주문형 비디오 시스템을 위한 버퍼링 패칭 기법의 설계 및 평가

(Design and Evaluation of a Buffering Patching Technique for VOD Systems)

하 숙 정 † 배 인 한 ††
(Sook-Jeong Ha) (Ihn-Han Bae)

요 약 VOD 서비스는 멀티미디어의 특성인 높은 대역폭과 긴 재생시간으로 인해 비디오 서버의 네트워크 자원 소비가 많다. 비디오 서버의 네트워크 입출력 대역폭을 절약하기 위해 패칭이 제안되었다. 패칭은 True VOD를 실현하기 위해 멀티캐스트를 사용하여 비디오 스트림을 공유하고, 사용자에게 서비스 지연 없이 즉시 VOD 서비스를 제공한다. 패칭에서 통신 채널은 정규 채널로서 하나의 비디오 전체를 멀티캐스트하기 위해 또는 패칭 채널로서 비디오의 앞부분만을 멀티캐스트하기 위해 사용된다. 본 논문에서는 패칭에서 사용되는 정규 채널의 사용 시간을 단축시키기 위해 부정규 채널과 정규 채널로 구분하는 버퍼링 패칭 기법을 제안한다. 제안하는 기법에서 부정규 채널은 정규 채널과 달리 비디오의 전체 데이타 중에서 버퍼링 윈도우에 해당하는 비디오의 끝 부분을 전송하지 않으며, 이 비디오의 끝 부분은 최근의 정규 채널로부터 다운로드되어 사용자 버퍼에 저장된다. 제안하는 기법의 성능을 패칭과 비교하기 위해 시뮬레이션을 수행하였으며 시뮬레이션 결과 제안한 기법이 사용자 스테이션에서 버퍼링되는 비디오 데이타의 양은 패칭보다 많지만, 이탈율, 평균 서비스 지연시간, 공평성에 있어서 패칭보다 우수함을 보여주었다.

키워드 : VOD 서비스, 패칭, 멀티캐스트, 정규 채널, 부정규 채널, 버퍼링, 버퍼링 윈도우

Abstract Video on Demand(VOD) services cause high resource consumption in a video server, because multimedia are characterized by continuous playback, a high bandwidth requirement, and long playback duration. Patching has been proposed to save the network I/O bandwidth of a video server. To achieve true VOD, patching uses multicasting to share video streams, thereby providing immediate VOD services to users without any service latency. A communication channel is used to either multicast the entire video as a regular channel or multicast only the leading portion of a video as a patching channel. This paper proposes a buffering patching technique that divides regular channels, as used in patching, into sub-regular channels and regular channels to shorten the holding time of the channels. In the proposed technique, the last portion of video data, corresponding to the size of the buffering window, is not transferred by sub-regular channels, but rather downloaded and buffered in the user buffer from the latest regular channel. When simulations were performed to compare the performance of the proposed technique with that of conventional patching, the results indicated that the proposed technique was superior in terms of the defection rate, average service latency, and fairness, although the amount of video data buffered at each user station was higher than that with patching.

Key words : Video-on-Demand service, Patching, Multicast, Regular channel, Sub-regular channel, Buffering, Buffering window

· 본 연구는 2003학년 대구가톨릭대학교 일반 연구비 지원에 의한 것임
† 정 회 원 : 경북대학교 전자전기컴퓨터학부 교수
sjha@m80.knu.ac.kr
†† 비 회 원 : 대구가톨릭대학교 컴퓨터정보통신공학부 교수
ihbae@cuth.cataegu.ac.kr
논문접수 : 2003년 1월 20일
심사완료 : 2003년 7월 31일

1. 서 론

멀티미디어의 대표적인 형태인 주문형 비디오(Video-on-Demand) 서비스는 향후 가장 각광받게 될 서비스 중 하나로 디지털 비디오 도서관, 주문형 뉴스, 원거리 교육, 전자 상거래와 같은 많은 멀티미디어 응용을 위한

중요한 기술이다[1,2]. 전형적인 VOD 서비스는 서버 상에 대량의 비디오들이 저장되어 있으며, 원거리의 사용자가 장소와 시간에 제한 받지 않고 자신이 원하는 비디오를 서버에 요청함으로써 서버로부터 전송되어 재생되는 비디오를 보도록 해준다. 비디오 스트림은 비록 압축이 되었더라도 대역폭 요구량이 높으며, 재생시간이 길고, 엄격한 응답시간을 요구하며, 연속적으로 재생되어야 하므로 전송 비용이 많이 든다. 하나의 비디오 스트림을 유지하기 위해 필요한 시스템 자원을 비디오 채널이라고 하는데 일반적으로 VOD 시스템에서 비디오 서버의 가용 채널 수는 통신 대역폭에 의해 결정되므로 한정되어 있다[2,3]. 비디오 서버의 가용 채널수가 한정되어 있으므로 여러 사용자들의 비동기적 비디오 요청에 대해 대용량의 비디오 스트림들을 경제적이며 적은 서비스 지연시간으로 보다 많은 요청을 서비스할 수 있는 기법이 요구되었으며, 이를 위해 여러 사용자들이 멀티캐스트를 통해 비디오 스트림을 공유하는 방법들이 제안되었다[2-6].

특히 패칭[2,3]은 새롭게 서비스할 사용자를 기존의 멀티캐스트에 합병시킬 수 있다. 즉 기존 멀티캐스트의 패칭 윈도우 안에 존재하는 새로운 사용자 요청에 대해서 서버는 기존 멀티캐스트에서 이미 전송해버린 비디오 앞부분만을 새로운 채널을 통해 전송하며 이와 동시에 사용자가 뒷부분의 비디오 스트림을 연속적으로 상영할 수 있도록 기존의 멀티캐스트에 합병시킨다. 이와 같이 패칭은 많은 사용자들을 기존의 멀티캐스트에 합병시킴으로써 멀티캐스트의 효율을 증가시키며 더욱이 비디오 요청을 일정기간 동안 모아두는 배칭(batching)[4]과는 달리 사용자가 서비스 지연시간 없이 서비스를 받을 수 있으므로 true VOD를 성취할 수 있다.

본 논문에서는 패칭에서 사용되는 정규 채널의 사용시간을 단축시키기 위해 정규 채널을 비디오 전체 데이타를 전송하는 정규 채널(regular channel)과 비디오 전체 데이타 중에서 버퍼링 윈도우 크기에 해당하는 비디오의 마지막 데이타를 제외하고 전송하는 부정규 채널(subregular channel)로 구분하는 버퍼링 패칭 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 true VOD 서비스를 위해 멀티캐스트를 이용하는 패칭을 소개하고, 3장에서는 패칭의 성능을 개선하기 위해 부정규 채널을 사용하여 정규 멀티캐스트 스트림 중에서 버퍼링 윈도우만큼의 마지막 비디오 스트림을 공유하는 버퍼링 패칭을 제안하고, 4장에서는 제안하는 기법을 시뮬레이션을 통하여 평균 서비스 지연시간, 이탈률, 공평성, 사용자 스테이션에서 버퍼링된 데이타의 양에 대해 패칭의 성능과 비교한다. 그리고 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

VOD 서비스의 성능을 개선하기 위해 멀티캐스트를 통해 하나의 비디오 스트림을 여러 사용자가 공유함으로써 비디오 서버의 네트워크 대역폭 요구를 감소시키려는 배칭, 피기백킹(piggybacking)[5,6], 패칭이 제안되었다. 배칭은 비디오 서버에 도착한 사용자 요청을 즉시 서비스하는 것이 아니라 배칭 윈도우라는 짧은 기간 동안 일단 모아 두었다가 동일한 비디오의 요청들에 대해 한꺼번에 비디오 전체 데이타를 멀티캐스트하는 기법이다. 배칭은 서버의 한 채널상으로 멀티캐스트되는 비디오 데이타를 많은 사용자들이 공유함으로써 비디오 서버의 대역폭 요구를 감소시킬 수 있지만 요청이 도착하자마자 서비스되지 않고 일정 기간동안 요청이 모일 때까지 기다리게 되므로 사용자의 서비스 지연시간이 길다. 피기백은 진행 중인 비디오 스트림의 재생율을 조정하여 동일 비디오에 대한 여러 스트림이 하나로 합병되도록 하는 기법이다. 피기백은 배칭보다 서비스 지연시간이 짧지만, 피기백률의 변화가 ±5% 이내여서 합병할 수 있는 스트림의 수가 제한적이며 특별한 하드웨어 환경이 필요하다[6]. 이러한 서비스 지연시간과 멀티캐스트 스트림의 공유간의 상반관계를 해결하기 위해 패칭이 제안되었다.

패칭에서 서버의 통신 대역폭의 대부분은 비디오 하나를 재생율에 맞게 전송할 수 있는 논리적 채널의 집합으로 구성되며, 서버의 나머지 대역폭은 서비스 요청과 서비스 통보 같은 제어 메시지를 전송하기 위해 사용된다. 한 개의 비디오 데이타 전체를 멀티캐스트하기 위해 사용되는 채널은 정규 채널, 비디오의 시작에서부터 일부분만을 멀티캐스트하는 채널은 패칭 채널 역할을 한다고 말한다. 정규 채널과 패칭 채널을 통해 사용자가 수신하는 데이타 스트림들은 각각 정규 스트림과 패칭 스트림이라고 한다. 정규 멀티캐스트가 시작된 이후로 특정 기간 안에 도착한 새로운 사용자 요청에 대해서는 패칭 멀티캐스트가 수행되는데 이 특정 기간을 패칭 윈도우라고 한다[3]. 즉 동일 비디오에 대해 최근의 정규 멀티캐스트가 시작된 후 패칭 윈도우 안에 도착하여 서비스되는 사용자 요청에 대해서는 패칭 멀티캐스트가 수행된다.

사용자가 비디오 서버에게 비디오 i를 요청하면 서버는 정규 채널을 통해 요청된 비디오 i의 전체 데이타를 사용자에게 전송한다. 그리고 이 정규 채널로 비디오 i의 서비스가 시작된 시점인 t_1 이후로부터 패칭 윈도우 크기인 PW만큼의 시간이 지나가기 전인 $t_2 = t_1+PW$

이전에 도착한 동일 비디오 i에 대한 새로운 요청을 서비스할 때는 현재 진행중인 정규 멀티캐스트에 새 요청을 합병시켜 정규 채널상의 비디오 스트림을 공유하도록 한다. 그러나 새로운 사용자가 정규 채널로부터 수신할 수 있는 비디오 스트림은 t_2시점부터 정규 멀티캐스트가 끝날 때까지의 비디오 데이타이다. 그러므로 패칭에선 새로운 사용자의 서비스를 시작하는 시점에서 이미 전송되어 버린 정규 채널상의 비디오 스트림 즉 두 요청간의 스큐에 해당하는 비디오의 시작부터 (t_2-t_1) 만큼의 스트림을 새로운 채널(패칭 채널)로 전송하여 사용자 스테이션에서 즉시 상영되도록 한다. 여기서 스큐란 정규 멀티캐스트의 시작 시간과 새 요청의 서비스 시작 시간과의 시간 차이이다. 이 때 사용자 스테이션에서는 패칭 스트림을 수신하여 상영하면서 패칭 스트림 이후의 비디오 스트림을 연속으로 상영하기 위해 t_2 이후부터 로컬 버퍼에 정규 채널상의 비디오 스트림을 미리 버퍼링한다. 이와 같이 패칭은 새로운 요청에 대해 새로운 채널을 할당받아 비디오 i의 전체 데이타를 전송하는 것이 아니라 스큐에 해당하는 비디오 i의 앞부분만을 전송하고 이미 멀티캐스트되고 있는 정규 스트림을 공유하도록 함으로써 새로운 패칭 채널의 사용 시간을 최소화하고 정규 채널로 서비스할 수 있는 요청의 개수를 증가시켜 사용자 요청 당 필요한 시스템 비용을 감소시키는 것이 주목적이다.

그림 1은 비디오의 길이가 L분, 패칭 윈도우가 (t_5-t_1) 분, 현재 가용 채널이 3개라고 가정할 때 차례대로 t_1, t_2, t_3에 도착한 동일 비디오 i에 대한 3개의 요청 R_1, R_2, R_3를 서비스하기 위해 할당된 채널과 각 채널이 전송하는 데이타를 보여준다. 먼저 t_1에 도착한 요청 R_1은 정규 채널 RCH_1으로 서비스된다. 사용자 요청 R_2가 t_2에 도착했을 때 요청 R_2는 최근에 RCH_1으로 t_1에 시작된 정규 멀티캐스트의 패칭 윈도우 안에 즉 t_1과 t_5 사이에 있으므로 비디오 서버는 패칭 채널 PCH_1으로 스큐에 해당하는 비디오의 시작부분부터 (t_2-t_1)까지의 비디오 스트림을 전송한다. 즉 비디오 서버는 요청 R_2를

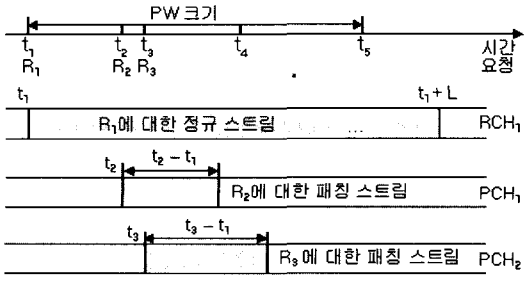


그림 1 패칭에서의 정규 채널과 패칭 채널

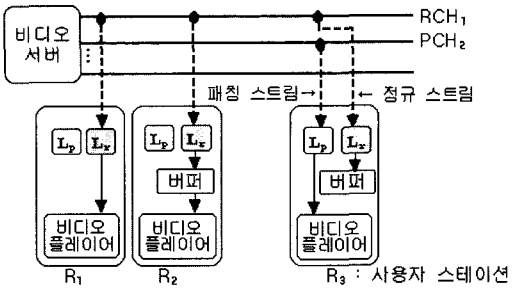


그림 2 그림 1의 t_4시점에서 사용자 스테이션의 상태[2]

위해 비디오 전체 재생 시간 L분 동안 PCH_1을 사용하여 비디오의 전체 데이타를 전송하는 것이 아니라, t_2 이후부터는 정규 채널 RCH_1으로의 멀티캐스트에 요청 R_2의 사용자를 합병시켜 비디오 데이타를 공유하며 이미 정규 채널 RCH_1으로 전송된 t_1부터 t_2까지의 비디오 데이타만 패칭 채널 PCH_1을 사용하여 전송함으로써 채널 PCH_1의 사용 시간을 최소화한다.

그림 2는 그림 1의 t_4 시점에서 각 사용자 스테이션의 상태를 보여준다. 사용자 스테이션의 데이타 로더 L_r과 L_p는 각각 정규 채널과 패칭 채널로 전송되는 정규 스트림과 패칭 스트림을 다운로드하는 책임을 지고 있으며 비디오 플레이어는 버퍼에 저장된 비디오 프레임을 인출하여 재조립한 후 스크린에 상영하는 기능을 제공한다[2,3]. 요청 R_1의 사용자 스테이션에서는 L_r이 정규 채널 RCH_1으로부터 다운로드한 비디오 프레임을 비디오 플레이어가 상영하고 있다. 요청 R_2의 사용자 스테이션은 패칭 채널 PCH_1으로부터 L_p가 다운로드한 비디오 프레임을 모두 상영한 후, 버퍼에 저장해둔 정규 스트림을 상영하면서 계속해서 L_r이 정규 스트림을 다운로드하고 있다. 요청 R_3의 사용자 스테이션에서는 L_p가 패칭 채널 PCH_2로부터 패칭 스트림을 다운로드하며 비디오 플레이어가 이 패칭 스트림을 상영하면서 동시에 정규 채널 RCH_1으로부터 정규 스트림을 버퍼에 저장한다.

이처럼 패칭은 진행중인 멀티캐스트에 새로운 사용자를 합병시킴으로써 채널의 처리력이 배칭보다 높을 뿐만 아니라 사용자의 요청에 대해 즉시 서비스함으로써 대기 시간을 제거할 수 있지만, 사용자 스테이션은 배칭과 달리 정규 스트림과 패칭 스트림을 동시에 받기 위해서 파일 재생율의 두 배의 대역폭이 필요하고 패칭 멀티캐스트가 수행되는 동안 정규 스트림을 버퍼에 저장하기 위해 추가적인 버퍼 공간이 요구된다. 그러나 최근의 사용자 시스템은 충분한 대역폭과 대용량의 디스크 공간을 가지고 있으므로 이러한 요구를 충분히 수용할 수 있다[2].

3. 버퍼링 패칭 기법

패칭 기법에서 패칭 채널은 동일 비디오 요청에 대해 최근의 정규 멀티캐스트 시작 시간과 새 요청 서비스 시작 시간의 차이에 해당하는 비디오 앞부분만을 전송하기 위해 사용된다. 이 때 패칭 멀티캐스트로 서비스되는 사용자 스테이션에서는 패칭 스트림을 수신하면서 동시에 정규 스트림을 수신하여 로컬 버퍼에 저장해둔다. 그러므로 패칭은 사용자 스테이션이 두 채널로부터 동시에 비디오 스트림을 전송 받을 수 있을 만큼의 네트워크 대역폭을 가질 것을 요구한다. 여기서 한가지 관찰할 수 있는 특징은 사용자 스테이션에서 패칭 채널로부터 스큐에 해당하는 패칭 스트림을 수신하는 동안은 정규 채널로도 정규 스트림을 수신해야 하지만 패칭 스트림을 모두 수신한 후에는 정규 채널로만 나머지 정규 스트림을 수신하면 된다는 것이다. 즉 패칭 스트림을 수신하기 위해 필요했던 네트워크 대역폭이 더 이상은 필요하지 않다는 것이다. 그러므로 본 논문에서는 사용자 스테이션에서 가지고 있는 네트워크 대역폭을 충분히 활용하여 하나의 비디오 스트림만을 수신하고 있을 때 동일 비디오 스트림의 마지막 일부분을 미리 수신하여 버퍼에 저장해둠으로써 기존 패칭에서의 일부 정규 채널이 전송해야 하는 비디오 데이타의 양을 감소시켜 가용상태가 되는 시점을 앞당길 수 있는 버퍼링 패칭을 제안한다.

제안하는 기법에서는 정규 멀티캐스트 외에 부정규 멀티캐스트 개념이 추가되며, 정규 멀티캐스트가 아닌 부정규 또는 패칭 멀티캐스트로 서비스되는 모든 사용자 스테이션에서는 가장 최근에 정규 멀티캐스트되고 있는 동일 비디오의 마지막 일부분을 미리 버퍼링한다. 정규 멀티캐스트는 패칭과 같이 특정 비디오의 데이타 전체를 전송하지만 부정규 멀티캐스트는 비디오 전체 데이타 중에서 버퍼링 윈도우만큼의 마지막 부분을 전송하지 않는다. 버퍼링 윈도우란 최근에 진행되고 있는 정규 멀티캐스트 기간 중에서 부정규나 패칭 멀티캐스트로 서비스되는 사용자가 정규 채널로 전송되는 비디오 데이타를 미리 버퍼링하는 기간으로 버퍼링 윈도우 크기가 BW라고 한다면 정규 멀티캐스트가 끝나기 BW분 전부터 끝날 때까지의 정규 스트림이 사용자의 로컬 버퍼에 저장된다. 그러므로 부정규 채널은 비디오의 길이가 L분이라면 비디오의 처음부터 $(L-BW)$분까지의 데이타만 전송하며 비디오의 마지막 BW분의 데이타는 최근의 정규 멀티캐스트를 통해 사용자 스테이션에서 버퍼링된다. 즉 사용자 스테이션에서는 부정규 채널로부터 부정규 스트림을 수신하여 상영하면서 나머지 대역폭을 활용하여 최근의 정규 채널로부터 정규 스트림의 마지막 BW분을 미리 로컬 버퍼에 저장해두었다가 부정규 스트림을 모두 상영한 후 버퍼에 저장해둔 BW분의 비디오 스트림을 상영함으로써 연속적으로 비디오를 상영한다.

버퍼링 패칭 기법에서 패칭 멀티캐스트는 현재 서비스할 사용자 요청이 최근의 정규 또는 부정규 멀티캐스트의 패칭 윈도우 안에 존재할 때 발생하며 두 가지 멀티캐스트 중에서 보다 최근의 멀티캐스트와 패칭이 수행된다. 이 때 서버는 최근의 정규 또는 부정규 멀티캐스트와의 스큐에 해당하는 비디오 시작 부분을 패칭 채널을 통해 전송하며 이와 동시에 사용자 요청을 정규 또는 부정규 멀티캐스트에 합병시켜 비디오 스트림을 공유하도록 한다. 특히 사용자 요청이 부정규 멀티캐스트에 합병된 경우에는 버퍼링 윈도우만큼의 비디오의 마지막 스트림을 최근의 정규 멀티캐스트로부터 미리 버퍼링할 수 있도록 한다. 버퍼링 패칭에서 부정규 멀티캐스트는 현재 서비스할 요청이 최근의 정규 또는 부정규 멀티캐스트의 패칭 윈도우 밖에 있을 때 발생하며 부정규 채널상으로 버퍼링 윈도우만큼의 마지막 부분을 제외한 비디오 데이타가 전송된다. 그리고 버퍼링 윈도우만큼의 비디오 스트림은 패칭 멀티캐스트와 마찬가지로 최근의 정규 멀티캐스트로부터 버퍼링된다.

그림 3과 그림 4는 패칭 윈도우의 크기 PW와 버퍼링 윈도우의 크기 BW는 2분, 비디오의 길이는 10분, 비디오 i에 대한 요청 R_1, R_2, R_3, R_4의 도착시간은 차례대로 0, 1, 4, 5분일 때 각각 패칭과 버퍼링 패칭을 사용하는 경우 비디오 서버에서 할당된 채널과 전송되는 비디오 데이타를 보여준다. 그림 3에서 요청 R_1은 현재 진행중인 정규 멀티캐스트가 없으므로 정규 채널

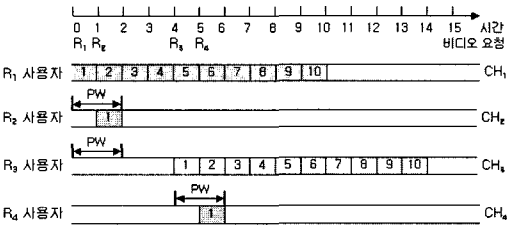


그림 3 패칭 예

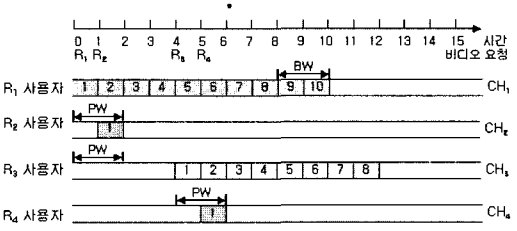


그림 4 버퍼링 패칭 예

CH_1으로 블록 10까지 비디오 i의 전체 데이타가 전송된다. 요청 R_2는 정규 채널 CH_1으로 진행중인 최근의 정규 멀티캐스트의 패칭 윈도우 안에 있으므로 새로운 패칭 채널 CH_2로 스큐 1분에 해당하는 블록 1만을 전송하고 블록 2부터는 정규 채널 CH_1으로부터 공유하도록 한다. 요청 R_3는 요청 R_1을 위한 최근의 정규 멀티캐스트의 패칭 윈도우를 벗어났기 때문에 서버는 새로운 정규 채널 CH_3로 비디오 전체 데이타를 전송하므로 채널 CH_3는 14분에 가용상태가 된다. 요청 R_4는 정규 채널 CH_3로 진행중인 최근의 정규 멀티캐스트의 패칭 윈도우 안에 있으므로 새로운 패칭 채널 CH_4로 스큐 1분에 해당하는 블록 1만을 전송한다.

그러나 버퍼링 패칭을 사용하는 그림 4에서 요청 R_3의 사용자에 대해 서버는 이 요청이 최근의 요청 R_1을 위한 정규 멀티캐스트의 패칭 윈도우는 벗어났지만 이 정규 멀티캐스트가 아직 진행 중이므로 정규가 아닌 부정규 멀티캐스트를 수행한다. 그래서 새로운 부정규 채널 CH_3로 버퍼링 윈도우에 해당하는 블록 9, 10을 제외한 블록 1에서 8까지의 비디오 데이타를 전송하고 사용자 스테이션에서는 블록 9, 10을 8분부터 최근의 정규 채널 CH_1으로부터 버퍼링한 후 상영한다. 요청 R_4에 대해서는 이 요청이 최근의 정규 멀티캐스트의 패칭 윈도우는 벗어났으나 요청 R_3를 위한 최근의 부정규 멀티캐스트의 패칭 윈도우 안에 존재하므로 이 부정규 멀티캐스트와의 패칭 멀티캐스트가 수행된다. 그래서 새로운 패칭 채널 CH_4로 스큐에 해당하는 블록 1을 전송하며 이와 동시에 부정규 채널 CH_3로 전송되고 있는 부정규 스트림을 공유하도록 한다. 그리고 요청 R_4의 사용자 스테이션에서는 8분부터는 정규 채널 CH_1으로부터 버퍼링 윈도우에 해당하는 블록 9, 10을 버퍼링한다. 그림 3에서 볼 수 있듯이 패칭에서 정규 채널 CH_3는 비디오 전체 데이타를 전송하므로 14분에 가용상태가 되지만 그림 4와 같이 버퍼링 패칭을 사용했을 때는 부정규 채널 CH_3가 가용 상태가 되는 시간이 12분으로 버퍼링 윈도우 크기에 해당하는 만큼 더 빨리 다른 요청을 서비스하기 위해 사용될 수 있음을 알 수 있다.

비디오의 길이는 17분, 버퍼링 윈도우의 크기 BW는 2분, 패칭 윈도우의 크기 PW는 3분이라고 가정하자. 서비스할 새로운 요청에 대해 항상 정규 멀티캐스트가 수행되어야 하는 경우는 그림 5에서의 요청 R_1과 같이 현 시점에서 동일 비디오에 대해 진행중인 정규 멀티캐스트가 없을 때와 요청 R_2와 같이 현 시점이 최근 정규 멀티캐스트에서 이미 버퍼링 윈도우에 해당하는 비디오 데이타가 전송된 후일 때이다. 후자의 경우는 사용자가 부정규나 패칭 멀티캐스트로 서비스된다면 최근의 정규 멀티캐스트로부터 버퍼링 윈도우에 해당하는 정규 스트림을 버퍼링해야 하는데 서비스 시작 시점이 이미 버퍼링 윈도우의 데이타가 전송된 후이기 때문에 버퍼링 윈도우에 해당하는 데이타를 모두 버퍼링하지 못하므로 전체 비디오 데이타를 연속적으로 재생할 수가 없기 때문이다.

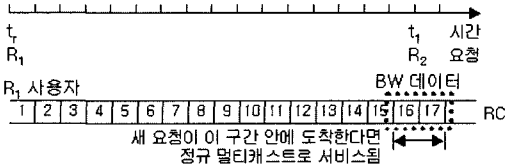


그림 5 새로운 정규 멀티캐스트가 발생하는 경우

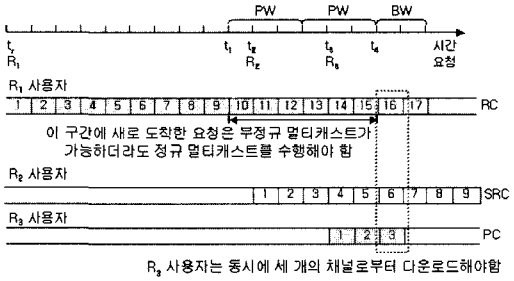


그림 6 부정규 멀티캐스트가 불가능한 경우

그림 6은 비디오 서버가 특정 요청을 부정규 멀티캐스트로 서비스함으로써 나중에 이 부정규 멀티캐스트의 패칭 윈도우 안에 도착한 새로운 요청을 패칭 멀티캐스트로 서비스하더라도 사용자 스테이션은 대역폭 부족으로 서버로부터 멀티캐스트되는 비디오 데이타를 동시에 수신할 수 없는 경우를 보여준다. 요청 R_2는 최근의 요청 R_1을 위한 정규 멀티캐스트의 패칭 윈도우를 벗어났으므로 일단 부정규 멀티캐스트로 서비스한다고 가정하자. 그렇다면 요청 R_3는 요청 R_2를 위한 최근의 부정규 멀티캐스트의 패칭 윈도우 안에 존재하므로 패칭 멀티캐스트로 서비스될 것이며 요청 R_3 사용자 스테이션에서는 t_4부터는 정규 스트림, 부정규 스트림, 패칭 스트림을 동시에 수신해야 하는 상황이 발생한다. 이는 정규 스트림의 버퍼링 윈도우에 해당하는 데이타 전송이 시작되기 전에 요청 R_3를 위한 패칭 스트림의 전송이 완료되지 않기 때문이다. 패칭이나 버퍼링 패칭 모두 사용자 스테이션의 대역폭이 최소 두 개의 채널로부터 동시에 비디오 스트림을 수신할 수 있다는 가정을 하고 있다. 이러한 가정 하에서 세 개의 채널로부터 동시에 스트림을 수신할 수 없는 사용자가 있을 수 있으므로 이와 같은 상황은 발생하지 않아야 한다.

최근의 정규 멀티캐스트의 시작 시간이 t_r, 종료 시간이 t_e, 최근의 부정규 멀티캐스트의 시작시간이 t_{sr}이라

고 하자. 이 부정규 멀티캐스트와 패칭이 가능한 새로운 요청 R의 가장 늦은 도착시간은 $t_{sr}+PW$이며, 이 요청 R에 대해 PW만큼의 패칭 스트림이 전송되어야 하므로 패칭 멀티캐스트의 종료시간은 $t_{sr}+2*PW$이다. 만일 이 시점이 최근의 정규 멀티캐스트에서 버퍼링 윈도우에 해당하는 데이타를 전송하는 시점 t_e-BW보다 더 늦으면 이 요청 R의 사용자 스테이션에서는 정규, 부정규, 패칭 스트림을 동시에 수신해야 하는 상황이 발생할 수 있다. 그러나 t_{sr}이 최근의 정규 멀티캐스트의 종료 시점보다 $(2*PW+BW)$이상 앞선 시점이라면 즉 $t_{sr}<=t_e-(2*PW+BW)$일 때는 이러한 상황이 발생하지 않는다. 그러므로 버퍼링 패칭에서는 새로운 요청이 최근의 부정규 멀티캐스트와의 패칭 멀티캐스트로 서비스됨으로써 세 개의 스트림을 동시에 수신하는 상황이 발생하지 않도록 $t_r+L-(2*PW+BW)$ 시점부터는 새로운 요청에 대해 부정규 멀티캐스트가 가능하더라도 새로운 정규 멀티캐스트를 수행한다.

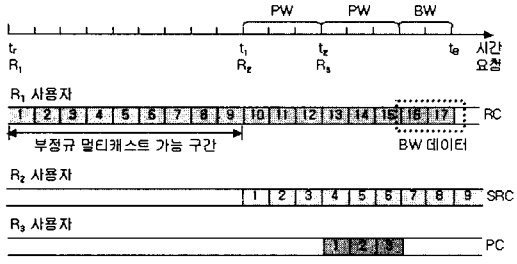


그림 7 부정규 멀티캐스트가 가능한 구간

t : 현재 시각
CH : 요청을 서비스하기 위해 디스패치된 가용 채널
PW : 패칭 윈도우 크기(분)
BW : 버퍼링 윈도우 크기(분)
L : 비디오의 전체 재생 시간(분)
$RM_i(t_r)$: t_r에 시작된 비디오 i에 대한 최근의 정규 멀티캐스트
$SRM_i(t_{sr})$: t_{sr}에 시작된 비디오 i에 대한 최근의 부정규 멀티캐스트

1. if ($t <= (t_r + PW)$) then
   // 최근의 정규 멀티캐스트와의 패칭이 수행
   채널 CH를 패칭 채널로 사용하여 $RM_i(t_r)$과의 스큐 $(t-t_r)$ 만큼의 비디오 시작 데이타를 전송
2. else if ($t <= (t_{sr} + PW)$) then
   // 최근의 부정규 멀티캐스트와의 패칭이 수행
   채널 CH를 패칭 채널로 사용하여 $SRM_i(t_{sr})$과의 스큐 $(t-t_{sr})$ 만큼의 비디오 시작 데이타를 전송
3. else if ($t > ((t_r + L) - (2*PW+BW))$) then
   // 현재 시각이 최근의 정규 멀티캐스트가 종료되기 $(2*PW+BW)$이내이므로 부정규가 아닌 정규 멀티캐스트를 수행
   채널 CH를 정규 채널로 사용하여 비디오 데이타 전체를 전송
4. else
   // 부정규 멀티캐스트가 수행
   채널 CH를 부정규 채널로 사용하여 버퍼링 윈도우에 해당하는 비디오의 마지막 부분을 제외한 비디오 데이타를 전송

그림 8 버퍼링 패칭의 멀티캐스트 선택 알고리즘

그림 7은 t_r에 시작된 최근의 정규 멀티캐스트가 존재할 때 가장 늦게 발생할 수 있는 부정규 멀티캐스트와 이 부정규 멀티캐스트와의 패칭 멀티캐스트가 가능한 가장 늦은 요청을 보여준다. 요청 R_1을 위해 정규 채널 RC로 정규 스트림이 전송되고 있다. t_1에 도착한 요청 R_2는 진행중인 최근의 정규 멀티캐스트의 패칭 윈도우를 벗어났으며, 진행중인 부정규 멀티캐스트도 존재하지 않으며, $t_1<=t_e-(2*PW+BW)$이므로 새로운 부정규 채널 SRC상으로 부정규 멀티캐스트가 수행된다. 시간 t_2에 도착한 요청 R_3에 대해서는 최근의 부정규 멀티캐스트의 패칭 윈도우 끝에 존재하므로 이 부정규 멀티캐스트와의 패칭이 수행된다. 이 패칭 멀티캐스트가 종료된 후 정규 채널 RC로부터는 버퍼링 윈도우에 해당하는 스트림을, 부정규 채널 SRC로부터는 부정규 스트림을 동시에 다운로드할 수 있다. 버퍼링 패칭에서 비디오 i에 대한 사용자 요청 R을 서비스하기 위해 비디오 서버가 정규 멀티캐스트, 부정규 멀티캐스트 그리고 패칭 멀티캐스트 중에서 어떤 멀티캐스트를 수행할지는 그림 8과 같은 알고리즘에 의해 결정된다.

4. 시뮬레이션 및 성능 평가

본 논문에서는 시뮬레이션을 통하여 3장에서 제안한 부정규 멀티캐스트를 사용하는 버퍼링 패칭 기법과 패칭 기법의 성능을 비교하여 평가한다. 성능 평가에서 고려된 평가 항목은 VOD 서버의 성능 평가에 주로 사용되는 비디오 요청 도착율에 따른 사용자의 이탈율, 평균 서비스 지연시간, 공평성, 버퍼링된 비디오 데이타의 양이다. 시뮬레이션은 *IBM PC-Pentium IV*에서 *C++*로 작성한 프로그램을 사용하여 수행되었으며 사용된 파라미터는 표 1과 같다. 비디오 관람 패턴에 대한 통계에 의하면 비디오의 인기도는 특정 비디오에 편중되며 이런 지역성은 *Zipf* 법칙을 사용하여 표현할 수 있기 때문에 비디오 i의 인기도 p_i는 식 1과 같이 *Zipf* 법칙을 따른다고 가정한다[7,8].

$$p_i = \frac{1}{i^z \sum_{j=1}^{N} \frac{1}{j^z}} \quad (1)$$

여기서 N은 전체 비디오의 수이며 z는 스큐 인자(skew factor)로서 VOD 응용에서 일반적으로 나타나

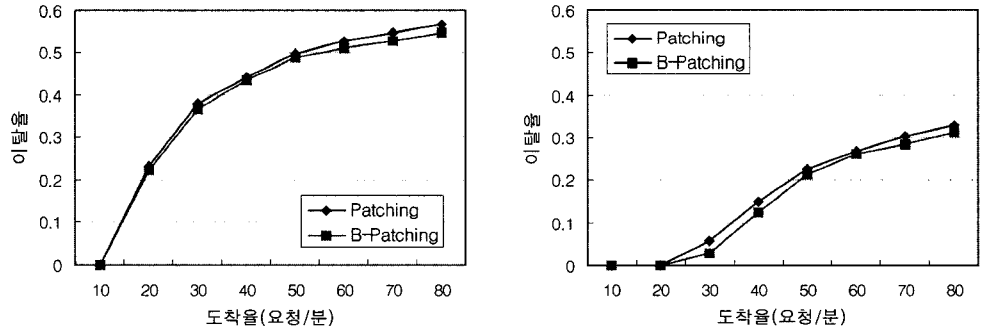
표 1 시뮬레이션 파라미터

파라미터	값
비디오 개수	100개
서버 네트워크 채널	600, 900채널
패칭 윈도우	5분
버퍼링 윈도우	1, 3, 5분
도착율(요청/분)	10~80 (포아송)
비디오 길이	90분
이탈시간	1분~5분 (랜덤)
시뮬레이션 기간	360분

는 0.7을 사용하였다[3]. 비디오의 길이는 *Cai*[3]에서와 마찬가지로 일반적인 영화 길이인 90분으로 설정하였으며 비디오를 요청한 사용자가 서비스를 기다리다가 이탈하는 시간은 관람자들의 일반적인 행동을 반영하여 1분~5분으로 랜덤하게 설정하였다.

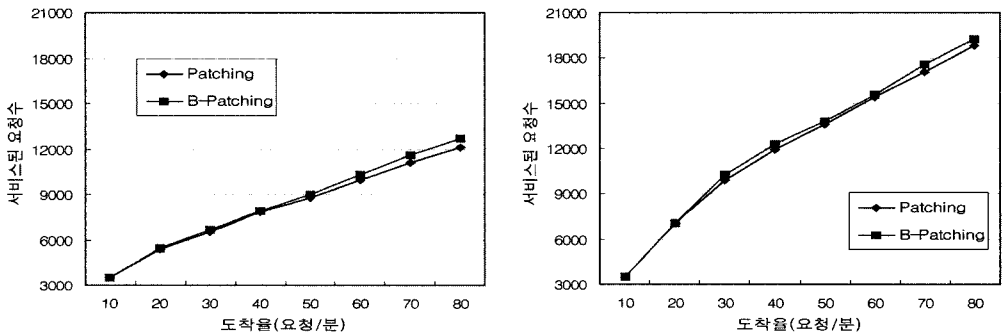
패칭 윈도우의 크기는 *Cai*[3]에서 얻어진 5분으로 설정하였으며 사용자 스테이션은 패칭과 버퍼링 패칭에서 필요한 비디오 스트림을 저장할 수 있는 충분한 버퍼공간을 가진다고 가정한다. 또한 사용자는 최소한 두 개의 비디오 스트림을 동시에 수신할 수 있는 충분한 대역폭을 가지며 비디오를 시작부터 끝까지 순차적으로 본다고 가정한다. 두 기법의 성능은 시뮬레이션 중단으로 인해 서비스되지 못한 사용자들을 제외시키기 위해 처음부터 350분 동안의 사용자 요청에 대해서만 비교 평가되었다.

그림 9와 그림 10은 *Patching*과 *B-Patching*은 각각 패칭과 제안한 버퍼링 패칭을 나타낼 때, 두 패칭 기법의 요청 도착율에 따른 이탈율과 실제로 서비스를 받은 사용자 수를 보여준다. 이탈율은 시뮬레이션 기간동안 도착한 전체 비디오 요청 중에서 대기 큐에서 서비스를 기다리다가 이탈한 사용자 요청이 차지하는 비율로서 이탈한 사용자 요청 수를 전체 요청 수로 나눈 값이다. 그림 9에서 두 가지 기법 모두 이탈율이 0인 경우를 제외하고, 버퍼링 패칭의 이탈율이 비디오 서버의 채널 수가 600일 때는 평균 3.2%, 채널 수가 900일 때 평균 14.7%만큼 개선되었음을 알 수 있다. 이것은 패칭에서는 패칭 채널 외의 모든 정규 채널이 비디오 전체 재생



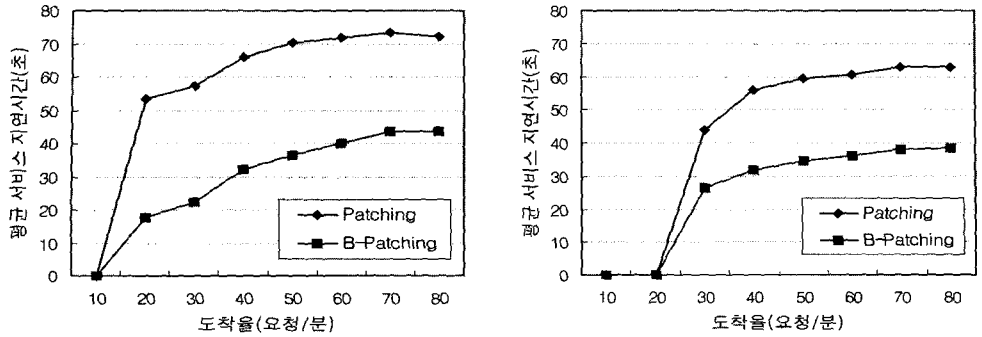
(a) 비디오 서버 채널 수 = 600 (b) 비디오 서버 채널 수 = 900

그림 9 요청 도착율에 따른 이탈율



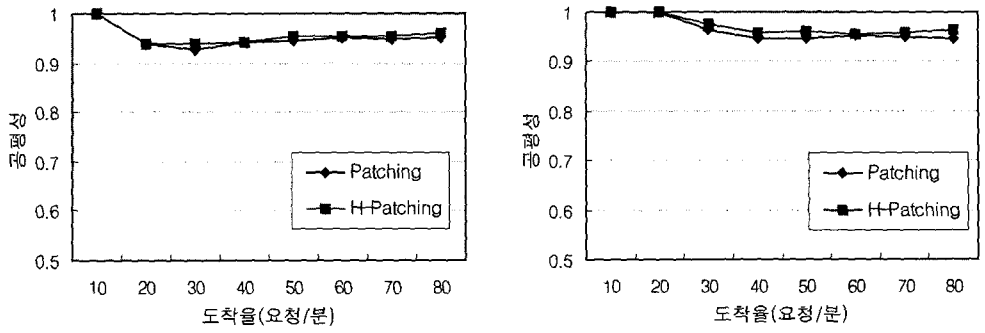
(a) 비디오 서버 채널 수 = 600 (b) 비디오 서버 채널 수 = 900

그림 10 요청 도착율별로 서비스된 요청의 수



(a) 비디오 서버 채널 수 = 600 (b) 비디오 서버 채널 수 = 900

그림 11 요청 도착율에 따른 평균 서비스 지연시간



(a) 비디오 서버 채널 수 = 600 (b) 비디오 서버 채널 수 = 900

그림 12 요청 도착율에 따른 공평성

시간동안 사용중인 것과는 달리 버퍼링 패칭에서는 부정규 멀티캐스트를 수행함으로써 패칭에서는 정규 채널로 사용되었던 많은 부정규 채널의 사용시간이 버퍼링 윈도우만큼 단축되었기 때문이다.

그림 11은 두 패칭 기법의 비디오 요청 도착율에 따른 평균 서비스 지연시간 결과를 보여준다. 평균 서비스 지연시간은 서비스를 받은 사용자들이 비디오를 요청한 후 서비스를 받을 때까지 대기 큐에서 기다린 평균 시간이다. 그림 11에서 보여 주듯이 제안한 버퍼링 패칭의 평균 서비스 지연시간이 비디오 서버의 채널 수가 600일 때는 평균 49.2%, 채널 수가 900일 때는 평균 40.5%만큼 많이 개선되었음을 알 수 있다.

그림 12는 비디오 요청 도착율에 따른 두 가지 기법의 공평성(*fairness*) 결과를 보여준다. 공평성은 모든 비디오 요청에 대한 평균 이탈율이 d이고 비디오 i에 대한 요청 이탈율이 d_i라고 할 때 식 2와 같이 계산된다[3].

$$fairness = 1 - \sqrt{\frac{\sum_{i=1}^{N}(d_i - d)^2}{N-1}} \qquad (2)$$

공평성은 두 가지 패칭 기법이 비슷한데 이는 패칭과 버퍼링 패칭 모두 가용 채널이 디스패치되었을 때 서비스할 요청을 스케줄하는 방법이 요청이 도착한 순서대로이며 특정 요청에 대해 우선적인 서비스를 제공하지 않는 기법이기 때문이다. 버퍼링 패칭의 공평성이 패칭보다 아주 조금 좋게 나타난 것은 버퍼링 패칭의 이탈율 개선에 기인한 것으로 패칭에서 이탈율이 높았던 비디오가 버퍼링 패칭에서는 이탈율이 낮아져 이탈율의 편차가 감소되었기 때문이다.

그림 13은 두 가지 패칭 기법에 대해 비디오 서버의 채널 수가 600일 때 사용자 스테이션에서 버퍼링한 평균 비디오 데이타의 양을 비교한 결과를 보여 준다 버퍼링 패칭이 버퍼링 윈도우의 크기가 5분일 때는 평균 246.0초, 버퍼링 윈도우의 크기가 3분일 때는 172.7초, 버퍼링 윈도우의 크기가 1분일 때는 57.2초에 해당하는 비디오 프레임을 패칭보다 많이 버퍼링하였는데 이 값들은 모두 버퍼링 윈도우의 크기와 비슷하다. 이는 버퍼링 패칭에서는 정규 멀티캐스트가 아닌 부정규나 패칭

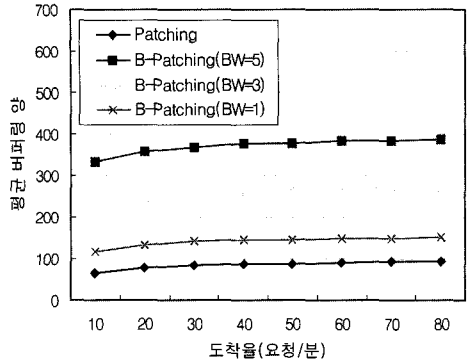


그림 13 버퍼링 윈도우 크기와 요청 도착율에 따른 버퍼링된 데이타 양

멀티캐스트로 서비스되는 모든 사용자들이 버퍼링 윈도우에 해당하는 비디오의 마지막 데이타를 정규 채널로부터 버퍼링하기 때문이다.

시뮬레이션을 통한 성능 평가 결과 버퍼링 패칭이 평균 서비스 지연시간, 이탈율, 공평성에 있어서 패칭보다 우수하였으나 사용자 스테이션에서 버퍼링해야 하는 데이타의 양이 패칭보다 평균적으로 버퍼링 윈도우 크기만큼 증가함을 알 수 있었다. VOD 시스템의 비용이 높은 이유는 대부분이 네트워크 비용에 기인하고 있으므로 VOD를 설계할 때는 네트워크의 총 대역폭을 완전히 이용하는 것이 중요하며 현재 사용자 시스템은 대용량의 디스크 공간을 확보하고 있으며 추가로 디스크 공간을 확보한다하더라도 그 비용은 아주 적다[2]. 실제로 VOD 성능의 저하는 사용자 스테이션의 디스크 공간 부족보다는 비디오 서버의 네트워크 부하에 기인하고 있다고 할 수 있다. 그러므로 서버의 네트워크 대역폭을 최대한 이용하여 비디오 서버의 성능을 높일 수 있는 버퍼링 패칭이 더 좋은 방법이라고 할 수 있다.

5. 결 론

VOD 서비스의 성능을 높이기 위해서는 비디오 서버의 네트워크 대역폭에 의해 결정되는 한정된 채널 자원을 효율적으로 사용하면서도 서비스 지연시간과 이탈율을 감소시킬 수 있는 방법이 필요하다. 멀티캐스트 기법은 새로운 비디오 요청들에 대하여 이미 전송중인 동일 비디오 스트림을 공유함으로써 서버의 네트워크 대역폭 요구를 감소시키는 우수한 기법이다. 특히 패칭은 사용자 스테이션에서 두 개의 비디오 스트림을 동시에 수신할 수 있는 대역폭을 요구하지만 서비스 지연시간 없이 사용자 요청을 즉시 서비스할 수 있어 true VOD 서비스를 제공한다.

본 논문에서는 패칭에서 사용되는 사용자 스테이션의 네트워크 대역폭을 최대한 활용하고 비디오 서버의 네트워크 대역폭을 효율적으로 활용하기 위해 비디오 전체 데이타 중에서 버퍼링 윈도우에 해당하는 마지막 부분을 제외한 비디오 데이타를 전송하는 부정규 멀티캐스트 개념을 추가한 버퍼링 패칭 기법을 제안하였다. 동일한 비디오 요청에 대해 정규 멀티캐스트로 서비스되지 않는 모든 요청들은 이미 전송중인 정규 스트림 중에서 버퍼링 윈도우 크기에 해당하는 마지막 부분을 자신의 로컬 버퍼에 버퍼링한다. 패칭에서는 비디오 전체 길이 동안 사용되었던 정규 채널의 일부를 부정규 채널로 사용하면서 부정규 채널의 사용 시간을 버퍼링 윈도우만큼 단축시킬 수 있었다. 제안한 버퍼링 패칭의 성능을 시뮬레이션을 통해 평가한 결과 사용자 스테이션에서 버퍼링하는 데이타의 양은 증가하였으나, 평균 서비스 지연 시간, 이탈율, 공평성에 있어서는 패칭보다 우수함을 알 수 있었다.

결론적으로 본 논문이 제안한 버퍼링 패칭은 정규 멀티캐스트로 서비스되지 않는 모든 요청들이 버퍼링 윈도우만큼의 비디오 스트림을 공유함으로써 네트워크 부하를 감소시켜 패칭보다 VOD 서버의 성능을 향상시킬 수 있는 방법이며 향후 사용자 스테이션에서 버퍼링되는 데이타의 양을 감소시킬 수 있는 방법을 연구하고자 한다.

참 고 문 헌

[1] Jani Huoponen and Thorsten Wagner, "Video on Demand A Survey," telecommunication Networks Project, I, http://fiddle.visc.vt.edu/courses/ee4984/Projects1996/huoponen_wagner/huoponen_wagner.html, 1996.

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

[3] Y. Cai, K. Hua, and K. Vu, "Optimizing Patching Performance," *In Proc. SPIE/ACM Conference on Multimedia Computing and Networking*, pp. 204-215, 1999.

[4] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Polices for an On-Demand Video Server with Batching," *In Proc. of the 2nd ACM Multimedia Conference*, pp. 25-32, 1994.

[5] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "On Optimal Piggyback Merging Policies for Video-On-Demand Systems," *In Proc. 1996 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pp. 200-209, 1996.

[6] L. Golubchik, J. Lui, and R. Muntz, "Adaptive Piggybacking: Arrival Technique for Data Sharing in Video-on- Demand Service," ACM Multimeda

Systems, Vol.4, No.3, pp.140-155, 1996.
[7] A.Chervenak, D. Patterson, and R. Katz, "Choosing the Best Storage System for Video Service," *In Proc. of ACM Multimedia 95*, pp. 109-119, Aug. 1995.
[8] Christer Samuelesson, "Relating Turing's Formula and Zipf's Law," Proceedings of the Fourth Workshop on Very Large Corpora, 1996.

하 숙 정
정보과학회논문지 : 시스템 및 이론
제 30 권 제 5 호 참조

배 인 한
정보과학회논문지 : 시스템 및 이론
제 30 권 제 5 호 참조