주문형 비디오 시스템을 위한 빠른 광범위한 비디오 배포 기법

권 혁 민†

요 약

주문형 비디오 시스템의 성능은 그들이 채택하고 있는 스케쥴링 기법에 의해 크게 영향을 받는다고 알려져 있다. 방송에 기초한 스케쥴링 기법은 방대한 규모의 클라이언트에게 인기 비디오를 배포하기 위한 효과적인 기술로서 큰 관심을 끌고 있다. 방송 스케쥴링 기법의 주된 동기는 그들은 확장성이 매우 좋으며 아주 적당한 수준의 대역폭을 요구한다는 것이다. 본 논문은 이 주제를 연구하여 FUVD로 명명된 새로운 방송 스케쥴링 기법을 제안한다. FUVD 기법은 사용자의 요청에 응답하여 동적으로 비디오 방송 스케쥴을 구성하고 이 스케쥴에 따라 비디오 세그먼트를 방송한다. 본 논문은 시뮬레이션 방법을 통하여 FUVD 기법의 성능을 평가한다. 시뮬레이션 결과에 의하면 FUVD 기법은 UD, CBHD, 그리고 NPB 기법보다 평균 응답시간에 있어서 더 우수한 성능을 보인다.

A Fast Universal Video Distribution Protocol For Video-On-Demand Systems

Hyeok Min Kwon†

ABSTRACT

The performance of video-on-demand(VOD) systems is known to be mainly dependent on a scheduling mechanism which they employ. Broadcast-based scheduling schemes have attracted a lot of attention as an efficient way of distributing 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 studies this issue and proposes a new broadcasting scheduling mechanism, named fast universal video distribution(FUVD). FUVD scheme dynamically constructs a video broadcasting schedule in response to client requests, and broadcasts video segments according to this schedule. This paper also evaluates the performance of FUVD on the basis of a simulation approach. The simulation results indicate that FUVD protocol shows a superior performance over UD, CBHD, and NPB in terms of the average response time.

키워드 : 주문형 비디오(Video-on-demand), 비디오 방송 기법(Video Broadcasting Protocol), 비디오 스케쥴링 기법(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 전송할 수 있어야 한다. 만일 비디오 서버가 이용할 수 있는 비디오 채널을 모두 사용하고 있다면 새로운 사용자의 요구는 즉시 만족될 수 없다. 비디오는 그 재생시간이 매우

† 정 회 원 : 세명대학교 소프트웨어학과 교수
논문접수 : 2004년 8월 20일, 심사완료 : 2004년 10월 1일

길기 때문에 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 기법들이 제안되었다[1-17].

이 기법들은 비디오 채널을 공유하기 위한 방식에 따라 일괄처리(batching)[6-7], 패칭(patching)[5, 17] 및 피기백(piggyback)[8, 9], 그리고 방송(broadcast) 기법으로[2-4, 10-16] 분류할 수 있다. 일괄처리 기법은 어느 정도의 시간동안 비디오 요청을 모아 두었다가 동일 비디오에 대한 다수의 요청들을 하나의 멀티캐스트(multicast) 스트림으로 서비스하는 방식으로 다수의 사용자들이 채널을 동시에 공유하는 것이 가능하다. 그러나 일괄처리 기법에서 각 채널은 비디오 스트림 전체를 처음부터 끝까지 다 전송해야 하므로 서버가 동시에 전송할 수 있는 배치의 수가 많이 제한된다는 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 한 채널로 전송될 수 있을 것이다. 피기백 기법은 이와 같은 방법으로 여러 스트림을 하나의 스트림으로 병합하여 채널의 공유도를 향상시킨다. 그러나 피기백 기법에서 스트림간의 재생율의 차이는 사용자가 인지하지 못할 정도로만 조정해야 하므로 병합될 수 있는 스트림의 수가 극히 제한되며, 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 요청에 대한 시작시간과의 차이에 해당되는 앞 부분만의 스트림으로 구성되는 패칭 스트림을 새로운 채널을 할당하여 전송한다. 그리고 그 이후의 나머지 스트림은 정규 스트림을 공유하는 방법을 통하여 채널의 공유도 높인다. 일괄처리, 피기백, 그리고 패칭 기법은 서로 대안적으로 사용될 수도 있지만, 일괄처리 기법에 적절한 피기백이나 패칭 기법을 접목하여 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 비디오를 모든 사용자에게 주기적으로 방송하는 정적 방송 스케쥴링 기법들이 제안되었다[10-14]. 이 기법들은 비디오 스트림을 일정한 수의 채널을 사용하여 미리 정해진 스케쥴대로 주기적으로 방송한다. 방송 기법은 비디오 요청율이 아무리 높더라도 항상 일정한 수의 채널을 필요로 할뿐만 아니라 일정한 응답시간의 성능을 보이기 때문에 인기가 있는 일부 비디오를 스케쥴링하는데 매우 바람직하다. 그렇지만, 정적 방송 기법은 사용자가 비디오를 시청하지 않더라도 항상 다수의 채널을 통하여 비디오를 방송하므로 비디오 채널의 낭비를 유발할 가능성이 있다.

정적 방송 기법은 인기 비디오를 스케쥴링하는데 매우 적합하지만, 비인기 비디오에 이 기법을 적용하면 심각한 채널의 낭비를 가져오게 된다. 따라서 인기 비디오와 비인기 비디오를 적절하게 구분하여 인기 비디오는 방송 기법으로 스케쥴링하고 비인기 비디오는 요청-응답 형태의 일괄처리나 패칭 기법으로 스케쥴링하면 VOD 시스템은 바람직한 성능 특성을 보일 것이다. 그러나 현실적으로는 아이들이 많이 시청하는 시간대, 어른들이 많이 시청하는 시간대에 따라 인기 비디오의 순서가 바뀌는 경향이 있다. 또한 신작 비디오가 나오고 날짜가 경과함에 따라 그 순위는 계속 변하기 때문에 이를 지속적으로 파악하여 스케쥴링에 반영하기가 쉽지는 않다. 본 논문은 이와 같은 환경을 고려하여 사용자의 요청에 따라 꼭 필요한 비디오 데이타만을 방송하는 동적 방송 스케쥴링 기법을 연구하여 FUVD(fast universal video distribution)로 명명된 새로운 기법을 제안한다. FUVD 기법은 비디오 요청율이 낮을 경우에는 한 비디오를 위해 필요로 하는 평균 채널 수 및 사용자의 평균 대기시간의 성능이 우수하다. 그리고 비디오 요청율이 높아지더라도 필요로 하는 채널 수가 일정 수준을 넘지 않으며 사용자의 대기시간이 어느 이상을 초과하지 않는다는 것을 보장한다.

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

2. 관련 연구 : 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 서버가 일방적으로 정해진 스케쥴대로 비디오를 방송하는 정적 기법과 사용자의 요청에 응답하여 필요에 따라 비디오를 방송하는 동적 기법으로 분류할 수 있다. SB(staggered broad-

casting)[16], FB(fast broadcasting)[14], PB(pagoda broadcasting)[11], 그리고 NPB(new pagoda broadcasting)[12] 기법은 정적 기법으로 분류되는 대표적인 것들이다. 이 중에서 가장 간단한 것은 SB 기법인데, 이 기법은 동일 비디오를 일정한 시차를 두고 다수 개의 채널을 통하여 지속적으로 방송한다. 한 비디오를 위해 k개의 채널을 사용하고 비디오 재생시간이 120분이면, 한 비디오는 매 120/k분마다 방송이 시작될 수 있을 것이다. 따라서 사용자의 최대 대기시간은 120/k분이 된다. SB 기법에서는 사용자의 셀톱 박스가 간단하다는 장점이 있지만 적당한 수준의 응답시간을 얻기 위해서는 많은 채널을 필요로 한다는 단점이 있다.

SB 기법 이후에 효율적인 정적 방송 기법들이 많이 제안되었는데[10-14], 이들은 각 비디오 스트림을 일련의 다수의 세그먼트(segment)로 분할하고 이들을 다수의 채널을 통하여 방송한다. 이들은 비디오의 연속적인 시청이 가능하도록 각 세그먼트의 방송 스케쥴을 구성한다. 이 정적 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 동일 크기로 분할하는 방법이 있다. 전자의 방법을 사용하는 대표적인 기법으로는 Pyramid와[13] Skyscraper 기법이[10] 있는데, 이들은 동일한 응답시간을 보장하기 위해서는 다음에 논의할 동일 크기로 비디오를 분할하는 기법들보다 더 많은 비디오 채널을 필요로 한다[12].

FB, PB, 그리고 NPB 기법은 동일 크기의 세그먼트로 비디오를 분할한다. 한 비디오를 위해 k개의 채널을 할당한다면, FB 기법은 비디오를 2^{k-1}개의 세그먼트로 분할하고 이들을 재생 순서대로 S_1에서 $S_{2^{k-1}}$로 명기한다. 그리고 첫 번째 채널은 S_1만을 지속적으로 방송하고 두 번째 채널은 S_2와 S_3을 방송한다. 이를 일반화하여 표현하면 각 j번째 채널은 $S_{2^{j-1}}$~$S_{2^{j}-1}$를 주기적으로 방송한다. 예를 들어 한 비디오를 위해 3개의 채널을 할당한다면 비디오 스트림은 7개의 세그먼트로 분할되고 다음과 같이 각 세그먼트를 스케쥴링한다.

〈표 1〉 FB 기법의 세그먼트 스케쥴링

1st 채널	S_1	S_1	S_1	S_1
2nd 채널	S_2	S_3	S_2	S_3
3rd 채널	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 NPB는 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 2k개의 채널을 사용하면 비디오를 $4(5^{k-1})-1$개의 세그먼트로 분할하고, 2k+1개의 채널을 사용하면 $2(5^{k})-1$개로 분할한다. PB는 FB 기법보다 동일 채널에 더 많은 수의 세그먼트를 할당하므로 사용자의 평균 대기시간을 줄이는 것이 가능하다. 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〉 정적 방송 기법의 비교

기법 / 채널수	세그먼트의 수(개)			평균 응답시간(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 7개의 채널을 사용하면, 사용자는 평균적으로 30초 이내에 해당 비디오를 시청할 수 있다. SB 기법은 이 성능을 내기 위해서는 120개의 채널을 사용해야 한다는 점과 비교하면, 비디오 채널을 매우 효율적으로 사용함을 알 수 있다. 그렇지만 이 정적 방송 기법들은 사용자가 비디오를 시청하지 않더라도 항상 다수의 채널을 통하여 비디오를 방송하므로 비디오 채널의 낭비를 유발할 가능성이 있다.

이 단점을 극복하기 위하여 사용자의 요청에 응답하여 필요에 따라 방송 스케쥴을 구성하는 동적 방송 기법들이 제안되었는데, 대표적인 기법으로는 UD(universal distribution)[3], CBHD(channel-based heuristic distribution)[2], DHB(dynamic heuristic broadcasting)[4], DSB(dynamic skyscraper broadcasting)[15] 기법들이 있다. UD 기법은 FB에 기초하여 다음과 같이 동적으로 세그먼트를 채널에 할당한다. 각 채널의 한 슬롯(slot) 구간에서는 하나의 세그먼트 전송이 가능하다.

〈표 3〉 UD 기법의 세그먼트 스케쥴링

슬롯 번호	0	1	2	3	4	5	6	7	8
1st 채널	-	S_1	-	-	S_1	S_1	-	-	-
2nd 채널	-	-	S_2	S_3	-	S_2	S_3	-	-
3rd 채널	-	-	-	-	S_4	S_5	S_6	S_7	S_4

슬롯 0에서 새로운 비디오 요청이 도착하면, UD 기법은 슬롯 1부터 이텔릭체로 표현된 세그먼트들을 차례로 스케쥴링하는데, S_1 이후의 세그먼트들은 공유도를 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_3을 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 비디오 요청이 발생할 경우에만 필요한 세그먼트를 방송한다. 따라서 비디오 요청율이 시간당 60 이하를 유지할 경우에는 정적 기법보다 더 우수한 AVG_CH_NO의 성능을 보인다[3]. 그러나 요청율이 200 이상이 되면 거의 모든 채널들이 포화 상태가 되어 FB 기법과 거의 동일한 성능을 낸다.

[2]에서 제안된 CBHD 기법은 한 비디오를 위해 k개의 채널을 할당하면, 비디오를 (2^k-1)m개의 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_j의 방송을 담당하는 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 빈 슬롯을 찾아 S_j를 할당한다. 만일 빈 슬롯이 다수일 경우에는 가장 뒷 슬롯에 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[2]. 그러나 이 이점은 대부분 사용자의 평균 대기시간을 희생시키면서 얻는 대가이다. 또한 CBHD 기법에서는 S_1을 수신하더라도 사용자가 바로 비디오 시청이 가능한 것이 아니기 때문에 비디오 시청 시작 시간을 적절하게 판단할 수 있어야 한다. [4]에서 연구된 DHB는 CBHD 기법보다 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. [15]에서 제안된 DSB 기법은 사용자 셋톱 박스가 동시에 수신해야 하는 스트림의 수가 [2-4]의 연구에서 개발된 다른 동적 기법들보다 작다는 장점이 있다. 그러나 DSB 기법은 다른 동적 기법들보다는 더 높은 통신 대역폭을 필요로 한다고 알려져 있다[3].

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[2-4] 정적 기법에 비해 동일하거나, 오히려 더 나쁜 평균 응답시간의 성능을 보인다. 이는 비디오 요청율이 낮을 경우에도 마찬가지이다. 본 논문은 이와 같은 단점을 개선하여 비디오 요청율이 낮을 경우에는 매우 우수한 평균 대기시간의 성능을 보일 수 있는 FUVD(fast universal video distribution) 기법으로 명명된 새로운 동적 방송 스케쥴링 기법을 제안한다. FUVD 기법은 다음과 같은 사항을 기본 전제로 한다.

① 본 논문은 일정한 비트율(constant-bit-rate : CBR)을 갖는 비디오를 가정한다. 각 비디오는 동일 크기의 세그먼트로 분할하는데, 세그먼트의 재생시간은 d로 표현한다. 본 논문은 CBR 비디오를 가정하므로 각 세그먼트의 재생시간은 동일하다. 각 세그먼트는 비디오의 재생 순서대로 S_1부터 순차적으로 명기한다.

② 한 비디오를 위해 최대 k개의 비디오 채널을 할당하는데, 각 채널은 CH_1에서 CH_k로 표현한다. 각 채널의 대역폭은 비디오 데이타의 소비율(consumption rate)과 동일하다. 각 채널은 세그먼트 단위로 스케쥴링하기 위해 d와 동일한 시간 간격을 갖는 슬롯으로 분할되는데, 각 슬롯은 시간 순서에 따라 지속적으로 증가하는 일련번호로서 표현한다.

③ 사용자의 비디오 요청은 즉시 서버에 도착하며, 사용자는 어떤 세그먼트의 수신이 시작됨과 동시에 이를 재생시킬 수도 있다고 가정한다.

FUVD 기법은 하나의 비디오 스트림을 (2^k-1)m개의 세그먼트로 분할한다. 그리고 각 CH_i는 $S_{(2^{i-1}-1)m+1}$~$S_{(2^i-1)m}$까지 총 2^{i-1}m개의 세그먼트들의 방송을 담당한다. 여기서 m은 양의 정수로서 성능을 위해서 조정될 수 있는 변수이다. FUVD는 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 해

당 비디오의 시청을 시작하여 그 비디오의 종료시까지 연속적인 시청이 보장된다. 이 두 가지 목표를 달성하기 위해서는 어떤 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.

(그림 2)에 제시된 FUVD 기법의 의사 코드는 슬롯 r에서 도달한 비디오 요청을 스케쥴링하는 과정을 보이고 있다. 동일 슬롯 구간에서 다수의 요청이 도달할 경우에는 한번만 스케쥴링하면 된다. FUVD 기법은 어떤 요청을 처리하기 위하여 우선 각 채널의 스케쥴링 시작위치를 결정하는데, 이 정보를 S_SLOT[i]에 저장한다. 각 S_SLOT[i]에는 CH_i가 방송을 담당하는 첫번째 세그먼트, 즉 $S_{(2^{i-1}-1)m+1}$이 스케쥴링되는 슬롯 번호를 저장하는데, 이 값은 어떤 요청을 스케쥴링하는 과정에서 특정 규칙에 따라 변경된다. 각 S_SLOT[i]가 결정되면, 모든 세그먼트는 자기가 속한 채널의 S_ SLOT[i]부터 순차적으로 자신의 위치가 결정된다. 슬롯 r에서 처음으로 비디오가 요청되었다면, 각 S_SLOT[i]는 $r+(2^{i-1}-1)*m+1$로 설정된다. S_SLOT[1]은 해당 요청을 위한 서비스 시작시간을 의미하기도 한다. CH_1을 제외한 다른 채널의 시작위치는 비디오의 연속적인 재생에 문제가 없는한 가급적 지연시키는데, 이는 세그먼트들의 공유도를 높이기 위함이다. 본 논문은 (S_SLOT[i]%m) 값을 start_skew로 표현하는데, 이 값은 모든 채널에서 동일해야 하며 처음에는 0으로 초기화한다. 만일 S_SLOT[i]가 m의 정수배 단위로 증가하면, start_skew는 변경되지 않는다. 어떤 조건들이 만족되면 start_skew가 변경될 수 있는데, 이 때는 모든 채널의 start_skew가 일치하도록 각 채널의 S_SLOT[i]가 재설정되어야 한다. FUVD 기법의 이해를 위해 m을 4로 가정하고 채널 1의 스케쥴링 과정을 살펴 보자.

〈표 4〉 CH1의 세그먼트 스케쥴링

슬롯 번호	0	1	2	3	4	5	6	7	8
채널 1	-	S_1	S_2	S_3	S_4	S_1	S_2	S_3	S_4

슬롯 0에서 비디오가 요청되면, 슬롯 1~4에 S_1~S_4를 스케쥴링한다. 슬롯 1에서 도달한 요청은 슬롯 1의 S_1을 공유할 수 없다. 따라서 FUVD 기법은 슬롯 5에 S_1을 스케쥴링한다. 그리고 사용자 셀톱 박스는 슬롯 2~4에서 방송되는 세그먼트들을 수신하여 자신의 버퍼 영역에 저장하고 슬롯 5에서 S_1이 도착하면 비디오 시청을 시작하게 된다. 슬롯 2에서 도착한 요청을 위해서는 슬롯 6에 S_2만을 할당하면 된다. 이와 마찬가지 방법으로 슬롯 3과 4에서의 요청을 위해서는 각각 슬롯 7과 8에 S_3과 S_4를 할당하면 된다. <표 4>에서 1~4의 슬롯 구간에서는 S_SLOT[1]을 5로 설정함에 의해, 이 구간에서 도달한 요청들을 모두 슬롯 5부터 스케쥴링이 시작되도록 한다. 슬롯 8까지 스케쥴링된 상태에서 5~8의 슬롯 구간에서 새로운 요청이 도달하면, S_SLOT[1]은 9로 변경된다.

이와 같이 S_SLOT[1]이 4의 정수배 단위로 증가되면 start_skew는 변경되지 않는다. 이 경우에는 다른 채널의 스케쥴링 상태를 고려하지 않고 CH_1을 독자적으로 스케쥴링하는 것이 가능하다. 그러나 start_skew를 변경시키기 위해서는 다른 채널의 상태를 고려해야 한다. 이에 대한 이해를 돕기 위해 다음 표를 살펴보자.

〈표 5〉 start_skew의 변경

슬롯번호	0	1	2	3	4	5	6	7	8	9	10	11	12
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}

현재 start_skew는 1이며, 슬롯 0에서 비디오가 요청되어 위와 같이 세그먼트들이 스케쥴링되었다고 가정하자. start_skew가 변경되지 않는다면, 슬롯 1~4에서 도달한 요청을 위한 S_SLOT[1]은 슬롯 5가 되며, 5~8에서 도달한 요청을 위한 시작위치는 9가 된다. 슬롯 5에서 서비스가 시작되는 비디오는 CH_2의 S_5~S_{12}를 공유할 수 있고, 필요하면 슬롯 5~8에 S_1~S_4를 스케쥴링할 수 있다. 그리고 슬롯 9에서 시작되는 비디오는 S_5~S_8의 공유가 불가능할 수도 있지만, 슬롯 13을 CH_2의 새로운 시작위치로 설정하여 S_5~S_8을 제 때에 스케쥴링하는 것이 가능하다. 따라서 start_skew가 변경되지 않는다면, 어떤 경우라도 S_{12}까지는 연속적인 비디오 시청이 가능하다.

슬롯 9에서 새로운 요청이 도달했다고 가정하자. 이 경우에는 슬롯 10을 CH_1의 새로운 시작위치로 설정하고, 14를 CH_2의 시작위치로 설정하여 필요한 세그먼트들을 스케쥴링하면 슬롯 10부터 즉시 서비스가 시작되기 때문에 빠른 응답시간의 성능을 기대할 수 있다.[1] <표 5>와 같이 스케쥴링된 상태에서 슬롯 5에서 새로운 요청이 도착했다고 가정하자. CH_1만을 고려한다면 슬롯 6~9에 S_1~S_4를 스케쥴링하는 것이 가능하지만, 슬롯 10에서 S_5의 재생이 불가능하므로 연속적인 비디오 시청이 이루어지지 않는다. 만일 <표 5>에서 CH_2의 슬롯 9 이후가 비어 있다면,[2] 슬롯 6을 CH_1의 새로운 시작위치로 설정하고 슬롯 10을 CH_2의 시작위치로 설정하면 필요한 세그먼트들을 시간에 맞게 스케쥴링하는 것이 가능하다. 이와 같이 어떤 요청을 예정보다 빠르게 서비스할 수 있는 경우에 한하여 start_skew 값이 변경되는 것이다. start_skew 값의 변경 가능성을 파악하기 위해서는

1) 만일 start_skew를 변경하지 않는다면, 이 서비스는 슬롯 13부터 시작될 것이다.

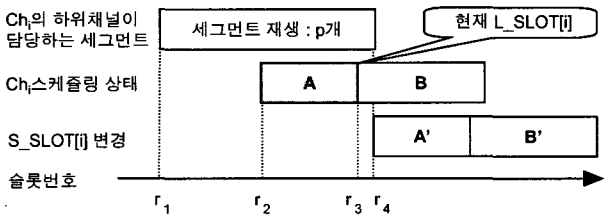
2) 이와 같은 상황은 슬롯 0에서 도착한 요청을 처리할 때, 슬롯 1~4에 이미 S9~S12가 할당되어 있을 때 발생한다. 물론 이들은 슬롯 0 이전에 도착한 요청을 위해서 스케쥴링된 세그먼트이다.

CH_1과 CH_2 뿐만 아니라 다른 모든 채널의 상태도 살펴야 한다. 이를 위해서 각 채널에서 이미 스케쥴링된 맨 뒤의 슬롯 번호가 필요하다. FUVD 기법은 이 정보를 L_SLOT[i]에 유지 관리하는데, 이 값은 처음에는 0으로 초기화한다.

위의 설명을 일반화하기 위하여 CH_i를 살펴보자. CH_i 이전의 채널들은 총 $p=(2^{i-1}-1)m$개의 세그먼트를 방송하고, CH_i는 p+m개의 세그먼트의 방송을 담당한다. 현재 CH_i의 S_SLOT[i]을 s+1이라고 가정하자. 물론 이 값은 슬롯 s-m+1~s-p 구간에서 도달한 요청을 스케쥴링할 때 설정된 값이며, 이 요청은 s-p+1부터 서비스가 시작될 것이다. 그러므로 그 싯점부터 CH_i 이전의 채널들이 담당하는 p개의 세그먼트를 다 재생하고 나면 슬롯 s+1에 도달하게 되어 필요하면 S_{p+1}부터 차례대로 수신하는 것이 가능하여 연속적인 비디오 시청이 가능하다. S_SLOT[i]부터 CH_i가 담당하는 세그먼트들이 <표 6>과 같이 순서대로 스케쥴링되는데, 만일 s+1 이전에 스케쥴링된 세그먼트의 공유가 가능할 경우에는 그 세그먼트를 위한 슬롯은 비워 있게 된다. S_SLOT[i]의 변경 과정을 이해하기 위해 <표 6>을 살펴 보자.

슬롯 s-p+1~s까지 진행되는 동안 S_SLOT[i]가 변경되지 않았고, 슬롯 s+1에서 새로운 요청이 도달했다고 가정하자. 이 요청에 대한 서비스는 늦어도 s+m+1부터는 시작될 수 있다. 우선 start_skew가 변경되는 조건을 살펴 보자. 이 요청의 서비스가 슬롯 s+i(i=2~m)에서 시작된다면 start_skew가 변경될 것이다. 이를 위해서는 s+i+p 이후의 슬롯이 비워 있어 s+i+p를 CH_i의 새로운 스케쥴링 시작위치로 설정할 수 있어야 한다. 이를 일반화하여 표현하면, 어떤 슬롯 r에서 도달한 요청은 늦어도 r+m-(r-start_skew)%m 슬롯부터는 서비스가 시작될 수 있다. 이 경우에는 start_skew가 변경되지 않는다. start_skew가 변경되면서 예정보다 일찍 서비스가 시작될 수 있는 슬롯은 r+start(start=1~r+m-1-(r-start_skew)%m)가 된다. r+start 슬롯부터 서비스가 시작되기 위해서는 자신보다 하위 채널에서 담당하는 p개의 세그먼트를 재생하는데 필요한 시간 이후의 슬롯, 즉 r+start+p부터는 전부 비워 있어야 한다. 모든 채널이 이 조건을 만족하면, 각 채널의 S_SLOT[i]의 값이 (r+start+p)로 설정된다. 그리고 start_skew 값도 다시 설정된다. 다수의 start 값이 이 조건을 만족하면 가장 작은 start를 선택한다.

start_skew를 변경시킬 수 없다면, m 슬롯 단위로 건너뛰면서 서비스가 시작된다. 이 경우에 s+1~s+m에서 도달한 요청을 위한 서비스 시작시간은 s+m+1이 될 것이다. 이 구간에서 도달한 요청은 S_{p+1}~S_{p+m}의 공유가 불가능할 수도 있지만, s+p+m+1을 CH_i의 새로운 시작위치로 설정하는 것이 가능하기 때문에 그 위치부터 적절하게 S_{p+1}~S_{p+m}을 스케쥴링하면 비디오의 연속적인 시청이 가능하다. start_skew가 변경되지 않으면 각 채널은 자신의 S_SLOT[i]를 적절하게 변경시키기만 하면, 다른 채널의 상태를 고려하지 않고 자신이 담당하는 세그먼트를 스케쥴링하는 것이 가능하다. 비디오 요청율이 높을 경우에는 보통 start_skew가 변경되지 않으며, S_SLOT[i]는 일반적으로 p+m 단위로 다시 설정되는데, 세그먼트들의 공유도를 높이기 위하여 다음 그림과 같이 이보다 빠르게 변경될 수도 있다.



(그림 1) 스케쥴링 시작위치 변경

S_SLOT[i]가 r_2인 상태에서 새로운 요청이 도달했는데, 서비스 시작이 r_1로 결정되었다고 하자. 그리고 (그림 1)과 같이 현재 S_SLOT[i]는 r_2라고 가정하고, L_SLOT[i]가 CH_i 이전의 채널들이 담당하는 p개의 세그먼트 재생이 끝나는 r_4 이전이라고 가정하자. 이 경우에 S_SLOT[i]를 변경하지 않고 A, B 부분에 필요한 세그먼트들을 스케쥴링하더라도 비디오의 연속적인 시청에는 문제가 없다. 그렇지만, FUVD 기법은 위와 같은 조건이 성립하면 S_SLOT[i]를 r_4로 변경하고 A, B 부분 대신 A′, B′ 부분에 필요한 세그먼트들을 스케쥴링한다. 연속적인 비디오 시청에 문제가 없는한 세그먼트를 가급적 뒤에 스케쥴링하는 것이 더 많은 요청들이 이들을 공유할 수 있다는 점에 유의해야 한다.

이와 같이 어떤 요청이 도달하면, FUVD 기법은 우선 각 채널의 S_SLOT[i]를 먼저 결정한다. 그리고 나서 각 요청들을 위해 필요한 세그먼트들을 스케쥴링한다. 각 S_SLOT[i]가 결정되면 모든 세그먼트의 위치가 결정되는 것이기 때문에 이를 스케쥴링하는 것은 간단하다. 만일 어떤 세그먼트가 이미 공유 가능하게 스케쥴링되어 있다면 그 세그먼트는 다시 스케쥴링할 필요가 없다. S_SLOT[i]를 쉽게 결정할 수 있도록 채널에 어떤 세그먼트가 새로이 스케쥴링될 때마다 필요하면 L_SLOT[i]가 다시 설정된다.

〈표 6〉 채널 i의 세그먼트 스케쥴링

슬롯 번호	s	s+1	s+2	…	s+m	s+m+1	…	s+p	s+p+1	…	s+p+m
채널 i	?	S_{p+1}	S_{p+2}	…	S_{p+m}	S_{p+m+1}	…	S_{2p}	S_{2p+1}	…	S_{2p+m}
세그먼트 수	?	m개				p개 = $(2^{i-1}-1)m$개					

(그림 2)의 알고리즘에 따라 비디오 요청을 스케쥴링하면, 모든 요청들에 대하여 대기시간이 md를 초과하지 않는다는 것을 보장하면서 연속적인 비디오 시청이 가능하도록 모든 세그먼트들을 스케쥴링할 수 있다. 이에 대한 형식적인 증명이 필요하나 논문의 분량상 이를 생략한다. 본 논문에서는 FUVD 기법을 이용하여 다양한 환경에서 1억 개 이상의 비디오 요청들을 스케쥴링했는데, 모든 요청들에 대하여 위의 사항이 만족되도록 스케쥴링하는 것이 항상 가능하였다는 사실로 증명을 대신한다.

```
가정 : 1) 비디오당 k개의 채널을 할당 : 한 비디오는 (2^k-1)m개의 세그
        먼트로 분할
      2) 새로운 비디오 요청이 슬롯번호 r에서 도착함

알고리즘 :
  skew_change_flag = FALSE ;
  for (start=1 ; start < (m-(r-start_skew)%m) ; start++) {
     skew_change_flag = TRUE ;
     for (i=1 ; i <= k ; i++) {
           if ( (r + start + (2^(i-1)-1)*m) <= L_SLOT[i] )
           {   skew_change_flag = FALSE ;    break ;      }
     }
     if (skew_change_flag == TRUE)     break ;
  }
  if (skew_change_flag == TRUE) {
     for (i=1 ; i<=k ; i++) S_SLOT[i] = r + start + (2^(i-1)-1)*m ;
     start_skew = S_SLOT[1] % m ;
  }
  else {
     for (i=1 ; i<=k ; i++) {
           start_slot_limit = r + 2^(i-1)*m - (r - start_skew) % m
           if (L_SLOT[i] < start_slot_limit)
              S_SLOT[i] = start_slot_limit ;
     }
  }
  for (i=1 ; i <= k ; i++) {
     for (j=(2^(i-1)-1)*m+1 ; j<=(2^i-1)*m ; j++) {
           search for a previously scheduled Sj in slots r+1 to
           L_SLOT[i] of CHi ;
           if (Sj is not found) {
              insert_slot_no = S_SLOT[i] + j - (2^(i-1)-1)*m -1
              schedule Sj in insert_slot_no ;
              if (insert_slot_no > L_SLOT[i])
                 L_SLOT[i] = insert_slot_no ;
           }
     }
  }
```

(그림 2) FUVD 기법의 의사코드

4. 성능 평가 모델 및 성능 결과

이 장은 FUVD 기법의 성능 파악을 위하여 성능평가 모델을 제시하고 그 성능 결과를 제시하고 분석한다. 성능 비교 대상으로는 가장 대표적인 동적 방송 기법으로 알려진 CBHD와 UD 기법을 선정하였고, 정적 방송 기법들 중에서 성능이 우수한 것으로 알려진 NPB 기법을 선정하였다. 성능평가 모델은 MCC에서 개발한 CSIM[18] 언어를 이용하여 구현하였다.

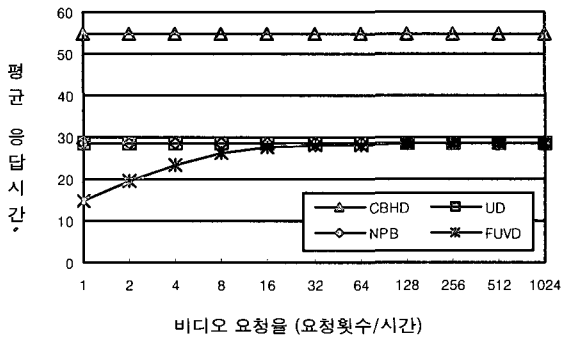
4.1 성능 평가 모델

본 성능평가 모델은 다수의 비디오 채널을 갖는 단일 비디오 서버와 다수의 클라이언트로 구성된다. 본 모델에서 사용하는 비디오는 CBR 형태의 가상 비디오로서 그 재생시간을 120분으로 설정하며, 비디오 서버의 스케쥴링 부담(overhead)은 고려하지 않는다. 그리고 본 성능 평가에서는 다른 연구들과[2-4] 마찬가지로 클라이언트가 보내는 요청 정보는 즉시 서버에 도달한다고 설정한다. 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 않기 때문에 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 1~1024개의 비디오가 요청된다고 설정한다.

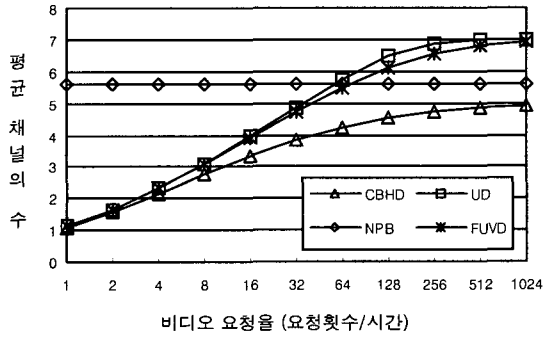
본 실험은 복사 방법(replication approach)으로[19] 실시되었는데, 실험 시작시의 초기 편향(initial bias)을 제거하기 위하여 초기 5000개의 서비스의 결과는 무시하였다. 이 절에서 제시된 결과 값은 5개의 서로 다른 임의 수를 사용하여 실시된 실험 결과의 평균값으로, 각 실험은 백만 개의 비디오 서비스가 이루어질 때까지 실시하였다.

4.2 성능 결과 및 분석

이 절은 각 기법의 성능 결과를 제시하고 분석한다. 비디오 요청율을 변화시키면서 살펴 본 평균 응답시간 및 AVG_CH_NO의 성능 결과가 각각 (그림 3)과 (그림 4)에 제시되어 있다. 이 결과에서 k는 하나의 비디오가 사용할 수 있는 최대 채널의 수를 의미하며, m은 FUVD와 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.



(그림 3) 평균 응답시간(초)(k=7, m=16)



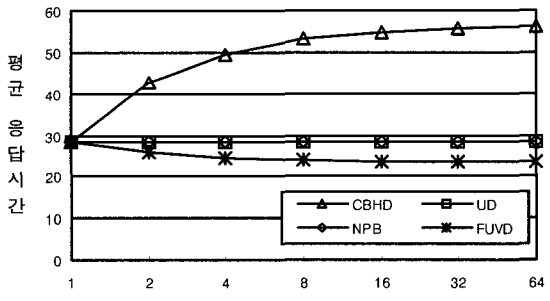
(그림 4) 평균 채널의 수(개)(k=7, m=16)

k를 7로 설정하면, UD 기법은 한 비디오 스트림을 127개의 세그먼트로 분할하고, FUVD와 CBHD 기법은 127*m개, 즉 2032개로 분할한다. 이 동적 기법들은 이 환경에서 사용자의 최대 대기시간이 57초가 넘지 않는다는 것을 보장한다. 정적 기법인 NPB는 이 성능을 보장하기 위해서는 한 비디오를 127개의 세그먼트로 분할해야 한다. 그림에서 제시된 NPB 기법의 성능은 이 경우의 성능을 표시한 것이다. NPB는 정적 방송 기법이 지니는 특성으로 인하여 그림에서 보는 것과 같이 비디오 요청율에 관계없이 항상 일정한 평균 응답시간의 성능과 일정한 AVG_CH_NO의 성능을 보인다.

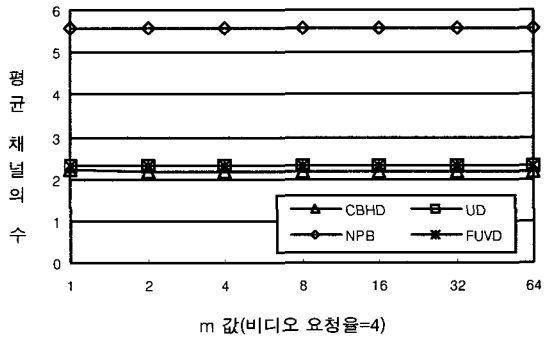
(그림 3)의 성능 결과를 살펴 보면, FUVD는 다른 기법에 비하여 비디오 요청율이 낮을 경우에는 우수한 평균 응답시간의 성능을 보인다. 이는 FUVD 기법은 start_skew가 변경 가능하면 다른 기법보다는 더 이른 시간에 서비스를 시작할 수 있기 때문이다. 그러나 비디오 요청율이 증가하면 start_skew를 변경시킬 수 있는 확률이 작아지게 되어 사용자의 대기시간은 늘어나게 된다. 그렇지만 비디오 요청율이 아무리 높아지더라도, 모든 슬롯에서 start_skew 값을 동일하게 유지하므로 S_1은 최소한 m 슬롯 이내에 방송될 수 있다. 따라서 비디오 요청율이 점점 높아지게 되면, FUVD 기법의 평균 응답시간은 궁극적으로 $\frac{1}{2}$md 정도로 수렴하게 될 것이다. 여기서 d는 한 세그먼트의 재생시간을 의미한다.

FUVD를 제외한 다른 모든 기법들은 비디오 요청율에 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 3)에서 보는 것과 같이 $\frac{1}{2}$d 정도의 평균 응답시간의 성능을 보인다. UD와 NPB 기법의 $\frac{1}{2}$d와 FUVD의 $\frac{1}{2}$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 대기시간의 성능을 보인다. 따라서 (그림 3)에서 보는 것과 같이 CBHD는 다른 기법에 비해 사용자의 평균 대기시간이 매우 길다는 단점이 있다.

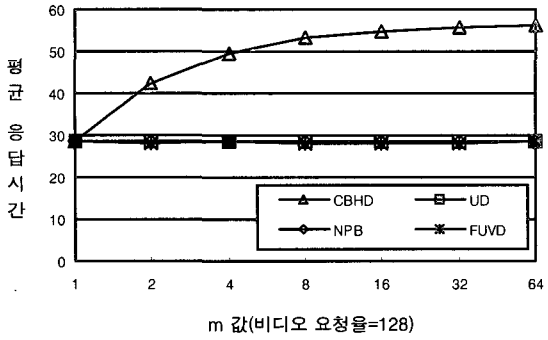
(그림 4)의 AVG_CH_NO 성능을 살펴보면, 비디오 요청율이 시간당 64 이하로 유지되면 모든 동적 기법들은 NPB보다 훨씬 우수한 성능을 보인다. 동적 방송 기법들은 사용자의 요청이 있을 경우에만 방송 스케쥴을 구성하여 꼭 필요한 비디오 세그먼트만을 방송하기 때문이다. 동적 기법들 중에서 CBHD는 다른 기법에 비해서 매우 우수한 성능을 발휘하는데, 이는 (그림 3)에서 살펴보는 것과 같이 대부분 평균 대기시간의 성능을 희생하여 얻는 대가이다. FUVD 기법은 서브 세그먼트의 개념을 사용하여 S_1을 수신하기 이전에 다른 세그먼트를 수신하는 것이 가능하기 때문에 UD 기법보다는 더 우수한 AVG_CH_NO의 성능을 보일 수 있다. 그러나 S_1을 수신하기 이전에 공유 가능한 세그먼트의 수는 그리 많은 것은 아니다. 따라서 그들의 성능 차이가 그다지 크지는 않다. 또한 FUVD 기법은 UD보다 더 빨리 사용자의 요청을 서비스할 수 있다는 사실도 그 차이를 줄이는데 어느 정도 일조를 한다.



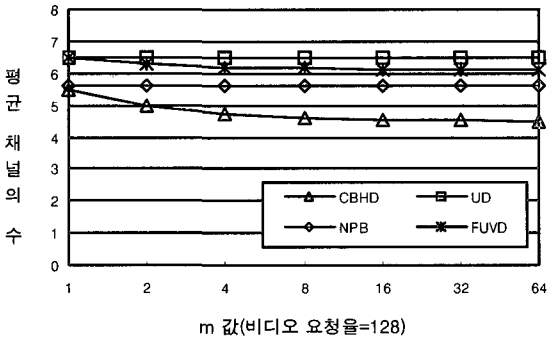
(그림 5) 평균 응답시간(초)



(그림 6) 평균 채널 수(개)



(그림 7) 평균 응답시간(초)



(그림 8) 평균 채널 수(개)

m 값을 변경시키면서 살펴본 성능 결과가 (그림 5)~(그림 8)에 제시되어 있다. 이 실험에서 k 값은 7로 설정하였다. NPB와 UD 기법은 서브 세그먼트 개념을 사용하지 않기 때문에 m 값을 적용할 수 없다. 따라서 이들은 항상 일정한 성능을 보이는데 비교를 위하여 그 성능을 제시하고 있다. 비디오 요청율은 낮은 요청율을 대표하는 값으로 4를 선택했으며, 높은 요청율의 대표 값으로는 128을 선정하였다. 비디오 요청율을 4로 설정한 (그림 5)의 성능 결과를 살펴보면, m 값의 증가에 따라 FUVD 기법에서는 평균 응답시간의 성능이 개선되는데 비하여, CBHD 기법의 응답시간 성능은 급격하게 악화된다. CBHD 기법에서 평균 응답시간의 성능은 비디오 요청율에는 전혀 영향을 받지 않는다. 따라서 CBHD 기법은 (그림 5)와 (그림 7)에서 동일한 성능 추이를 보이는 것이다. 그러나 FUVD 기법에서는 비디오 요청율을 128로 설정하면, 모든 채널들이 거의 포화 상태가 되어 start_skew의 변경 가능성이 매우 작아지게 된다. 따라서, (그림 7)에서 보는 바와 같이 FUVD는 m 값에 관계없이 UD나 NPB 기법과 거의 동일한 응답시간의 성능을 보인다. (그림 6)을 살펴보면, 비디오 요청율이 낮을 경우에는 모든 동적 기법들은 정적 기법에 비하여 훨씬 우수한 AVG_CH_NO의 성능을 보임을 알 수 있다. 비디오 요청율을 4로 고정하고 m 값을 증가시키면, CBHD와 FUVD 기법은 평균 채널의 수가 줄어들기는 하지만 그 차이가 매우 작아 구분이 되지 않는 실정이다. 그러나 비디오 요청율이 높을 경우에는 (그림 8)에서 보는 것과 같이 m 값의 증가에 따라 FUVD와 CBHD 기법에서는 AVG_CH_NO의 성능이 우수해 진다.

m 값을 크게 설정하면 스케쥴링해야 할 세그먼트의 수가 늘어나므로 스케쥴러의 부담이 커진다. FUVD 기법에서 m 값이 8 이상으로 증가하면 응답시간 및 AVG_CH_NO의 성능이 극히 미약하게 개선된다. 따라서 FUVD 기법에서는 m 값을 8 이내로 설정하는 것이 바람직한데, 그 중에서 4 정도로 설정하는 것이 스케쥴링 부담의 증가 대비 성능 개선 효과가 가장 크다고 판단된다. CBHD 기법에서도 m 값이 8 이상이 되면 AVG_CH_NO의 성능이 거의 개선되지 않는다. CBHD 기법에서 AVG_CH_NO의 성능과 평균 응답시간의 성능이 절충(tradeoffs) 관계에 있기 때문에 VOD 서비스 환경을 고려하여 m 값을 8 이내에서 적절하게 설정하는 것이 바람직할 것이다. 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 최대 대기시간도 k가 1씩 감소될 때마다 거의 2배로 증가한다.

5. 결 론

VOD 환경에서 비디오 서버의 통신 대역폭을 다수의 사용자들이 효율적으로 공유하기 위한 스케쥴링 기법은 사용자에게 고품질 및 저비용의 VOD 서비스를 제공하기 위한 가장 핵심적인 기술이다. 본 논문은 사용자의 요청에 따라 꼭 필요한 비디오 세그먼트만을 방송하는 동적 방송 스케쥴링 기법에 관한 연구를 진행하여 FUVD로 명명된 새로운 기법을 제안했다. FUVD 기법은 방송 방식에 기초하기 때문에 비디오 요청율이 아무리 높더라도 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 FUVD 기법의 성능과 UD, CBHD, 그리고 NPB 기법의 성능을 비교하였다. FUVD 기법은 바람직한 평균 응답시간의 성능을 보이는데, 특히 비디오 요청율이 낮을 경우에는 다른 기법에 비해 매우 우수한 평균 응답시간의 성능을 발휘한다. 비디오 요청율이 높아지더라도 다른 기법과 동일하거나 또는 더 우수한 응답시간의 성능을 보인다. 이는 FUVD 기법은 가능한 빨리 비디오 요청에 대한 서비스가 시작될 수 있도록 비디오 세그먼트의 방송 스케쥴을 구성하기 때문이다. CBHD는 평균 응답시간을 희생한 대가로 다른 기법보다 항상 우수한 AVG_

CH_NO의 성능을 보인다. FUVD와 NPB 기법의 AVG_CH_NO 성능을 비교하면, 비디오 요청율이 낮을 경우에는 FUVD 기법이 훨씬 우수한 성능을 보인다. 그러나 그 요청율이 높아지면 NPB 기법이 더 우수한 성능을 보이기 시작하는데 그 차이가 일정 수준을 초과하지는 않는다. FUVD는 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 셀톱 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, Anchorage, AK, pp.508-517, 2001.

[2] 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, Lausanne, Switzerland, 2002.

[3] 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, Vol.1, pp.49-52, New York, NY, 2000.

[4] 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 Computing Systems, pp.657-664, Mesa, AZ, 2001.

[5] 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, Bristol, United Kingdom, 1998.

[6] 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, San Francisco, CA, 1994.

[7] 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.

[8] L. Golubchik, J. C. S. Lui and R. Muntz, "Reducing I/O Demand in Video-On-Demand Storage Servers, Proc. ACM SIGMETRICS Conf. on Measurement and modeling of comp. syst. pp.25-36, Ottawa, Canada, 1995.

[9] 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, Philadelphia, PA, 1996.

[10] K. A. Hua and S. Sheu, "Skyscraper Broadcasting : A New Broadcasting Scheme for Metropolitan Video-On-Demand Systems," ACM SIGCOMM Comp. Comm. Review, Vol.27, No.4, pp.89-100, 1997.

[11] 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, San Jose, CA, 1999.

[12] 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, Boston-Natick, MA, 1999.

[13] S. Viswanthan and T. Imielinski, "Metropolitan Area Video-On-Demand service Using Pyramid Broadcasting," Multimedia Systems, Vol.4, No.4, pp.197-208, 1996.

[14] 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.

[15] 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, Istanbul, Turkey, 1998.

[16] 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, 1996.

[17] 이주영, 하숙정, 배인한, "True VOD 시스템을 위한 채널 예약 패칭 방법의 설계 및 평가", 정보처리학회논문지B, 제9-B권 제6호, Dec., 2002.

[18] H. Schwetman, 'CSIM Users' Guide for Use with CSIM Revision 16', Microelectronics and Computer Technology Corporation, 1992.

[19] A. M. Law and W. D. Kelton, 'Simulation Modeling & Analysis,' McGraw-Hill, 1991.

권 혁 민

e-mail : hmkwon@semyung.ac.kr

1984년 서울대학교 제어계측공학과(학사)

1994년 한국과학기술원 정보및통신공학과 (공학석사)

1998년 한국과학기술원 정보및통신공학과 (공학박사)

1984년~1991년 대우전자 중앙연구소 컴퓨터개발부 선임연구원

1999년~현재 세명대학교 소프트웨어학과 조교수

관심분야 : 트랜잭션 처리, 분산/병렬 데이타베이스, 이동 컴퓨팅