dynamic broadcasting scheme)로 명명된 새로운 동적 방송 스케쥴링 기법을 제안한다. FDBS는 다음과 같은 사항을 기본 전제로 한다.

a) 본 논문은 일정한 비트율(constant-bit-rate: CBR)을 갖는 비디오를 가정한다.
b) 한 비디오를 위해 k(k≥1) 개의 비디오 채널을 할당하는데, 각 채널은 CH_1에서 CH_k로 표현한다. 각 채널의 대역폭은 비디오 데이타의 소비율(consumption rate)과 동일하다.
c) 각 비디오는 동일 크기의 세그먼트로 분할되는데, 세그먼트의 재생시간은 d로 표현한다. 본 논문은 CBR 비디오를 가정하므로 각 세그먼트의 재생시간은 동일하다. 각 세그먼트는 비디오의 재생 순서대로 S_1부터 순차적으로 명기한다.
d) 각 채널은 세그먼트 단위로 스케쥴링하기 위하여 d와 동일한 시간 간격을 갖는 슬롯(slot)으로 분할되는데, 각 슬롯은 시간 순서에 따라 지속적으로 증가하는 일련번호로 표현한다.
e) 사용자의 비디오 요청은 즉시 서버에 도착하며, 사용자는 어떤 세그먼트의 수신이 시작됨과 동시에 이를 재생시킬 수 있다고 가정한다.

FDBS는 하나의 비디오를 위해 k 개의 채널을 사용하면, 비디오 스트림을 $(2^k-1)m$ 개의 세그먼트로 분할한다. 그리고 CH_1은 S_1~S_m까지 m 개의 세그먼트들의 방송을 담당하고, CH_2는 S_{m+1}~S_{3m}까지 $2m$ 개의 세그먼트들을 방송한다. 이를 일반화하여 표현하면 각 CH_i는 $S_{(2^{i-1}-1)m+1}$~$S_{(2^i-1)m}$까지 총 $2^{i-1}m$ 개의 세그먼트들을 방송한다. 여기서 m은 양의 정수로서 성능을 위해서 조정될 수 있는 변수이다. FDBS는 다음의 두 가지 목표가 충족되도록 각 비디오 세그먼트를 슬롯에 스케쥴링한다. 첫째, 비디오 요청율에 관계없이 사용자의 대기시간이 md가 초과하지 않는다는 것을 보장한다. 둘째, 사용자는 S_1의 수신이 시작됨과 동시에 해당 비디오의 시청을 시작하여 그 비디오의 종료시까지 연속적인 시청이 보장된다.

표 4 FDBS의 기본 개념

슬롯번호	3	4	5	6	7	8	9	10	11	12	13	14	15	16
채널 1	-	S_1	S_2	S_3	S_4	-	-	-	-	-	-	-	-	-
채널 2	-	-	-	-	-	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	-

표 5 스케쥴링 이상 현상

슬롯번호	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
채널 1	S_1	S_2	S_3	S_4	S_1	S_2	S_3	-	S_1	S_2	S_3	S_4	-	-	-	-	-	-	-	-
채널 2	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	-	S_5	S_6	S_7	-	-	-	S_8	S_{12}	S_9	S_{10}	S_{11}	-

이 두 가지 목표를 달성하기 위해서는 비디오 요청이 도달하면 m 슬롯 이내에 S_1을 스케쥴링해야 하며, S_1 이후의 세그먼트들은 그들의 재생이 시작되기 전에 이미 다운로드가 완료될 수 있거나, 또는 최소한 재생이 시작되는 시점에서는 수신될 수 있도록 스케쥴링되어야 한다. FDBS는 어떤 비디오 요청을 스케쥴링하기 위해서는 우선 그 요청을 위한 방송 시작 슬롯을 결정하는데, 이 정보를 start_slot에 저장한다. 본 논문은 설명의 간단성을 위하여 방송 시작위치는 슬롯번호가 m의 정수배인 경우로만 한정한다. 이 제약은 추후에 응답시간의 개선을 위하여 해제될 것이다. 본 논문은 설명의 편의를 위해 다음의 두 가지 용어를 정의한다.

용어 1. (req_s): 슬롯 s에서 도달한 비디오 요청을 req_s로 표현한다.

용어 2. ($start\_slot_s$): 슬롯 s에서 비디오 요청이 도달하면, 우선 이 요청을 위한 스케쥴링 시작위치를 결정하는데 이 시작위치를 $start_slot_s$로 표현한다. $start_slot_s$는 req_s를 위해 세그먼트 S_1이 스케쥴링되는 위치 및 req_s의 서비스 시작시간을 의미한다.

FDBS는 다음의 스케쥴링 규칙에 따라 비디오 세그먼트들을 적절한 슬롯에 할당한다. 비디오 방송 서버는 해당 슬롯이 시작되는 시간이 되면 그 슬롯에 할당된 세그먼트가 존재할 경우 이를 방송하기 시작한다.

규칙 1. req_s를 위한 스케쥴링 시작위치 $start_slot_s$는 $(s+m-s\%m)$으로 결정된다. 이 규칙은 $start_slot_s$를 m의 정수배로 한정한다. □

규칙 2. 규칙 1에 의해 어떤 요청의 스케쥴링 시작 슬롯을 결정하고 그 시간부터 연속적인 비디오 시청이 가능하도록 필요한 세그먼트들을 스케쥴링한다. □

규칙 3. req_s를 위해 어떤 세그먼트 S_i를 스케쥴링하려고 할 때, $(s+1)\sim(start_slot_s+j-1)$의 슬롯 구간에 S_j가 이미 할당되어 있으면 S_i를 다시 스케쥴링하지 않고 기존의 세그먼트를 공유한다. □

FDBS의 기본 개념 및 스케쥴링 규칙들에 대한 이해를 위하여 m은 4로, k는 2로 가정하고 스케쥴링 과정을 살펴보자.

이 경우에 한 비디오 스트림은 $S_1\sim S_{12}$까지 총 12 개의 세그먼트로 분할된다. 만일 비디오의 재생시간이 120분이면 한 슬롯 구간은 10분에 해당하는 시간을 의미한다. req_3은 규칙 1에 의해 슬롯 4가 스케쥴링 시작위치로 결정된다. 따라서 슬롯 4에 S_1을 할당하고 $S_2\sim S_4$는 규칙 2에 따라 슬롯 5~7에 스케쥴링한다. $S_5\sim S_{12}$는 CH_2의 슬롯 4~8 구간의 어디서부터 스케쥴링을 시작하더라도 규칙 2를 만족한다. 그런데 FDBS는 세그먼트의 공유도를 높이기 위하여 표 4와 같이 슬롯 8부터 $S_5\sim S_{12}$를 할당한다. 표 4와 같이 스케쥴링된 상태에서 슬롯 5에서 새로운 요청이 도달했다고 가정하자. $start_slot_5$는 8이므로 슬롯 8이 스케쥴링 시작위치로 결정되어 슬롯 8과 9에 각각 S_1과 S_2를 할당한다. 그리고 나머지 세그먼트들은 이미 스케쥴링되어 있는 세그먼트들을 공유한다(규칙 3 참조).

FDBS는 연속적인 비디오 시청에 문제가 없는 한 세그먼트들의 방송을 가급적 지연시키는데, 이는 세그먼트들의 공유도를 높이기 위함이다.[1] 그렇지만 무조건 세그먼트의 방송을 지연시키면 현재는 연속적인 비디오 서비스가 가능할 지라도 추후에는 그 서비스가 불가능해 질 수도 있다. 이에 대한 이해를 돕기 위하여 다음의 예를 살펴보자.

(예 1: 스케쥴링 이상 현상) 현재 슬롯은 2라고 가정한다. 그리고 표 5에서 이태릭체로 표시된 세그먼트들이 이미 스케쥴링되어 있다고 가정하자.

슬롯 2에서 새로운 요청이 도달하면, 스케쥴링 규칙 1~3에 따라 회색 바탕으로 표시된 세그먼트들이 스케쥴링될 것이다. 그리고 시간이 경과해서 슬롯 7에서 새로운 요청이 도달하면, FDBS는 진한 글꼴로 표시된 세그먼트들을 각 슬롯에 할당할 것이다. 지금까지는 비디오의 연속적인 서비스가 가능하다. 표 5와 같이 스케쥴링된 상태에서 슬롯 11에서 새로운 요청이 도달했다고 가정해 보자. 규칙 1에 의해 $start_slot_{11}$은 12로 결정되어 $S_1\sim S_4$를 슬롯 12~15에 차례대로 할당할 것이다. 그리고 CH_2에는 $S_5\sim S_7$를 스케쥴링해야 하는데, 연속적인 시청을 위해서는 이들이 12~18의 슬롯 구간에 할당되

1) 표 4에서 슬롯 4부터 $S_5\sim S_{12}$를 할당했다면, req_5는 S_5와 S_8의 공유가 불가능할 것이다.

어야 한다. 그러나 이 구간에는 빈 슬롯이 2 개밖에 없기 때문에 이들을 제 때에 스케쥴링하는 것이 불가능하다. 즉, 이 경우에 req_{11}을 위해서는 연속적인 비디오 시청이 이루어지도록 세그먼트들을 스케쥴링하는 것이 불가능하다. □

예 2의 문제점은 req_7을 처리할 때 S_8을 슬롯 14에 할당했기 때문에 발생하는 것이다. 만일 S_8을 슬롯 14 대신에 11에 할당했다면 이런 문제는 발생하지 않는다.[2] 이는 각 세그먼트를 스케쥴링할 때, 현재는 연속적인 비디오 시청이 가능하게 스케쥴링될 수 있더라도 미래의 요청을 위해 어떤 제약이 더 필요하다는 것을 의미한다. S_1은 $m*i(i=1\sim\infty)$ 슬롯에만 할당되듯이, 각 세그먼트 S_j가 스케쥴링될 수 있는 슬롯을 $(m*i+(j-1)\%m)$으로 한정한다면 예 1과 같은 문제점은 발생하지 않는다. 따라서 FDBS는 다음과 같은 규칙을 더 필요로 한다.

규칙 4. req_s를 위해 어떤 세그먼트 S_j의 스케쥴링이 필요할 때, S_j가 할당될 수 있는 슬롯후보들을 $q+xm(x$는 정수로서 $0\sim\lfloor (j-1)/m \rfloor)$으로 한정한다. 이 슬롯후보들 중에서 가장 맨 뒤의 빈 슬롯을 찾아 S_j를 할당하는 슬롯으로 선정한다. 여기서 q는 수식 $[start_slot_s+(j-1)\%m]$을 의미한다. □

규칙 4에 따라 스케쥴링하면 규칙 2를 자동적으로 만족하므로(보조정리 1의 증명 참조) 규칙 2는 더 이상 필요하지 않다. FDBS가 스케쥴링 규칙 1, 3, 4를 준수하면 모든 요청에 대하여 FDBS의 설계 목표가 충족되는데, 이를 증명 과정을 통하여 입증한다. FDBS는 다음과 같은 성질을 지닌다.

성질 1. 각 채널 i는 총 $2^{i-1}m$ 개의 세그먼트들의 방송을 담당하기 때문에 $CH_1\sim CH_i$는 총 $(2^i-1)m$ 개의 세그먼트들을 방송한다. □

성질 2. 스케쥴링 규칙 3으로 인하여 현재 슬롯이 s라면, s+1 이후의 슬롯에 동일 세그먼트가 2 개의 슬롯을 차지하지는 않는다. □

보조정리 1: 모든 세그먼트 S_j가 규칙 4에 따라 스케쥴링될 수 있다면, S_j는 S_j의 재생이 시작되기 전에 이미 다운로드가 완료되거나, 또는 최소한 S_j의 재생이 시작되는 시점에서는 수신이 시작되어 연속적인 비디오 시청이 가능하다.

증명. req_s는 $start_slot_s$부터 그 서비스가 시작된다. 따라서 각 S_j를 $start\_slot_s\sim start\_slot_s+j-1$ 슬롯 구간에 할당할 수 있다면 S_j를 제 때에 수신하는 것이 가능하여 연속적인 비디오 시청이 보장된다. 규칙 4에 따르면, 각 S_j는 $q+xm(x=0\sim\lfloor (j-1)/m \rfloor$, $q=start\_slot_s+(j-1)\%m)$ 슬롯들 중에서 맨 뒤의 빈 슬롯에 스케쥴링될 수 있는데, 이 슬롯은 $start\_slot_s\sim start\_slot_s+j-1$ 슬롯 구간에 포함된 것이다. 따라서 각 S_j를 규칙 4에 따라 스케쥴링할 수 있다면, 이들을 제 때에 수신하는 것이 가능하여 연속적인 비디오 시청이 보장된다. □

모든 세그먼트들을 규칙 4에 따라 슬롯에 할당할 수 있다면 연속적인 비디오 시청이 보장된다. 그런데 이것이 가능하기 위해서는 공유 가능한 S_j가 존재하지 않으면, S_j를 스케쥴링하기 위한 빈 슬롯이 존재해야 한다. 지금부터는 이에 대한 증명을 시작한다.

정리 1. 스케쥴링 규칙 1, 3, 4를 준수하면, 한 비디오를 위해 $k(k\geq 1)$ 개의 채널을 사용할 때 사용자의 대기시간이 md를 초과하지 않는다는 것을 보장하면서 연속적인 비디오 시청이 가능하도록 모든 세그먼트들을 스케쥴링하는 것이 항상 가능하다.

증명 1. 우선 사용자의 대기시간이 md를 초과하지 않는다는 사실을 증명한다. 규칙 1에 의해 어떤 임의의 req_s의 스케쥴링 시작위치는 k 값에 관계없이 $(s+m-s\%m)$으로 결정된다. 따라서 사용자의 대기시간은 최대 $(m-s\%m)d$가 된다. 그러므로 어떤 슬롯에서 비디오가 요청되더라도 이는 md 시간 이내에 서비스된다. □

증명 2. 연속적인 시청이 보장되도록 세그먼트들을 스케쥴링하는 것이 가능하다는 사실은 귀납법을 이용하여 증명한다. 현재 슬롯 s에서 새로운 비디오 요청이 도달했다고 가정하자. 우선, 기본 단계로서 k=1인 경우를 증명한다. 이 경우에 FDBS는 비디오 스트림을 $S_1\sim S_m$까지 총 m 개의 세그먼트들로 분할하고 한개의 채널만을 사용한다. $start_slot_s$는 규칙 1에 의해 $(s+m-s\%m)$으로 결정되는데, $start\_slot_s\%m$ 값은 0임에 유의해야 한다. 각 세그먼트 $S_i(i=1\sim m)$가 할당될 수 있는 슬롯은 규칙 4에 따른다면 $start_slot_s+i-1$로 결정된다. 그런데 규칙 1에 의해 스케쥴링 시작 슬롯은 m의 정수배로 한정되므로 $start\_slot_s+1\sim start\_slot_s+m-1$ 구간에는 어떤 다른 요청 req_r을 위한 새로운 $start_slot_r$이 존재할 수 없다. 또한, $S_i(i=1\sim m)$들 중에서는 i%m 값이 동일한 서로 다른 세그먼트들은 존재하지 않는다. 그러므로 어떤 특정 S_i가 할당되어야 하는 $start_slot_s+i-1$의 슬롯에는 다른 세그먼트가 할당되는 것은 불가능하다. 따라서 공유 가능한 $S_i(i=1\sim m)$가 존재하지 않을 경우에는 S_i를 할당하기 위한 슬롯은 비워 있을 수 밖에 없다. 따라서 $S_i(i=1\sim m)$의 스케쥴링이 필요할 경우에는 규칙 4에 따라 이를 스케쥴링하는 것이 항상 가능하

2) req_{11}을 위해 $S_1\sim S_4$를 슬롯 12~15에 할당하고 CH_2의 슬롯 12~14에 $S_5\sim S_7$를 할당하고, 그리고 슬롯 19에 S_8을 할당하면 된다.

다. 규칙 4에 따라 모든 세그먼트들을 스케쥴링하면 보조정리 1에 의해 연속적인 시청이 보장된다.

귀납법 증명의 다음 단계로서 k=i(i≥1)인 경우에 연속적인 시청이 가능하도록 모든 세그먼트들을 스케쥴링 할 수 있다고 가정하자. 마지막 단계로 k=i+1인 경우에 이를 증명한다. 현 시점 s에서 CH_1~CH_i와 CH_{i+1}은 표 6과 같이 회색 바탕으로 표시된 지점까지 스케쥴링되어 있을 수 있는데, 각 구간에 표시된 숫자는 그 구간에 할당되어 있는 세그먼트의 수를 의미한다. 여기서 p는 CH_1~CH_i가 방송하는 세그먼트들의 수 $(2^i-1)m$을 의미한다. 이 표에서 각 슬롯 구간의 시작 슬롯은 *슬롯번호%m* 값이 0이다. 이는 $start_slot_s$는 T 구간의 시작 슬롯과 일치한다는 것을 의미한다.

표 6 k가 i+1인 경우의 스케쥴링 상태

슬롯번호	..s..	T 구간	U 구간	V 구간
CH_1~CH_i	0~m	0~p		
CH_{i+1}	0~m	0~p	0~m	0~p

CH_{i+1}이 방송하는 세그먼트들($S_{(2^i-1)m+1}$~$S_{(2^{i+1}-1)m}$) 중에서 *세그먼트번호%m*의 값이 동일한 세그먼트는 2^i 개이다. 이는 규칙 4로 인하여 2^i 개의 세그먼트들이 같은 슬롯을 차지하려고 경쟁한다는 것을 의미한다. 우선 설명의 이해를 돕기 위하여 특정 세그먼트 $S_{(2^i-1)m+1}$만을 고려하자. 규칙 4에 따르면 $S_{(2^i-1)m+1}$이 할당될 수 있는 슬롯후보는 $start\_slot_s+mx$(x=0~2^i-1)(이 슬롯들은 표 6에서 T 또는 U 구간에 속한 슬롯들임)로서 총 2^i 개이다. 그런데 이 2^i 개의 슬롯에 할당될 수 있는 세그먼트 후보의 수도 2^i 개이다. 따라서 이 2^i 개의 슬롯에 이미 스케쥴링된 $S_{(2^i-1)m+1}$이 존재하지 않는다면, 성질 2로 인하여 빈 슬롯이 존재하게 마련이다. 그러므로 공유 가능한 $S_{(2^i-1)m+1}$이 존재하지 않을 경우에는 이를 규칙 4에 따라 스케쥴링할 수 있다.

이의 일반화를 위해 CH_{i+1}이 방송하는 모든 S_j($j=(2^i-1)m+1$~$(2^{i+1}-1)m$)를 살펴보자. S_j가 할당될 수 있는 슬롯후보의 수는 총 ($\lfloor (j-1)/m \rfloor + 1$) 개인데(규칙 4), 이 수는 j의 값에 따라 2^i~$2^{i+1}-1$ 값을 가진다. 이 2^i~$2^{i+1}-1$ 개의 슬롯들에 할당될 수 있는 세그먼트 후보 수는 2^i 개이므로 S_j가 스케쥴링되어 있지 않다면 성질 2로 인해 빈 슬롯은 존재할 수 밖에 없다. 따라서 공유 가능한 S_j($j=(2^i-1)m+1$~$(2^{i+1}-1)m$)가 존재하지 않는다면 S_j를 규칙 4에 따라 스케쥴링할 수 있다. 이는 보조정리 1에 의해 CH_{i+1}이 방송하는 $S_{(2^i-1)m+1}$~$S_{(2^{i+1}-1)m}$의 연속적인 시청이 보장된다는 것을 의미한다. 귀납법 가설(induction hypothesis)에 의해 CH_1~CH_i가 방송하는 S_1~$S_{(2^i-1)m}$까지는 연속적인 비디오 시청이 가능하도록 스케쥴링될 수 있다. 따라서 i+1 개의 채널들이 방송하는 모든 세그먼트들을(S_1~$S_{(2^{i+1}-1)m}$) 연속적인 시청이 보장되도록 스케쥴링하는 것이 항상 가능하다.

증명 1과 증명 2에 의해 하나의 비디오를 위하여 k 개(k≥1)의 채널을 사용할 때 정리 1은 항상 성립한다. □

본 논문은 어떤 요청의 스케쥴링 시작 슬롯을 규칙 1에 의해 m의 정수배로 한정했는데, 지금부터는 이 제한을 해제한다. 본 논문은 start_slot과 m의 정수배간의 전이(skew), 즉 (*start_slot%m*) 값을 start_skew로 표현하는데 편의상 처음에는 0으로 초기화한다. 한번 설정된 start_skew 값은 비디오 요청율이 높을 경우에는 보통 변경되지 않는다. 만일 현재의 start_skew가 변경되지 않으면 m 슬롯 단위로 건너뛰면서 서비스가 시작되는데, 이 경우에 req_s의 스케쥴링 시작위치는 $(s+m-(s-start_skew)\%m)$으로 설정된다. 어떤 조건이 만족되면 start_skew를 변경시킬 수 있는데, 이는 응답시간의 개선을 위해 예정보다 빠르게 서비스가 시작되도록 스케쥴링한다는 것을 의미한다. start_skew를 변경시키기 위한 조건을 파악하기 위하여 표 7을 살펴보자.

표 7 start_skew의 변경

슬롯번호	...s....	T 구간	.....
CH_1~CH_{i-1}	?	U 영역: $p_i=(2^{i-1}-1)m$	
CH_i	?	V 영역	W 영역

현재 슬롯 s에서 새로운 요청이 도달했다고 가정하자. 슬롯번호에서 s를 포함하고 있는 구간과 T 구간은 m 개의 슬롯 단위로 표시한 것으로, T 구간의 시작은 슬롯번호가 s+y(y=$m-(s-start\_skew)\%m$)라고 가정한다. req_s는 최소한 슬롯 $s+y$부터는 서비스가 시작될 수 있다. 이 경우에는 start_skew 값이 변경되지 않는다. req_s를 위해 $s+y$ 이전에 스케쥴링을 시작하면 start_skew가 변경된다. 즉, $start_slot_s$를 $s+start$(start=1~$y-1$)로 설정하면 start_skew가 변경된다(U 영역의 시작 슬롯은 $s+start$를 표시한 것임). req_s를 위해 $s+start$부터 스케쥴링을 시작하기 위해서는 각 CH_i는 자신보다 하위 채널들이 방송하는 $p_i=(2^{i-1}-1)m$ 개의 세그먼트를 재생하는데 필요한 시간 이후의 슬롯, 즉 $s+start+p_i$부터는 전부 비워 있으면 된다. 그렇다면 CH_i가 방송하는 세그먼트들은 $s+start+p_i$ 이후에(W 영역 해당) 규칙 4에 따라 스케쥴링하는 것이 항상 가능할 것이다. 모든 채널들이 이 조건을 만족하면, $start_slot_s$는 $s+start$로 설정되며 start_skew가 변경된다. 만일 다수의 start 값이 이 조건을 만족하면 가장 작은 start 값을 선택한다.

이와 같이 start_skew가 변경될 경우에는 CH_i는 V 영역(V 영역의 시작 슬롯은 $s+1$임)에 할당된 세그먼트

들을 공유는 하지만 이 영역에 req_s를 위해 다른 세그먼트들을 스케쥴링하지는 않는다. 규칙 4에 따르면 당연히 CH_i가 방송하는 세그먼트들은 W 영역에 할당될 것이다. start_skew가 변경될 경우에도 V 영역에 req_s를 위해 새로운 세그먼트들을 할당하는 것을 허용한다면, start_skew를 변경시키기 위한 조건을 완화시키는 것은 가능하다. 그러나 이 경우에는 이전에 할당된 세그먼트들과 현재 스케쥴링되는 세그먼트들은 start_skew 값이 다르기 때문에 훨씬 더 복잡한 알고리즘을 필요로 한다. 따라서 본 논문은 이를 고려하지는 않는다. start_skew 값의 변경 가능성을 파악하기 위해서 각 CH_i에서 이미 스케쥴링된 맨 뒤의 슬롯번호가 필요하다. FDBS는 이 정보를 LAST_SLOT[i]에 유지 관리하는데, 이 값은 처음에는 0으로 초기화한다.

위에서 설명한 start_skew를 고려하기 위하여 기존에 정의한 스케쥴링 규칙 1은 다음과 같이 변경되어야 한다. 규칙 2와 규칙 4는 그대로 사용하면 된다.

최종규칙 1: 이 규칙에서 start는 $1 \sim m-1-(s-start\_skew)\%m$ 구간의 수를 의미하며, p_i는 각 CH_i에서 자신보다 하위 채널들이 방송하는 세그먼트들의 수 $(2^{i-1}-1)$ m을 의미한다. 임의의 req_s를 위한 스케쥴링 시작위치, $start_slot_s$는 각 CH_i에서 LAST_SLOT[i]가 $s+start+p_i$보다 작다면 $s+start$로 설정된다. 다수의 start 값이 이 조건을 만족하면 가장 작은 start 값을 선택한다. 만일 이 조건을 만족하지 않는 채널이 존재하면 $start_slot_s$는 $s+m-(s-start_skew)\%m$으로 설정된다.

최종규칙 1, 규칙 3, 그리고 규칙 4에 대한 의사 코드가 그림 1에 제시되어 있는데, 이는 비디오당 k 개의 채널을 사용한다고 가정하고 슬롯 s에서 도달한 비디오 요청을 스케쥴링하는 과정을 보이고 있다. 이 스케쥴링 규칙들을 준수하면 사용자의 대기시간이 md를 초과하지 않는다는 것을 보장하면서 연속적인 비디오 시청이 가능하도록 모든 세그먼트들을 스케쥴링할 수 있다는 사실의 증명은 복잡하기는 하지만 start_slot을 m의 정수배로 한정한 경우를 확장하면 되므로 이를 생략한다.

4. 성능 평가 모델 및 성능 결과

이 장은 FDBS 기법의 성능 파악을 위하여 성능평가 모델을 제시하고 그 성능결과를 제시하고 분석한다. 성

```
가정: 1) 비디오당 k 개의 채널을 할당(한 비디오는 (2^k-1)m 개의 세그먼트로 분할).
      2) 채널 i는 S_(2^(i-1)-1)m+1 ~ S_(2^i-1)m 세그먼트들의 방송을 담당.
      3) 새로운 비디오 요청이 슬롯번호 s에서 도착함.
알고리즘:
  skew_change_flag = FALSE;
  for (start=1; start < (m-(s-start_skew)%m); start++) {
     skew_change_flag = TRUE;
     for (i=1; i <= k; i++) {    // i는 채널 번호를 의미
          if ( (s + start + (2^(i-1)-1)*m) <= LAST_SLOT[i] )
          {    skew_change_flag = FALSE;     break;    }
     }
     if (skew_change_flag == TRUE)     break;
  }
  if (skew_change_flag == TRUE) {
     start_slot = s + start;
     start_skew = start_slot % m;
  }
  else    start_slot = s + m - (s - start_skew) % m;
  for (i=1; i <= k; i++) {          // i는 채널 번호를 의미
     for (j=(2^(i-1)-1)*m+1; j<=(2^i-1)*m; j++) {  // j는 세그먼트 번호를 의미
          search for a previously scheduled S_j in slots s+1 to LAST_SLOT[i] of CH_i;
          if (S_j is not found) {
             for (n=start_slot+j-1; n >= start_slot; n=n-m)
                  if (slot n of channel i is empty)   break;
             schedule S_j in slot n of channel i;
             if (n > LAST_SLOT[i])    LAST_SLOT[i] = n;
          }
     }
  }
```

그림 1 FDBS의 의사코드

능 비교 대상으로는 가장 대표적인 동적 방송 기법으로 알려진 UD와 CBHD 기법을 선정하였다. 그리고 정적 방송 기법은 성능이 우수한 것으로 알려진 NPB 기법을 선정하였다. 성능 평가 모델은 MCC에서 개발한 CSIM [17]을 이용하여 구현하였다.

4.1 성능 평가 모델

본 성능평가 모델은 다수의 비디오 채널을 갖는 단일 비디오 서버와 다수의 클라이언트로 구성된다. 본 모델에서 비디오는 재생시간을 120분으로 설정하며, 비디오 서버의 스케쥴링 부담(overhead)은 고려하지 않는다. 그리고 클라이언트가 보내는 비디오 요청 정보는 즉시 서버에 도달한다고 가정한다. 본 성능 평가에서 중요한 성능 지수는 한 비디오를 방송하기 위해서 필요한 평균 채널 수인 AVG_CH_NO와 비디오를 요청하고 나서 이를 시청하기까지 걸리는 평균 응답시간 또는 대기시간이다. 방송 스케쥴링 기법의 특성상 하나의 비디오를 위하여 필요한 최대 채널이 확보되기만 하면 한 비디오를 시청하는데 필요한 AVG_CH_NO 및 평균 대기시간의 성능이 다른 비디오의 존재 여부에 영향을 받지 않는다. 따라서 본 논문은 비디오를 한 개로 고정한다. 본 논문에서 클라이언트들은 이들의 비디오 요청 시간간의 간격이 지수 분포(exponential distribution) 형태를 갖는 단일 요청 스트림으로 모델링되었다. 비디오 요청율은 로그 스케일로 증가되는데 시간당 1~1024 개의 비디오가 요청된다고 설정한다.

본 실험은 복사 방법(replication approach)으로[18] 실시되었는데, 실험 시작시의 초기 편향(initial bias)을 제거하기 위하여 초기 5000개의 비디오 서비스의 결과는 무시하였다. 이 절에서 제시된 결과 값은 5개의 서로 다른 임의 수를 사용하여 실시된 실험 결과의 평균값으로, 각 실험은 백만 개의 비디오 서비스가 이루어질 때까지 실시하였다.

4.2 성능 결과 및 분석

이 절은 각 기법의 성능 결과를 제시하고 이를 분석한다. 비디오 요청율을 변화시키면서 살펴 본 평균 응답시간 및 AVG_CH_NO의 성능 결과가 각각 그림 2와 그림 3에 제시되어 있다. 이 결과에서 k는 하나의 비디오가 사용할 수 있는 최대 채널의 수를 의미하며, m은 FDBS와 CBHD 기법만이 사용하는 값으로 성능 튜닝을 위해 조정될 수 있는 값이다. CBHD를 제외한 다른 모든 기법들은 S_1이 수신됨과 동시에 해당 비디오의 연속적인 시청이 가능하다.

k를 7로 설정하면, UD 기법은 한 비디오 스트림을 127 개의 세그먼트로 분할하고, FDBS와 CBHD 기법은 127*m 개, 즉 508 개로 분할한다. 이 동적 기법들은 이 환경에서 사용자의 대기시간이 57초가 넘지 않는다는 것을 보장한다. 정적 기법인 NPB는 이 성능을 보장하기 위해서는 한 비디오를 127 개의 세그먼트로 분할해야 한다. 그림에서 제시된 NPB 기법의 성능은 이 경우의 성능을 표시한 것이다. NPB 기법은 정적 방송 기법이 지니는 특성으로 인하여 그림에서 보는 것과 같이 비디오 요청율에 관계없이 항상 일정한 평균 응답시간의 성능과 일정한 AVG_CH_NO의 성능을 보인다.

그림 2의 성능 결과를 살펴보면, FDBS는 다른 기법에 비하여 비디오 요청율이 낮은 환경에서 우수한 평균 응답시간의 성능을 보인다. 이는 FDBS는 start_skew가 변경 가능하면 다른 기법보다는 더 이른 시간에 서비스를 시작할 수 있기 때문이다. 그러나 비디오 요청율이 증가하면 start_skew를 변경시킬 수 있는 확률이 작아지게 되어 사용자의 대기시간은 늘어나게 된다. 그렇지만 그 요청율이 아무리 높아지더라도, 각 비디오 요청은 md 시간 이내에 그 서비스가 시작될 수 있다. 따라서 비디오 요청율이 점점 높아지게 되면, FDBS 기법의 평균 응답시간은 점점 길어져 궁극적으로 $\frac{1}{2}$md 정도로 수렴하게 될 것이다. 여기서 d는 한 세그먼트의 재생시간을 의미한다.

FDBS를 제외한 다른 모든 기법들은 비디오 요청율과 관계없이 항상 일정한 평균 응답시간의 성능을 보인다. UD와 NPB 기법에서 CH_1은 S_1만을 방송하며 사용자는 S_1이 수신되기 시작하면 즉시 비디오 시청이 가능하다. 따라서 이 기법들은 그림 2에서 보는 것과 같이 항상 $\frac{1}{2}$d 정도의 평균 응답시간의 성능을 보인다. UD와 NPB 기법의 $\frac{1}{2}$d와 FDBS의 $\frac{1}{2}$md는 그들의 세그먼트 수의 차이를 고려하면 그 값이 동일하다. 따라서 비디오 요청율이 어느 이상으로 증가하게 되면, 이 기법들은 거의 동일한 평균 응답시간의 성능을 보이는 것이다. CBHD 기법은 연속적인 비디오 시청을 위해서는 S_1이 방송되더라도 즉시 비디오를 시청할 수 있는 것이 아니라, 사용자가 비디오를 요청하고 나서 (m-1)~m 슬롯이 지나간 후에 비디오 시청을 시작해야 한다. 그러므로 CBHD 기법은 평균적으로 (m-$\frac{1}{2}$)d 정도의 응답시간의 성능을 보인다. 따라서 그림 2에서 보는 것과 같이 CBHD는 다른 기법에 비해 사용자의 평균 응답시간이 매우 길다는 단점이 있다.

그림 3의 AVG_CH_NO 성능을 살펴보면, 비디오 요청율이 시간당 64 이하로 유지되면 모든 동적 기법들은 NPB보다 훨씬 우수한 성능을 보인다. 동적 방송 기법들은 사용자의 요청이 있을 경우에만 방송 스케쥴을 구성하여 꼭 필요한 비디오 세그먼트만을 방송하기 때문이다. 동적 기법들 중에서 CBHD는 다른 기법에 비해서 세그먼트들의 방송을 더 많이 지연시키기 때문에 훨씬 우수한 AVG_CH_NO의 성능을 보이는데, 이는 그림 2

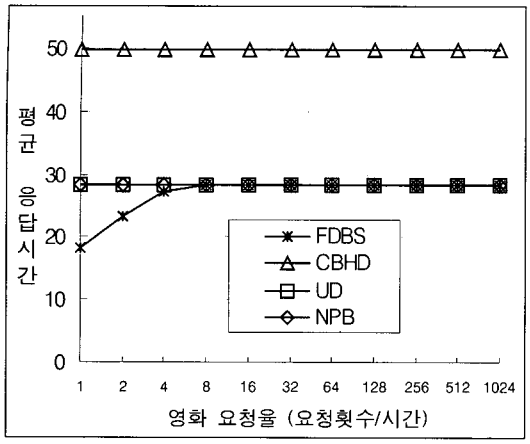

그림 2 평균 응답시간(초) (k=7, m=4)

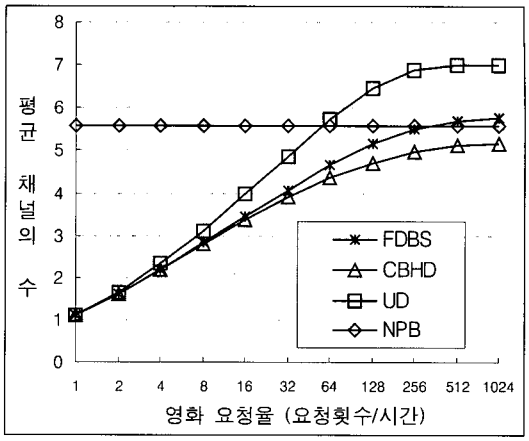

그림 3 평균 채널의 수(개) (k=7, m=4)

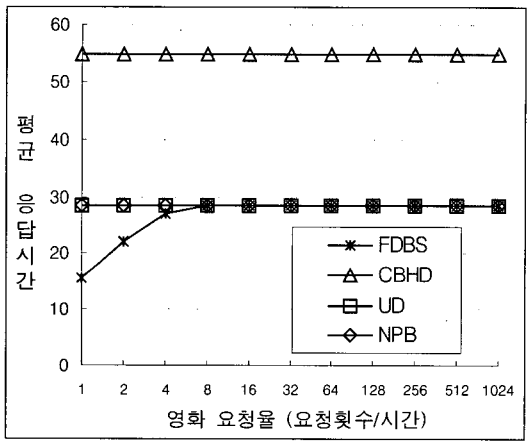

그림 4 평균 응답시간(초) (k=7, m=16)

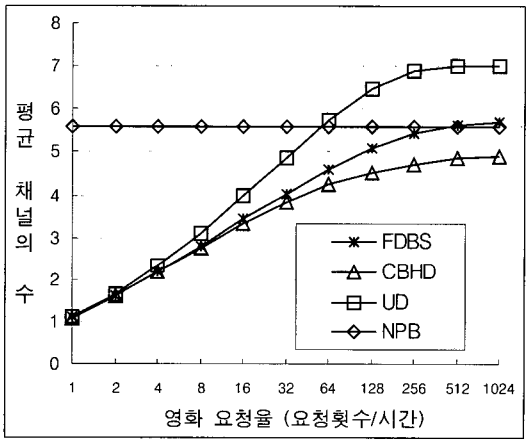

그림 5 평균 채널의 수(개) (k=7, m=16)

에서 살펴보는 것과 같이 대부분 평균 응답시간의 성능을 희생하여 얻는 대가이다. FDBS는 UD와는 다르게 서브 세그먼트의 개념을 사용하여 S_1을 수신하기 이전에 다른 세그먼트를 수신하는 것이 가능하다. 그리고 FDBS는 UD 기법에 비해 평균 대기시간의 성능을 희생시키지 않는 한도에서 각 세그먼트들의 방송을 더 지연시키는 특성을 가지고 있다. 따라서 FDBS는 UD 기법에 비하여 매우 우수한 AVG_CH_NO의 성능을 보일 수 있는데, 비디오 요청율이 64~1024 구간에서 21%~25% 정도의 향상된 성능을 보인다.

m을 16으로 설정하고 실험한 성능 결과가 그림 4와 그림 5에 제시되어 있다. 이 그림에서 성능의 변화 추이는 그림 2와 그림 3의 결과와 거의 유사하다. FDBS는 그림에서 보는 바와 같이 m 값이 커지면 더 빠른 응답시간의 성능을 보일 수 있고, 또한 AVG_CH_NO의 성능도 그 차이가 크지는 않지만 더 우수해 진다. 반면, CBHD 기법은 m 값이 커지면 사용자의 평균 대기시간은 길어지게 된다. 그러나 이에 대한 대가로 AVG_CH_NO의 성능은 더 우수해 진다. 본 논문에서는 m을 64로 설정하고도 실험을 실시했는데 그 성능 결과는 m이 16인 경우와 비교하여 그 차이가 극히 적은 편인데 이를 정리하면 다음과 같다. 첫째 FDBS에서는 AVG_CH_NO의 성능이 거의 차이가 없고, CBHD 기법은 2% 이내의 향상된 성능을 보인다. 둘째, FDBS는 비디오 요청율이 1~8인 경우에 평균 응답시간의 성능이 4% 이내의 정도로 향상되며, CBHD 기법은 모든 요청율에서 2% 정도 저하된 성능을 보인다.

본 논문은 k를 6과 5로 설정하고 실험을 실시했는데, 각 기법들간의 상대적인 성능 결과는 k가 7인 경우와 동일하며, 비디오 요청율에 따른 성능의 변화 추이 또한 k가 7인 경우와 거의 유사하다. 모든 동적 기법들은 k가 1씩 감소할 때마다 사용자의 평균 대기시간은 거의

2배 정도로 길어지게 되는데, 이는 채널이 하나 감소할 때마다 채널에 할당될 수 있는 세그먼트의 수가 거의 반으로 줄기 때문이다. 물론 사용자의 최대 대기시간도 k가 1씩 감소될 때마다 거의 2배로 증가한다.

5. 결 론

VOD 환경에서 비디오 서버의 통신 대역폭을 다수의 사용자들이 효율적으로 공유하기 위한 스케줄링 기법은 사용자에게 고품질 및 저비용의 VOD 서비스를 제공하기 위한 가장 핵심 기술이다. 본 논문은 사용자의 요청에 따라 꼭 필요한 비디오 세그먼트만을 방송하는 동적 방송 스케줄링 기법에 관한 연구를 진행하여 FDBS로 명명된 새로운 기법을 제안했다. FDBS 기법은 방송 방식에 기초하기 때문에 비디오 요청율이 아무리 높아지더라도 사용자의 최대 대기시간을 항상 일정 시간 이내로 유지할 수 있고, 최대로 필요로 하는 대역폭이 일정 수준을 초과하지 않는다는 매우 바람직한 특성을 지니고 있다.

본 논문은 시뮬레이션을 통하여 FDBS의 성능과 UD, CBHD, 그리고 NPB 기법의 성능을 비교하였다. 성능평가 결과에 의하면, FDBS는 적당한 수준의 통신 대역폭을 요구하면서도 매우 바람직한 평균 응답시간의 성능을 보인다. 특히 비디오 요청율이 낮은 환경에서 다른 기법에 비해 매우 우수한 평균 응답시간의 성능을 발휘한다. 그리고 비디오 요청율이 높아지더라도 다른 기법과 동일하거나 또는 더 우수한 응답시간의 성능을 보인다. 이는 FDBS는 가능한 빨리 비디오 요청에 대한 서비스가 시작되면서도 비디오 세그먼트들의 공유도가 높도록 방송 스케줄을 구성하기 때문이다. CBHD는 평균 응답시간을 희생한 대가로 다른 기법보다 우수한 AVG_CH_NO의 성능을 보인다. FDBS와 NPB 기법의 AVG_CH_NO 성능을 비교하면, 비디오 요청율이 낮은 환경에서는 FDBS가 훨씬 우수한 성능을 보인다. 그러나 그 요청율이 극히 높아지면 NPB 기법이 더 우수한 성능을 보이는데 그 차이가 그다지 크지는 않다. FDBS는 UD 기법보다는 항상 우수한 AVG_CH_NO의 성능을 보인다.

본 논문의 미래 연구로서 VBR(variable-bit-rate) 비디오를 위한 방송 기법에 대한 연구를 할 예정이다. 그리고 사용자 셋톱 박스의 캐쉬 메모리를 활용하여 VOD 서비스의 질을 향상시킬 수 있도록 하는 방안에 대한 연구를 진행할 예정이다.

참 고 문 헌

[1] A. Hu, "Video-on-Demand Broadcasting Protocols: A Comprehensive Study," Proc. of INFOCOM 2001, pp.508-517, 2001.

[2] A. Dan, D. Sitaram and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," Proc. of ACM Multimedia 94, pp. 15-23, 1994.

[3] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "The Maximum Factor Queue Length Batching Scheme for Video-on-Demand Systems," IEEE Trans. on Comp. Vol. 50, No. 2, pp. 97-110, 2001.

[4] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A Multicast Technique for True Video-on-Demand Services," Proc. of ACM Multimedia 98, pp. 191-200, 1998.

[5] L. Golubchik, J. C. S. Lui, and R. Muntz, "Reducing I/O Demand in Video-On-Demand Storage Servers," Proc. of ACM SIGMETRICS Conf. on Measurement and Modeling of Comp. Syst. pp. 25-36, 1995.

[6] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "On Optimal Piggyback Merging Policies for Video-On-Demand Systems," Proc. of ACM SIGMETRICS Conf. on Multimedia Systems, pp. 253-258, 1996.

[7] J.-F. Paris, S. W. Carter, and D. D. E. Long, "A Universal Distribution Protocol for Video-on-Demand," Proc. of Int. Conf. on Multimedia and Expo 2000, Vol. 1, pp. 49-52, 2000.

[8] Q. Zhang and J.-F. Paris, "A Channel-Based Heuristic Distribution Protocol for Video-on-Demand," Proc. of 2002 IEEE Int. Conf. on Multimedia and Expo, Vol. 1, pp. 245-248, 2002.

[9] S. R. Carter, J.-F. Paris, S. Mohan, and D. D. E. Long, "A Dynamic Heuristic Broadcasting Protocol for Video-on-Demand," Proc. of Int. Conf. on Distributed Comp. Syst., pp. 657-664, 2001.

[10] D. L. Eager and M. K. Vernon, "Dynamic Skyscraper Broadcasts for Video-on-Demand," Proc. of Int. Workshop on Advances in Multimedia Information Systems, pp. 18-32, 1998.

[11] K. C. Almeroth and M. H. Ammar, "The Use of Multicast Delivery to Provide a Scalable and Interactive Video-On-Demand Service," IEEE Journal on Selected Areas in Communications, Vol. 14, No. 5, pp. 1110-1122, 1996.

[12] L. Juhn and L. Tseng, "Fast Data Broadcasting and Receiving Scheme for Popular Video Service," IEEE Trans. on Broadcasting, Vol. 44, No. 1, pp. 100-105, 1998.

[13] J.-F. Paris, S. W. Carter, and D. D. E. Long, "A Hybrid Broadcasting Protocol for Video on Demand," Proc. of 1999 Multimedia Computing and Networking Conf., pp. 317-326, 1999.

[14] J.-F. Paris, "A Simple Low-Bandwidth Broadcasting Protocol for Video-on-Demand," Proc. of Int. Conf. on Computer Communications and Networks, pp. 118-123, 1999.

[15] K. A. Hua and S. Sheu, "Skyscraper Broad-

casting: A New Broadcasting Scheme for Metropolitan Video-On-Demand Systems," ACM SIGCOMM Comp. Comm. Review, Vol. 27, No. 4, pp. 89-100, 1997.
[16] S. Viswanthan and T. Imielinski, "Metropolitan Area Video-On-Demand service Using Pyramid Broadcasting," Multimedia Systems, Vol. 4, No. 4, pp. 197-208, 1996.
[17] H. Schwetman, 'CSIM Users' Guide for Use with CSIM Revision 16,' Microelectronics and Computer Technology Corporation, 1992.
[18] A. M. Law and W. D. Kelton, 'Simulation Modeling & Analysis,' McGraw-Hill, 1991.

권 혁 민
1984년 서울대학교 제어계측공학과 학사 1994년 한국과학기술원 정보및통신공학과 석사. 1998년 한국과학기술원 정보및통신공학과 박사. 1984년~1991년 대우전자 중앙연구소 컴퓨터개발부 선임연구원. 1999년~현재 세명대학교 소프트웨어학과 조교수. 관심분야는 트랜잭션 처리, 분산/병렬 데이타베이스, 이동 컴퓨팅