True VOD 서비스를 위한 혼성 패칭 방법의 설계

o
하숙정*, 이경숙*, 배인한**
*경북대학교 전자전기컴퓨터학부, **대구가톨릭대학교 컴퓨터정보통신공학부

Design of Hybrid Patching Method for True VOD Services

o
Sook-Jeong Ha*, Kyung-Sook Lee*, Ihn-Han Bae**
* School of Electrical Engineering and Computer Science, Kyungpook National University
E-mail : sjha@m80.knu.ac.kr, lks8682@hotmail.com
** School of Computer & Information Communication Engineering, Catholic University of Daegu
E-mail : ihbae@cuth.cataegu.ac.kr

요 약

패칭은 새로운 비디오 요청들에 대하여 이미 전송중인 동일 비디오 스트림을 공유함으로써 비디오 서버의 네트워크 대역폭 요구를 감소시키며 true VOD를 지원하는 멀티캐스트 기법이다. 본 논문에서는 탐욕 패칭과 점진 패칭의 성능을 향상시키기 위해 두 가지 방법의 장점을 취할 수 있는 혼성 패칭 방법을 제안한다. 혼성 패칭은 탐욕 패칭을 기본으로 하고 있으나 새로운 요청과 동일 비디오에 대한 최근의 정규 멀티캐스트간의 스큐가 패칭 윈도우보다는 크지만 최근의 패칭 멀티캐스트와의 스큐가 패칭 윈도우보다 작은 경우에는 최근의 패칭 멀티캐스트를 정규 멀티캐스트가 되도록 확장함으로써 새로운 요청을 위한 패칭 채널의 시간을 대폭 감소시켜 VOD 시스템의 성능을 향상시키고자 한다.

1. 서론

멀티미디어의 대표적인 형태인 주문형 비디오 서비스는 향후 가장 각광받게 될 서비스 중 하나로 디지털 비디오 도서관, 주문형 뉴스, 원거리 교육과 같은 많은 멀티미디어 응용을 위한 중요한 기술이다[1, 2]. 전형적인 VOD(Video-on-Demand) 서비스에서 비디오 서버는 대량의 비디오들을 저장하고 있으며 원거리의 사용자는 장소와 시간에 제한 받지 않고 자신이 원하는 비디오를 서버에 요청함으로써 서버로부터 전송되는 비디오 스트림을 다운로드하여 재생한다. 하나의 비디오 스트림을 유지하는데 필요한 시스템 자원을 비디오 채널이라고 하는데 비디오 서버의 가용 채널 수는 통신 대역폭에 의해 결정되므로 한정되어 있다[2, 3]. 더욱이 비디오 스트림은 비록 압축이 되어있지만 대역폭 요구량이 높으며, 재생시간이 길고, 엄격한 응답시간을 요구하며, 연속적으로 재생되어야 하므로 비디오 서버의 자원 소비가 높다. 그러므로 여러 사용자들의 비동기적 비디오 요청을 적은 서비스 지연시간으로 보다 많은 요청을 서비스하기 위해 여러 사용자들이 멀티캐스트되는 비디오 스트림을 공유하는 방법들이 제안되었다[2, 3, 4, 5, 6].

특히 패칭[2, 3]은 새롭게 서비스할 사용자를 기존의 멀티캐스트에 합병시킴으로써 멀티캐스트되는 비디오 스트림의 공유를 증가시켜 비디오 서버가 새로 전송해야할 비디오 데이터를 감소시킨다. 더욱이 비디오 요청을 일정기간 동안 모아두는 배칭[4]과는 달리 패칭은 비디오를 요청한 사용자가 서비스 지연시간 없이 서비스를 받을 수 있으므로 true VOD를 성취할 수 있는 방법이다. 패칭은 패칭 채널로 전송할 비디오 데이터의 양을 결정하는 방법에 따라 탐욕(greedy) 패칭과 점진(grace) 패칭으로 구분된다. 본 논문에서는 패칭의 성능을 개선하기 위해 두 패칭 방법의 장점을 취할 수 있는 혼성 패칭을 제안한다. 제안하는 방법은 탐욕 패칭을 기본으로 하여 멀티캐스트 데이터의 공유를 최대화하며 최근의 패칭 멀티캐스트를 정규 멀티캐스트로 확장함으로써 새로운 요청을 위한 패칭 채널의 사용시간을 대폭 감소시킨다. 본 논문의 2장에서는 탐욕과 점진 패칭을 소개하고, 3장에서는 점진 패칭과 탐욕 패칭을 혼합한 혼성 패칭을 제안하고, 4장에서 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 배칭 윈도우라는 짧은 기간 동안 일단 모아 두

었다가 동일한 비디오의 요청들에 대해 한꺼번에 한 채널상으로 비디오 전체 데이터를 멀티캐스트한다. 그러나 일정 기간동안 요청이 모일 때까지 기다렸다가 서비스되므로 사용자의 서비스 지연시간이 길다. 피기백킹은 진행 중인 비디오 스트림의 재생율을 조정하여 동일 비디오에 대한 여러 스트림이 하나로 합병되도록 하는 기법으로 배칭보다 서비스 지연시간이 짧지만 피기백 율의 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, 나머지 대역폭은 서비스 요청과 서비스 통보 같은 제어 메시지를 전송하는데 사용된다[2]. 한 개의 비디오 데이터 전체를 멀티캐스트하는데 사용되는 채널은 정규 채널, 비디오의 시작에서부터 일부분만을 멀티캐스트하는 채널은 패칭 채널이라고 한다. 정규 채널과 패칭 채널을 통해 사용자가 수신하는 데이터 스트림들은 각각 정규 스트림과 패칭 스트림이라고 한다.

정규 멀티캐스트가 시작된 t_r 이후로 특정 기간 안에 새로 도착하여 t 시점에 서비스되는 요청에 대해서는 정규 멀티캐스트가 아닌 패칭 멀티캐스트가 수행된다. 즉 새로운 요청과 정규 멀티캐스트간의 스큐[2]인 $t-t_r$이 특정 기간보다 작으면 패칭 멀티캐스트가 수행된다. 이 특정 기간을 사용자 스테이션에서 정규 스트림을 버퍼링할 수 있는 정규 버퍼의 크기로 사용하는 방법을 점진 패칭이라고 하며 이 정규 버퍼의 크기를 패칭 윈도우 크기라고 한다. 반면에 정규 멀티캐스트 데이터의 공유를 최대화하기 위해 진행중인 정규 멀티캐스트가 존재하는 한 패칭 멀티캐스트를 수행하는 방법 즉 특정 기간이 정규 멀티캐스트되는 비디오의 전체 재생기간인 방법을 탐욕 패칭이라고 한다. 패칭을 구현하기 위한 비디오 서버와 사용자 스테이션의 알고리즘[2]에서 비디오 서버가 새로 디스패치된 가용 채널 *Free_Ch*가 전송할 비디오 데이터 부분을 결정하기 위해 호출하는 점진 패칭과 탐욕 패칭의 *GreedyPatching(Free_Ch, Regular_Ch)*과 *Grace Patching(Free_Ch, Regular_Ch)*을 그림 1에서 보여준다.

그림 1의 *GreedyPatching*에서 단계 2는 스큐가 PW보다 커서 버퍼에 스큐만큼의 정규 스트림을 버퍼링할 수는 없지만 최대 PW만큼의 정규 스트림의 마지막 일부분은 버퍼에 저장할 수 있는 경우이다. 현재 시간부터 최근 정규 멀티캐스트가 종료되기까지의 기간이 PW 이상이라면 PW만큼의 정규 스트림의 끝부분을 공유할 수 있으며 PW보다 작다면 현재 시간부터 정규 멀티캐스트의 종료까지 남은 시간에 해당하는 정규 스트림의 끝부분을 공유할 수 있다.

그림 2는 패칭을 사용하는 VOD 시스템의 예를 보여준다. 사용자 스테이션의 데이터 로더 L_r과 L_p는 각각 정규 채널과 패칭 채널로 전송되는 정규 스트림과

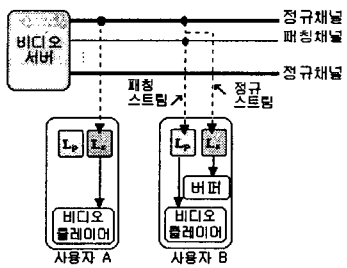
t	:current time
t_r	:start time of the latest regular multicast on regular channel *Regular_Ch* for video v
v	:the video currently multicast on channel *Regular_Ch*
PW	:size of patching window(i.e. size of the client buffer in playback time unit)
D	:portion of video data which should be multicast on channel *Free_Ch*(i.e. workload for channel *Free_Ch*)
l	:playback duration of video v(i.e. video length)
$v[t_d]$	:video data of video v from start to t_d time unit

GreedyPatching(*Free_Ch, Regular_Ch*)
1. if $(t-t_r < PW)$ $D=v[t-t_r]$
2. else
   $D = v[l - Min(PW, l - (t - t_r))]$

GracePatching(*Free_Ch, Regular_Ch*)과
1. if $(t-t_r < PW)$ $D=v[t-t_r]$
2. else
   -$D = v[l]$
   -Modify the service token as (*PID=null, RID=Free_Ch*) so that *Free_Ch* is designated to start a new regular multicast.

(그림 1) *Free_Ch*이 전송할 데이터 결정 알고리즘

패칭 스트림을 다운로드하는 책임을 지고 있으며 비디오 플레이어는 버퍼에 저장된 비디오 프레임을 인출하여 재조립한 후 스크린에 상영하는 기능을 제공한다.

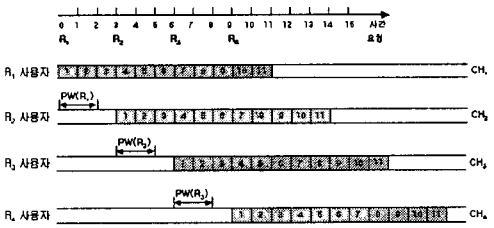


(그림 2) 패칭 시스템 예[2]

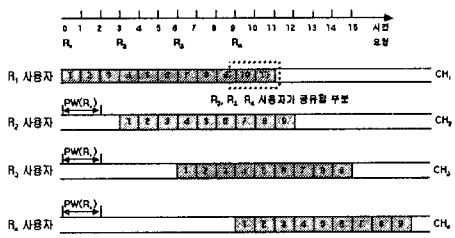
3. 혼성 패칭

그림 3과 그림 4는 비디오의 길이(l)가 11분, 비디오 i를 요청한 모든 사용자의 버퍼 크기 즉 패칭 윈도우의 크기(PW)가 2분, 비디오 i에 대한 4개의 요청 R_1, R_2, R_3, R_4가 차례대로 비디오 서버에 도착한다고 가정하였을 때 각각 탐욕과 점진 패칭이 보다 우수한 경우를 보여준다. 그림 3은 비디오 요청의 도착 간격이 패칭 윈도우보다 크면서 일정하게 도착하는 경우로 점진 패칭에서는 요청 R_2, R_3, R_4 모두 최근의 정규 멀티캐스트가 시작한 후 패칭 윈도우 만큼의 시간이 지난 후에 도착했으므로 모두 정규 멀티캐스트로 서비스된다. 그러나 그림 3(b)의 탐욕 패칭에서는 요청 R_2, R_3, R_4 모두 R_1을 위한 정규 멀티캐스트의 패칭 윈도우는 벗어났으나 아직 이 정규 멀티캐스트가

진행되고 있으므로 새로운 정규 멀티캐스트는 발생하지 않으며 비디오의 처음부터 (l-PW)분 동안의 데이터를 새로운 패칭 채널로 전송하며 정규 스트림의 마지막 PW분 동안의 데이터는 공유한다. 점진 패칭의 경우 4개의 요청을 위한 채널의 사용 시간은 11*4=44분이지만 탐욕 패칭은 11+9*3=38분으로 정규 스트림의 마지막 일부분이라도 공유하는 탐욕 패칭이 더 우수하다. 이와 같은 결과는 점진 패칭에서 새로운 요청이 최근의 멀티캐스트의 패칭 윈도우 안에 도착해야만 패칭 멀티캐스트가 수행되기 때문에 요청 도착간격이 패칭 윈도우보다 큰 경우에는 패칭의 이득을 전혀 얻지 못하기 때문이다.



(a) 점진 패칭

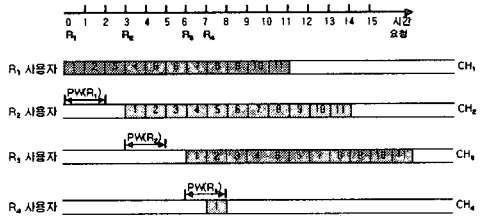


(b) 탐욕 패칭

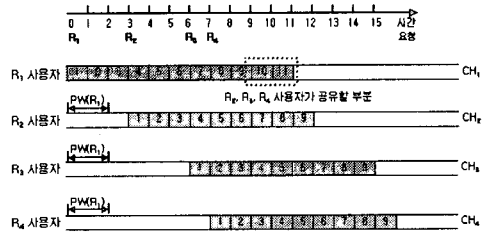
(그림 3) 탐욕 패칭이 우수한 예

그림 4는 그림 3과 달리 요청 R_3와 요청 R_4의 도착간격이 패칭 윈도우보다 작은 경우이다. 그림 4(b)에서처럼 탐욕 패칭은 요청 R_4를 서비스하고자 할 때 요청 R_1에 대한 정규 멀티캐스트가 진행 중이므로 마지막 PW분 동안의 비디오 데이터를 공유한다. 그러나 그림 4(a)의 점진 패칭에서는 요청 R_4의 서비스 시점이 요청 R_3를 위한 최근의 정규 멀티캐스트의 패칭 윈도우 안에 존재하므로 스큐인 1분 동안의 비디오 시작부분만 새로운 패칭 채널로 전송하며 비디오의 나머지 데이터는 채널 CH_4상의 정규 스트림을 공유하면 된다. 탐욕 패칭은 채널 사용 시간이 11+9*3=38분이지만 점진 패칭은 11*3+1=34분으로 탐욕 패칭보다 우수하다.

그림 4(a)에서 점진 패칭이 우수한 것은 요청 R_3와 요청 R_4의 도착 간격이 그림 3과는 달리 PW보다 작기 때문이다. 만일 그림 4(b)에서 요청 R_4를 서비스하는 시점에서 채널 CH_3로 전송되는 데이터가 PW만큼 더 추가되어 비디오 전체 데이터가 된다면 요청 R_4는 이 멀티캐스트와의 스큐가 PW보다 작으므로 1분 동안의 비디오 시작 부분만 멀티캐스트하면 될 것이다. 그렇게 되면 채널 사용시간이 11+9+11+1=32로 점진 패칭보다 더 우수해진다. 그러므로 본 논문에서는 탐욕 패칭에서 이미 진행 중인 패칭 멀티캐스트를 정규 멀티캐스트가 되도록 확장함으로써 점진 패칭에서의 이점을 얻을 수 있는 혼성 패칭 방법을 제안한다.



(a) 점진 패칭



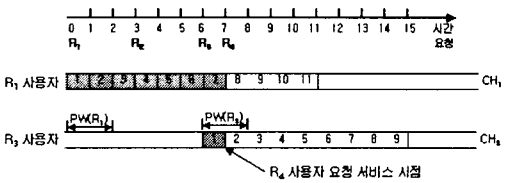
(b) 탐욕 패칭

(그림 4) 점진 패칭이 우수한 예

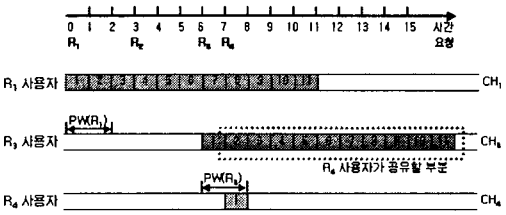
패칭에서는 비디오마다 가장 최근의 정규 멀티캐스트의 시작 시간 t_r을 유지하는데 혼성 패칭에서는 최근의 정규 멀티캐스트가 진행되는 동안 가장 최근에 발생한 패칭 멀티캐스트의 시작 시간 t_p 또한 유지한다. 새로운 요청의 서비스 시점 t가 최근의 정규 멀티캐스트의 패칭 윈도우를 벗어났더라도 최근 패칭 멀티캐스트의 패칭 윈도우를 벗어나지 않았다면 비디오 서버는 이 최근의 패칭 멀티캐스트를 정규 멀티캐스트가 되도록 확장하여 서비스를 계속한다. 이로 인해 패칭 채널의 사용 시간은 비디오의 끝까지 데이터를 전송하기 위해 $Min(PW,\ l-(t-t_r))$만큼 증가한다. 그러나 새로 확장된 패칭 멀티캐스트로 인해 새로운 요청은 비디오의 시작부터 l-$Min(PW,\ l-(t-t_r))$까지가 아닌 확장된 패칭 멀티캐스트와의 스큐만큼 즉 시작부터 $(t-t_p)$까지만 새로운 패칭 채널로 전송하면 되므로 최소 $(l\ -\ 2PW)$만큼의 채널 사용시간이 감소된다. 일반 비디오의 길이가 90분이고 패칭 윈도우가 5분이라고 가정한다면 증가되는 채널 사용시간은 최대 5분이며 감소되는 시간은 최소 80분이 된다.

그림 5(a)는 그림 4와 같은 상황으로 요청 R_4의 서비스를 시작하기 직전의 상태를 보여준다. 요청 R_3를 위해 패칭 멀티캐스트가 진행되어 현재 1분까지의 비디오 데이터를 전송한 상태에서 요청 R_4를 서비스해야한다. 탐욕 패칭이라면 그림 4(b)와 같이 요청 R_4에 대해서도 비디오의 처음부터 (l-PW)까지의 패칭 스트림을 채널 CH_4로 전송할 것이다. 그러나 혼성 패칭에서는 이 요청이 R_1을 위한 최근의 정규 멀티캐스

트의 패칭 윈도우는 벗어났지만 R_3를 위한 최근의 패칭 멀티캐스트의 패칭 윈도우는 벗어나지 않았음을 확인하고 비디오 서버는 요청 R_3에 대한 패칭 멀티캐스트가 비디오 끝까지의 데이터를 전송하도록 확장하며 최근 정규 멀티캐스트의 시작 시간과 사용 채널을 6분과 CH_3으로 수정한다. 그렇다면 요청 R_4는 새로운 최근의 정규 멀티캐스트의 패칭 윈도우 안에 있게 되므로 스큐에 해당하는 1분 동안의 데이터만 새로운 패칭 채널 CH_4로 전송한다. 결국 각 채널로 전송되는 데이터는 그림 5(b)와 같이 된다.



(a) 요청 R_4를 서비스하기 직전의 채널 상태



(b) 요청 R_4를 위해 확장된 요청 R_3의 스트림

(그림 5) 혼성 패칭 예

HybridPatching(*Free_Ch, Regular_Ch*)
t_p : Start time of the latest patching multicast on patching channel *Patch_Ch* which is started after the latest regular multicast for video v.
1. if ($(t-t_r) < PW$) $D = v[t-t_s]$
2. else if ($(t-t_p) < PW$)
   -Expand the portion of the video data which should be multicast on the patching channel *Patch_Ch* to the end of the video so that *Patch_Ch* is used as a regular channel from now on.
   -*Regular_Ch=Patch_Ch*
   -$D = v[t - t_p]$
3. else
   $D = v[l - Min(PW, l - (t - t_r))]$

(그림 6) 혼성 패칭 알고리즘

이와 같이 혼성 패칭은 새로 서비스할 요청이 최근의 정규 멀티캐스트의 패칭 윈도우를 벗어났다면 최근의 패칭 멀티캐스트와의 스큐를 확인하여 스큐가 패칭 윈도우보다 작다면 최근의 패칭 멀티캐스트가 최근의 정규 멀티캐스트가 될 수 있도록 확장하여 수정함으로써 새로운 요청이 최근의 정규 멀티캐스트의 패칭 윈도우 안에 들어오도록 함으로써 탐욕 패칭의 단점을 보완하여 점진 패칭에서와 같은 이득을 얻으며, 스큐가 패칭 윈도우보다 클 경우에는 탐욕 패칭과 같이 서비스함으로써 정규 멀티캐스트 데이터의 공유를 최대화시킨다. 혼성 패칭에서 비디오 서버가 새로 디스패치된 가용 채널 *Free_Ch*로 전송할 비디오 v의 데이터를 결정하기 위해 호출하는 *HybridPatching (Free_Ch, Regular_Ch)*가 그림 6에 있다.

4. 결론

VOD 시스템의 성능을 높이기 위해서는 비디오 서버의 한정된 채널 자원을 효율적으로 사용하여 서비스 지연시간과 이탈율을 감소시킬 수 있는 방법이 필요하다. 패칭은 새로운 비디오 요청들이 이미 전송중인 동일 비디오 스트림을 공유함으로써 서버의 네트워크 대역폭 요구를 감소시켜 true VOD를 지원하는 멀티캐스트 기법이다. 본 논문에서는 비디오 서버의 성능을 향상시키기 위해 탐욕 패칭과 점진 패칭의 장점을 취할 수 있는 혼성 패칭 방법을 제안하였다. 탐욕 패칭이 점진 패칭보다 성능이 좋지 못한 경우는 새로운 요청이 최근 정규 멀티캐스트의 패칭 윈도우를 벗어났지만 최근의 패칭 멀티캐스트와의 스큐는 패칭 윈도우보다 작을 때임을 발견하였으며 이를 해결하기 위해 최근의 패칭 멀티캐스트를 정규 멀티캐스트가 되도록 확장함으로써 새로운 요청을 위해 사용되는 패칭 채널의 시간을 대폭 감소시켰다. 향후 시뮬레이션을 통해 평균 서비스 지연시간, 공평성, 이탈율과 같은 항목에 대해 혼성 패칭, 탐욕 패칭, 점진 패칭에 대한 성능을 자세히 평가할 계획이다.

[참고문헌]

[1] Jani Huoponen and Thorsten Wagner, "Video on Demand A Survey," telecommunication Networks Project, 1, 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.