VOD 서버에서 클라이언트 버퍼 확장을 통한 효율적인 패칭 기법

이근정° 권춘자** 최황규*
* 강원대학교 전기전자정보통신공학부
** 강릉영동대학 사이버경찰과
phinex93@hanmail.net°, kwoncj@mail.kangwon.ac.kr**, hkchoi@kangwon.ac.kr*

Efficient Patching Scheme Based on Client Buffer Expanding for VOD Servers

Keun Jung Lee° Chun Ja Kwon** Hwang Kyu Choi*
* Dept. of Electrical and Computer Engineering, Kangwon National University
** Dept. of Cyber Police, Kangneung Yeongdong College

요 약

본 논문은 클라이언트 버퍼 확장 기법과 프록시 프리픽스 캐싱을 응용하여 기존의 패칭 기법의 성능을 향상시키기 위한 새로운 패칭 기법을 제안한다. 제안된 기법은 프록시에서 비디오의 프리픽스를 캐싱하여 패칭 윈도우 크기를 확장한다. 또한 클라이언트의 버퍼로 부터 오버플로우된 스트림을 공유하고 저장하기 위해 프록시 버퍼를 통해 클라이언트의 버퍼 공간을 확장한다. 프록시에서 프리픽스와 패칭 윈도우 크기가 확장되므로 기존의 패칭 기법보다 서버의 대역폭 요구량이 현저히 줄어들게 된다. 성능 평가를 통해 제안된 기법이 서버의 대역폭 요구량을 줄일 수 있음을 보인다.

1. 서 론

VOD(Video-On-Demand) 서비스는 온라인 영화, 전자 도서관, 원격 교육 등에 널리 활용되고 있다. 이러한 서비스는 서버의 높은 대역폭을 긴 시간동안 꾸준히 요구하는 것이 특징이다. 따라서 이러한 비디오 시스템은 서버내의 저장 공간에 대한 대역폭과 서버와 연결된 네트워크의 대역폭에 대한 병목현상이 발생한다.

이러한 대역폭의 문제를 해결하기 위하여 멀티캐스트와 브로드캐스트를 활용한 연구가 이루어 졌으며, 그 결과로 VOD 서버의 성능 향상을 가져왔다. 그중 패칭[1]은 멀티캐스트의 장점을 활용하는 가장 효과적인 기법이다. 그러나 기존의 패칭 기법은 클라이언트의 버퍼 또는 최적 패칭 윈도우의 크기에 의하여 그 성능이 제한된다.

본 논문에서는 기존의 패칭 기법의 성능을 개선하기 위하여 프록시 서버를 활용한 프리픽스 캐싱[3-8]과 클라이언트의 버퍼 확장 기법을 제안한다. 제안된 기법에서 프록시 서버는 비디오의 앞부분을 고정된 크기만큼 저장하고 서비스하는 프리픽스 캐싱을 수행하며, VOD 서버는 프록시 이후의 부분에 대해서만 전송을 수행한다.

따라서 최적 패칭 윈도우[2]의 크기는 프리픽스의 크기보다 크게 되며, 프록시의 프리픽스의 크기가 클수록 서버에 요구되는 네트워크의 소모량은 줄어든다. 또한 제안된 기법은 확장된 패칭 윈도우 크기를 수용하기 위하여 클라이언트의 버퍼 확장 기술을 제안한다. 즉, 클라이언트의 버퍼 크기가 확장된 패칭 윈도우의 크기를 수용하지 못하는 경우 프록시 서버의 버퍼를 추가로 할당하여 연장하도록 한다. 또한 이러한 기법이 VCR 기능을 효과적으로 지원할 수 있음을 보인다.

2. 관련연구

본 장에서는 본 논문과 관련된 패칭 기법과 프록시 캐싱 기법의 연구 결과에 대하여 알아본다. 패칭 기법[1]은 초기 요청에 대하여 정규 채널을 생성하고 이후의 요청은 동적으로 정규채널에 가입시키고, 지나간 부분은 패칭 채널을 통하여 전송받는 방식이다. 이러한 패칭 기법은 초기 지연시간이 존재하지 않으며 클라이언트의 버퍼링을 위한 저장 공간을 요구한다. 또한 새로운 클라이언트가 기존의 정규 채널에 참여할 수 있는 최대 시간 차이를 패칭 윈도우 크기로 정의하고 요청 시점이 그 크기를 지나면 새로운 정규 채널을 생성하도록 하였다.

패칭 기법은 그 적용 방법에 따라 크게 Greedy 패칭과 Grace 패칭 및 Optimal 패칭으로 구분된다[2]. 또한 [3]은 클라이언트의 버퍼 활용률을 최대로 높이기 위하여 GBR(General Buffer Reuse) 기법을 제안하고, 서버 대역폭을 최소로 하는 최적 패칭 윈도우 크기를 보였다.

VOD 시스템의 성능 향상을 위한 또 다른 방법은 프록시 캐싱 기법을 활용하는 것이다. 프록시를 활용한 기법으로는 먼저 프록시 프리픽스 캐싱을 적용하여 네트워크의 소모량을 최소로 하는 프록시 캐시의 최적 할당 기법이 제안되었다[5]. 또한 [6]은 네트워크의 소모량을 최소로 하기 위한 Joint Server Scheduling 및 스트림의 캐싱 방법을 제안하고 그에 따른 Tradeoff를 보였다. [4]는 단일 비디오 스트림의 최적의 프록시 캐시 정책을 제시하였다. 또한 Gradient-Descent-Based Cache 할당 방법과 이를 서로 다른 다수의 비디오 스트림에 적용하는 방법을 보였다. 마지막으로 [7]은 클라이언트의 요청률에 따른 최적의 Batched Patch Caching 방법을 제안하였다.

3. 버퍼 확장을 통한 패칭 기법

3.1 기본 동작

인터넷을 통하여 전송되는 VOD 서비스는 네트워크 거리에 따른 초기 지연시간이 발생한다. 프록시 프리픽스 캐싱[3-8]을 활용한 패칭 기법은 이러한 초기 지연시간을 해결하고 VOD 서버의 네트워크 부하를 줄이는 좋은 방법이다. 그러나 일반적인 패칭 기법의 성능은 클라이언트의 버퍼 크기 또는 최적 패칭 윈도우 크기에 의하여 제한된다. 따라서, 본 논문은 프록시 프리픽스를 활용하는 패칭 기법의 성능을 향상시키기 위하여 프록시 프리픽스 캐싱과 클라이언트 버퍼 확장 기술에 기반한 새로운 패칭 기법을 제안한다.

그림 1은 본 논문에서 제안된 기법의 개략적인 구성을 나타낸다. 그림 1 에서 모든 클라이언트들은 항상 비디오의 시작지점부터 재생을 요구하며 각 요청은 프록시를 통하여 이루어진다. 또한, 프록시 서버에 저장된 프리픽스 이후의 비디오 스트림은 VOD 서버로부터 전송받는다. 또한 모든 요청은 클라이언트로부터 프록시 서버를 통하여 이루어진다. 또한 각각의 정규 채널과 패칭 채널의 전송률은 비디오의 재생률과 같다고 가정한다. VOD 서버는 모든 비디오를 저장하고, 프록시 서버는 요청 빈도가 높은 비디오의 프리픽스를 저장한다. 더불어, 프록시 서버는 프리픽스 캐싱에 의하여 확장된 최적 패칭 윈도우의 크기를 수용하기 위하여 클라이언트들에 의하여 공유되는 버퍼를 추가로 할당한다.

각 프록시에 저장된 프리픽스의 시간 내에 도착한 요청은 프록시가 직접 서비스 하므로 서버의 부하를 요구하지 않는 반면 프리픽스 이후의 요청에 대해서만 서버의 추가적인 패칭 채널을 요구한다. 즉, 프리픽스 크기를 지난 시간에 대해서만 서버에 추가적인 패칭 채널을 요구하므로 서버에 대한 최적 패칭 윈도우의 크기는 프리픽스 크기를 초과하는 범위로 확장되게 된다.

또한, 클라이언트는 패칭 스트림을 재생하는 시간동안 자신의 버퍼에 정규 스트림을 저장하여야 한다. 만약 확장된 패칭 윈도우

본 논문은 2004년 정보통신 기초기술연구지원사업 (과제번호 04-기초004) 연구결과의 일부임

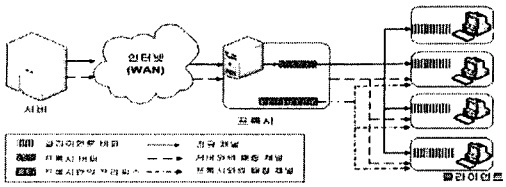

그림 1. 제안된 기법의 기본 개념도

의 크기가 클라이언트의 버퍼 공간을 초과하는 경우, 프록시 서버는 초과된 크기의 공간을 할당하여 정규 스트림을 저장한다. 즉, 정규스트림을 저장하기 위한 공간을 프록시 서버로 확장한다. 이때 프록시 버퍼에 저장된 정규스트림은 FIFO 방식으로 동작하여 클라이언트의 버퍼로 이동한다. 이렇게 프록시로 확장된 버퍼 공간은 패칭 윈도우에 속한 클라이언트들에 의하여 공유되므로 각 클라이언트의 추가적인 버퍼를 요구하지 않으면서 늘어난 패칭 윈도우 크기를 수용할 수 있다.

3.2 패칭 시나리오

본 논문의 패칭의 성능 향상 기법을 그림 2를 예제로 하여 각 클라이언트의 요청에 따른 패칭 방법을 보여준다. 클라이언트 $C=\{C_0, C_1, C_2, C_3, \ldots, C_n, \ldots\}$는 시간 $T=\{t_0, t_1, t_2, t_4, \ldots, t_n, \ldots\}$, $n \geq 0$에 비디오에 대한 재생을 요청한다. B_C와 B_P는 각각 클라이언트 버퍼 크기와 프록시 버퍼 크기를 나타낸다. V_P는 프록시 안에 저장된 프리픽스 크기를 나타내고, V_S는 VOD 서버로부터 전송되는 프록시에 없는 나머지 프리픽스 크기를 나타낸다. 또한 PW_{opt}는 제안된 기법의 최적의 패칭 윈도우 크기를 나타낸다.

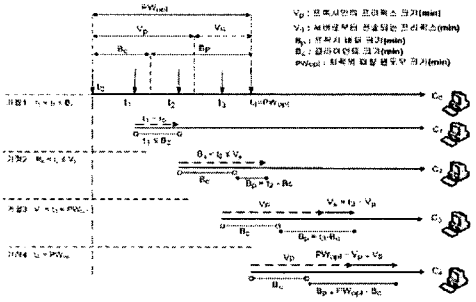

그림 2. 패칭의 성능 향상 기법의 시나리오

그림 2에서 정규 채널이 t_0의 시간에 시작된 정규 채널에 대하여 연속적인 클라이언트들이 참여한다고 가정한다. 즉, 그림 2의 각 클라이언트 C_i는 서로 다른 시각 t_i에 동일한 비디오를 요청한다고 가정하며, 요청 시간 t_i는 $t_0 < t_i \leq PW_{opt}$의 범위를 갖는다. 각 클라이언트 C_1, C_2, C_3와 C_4의 요청에 대하여, 클라이언트 C_1은 t_1 시간에 요청한다. 이때 $t_1 < B_C$이고 $t_1 < V_P$이다. 이 경우, 클라이언트의 버퍼는 $t_1 - t_0$의 크기를 저장하기에 충분하다. 다음으로, 클라이언트 C_2는 t_2 시간에 요청하고, 이때 $B_C < t_2 \leq V_P$ 범위 안에 있다. 이 경우, 클라이언트 C_2는 $t_2 - B_C$ 크기만큼의 프록시 버퍼를 필요로 한다. C_1과 C_2의 경우 모두 $t_2 < V_P$ 이므로 VOD 서버로부터 프리픽스의 전송을 필요로 하지 않는다. 이와 다르게 클라이언트 C_3의 요청시간 t_3는 $V_P < t_3 < PW_{opt}$의 범위 안에 있다. 따라서 프록시에 VOD 서버로부터 추가적인 프리픽스 부분에 해당하는 $V_S = t_3 - V_P$ 만큼의 크기를 전송받아야 한다. 마지막으로, 클라이언트 C_4는 t_4 시간에 요청하며, 이때 $t_4 = PW_{opt}$ 이다. 따라서 $V_S = PW_{opt} - V_P$ 만큼의 추가적인 프리픽스를 VOD 서버로부터 전송받아야 한다. 또한, 최적 패칭 윈도우의 크기를 지나게 되므로 클라이언트 C_4의 요청 이후에 패칭 윈도우는 닫히고 다음 클라이언트들을 위해 새로운 패칭 윈도우가 시작될 것이다.

또한 그림 2에서 최적 패칭 윈도우의 크기가 클라이언트의 버퍼 크기 B_C를 초과하므로 프록시 서버는 클라이언트의 버퍼를 확장하기 위하여 $PW_{opt} - B_C$의 버퍼공간을 할당한다. 즉, 프록시는 클라이언트의 크기가 패칭 윈도우 크기보다 작은 경우, 이를 수용하기 위하여 클라이언트의 모자라는 분량만큼 버퍼를 할당한다. 또한 이러한 프록시의 버퍼는 패칭 윈도우에 속한 클라이언트들에 의하여 공유된다. 확장된 최적 패칭 윈도우를 수용하기 위하여 할당된 프록시 버퍼는 버퍼가 가득 찬 후 FIFO 방식으로 관리된다.

3.3 VCR 기능 지원

본 논문에서 제안한 클라이언트 버퍼 확장 기법의 확장된 버퍼 공간에 의해 클라이언트에게 VCR 기능의 기본이 되는 이동 연산을 제공할 수 있다. VCR 기능 중 전진 이동, 후진 이동 및 일시 정지/재시작을 빠르게 수행할 수 있도록 버퍼에 저장된 스트림을 이용한다. 클라이언트 버퍼 공간과 프락시의 버퍼 공간으로 확장하여 패칭 윈도우 크기로 사용하므로 일정 구간으로의 이동을 효율적으로 제어할 수 있다. 각 클라이언트의 요청 시간과 현재 재생 위치와 이동하여야 할 위치 값이 프록시 서버에 관리되며, 아울러 패칭 윈도우안의 첫 번째 클라이언트의 재생 위치 및 패칭 윈도우 시작 시간과 종료 시간도 함께 기록된다. 이동 처리는 패칭 윈도우 안에서 수행되며 패칭 윈도우를 벗어나는 경우는 다른 패칭 윈도우 그룹에 포함되도록 요청을 넘김으로써 비디오 스트림의 이동 기능을 계속 제공할 수 있도록 한다.

재생 위치 이동의 종류 중 전진 이동에 대한 요청 처리 예를 그림 3에서 보인다. 비디오 스트림을 앞으로 건너뛰어 그 부분부터 보고자하는 경우이다. 패칭 윈도우는 클라이언트 버퍼 크기 B_C와 프록시 버퍼 크기 B_P로 확장되었다. 클라이언트 C_0, C_1, C_2는 같은 비디오를 보기 위해 동일한 패칭 윈도우 그룹에 속해 있다. 이 패칭 윈도우의 시작 시간은 t_s이며 종료 시간은 t_e이다. 클라이언트 C_0는 정규 스트림을 전송하는 패칭 윈도우의 첫 클라이언트가 되고 현재 정규 스트림이 전송되는 위치는 trs시간이 된다. 클라이언트 C_1은 tC_1a 시간에 도착하여 비디오를 요청하여 비디오의 처음부터 재생해 보면서 자신의 버퍼에 정규 스트림을 저장하였으므로 현재 자신의 버퍼에 저장된 스트림을 즉시 재생해 보고 있다. C_1은 현재 재생 위치 tC_1d에서 ①의 위치로 이동을 요청하였으므로 ①의 위치는 실제 클라이언트 버퍼에 저장된 스트림의 tC_1j 위치가 된다. 따라서 즉시 이동 처리를 제공하여 스트림이 재생될 수 있다. 클라이언트 C_2는 $B_C + tC_2a$ 시간에 도착하여 비디오를 요청하였으므로 비디오의 처음부터 재생해 보면서 자신의 버퍼와 프록시 버퍼에 정규 스트림을 저장해 나갈 수 있다. 프리픽스를 전부 재생하고 나면 버퍼에 버퍼링된 스트림을 재생하게 된다. 그런데, 현재 시점에서 C_2는 tC_2d에서 스트림을 재생해 보다가 재생 위치를 이동하고자 요청 하였으며 그 위치는 ②가 된다. 이 위치는 실제 C_2의 버퍼에 저장된 스트림의 위치 tC_2j가 된다.

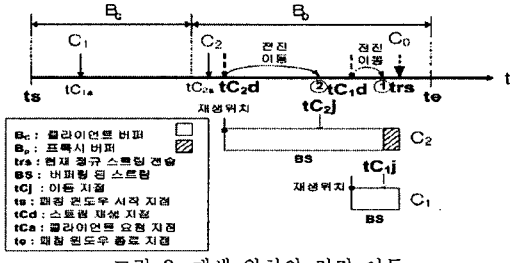

그림 3. 재생 위치의 전진 이동

그림 4는 후진 이동 기능의 예를 나타낸다. 전진 이동의 경우와 같은 환경이라고 가정한다. 클라이언트 C_1이나 C_2는 이전 스트림을 재생한 이후이므로 다시 패칭 채널을 통해 tC_1j나 tC_2j 위치부터 전송받아 재생해 보게 된다. 이때에도 정규 스트림은 자신의 버퍼와 프록시 버퍼로 확장되어 버퍼링이 계속 수행될 것이다. 전진과 후진 이동의 경우 정규 스트림을 현재 패칭 윈도우에서 전송받지 못할 경우 요청은 다른 패칭 윈도우 그룹에 포함되어 계속 서비스를 받게 된다.

VCR 기능중 일시 정지는 일정 시간 후에 재생을 계속한다는 가정 하에 잠시 멈추는 것으로 전송자체를 중단하는 것은 아니다. 따라서 언제든지 계속 서비스할 수 있도록 정규 스트림의 버퍼링

은 계속된다. 일시 정지된 시점에 대한 위치 정보는 프록시에 전송되어 재시작(resume)시 활용하도록 한다. 그러나 패칭 윈도우가 닫힐 때까지 재시작 요청이 없으면 완전 중단하고 나간 것으로 간주한다.

기존의 패칭 기법에서는 VCR 기능의 일부인 재생 위치 이동 기능을 수행하고자 할 때 패칭 윈도우 크기가 클라이언트의 버퍼 크기로 제한되어 서비스를 제공할 수 있는 범위가 한정되었다. 이에 반하여 제안된 기법에 의한 위치 이동 기능 수행 시 버퍼가 확장되었고 프리픽스도 프록시로부터 즉시 전송받으므로 더욱 빠르게 제어할 수 있다.

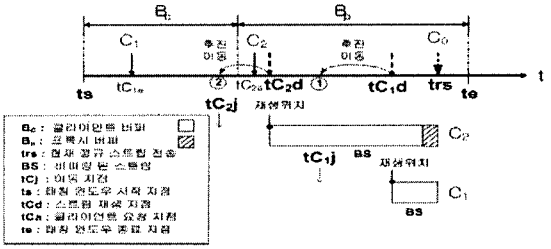

그림 4. 재생 위치의 후진 이동

4. 성능평가

클라이언트의 요청률은 평균 도착률 λ의 포아송 분포로 발생한다고 가정한다. t시간 동안 요청이 도착할 확률은 $\frac{(\lambda t)^k e^{-\lambda t}}{k!}$일 때 표 1의 파라미터를 이용하여 실험했다. 성능 분석은 버퍼 크기, 프리픽스 크기, 평균 도착 시간 등의 파라미터 값을 다양하게 변화시켰을 때 평균 서버 대역폭 요구량을 측정하여 분석한다.

표 1. 시뮬레이션 파라미터

Parameter	Default	Variation
비디오의 개수	1	N/A
비디오의 재생률 b(Mbps)	1.5	N/A
클라이언트의 버퍼 크기 B_C(minutes)	15	N/A
프록시의 버퍼 크기 B_P(minutes)	20	0-60
프리픽스 크기 V_P(minutes)	10	0-30
평균 요청 간격 1/λ (seconds)	50	5-95
비디오의 길이 μ (minutes)	120	N/A

그림 5에서, 프리픽스 크기가 각각 0분, 5분, 10분, 20분일 때 평균 서버 대역폭 요구량과 총 버퍼 크기의 효과를 보인다. 기존의 패칭 기법에서 최적의 패칭 윈도우 크기만큼 클라이언트 버퍼를 가지고 있을 때 요구되는 대역폭과 제안된 기법의 경우 총 사용 버퍼량은 프록시 버퍼까지 확장될 때 총 버퍼 크기 증가에 따른 평균 대역폭 요구량을 측정하였다. 총 버퍼 크기가 증가해도 최적의 패칭 윈도우 크기를 기점으로 최소가 되는 최적 패칭 윈도우 크기를 적용하므로 서버의 대역폭이 증가하지 않는다. 또한 프리픽스의 크기가 클수록 서버의 대역폭 요구량이 줄어든다.

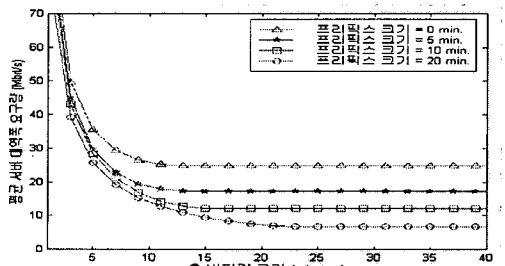

그림 5. 총 버퍼 크기의 효과

그림 6은 20초, 50초, 90초의 요청 간격과 함께 사용 버퍼에 따라 프리픽스 크기 증가시 각 경우의 총 버퍼 요구량을 구하였다. 사용 버퍼는 클라이언트 버퍼만을 사용하는 경우와 프록시와 클라이언트 버퍼를 모두 사용하는 경우로 비교해 본다. 점선으로 표시되는 부분들은 프리픽스 크기에 따라 요구되는 프록시와 클라이언트의 버퍼 총량을 나타낸다. 각 클라이언트의 버퍼 크기를 고정시킨 상태에서 프록시 버퍼를 공유하므로 프리픽스 크기가 계속 증가하여도 버퍼의 총 요구량은 크게 증가하지 않는다. 이에 반해 실선은 요청 간격 시간의 변화를 가질 때 클라이언트 버퍼만을 사용하는 경우의 총 버퍼 요구량을 나타낸다. 이 경우, 성능을 최적화하기 위해 각 클라이언트 버퍼는 급속히 증가함을 볼 수 있다. 즉 기존의 패칭 기법에서 클라이언트 버퍼로 제한된 경우 빈번하게 요청이 들어올 때 필요로 하는 버퍼량이 기하급수적으로 증가함을 보인다.

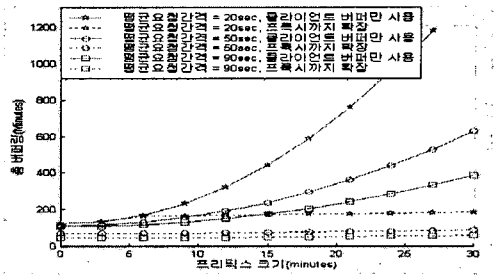

그림 6. 프리픽스 크기에 따른 총 버퍼 요구량

5. 결 론

본 논문에서는 패칭 기법을 이용, VOD 시스템의 성능을 향상시키기 위하여 프록시 프리픽스 캐싱과 클라이언트 버퍼 확장 기술을 이용한 패칭의 성능 향상 기법을 제안하였다. 기존 패칭 기법에서는 패칭 윈도우 크기는 클라이언트의 버퍼 크기의 제한을 받기 때문에 패칭의 효과가 클라이언트의 버퍼 크기에 한정되므로 최적의 패칭 윈도우 크기로 늘릴 수가 없다. 그러므로 제안된 기법은 프록시 안의 프리픽스 크기가 증가됨에 따라 최적의 패칭 윈도우 크기도 늘어난다. 즉, 클라이언트 버퍼보다 더 많은 스트림을 공유할 수 있도록 클라이언트 버퍼 공간을 프록시 버퍼로 확장하였다. 결과적으로 제안된 기법은 성능평가를 통하여 프리픽스 캐싱의 활용이 기존의 방법보다 평균 서버 대역폭이 줄어드는 것을 보였다.

[참고문헌]

[1] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A Multicast Technique for True Video-on-Demand Services", In Proc. of ACM Multimedia '98, Bristol, U.K., Sep. 1998.

[2.] Y. Cai, K. A. Hua and K. Vu, "Optimizing Patching Performance", In Proc. of SPIE's Conference on Multimedia Computing and Networking '99, San Jose, Jan. 1999.

[3] S. Sen, L. Gao, J. Rexford, and D. Towsley, "Optimal Patching Schemes for Efficient Multimedia Streaming", In Proc. NOSSDAV '99, Basking Ridge, NJ, June 1999.

[4] C. Venkatramani, O. Verscheure, P. Frossard, and K. W. Lee, "Optimal Proxy Management for Multimedia Streaming in Content Distribution Networks", NOSSDAV '02, Miami, FL, May 2002.

[5] B. Wang, S. Sen, M. Adler, and D. Towsley, "Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution", In Proc. of the IEEE Infocom, Vol. 3, New York, NY, June 2002.

[6] O. Verscheure, C. Venkatramani, P. Frossard, and L. Amini, "Joint Server Scheduling and Proxy Caching for Video Delivery," In Proc. of Sixth International Workshop on Web Caching and Content Distribution, Boston, MA, May 2001.

[7] P. Frossard and O. Verscheure, "Batched Patch Caching for Streaming Media", IEEE Communications Letters, Vol. 6, No. 4, April 2002.

[8] S. Sen, J. Rexford, and D. Towsley, "Proxy Prefix caching for Multimedia Streams", In Proc. of the IEEE Infocom, Vol. 3, 1998.