산업기술연구(강원대학교 산업기술연구소 논문집), 제24권 B호, 2004.
Journal of Industrial Technology, Kangwon Natl. Univ., Korea, No. 24 B, 2004.

VOD 시스템 상에서 P2P 프록시 기반의 패칭기법

A P2P Proxy Patching Scheme on VOD System

권 춘 자* 최 치 규** 최 황 규***
Kwon, Chun-Ja Choi, Chi-Kyu Choi, Hwang-Kyu

Abstract

The main bottleneck for a VOD system is bandwidth of storage or network I/O due to the high bandwidth requirements and long-lived nature of digital video. Patching is one of the most efficient techniques to overcome the bottleneck of the VOD system through the use of multicast scheme. In this paper, we propose a new patching scheme, P2P proxy patching, for improving the typical patching technique by jointly using the proxy prefix caching scheme and the P2P proxy. In our proposed scheme, each client play a role in a proxy for multicasting a regular stream to other clients that request the same stream. Due the use of the P2P proxy and the prefix caching, the server bandwidth is required significantly less than that of the typical patching technique. In the performance study, we show that our patching scheme can reduce the server bandwidth requirements compared with the existing patching techniques.

키워드 : *멀티캐스팅, Prefix 캐싱, P2P, 프록시, VOD*
Keywords : *multicasting, prefix cahcing, peer-to-peer, proxy, VOD*

1. 서론

최근 인터넷의 급속한 보급과 디지털 미디어의 제작 기술의 발전으로 실시간 VOD(Video On Demand)의 수요가 증가하고 있다. 또한 온라인 교육이나 영화 등 사용자의 다양한 요구에 의하여 인터넷을 통한 미디어의 실시간 스트리밍 기술의 응용분야가 점점 넓어지고 있다. 그러나 멀티미디어의 실시간 스트리밍은 대용량의 멀티미디어 저작물에 대한 연속적인 전송을 요구로 하므로 많은 양의 네트워크 대역폭을 지속적으로 요구한다. 이러한 문제로 인하여 서버-클라이언트의 중앙집중식 VOD 시스템은 서버에 부하가 집중되므로 다수의 사용자에 대한 서비스의 한계를 지닌다. 따라서 VOD 시스템의 전체 성능을 높이고 서버의 부하를 줄이기 위한 연구가 활발히 이루어졌다.

이와 같이 서버의 부하를 줄이기 위하여 기본적으로 사용자의 요청에 의하여 생성되는 미디어 스트림을 효율적으로 스케줄링 하는 방법이 필요하다. 더불어 브로드캐스팅과 멀티캐스팅과 같은 기법을 활용하여 미디어 스트림을 할당하고 스케줄링을 통하여 시스템의 성능을 높이는 연구가 활발히 이루어졌다[13][14]. 특히 멀티캐스팅을 통한 패칭(patching) 기법은 멀티캐스팅을 활용하는 방식으로서 하나의 정규채널에 대하여 추가적인 요청이 들어올 때, 패칭 채널을 통하여 지나간 부분(Prefix)만을 전송함으로서 서버의 대역폭 요구를 감소시키는데 탁월한 성과를 보였다[1][2]. 즉, 클라이언트가 기존의 정규채널로부터 멀티캐스팅을 통하여 진행 중인 스트림을 받아 버퍼링하는 동시에 패칭 채널로부터 지나간 부분만을 받아 재생하므로 초기 지연시간이 없이도 대

* 강원대학교 대학원 컴퓨터정보통신공학과 박사과정
** 강원대학교 대학원 컴퓨터정보통신공학과 석사과정
*** 강원대학교 전기전자정보통신공학부 교수, 공학박사

본 논문은 2004년도 정보통신부 지원 기초기술연구지원사업 연구결과의 일부임

역폭의 요구량을 현저히 줄일 수 있다. 또, 최근에는 패칭을 통한 VOD 시스템의 성능 향상을 위하여 프록시 서버를 함께 활용하는 기법에 대한 연구도 이루어졌다[15][16]. 그러나 이와 같은 패칭 기법을 적용한다고 하더라도 하나의 패칭 윈도우마다 서버와 클라이언트 사이의 새로운 통신 채널을 생성해야 하므로 VOD 서버에 대한 부하의 집중으로 인한 대역폭의 한계는 여전히 존재한다.

서버로 집중되는 부하를 줄이기 위한 또 다른 방법으로써 프록시(proxy) 서버를 이용하여 VOD 서버의 역할 중 일부를 대신하도록 하는 기법에 대한 연구도 활발히 이루어졌다[3][4][5][6][7]. 즉, 서버에 저장된 전체 비디오 중 앞 일부분(prefix)을 프록시 서버에 저장하여 VOD 서버의 부하를 덜어주는 방식이다. 그러나 프록시 서버 역시 버퍼 용량에 한계가 있으므로 서버에 대한 부하의 집중을 궁극적으로 해결하지는 못한다.

위와 같은 중앙 집중 방식의 한계를 극복하는 분산 환경의 시스템을 구축하기 위하여 클라이언트 시스템을 통하여 미디어 스트림을 캐싱하는 P2P(Peer-to-Peer) 기반의 미디어 스트리밍 기법에 대한 연구도 진행되었다[8][9][10][11]. 이러한 P2P 미디어 스트리밍 방식은 WAN 환경에 속한 다수의 피어로 이루어진 시스템들을 트리 또는 선형 구조로 구성한다. 또한 불규칙한 각각의 피어들의 행동에 따라 전체 시스템의 구조를 재구성하여야 하고 새로운 피어를 선택하여 미디어 스트림을 복구하여야 한다. 때문에 P2P 미디어 스트리밍 기법은 전체 시스템의 구조를 유지하기 위하여 피어들 간에 메시지를 교환해야 하는 추가적인 부담이 따르며 그에 따른 스트림의 전송 지연도 발생한다.

따라서 본 논문은 프록시 서버의 prefix 패칭 기법을 적용하는 서버-프록시-클라이언트로 이루어진 VOD 시스템 상에서 정규 채널을 생성하는 서버에 대한 부하의 집중을 줄이기 위하여 클라이언트 시스템의 버퍼링과 재전송을 통한 정규 스트림의 캐싱 기법인 P2P 프록시 패칭 기법을 제안하였다. 즉, 본 논문에서 제안하는 P2P 프록시 패칭 기법은 패칭 윈도우의 크기를 벗어나는 요청에 대하여 VOD 서버가 새로운 정규채널을 생성하는 대신 이전의 패칭 그룹에 속한 클라이언트로부터 정규 스트림을 전송받아 사용하여 서버로 집중되는 부하를 줄이는 기법이다. 또한 본 논문에서 제안하는 기법은 기존의 P2P 기법이 WAN 환경 상에서 멀티캐스트 트리 구조를 유지하기 위하여 여러 차례에 걸쳐 메시지를 교환하는 오버헤드의 단점을 줄이고자 하였다.

이를 위하여 본 논문은 프록시 서버를 기반으로 한 LAN 환경 내에서 클라이언트를 패칭 윈도우 단위로 묶어 패칭 그룹에 대한 선형 구조로 구성하였다. 또, 프록시 서버 내에 클라이언트의 상태 정보를 유지하는 인덱스를 유지하도록 하였다. 즉, 본 논문에서 제안하는 기법은 클라이언트의 상태 변화에 따른 시스템의 구조를 재구성하기 위하여 한 번의 메시지를 이용하여 프록시의 인덱스를 갱신한다. 더불어 본 논문에서 제안된 기법이 불안정한 클라이언트 상에서 정규 스트림을 재전송 수행하므로 클라이언트의 이탈로 인한 스트림의 복구과정이 필요하다. 본 논문에서는 스트림의 복구를 위한 각 클라이언트 시스템의 버퍼링 과정과 전송 중인 클라이언트의 이탈에 따른 복구과정을 함께 제안하였다. 마지막으로 성능 평가를 위하여 시뮬레이션 모델을 제안하고 prefix 크기와 비디오의 접근 패턴, 그리고 평균 요청 간격에 따른 서버의 대역폭 요구량을 측정하여 기존의 패칭 기법과 비교하였다.

본 논문은 2장에서 제안된 기법과 관련된 기존의 연구에 대하여 알아보고, 3장에서는 제안된 기법에 대한 자세한 설명이 이루어진다. 그리고 4장에는 제안된 기법과 기존의 패칭 기법에 대한 실험 결과를 비교가 이루어지며, 마지막으로 5장에서 결론을 도출하고자 한다.

2. 관련연구

VOD 시스템은 다수의 사용자에게 대용량 멀티미디어 데이터의 연속적인 스트리밍 서비스를 제공하여야 하므로 고성능의 서버 시스템을 필요로 한다. 그러나 급속히 증가하는 사용자의 수요를 고려할 때 고성능의 서버라고 할지라도 사용자의 증가에 따른 성능의 한계를 피할 수 없다. 따라서 서버로 집중되는 부하를 줄이기 위한 연구가 꾸준히 진행되어 왔으며, 주로 멀티캐스팅을 이용한 패칭 기법과 원거리의 서버로부터 일부의 데이터를 가져와 서버의 역할을 부담하는 프록시 서버를 활용하는 기법, 그리고 최근 들어 활발히 연구가 이루어지는 클라이언트 시스템을 활용하여 서버의 부하를 줄이는 P2P 미디어 스트리밍 기법에 대한 연구 분야로 나눌 수 있다. 각 연구 분야에 대한 내용은 다음과 같이 요약된다.

2.1 멀티캐스트 패칭 기법

멀티캐스트를 통한 미디어의 전송은 다수의 클라이언트에게 연속 매체를 제공하는 VOD 서버의 부하를 줄이기 위한 효과적인 기법이다. 그 중 대표적인 기법은 패칭으로서 멀티캐스팅 되는 정규 채널과 클라이언트의 요청에 대한 시간적인 차이를 클라이언트 시스템의 버퍼링을 통하여 보완하는 방식이다[1][2][15][16]. 즉, 정규 채널의 진행 중에 클라이언트의 요청이 들어오면 클라이언트 시스템이 정규 채널을 저장하는 동시에 서버로부터 지나간 앞부분(prefix) 스트림을 별도의 채널을 통해 전송받아 재생하는 방식이다. 따라서 이러

한 패칭 기법을 이용한다면 클라이언트의 요청에 대하여 초기 지연 시간 없이 prefix 부분에 대한 스트림만을 추가로 전송하므로 서버의 부하를 크게 줄일 수 있다.

이러한 패칭 기법은 크게 Greedy Patching, Grace Patching, Optimal Patching의 세 가지로 구분할 수 있으며 그 기준은 패칭 윈도우의 크기가 된다[2]. 예를 들어, Greedy Patching 기법은 비디오 스트림의 길이 전체를 패칭 윈도우의 크기로 하여 동일한 비디오 파일에 대하여 하나의 정규 채널만을 생성한다. 또한 Grace Patching 기법은 클라이언트 시스템의 버퍼 용량을 패칭 윈도우 크기로 정의한다. 따라서 최근에 생성된 정규 스트림과의 시간적인 차이가 클라이언트의 버퍼가 수용할 수 있는 범위를 벗어나면 새로운 정규 채널이 생성된다. 마지막으로 Optimal Patching 기법은 서버의 대역폭 요구량이 최소가 되는 패칭 윈도우의 크기를 Optimal Patching Window 크기로 정의한다. 따라서 Optimal Patching 기법은 서버의 대역폭 요구량이 최소가 되는 지점에서 새로운 정규 채널을 생성한다.

앞서 기술한 바와 같이 패칭 기법은 클라이언트 시스템이 버퍼링을 수행하므로 클라이언트 시스템의 버퍼 용량에 대한 제약이 따르며 일반적으로 Grace Patching이 가장 현실적인 기법이라고 할 수 있다. 때문에 패칭 기법을 적용한 VOD 시스템은 패칭 윈도우 크기의 시간마다 새로운 정규 채널을 생성하여야 하며 이것은 서버의 대한 부하로 작용한다.

2.2 프록시 캐싱 기법

실시간 스트리밍을 위한 VOD 시스템은 WAN 환경으로 연결되어 있으므로 원거리의 VOD 서버로부터의 미디어의 전송은 지연이 존재한다. 따라서 네트워크 거리가 가까운 LAN 환경에서 서버의 데이터를 캐싱하여 서비스하는 프록시 서버에 대한 연구가 진행되었다. 또한 이러한 프록시 서버는 캐싱을 통하여 서버 기능의 일부를 대신하므로 서버에 집중되는 부하를 분산하는 효과도 갖는다.

위와 같은 장점으로 인하여 Proxy Prefix Caching 기법과 같이 프록시 서버에 비디오 파일의 prefix를 저장해 두어 초기 지연시간을 줄이는 기법이 제안되었다[3][4]. 또한 Proxy Prefix Caching 기법의 최적화를 통하여 네트워크의 대역폭 사용 비용을 최소화 하는 기법에 대한 연구도 진행되었다[5][6]. 최근에는 프록시 서버를 Catching Agent로 활용하여 하나의 프록시가 다른 프록시 서버에게 미디어 스트림의 일부 또는 전체를 전송하여 서버의 역할을 대신하는 기법도 제안되었다[7].

그러나 위와 같은 프록시 캐싱 기법을 사용한다고 하더라도 프록시 서버의 버퍼 공간이 제한되어 있으므로 프록시 서버가 캐싱할 수 있는 비디오 파일의 크기는 제한적이다. 때문에 다수의 사용자에게 대용량의 스트림을 제공하는 VOD 서버에 대한 부하의 집중이 여전히 존재한다.

2.3 P2P 미디어 스트리밍 기법

프록시 서버가 안정적인 서버 시스템을 가상의 서버로 사용하는 것과는 달리 P2P 미디어 스트리밍 기법은 클라이언트 시스템을 통하여 미디어 스트림을 캐싱한다. 사용자의 수가 서버의 부하로 작용하는 중앙집중 방식의 서버-클라이언트 구조와는 달리 P2P 스트리밍 기법의 장점은 사용자의 수가 증가함에 따라 전체 시스템의 용량이 증가하는데 있다. 그러나 불안정한 클라이언트 시스템을 캐싱 서버로 활용하므로 클라이언트의 이탈에 따른 복구과정이 필요하다. 또한 시스템의 구조를 유지하기 위한 메시지 전송에 대한 오버헤드도 필요하다.

먼저 Chaining 기법은 다수의 클라이언트를 선형 구조로 연결한 후 버퍼링을 통하여 다른 클라이언트에 재전송하는 방식이다[9] 또한 ZIGZAG 기법과 같이 전체 시스템의 피어(peer)들을 멀티캐스트 트리 구조로 구성하여 피어들을 통한 미디어의 전송을 수행한다[11]. 그러나 이러한 기법은 트리구조를 유지하기 위하여 매번 클라이언트의 상태가 변화에 의하여 메시지를 교환하는 오버헤드가 따른다. 또한 미디어 스트림이 여러 클라이언트를 거쳐 전송되므로 스트리밍의 전송에 대한 지연이 존재한다. 때문에 인터넷상의 캐싱 시스템들 간의 네트워크 대역폭 소모를 줄여 성능을 개선하기 위한 연구도 진행되었다[10].

이와는 별도로 기존의 패칭 기법과 P2P 미디어 스트리밍을 응용하여 패칭 기법 상에서 prefix의 전송을 위하여 클라이언트로부터 캐싱을 수행하는 P2CAST 기법도 제안되었다. 그러나 이러한 기법 역시 서버가 정규 채널을 주기적으로 생성하므로 정규 스트림의 생성에 대한 부하가 요구된다[11].

따라서 본 논문에서는 Proxy Prefix Patching 기법의 VOD 시스템 상에서 클라이언트 시스템의 버퍼링과 재전송을 통한 정규 스트림의 캐싱 기법을 제안하여 서버로 집중되는 부하를 줄이고자 하였다.

3. P2P 프록시 패칭 기법

3.1 개 요

본 논문은 프록시 서버의 prefix 캐싱을 활용하는

패칭 기반의 VOD 시스템 상에서 VOD 서버에 의하여 제공되는 정규채널에 의한 서버의 부하를 줄이기 위하여 클라이언트 시스템상의 버퍼링을 통한 지연 전송 방식을 이용하는 P2P 프록시 패칭 기법을 제안한다. 먼저, 제안된 기법의 시스템 구성은 그림 1과 같이 인기도가 높은 비디오 파일의 앞부분(prefix)을 저장하는 프록시 서버와 그 뒷부분(suffix)를 제공하는 VOD 서버, 그리고 자신과 가장 가까운 프록시 서버에 연결되어 버퍼링을 수행하는 다수의 클라이언트 시스템으로 구성된다.

일반적으로 Proxy Prefix 패칭 시스템 상에서 VOD 서버는 prefix 크기를 초과하는 요청에 대하여 새로운 정규채널을 생성하여야 하며, 요청 빈도가 높은 비디오의 경우는 주기적으로 정규채널을 생성하여야 하는 부하가 요구된다. 따라서 본 논문은 서버의 빈번한 정규채널의 생성으로 인한 대역폭의 사용량을 줄이기 위하여 클라이언트 시스템에 기존의 정규 스트림을 버퍼링 하였다가 패칭 윈도우 이후의 요청을 하는 클라이언트에게 전송하는 P2P 프록시 패칭 기법을 제안한다.

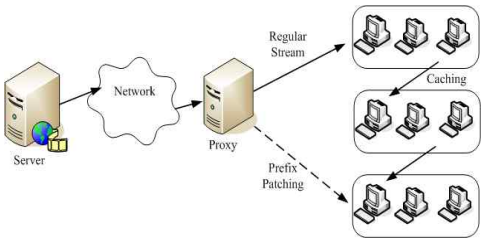


그림 1 P2P 프록시 패칭 기법의 시스템 구성

기존의 Grace Patching 기법을 적용하는 VOD 시스템 상에서 패칭 윈도우 크기는 클라이언트 시스템의 버퍼 크기와 일치하고 정규 스트림이 진행되는 동안 정규 스트림과 근접한 데이터가 클라이언트의 버퍼에 저장된다. 본 논문은 이점에 착안하여 패칭 윈도우를 벗어나는 새로운 요청이 들어온 경우 서버로부터 새로운 정규 채널을 생성하는 대신 이전의 패칭 윈도우에 속하는 클라이언트 시스템에 버퍼링된 정규채널의 비디오 데이터를 재전송 받아 사용하도록 한다.

즉, 하나의 프록시 서버에 연결된 다수의 클라이언트들이 특정 비디오에 대한 일련의 요청을 수행할 때, 각각의 요청에 대한 시간적인 순서에 의하여 패칭 윈도우를 단위로 하는 클라이언트의 그룹이 형성되며 이러한 그룹은 연속적으로 생성될 수가 있다. 이렇게 패칭 그룹이 연속적으로 생성되는 경우 기존의 패칭 기법은 패칭 윈도우 크기마다 하나의 정규채널을 생성하는 것에 반하여, 제안된 기법은 클라이언트의 버퍼링과 지연전송을 활용한 정규 스트림의 캐싱을 활용하여 연속적인 패칭 그룹이 지속되는 동안 서버로 부터의 정규 채널은 생성하지 않으며 대신 이전의 패칭 그룹에 속한 클라이언트에 버퍼링된 스트림을 전송받으므로 빈번한 정규채널의 생성으로 인한 서버의 부하를 줄일 수 있다. 다만 안정적인 서버가 아닌 불규칙한 클라이언트로부터 스트림을 제공받는 것이므로 클라이언트의 이탈에 따른 복구 방법과 동적으로 움직이는 다수의 클라이언트들에 대한 정보의 관리가 필요하다.

본 논문에서 제안하는 기법의 세부적인 내용은 정규 스트림의 캐싱을 위한 프록시 서버의 역할, 새로운 클라이언트의 참여에 대한 방법, 클라이언트의 버퍼링을 통한 재전송 과정, 그리고 클라이언트의 불규칙한 행동에 의한 이탈에 따른 복구 방법 등으로 이루어지며 상세한 수행과정은 다음과 같다.

3.2 프록시 서버를 이용한 인덱스 구성

일반적으로 프록시 서버를 활용하는 VOD 시스템의 경우 클라이언트로 향하는 스트림은 프록시 서버를 경유하여 전송된다. 즉, 프록시 서버는 클라이언트와의 세션에 대한 정보를 유지하고 상태에 대한 변화를 갱신하는 역할을 수행하여야 한다. 따라서 불안정한 클라이언트 시스템을 정규 스트림의 패칭 서버로 이용하는 본 논문에서는 클라이언트 시스템의 불규칙적인 행동으로 인한 대체 선택과정을 수행하기 위한 인덱스로서 프록시 서버에 유지되는 정보를 이용한다. 즉, 프록시 서버는 prefix의 패칭 스트림을 제공하는 역할과 더불어 인덱스 서버로서의 역할도 함께 수행한다. 이러한 인덱스는 기존의 프록시 서버가 스트림의 전송을 위하여 유지하는 각각의 클라이언트의 연결에 대한 세션 정보를 패칭 윈도우와 그에 속한 클라이언트의 리스트로 재구성 한 것이다. 그림 2는 자신과 연결된 모든 클라이언트들에 대한 세션 정보를 유지·관리하며 전송중인 클라이언트가 이탈한 경우 이를 대체하는 새로운 클라이언트를 선택하는 과정을 수행하기 위한 프록시상의 인덱스 구조를 나타낸 것이다.

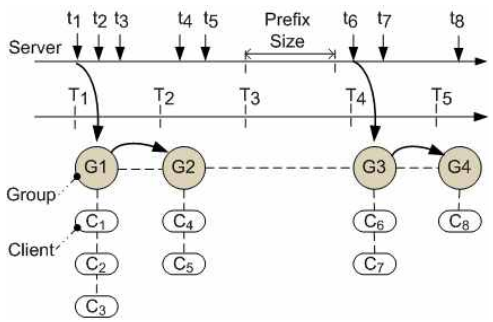


그림 2 P2P 프록시 패칭을 위한 인덱스 구성

그림 2는 시간의 변화에 따른 클라이언트의 요청과 프록시 서버에 형성되는 인덱스의 구조를 나타낸다. 여기에서 prefix 크기를 단위로 하는 그룹은 그 순서에 따른 리스트를 형성하며, 각각의 그룹은 prefix 크기와 동일한 패칭 윈도우에 속하는 클라이언트의 리스트로 이루어진다. 그룹에 대한 정보는 그룹의 시작 시간과 더불어 최소한 하나 이상의 클라이언트 인덱스를 포함하며 클라이언트 인덱스는 현재 재생중인 클라이언트를 가리킨다. 또한 각각의 그룹에 대한 정규 스트림이 서버로부터 전송되는 것인지 이전에 형성된 클라이언트의 그룹에 의하여 캐싱된 것인지에 대한 정보도 포함하여야 한다.

그림 2에서 G1은 서버로부터 전송된 정규 스트림이며 G2의 정규 스트림은 G1에 속하는 클라이언트의 버퍼에 저장되었다가 전송된다. 이 경우 G2의 클라이언트는 프록시 서버로부터 prefix를 패칭한다. 또한 G3는 이전의 그룹과의 시간적인 차이가 prefix 크기보다 크므로 G2의 클라이언트에서 캐싱할 수 있는 범위를 벗어나므로 서버로부터 다시 정규 스트림을 전송받는다. 마지막으로 G3와 G4에 대한 시간간격은 prefix 크기보다 작으므로 G4는 서버로부터 추가적인 정규 채널을 요구하지 않고 G3의 클라이언트로부터 캐싱된 정규스트림을 사용하는 것을 보여준다.

이와 같이 제안한 기법은 프록시 서버에서 클라이언트의 상태 정보를 기록하는 인덱스를 유지한다. 따라서 각각의 클라이언트가 자신이 원하는 요청을 프록시에 전달하면 프록시 서버는 클라이언트의 요청을 수행하는 동시에 서버에 유지되는 인덱스를 갱신한다. 때문에 시스템의 구조를 유지하기 위한 메시지의 교환이 단 한번 이루어지며 ZIGZAG 기법과 같은 트리구조 상의 peer들이 여러 차례 메시지를 교환하여야 하는 오버헤드를 줄일 수 있다. 또 ZIGZAG 기법과 같이 멀티캐스트 트리를 구성하는 P2P 시스템에서는 임의의 클라이언트의 이탈에 따른 스트림의 복구를 위해서는 다수의 peer들 간에 메시지를 교환하여야 하는 오버헤드와 그동안의 스트림의 지연이 발생한다. 그에 비하여 제안한 기법은 네트워크 거리가 가까운 프록시에서 관리하므로 메시지의 교환과 스트림의 복구를 위한 지연이 매우 짧다. 또한 이렇게 프록시 서버에 유지되어야 하는 인덱스 정보는 기존의 패칭 기법을 적용하기 위하여 필요한 클라이언트와의 세션정보를 이용하여 제안된 기법에 맞게 선형 구조로 재구성한 것이므로 인덱스 정보를 유지하기 위한 프록시 서버의 오버헤드 또한 매우 적다.

3.3 새로운 클라이언트의 참여

Grace Patching 기법에서 정의되는 패칭 윈도우의 크기는 클라이언트 시스템의 버퍼 용량에 의해 결정된다. 또한, 프록시 서버에서 패칭되는 prefix의 크기와 클라이언트 시스템에 버퍼링 되는 정규 스트림의 크기는 동일하다. 즉, 하나의 패칭 윈도우에 속하는 클라이언트의 집합으로 구성되는 캐싱 그룹에 의해 버퍼링되었다가 재전송이 가능한 정규 스트림의 크기는 prefix 크기와 동일하다.

따라서 새로운 클라이언트가 특정 비디오에 대한 요청을 하는 경우 가장 최근에 생성된 그룹의 시작 시간과 현재 클라이언트의 요청 시간을 비교하여 정규 채널을 사용하는 새로운 그룹을 생성할 것인지, 혹은 최근의 그룹으로부터 캐싱된 정규 채널을 재전송 받는 그룹을 생성할 것인지, 아니면 기존의 그룹에 참여할 것인지 결정하여야 한다. 즉, 최근 그룹의 시작시간과 클라이언트의 요구 시간의 차가 prefix의 크기보다 작다면 기존의 그룹에 참여하는 과정을 수행하고, 반대로 최근 그룹의 시작 시간과 현재의 요청 시간의 차가 prefix 크기보다 크다면 새로운 패칭 그룹을 생성한다. 또한 새로운 그룹을 생성하는 경우에도 이전 그룹의 종료 시간과 현재 요청 시간을 비교하여 이전의 패칭 그룹에 캐싱된 정규 스트림을 이용할 것인지 또는 서버로부터 새로운 정규채널을 요구할 것인지 결정한다. 이렇게 결정되는 패칭 그룹과 클라이언트의 관계는 프록시 서버 상에 유지되는 인덱스 정보와 일치한다.

그림 2에서 나타난 동일 비디오에 대하여 클라이언트 C_n의 요청들이 t_n 의 시간에 이루어진다고 할 때, 클라이언트 C1의 요청시간 t1에 대하여 이전의 그룹이 존재하지 않으므로 서버에 새로운 정규 채널을 요청한 후 패칭 그룹 G1을 생성한다. 계속해서 프록시 서버의 prefix가 멀티캐스트 되는 동안 클라이언트 C2와 C3에 의하여 t2와 t3의 시간에 요청이 이루어진 경우, 프록시는 패칭을 위하여 그룹의 시작시간 T1과 각 클라이언트의 요청 시간과의 차만큼의 분량에 해당하는 prefix를 각각의 클라이언트에 전송하고 패칭 그룹에 참여시킨다. 또, prefix의 크기가 패칭 윈도우의 크기와 일치하므로 G1의 시작 시간 T1으로부터 prefix 크기의 시간이 흐른 T2가 되면 그룹에 대한 추가 작업을 종료한다. 또 G1에 대한 prefix의 멀티캐스트가 끝나며 이후의 스트림은 서버로부터 정규 채널을 통하여 멀티캐스트 된다.

위와 같이 T2의 시점이 지나면 그룹 G1이 생성되고 G1에 속하는 클라이언트들의 버퍼에 최근의 정규 스트림의 내용이 버퍼링 된다. 따라서 이 후의 요청은 이전의 그룹 G1의 클라이언트로부터 정규 스트림을 재전송 받아 사용할 수 있다. 그림 2에서와 같이 t4의 시간에 클라이언트 C4에 의하여 요청이 이루어진 경우 프록시는 G1이 종료되었으므로 새로운 그룹 G2를 생성한다. 이때 G2는 G1이 종료된 시간 T2로부터 prefix 범위 이내의 시간에 이루어졌으므로 G1의 클라이언트로부터 캐싱된 스트림을 받아 정규 채널로 이용한다. 이때 프록시는 C4에게 T2와 t4의 분량에 해당하는 prefix를 전송하며 새로운 그룹 G2를

생성한다. 또한 G2의 시작시간은 G1에 버퍼링 된 스트림을 받아 사용하는 것이므로 미디어 스트림의 연속성을 위하여 이전 그룹 G1의 종료시간인 T2가 G2의 시작 시간이 된다. 즉, G1과 G2의 클라이언트 시스템에 캐싱되는 미디어가 패칭 윈도우 크기를 단위로 연속성을 가지며 이것은 연속된 패칭 그룹에 대하여 초기에 필요한 하나의 정규 채널만을 사용하여 VOD의 서비스가 가능하도록 한다. G2는 시작 시간 T2로부터 Prefix 크기의 시간이 지난 T3에 종료된다.

반대로 이전의 그룹이 이미 생성되어 있는 경우라고 하더라도 이전 그룹의 종료 시간과 새로운 요청이 이루어진 시간 차이가 prefix 크기보다 큰 경우 프록시로부터 패칭을 통하여 시간적인 차이에 대한 패칭을 수행할 수 없으므로 이전 그룹으로부터 정규스트림을 캐싱할 수 없다. 이런 경우는 그림 2의 G3에 나타난 바와 같이 서버로부터 새롭게 정규 채널을 할당받아 사용한다. 이때 G3의 시작 시간 T4는 C6의 요청이 이루어진 t6이며 이전의 그룹과는 단절된 상태이다. 또 그룹 G4는 G3로부터 정규 스트림을 캐싱하여 사용하는 새로운 그룹이 생성된 경우를 보여준다.

위와 같은 과정을 거쳐 그룹이 생성되면 각각의 그룹은 클라이언트 중 하나를 선택하여 이후의 그룹에 대한 캐싱 서버로 활용한다. 이때 각각이 클라이언트 시스템은 정규 채널을 저장하기 위하여 버퍼링 과정을 수행하며 그에 대한 내용은 다음과 같다.

3.4 클라이언트의 버퍼링과 재전송 과정

제안된 클라이언트의 지연전송을 이용한 정규 스트림의 캐싱 기법은 프록시를 이용한 prefix 패칭 시스템 상에서 클라이언트 시스템에서 정규 스트림을 버퍼링하여 이후의 그룹에 지연전송하는 기법이다. 따라서 클라이언트 시스템은 자신의 패칭 그룹에 대한 정규 스트림을 재전송하기 패칭 윈도우의 크기에 해당하는 분량의 스트림을 버퍼링을 요구한다. 즉, 기존의 패칭 기법이 재생을 위하여 정규 스트림과 요청 시간의 차에 해당하는 분량만을 버퍼링하는 것에 비하여 제안된 기법은 재생을 위한 버퍼링과 동시에 재전송을 위한 버퍼링을 함께 수행한다.

기존의 패칭 기법은 정규스트림과의 시간 차이만큼을 재생을 위하여 버퍼링하고 재생한 스트림을 소멸시켜버리지만, 제안된 기법은 재생된 스트림이라고 하더라도 재전송을 위하여 일정시간 버퍼에 남겨둔다. 따라서 각각의 패칭 그룹에 참여하는 모든 클라이언트는 패칭 윈도우 크기 분량의 정규 스트림을 저장하였다가 그만큼의 지연을 가지고 재전송한다. 그림 3은 제안된 기법 상에서 클라이언트의 버퍼링 구조를 보여주고 있다.

그림 3에서 나타난 그룹 K와 K+1은 연속적인 선후관계를 가지는 패칭 그룹으로서 각각 자신에게 속한 클라이언트들 중 하나를 선택하여 이어지는 그룹에게 멀티캐스트를 통하여 정규스트림을 전송하도록 하고 있다. 이때 각각의 클라이언트 버퍼는 버퍼링 중인 스트림의 재생이 이루어지는 지점을 경계로 재전송을 위하여 대기 중인 버퍼와 재생을 위하여 대기 중인 버퍼의 두 부분으로 나뉜다. 그림 3에 나타난 클라이언트 시스템의 버퍼 중 세로 빗금으로 표시된 부분은 이미 재생이 이루어졌지만 재전송을 위하여 버퍼에 남아있는 부분이다. 또한 그림 3에서 가로의 빗금으로 표시된 부분은 아직 재생이 이루지지 않은 부분을 나타낸다. 또 아직 재생이 이루어지지 않은 부분의 크기는 일반적인 패칭 기법에서 재생을 위하여 버퍼링을 수행하는 정규 스트림과 요청시간의 차이 값과 일치한다.

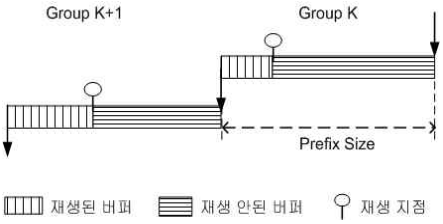


그림 3 클라이언트 시스템의 버퍼 구조

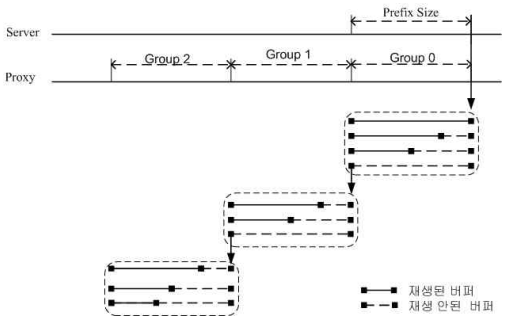


그림 4 패칭 그룹에 의한 정규 스트림 캐싱

그림 4는 각각의 패칭 그룹과 그에 속한 클라이언트들에 의하여 연속되는 그룹들간의 정규 스트림이 캐싱되는 과정을 보여준다. 여기서 하나의 그룹은 하나 이상의 클라이언트를 가지며 그 중 하나를 선택하여 이어지는 그룹에게 멀티캐스트를 통하여 정규스트림을 재전송하는 캐싱 서버의 역할을 부여한다. 또 같은 그룹에 속한 클라이언트라고 하더라도 서로의 요청시간에 따라 재생 지점과 전송을 위한 버퍼와 재생을 위한 버퍼 공간의 비율이 다르게 된다. 그러나 두 버퍼 공간의 비율과 상관없이 같은 그룹에 속한 모든 클라이언트는 prefix 크기와 동일한 분량의 정규 스트림에 대하여 버퍼링을 수행한다. 이와 같은

기법은 정규 스트림에 대한 캐싱 서버의 역할을 하는 클라이언트가 불규칙한 행동으로 인하여 그룹에서 이탈한 경우 그룹 내의 다른 클라이언트가 동일한 스트림을 버퍼링하고 있으므로 다른 클라이언트를 선택하여 캐싱 서버의 역할을 대신 수행하게 할 수 있다.

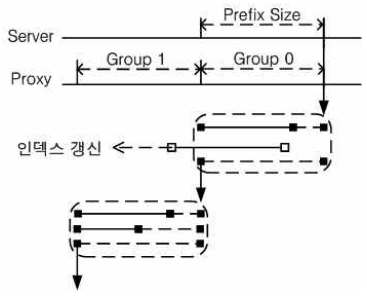
즉, 각 클라이언트의 이탈은 프록시 서버에 유지되는 인덱스 정보의 갱신을 요구한다. 또, 이렇게 유지되는 인덱스는 다음 그룹으로 정규스트림을 전송중인 클라이언트가 이탈한 경우 스트림을 복구하기 위한 대체선택과정을 수행하는데 이용된다. 이와 관련된 내용은 다음절에서 상세하게 기술한다.

3.5 클라이언트 이탈에 따른 복구

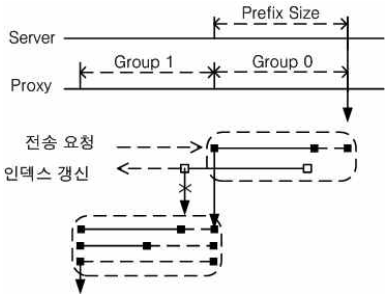
제안된 기법은 클라이언트 시스템을 정규 스트림에 대한 캐싱 서버로 활용하므로 안정적인 서버와는 달리 클라이언트의 불규칙한 행동이나 재생이 종료되어 멀티캐스팅 중인 클라이언트가 그룹에서 이탈하는 경우가 발생한다. 따라서 전송중인 클라이언트가 그룹에서 이탈하는 경우 그룹 내의 다른 클라이언트를 선택하여 정규 스트림을 전송하도록 하여야 한다. 또한 그룹 내의 모든 클라이언트가 그룹에서 이탈하여 재전송이 불가능하면 서버로부터 새로운 정규채널을 요청하고 그룹을 소멸시켜야 한다.

클라이언트의 이탈에 따른 복구 과정은 프록시 서버에 유지되는 인덱스를 이용해서 이루어진다. 즉, 각각의 클라이언트는 재생 중에 전송 그룹으로부터 이탈을 원하는 경우 프록시에 자신이 그룹을 떠남을 알린다. 이때 프록시는 이탈을 원하는 클라이언트가 재전송을 수행하는 클라이언트라면 자신의 인덱스를 검색하여 동일 그룹 내의 클라이언트를 선택하여 스트림의 전송을 대체한다. 그러나 동일한 그룹 안에 스트림의 전송을 대체할 클라이언트가 존재하지 않는 경우 해당 그룹을 소멸시키고 서버로부터 서비스 중인 비디오의 현재 시점부터 새로운 정규채널을 생성할 것을 요청한다.

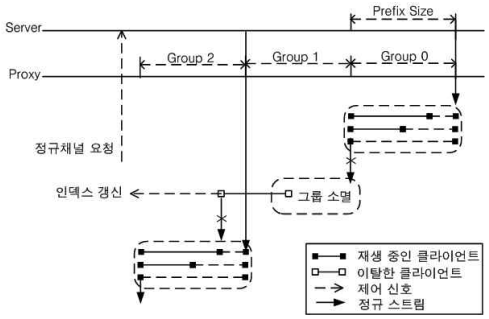
그림 5는 클라이언트의 이탈에 대한 정규 스트림의 복구 과정을 나타낸다. 그림 5의 a)는 재전송을 수행하지 않던 클라이언트가 그룹에서 이탈을 원하는 경우이다. 이탈을 원하는 클라이언트가 프록시에 이탈에 대한 신호를 보내면 프록시는 이에 대한 인덱스를 갱신하여 클라이언트의 상태와 인덱스의 내용을 동일하게 유지시킨다. 또 그림 5의 b)는 재전송을 수행중인 클라이언트가 이탈하는 경우에 대하여 그룹 내의 새로운 클라이언트를 선택하여 스트림의 전송을 대체하는 경우를 보여준다. 이때 프록시 서버는 이탈을 요청한 클라이언트의 신호에 대하여 인덱스를 갱신하고, 동시에 스트림의 전송을 대체할 클라이언트를 검색하여 스트림을 전송할 것을 요청한다. 마지막으로 그림 5의 c)는 그룹 내의 모든 클라이언트가 이탈한 경우를 나타낸다. 그림 5의 c)에서 이탈을 희망하는 클라이언트가 프록시에 이탈을 요청하면 프록시는 이탈에 대한 인덱스를 수정한다. 이때 해당하는 그룹의 클라이언트가 모두 사라졌으므로 스트림을 대체할 클라이언트를 선택할 수 없다. 이런 경우 프록시는 서버에게 비디오의 남은 분량에 대한 정규 스트림을 서버에 요청한다.



a) 재전송을 수행하지 않는 클라이언트의 이탈



b) 재전송을 수행 중인 클라이언트의 이탈



c) 클라이언트의 이탈로 인한 그룹의 소멸

그림 5 클라이언트의 이탈과 복구 과정

이와 같이 본 논문에서 제안된 기법은 빈번한 정규 채널의 생성으로 인한 서버의 부하를 줄이기 위하여 클라이언트의 재생 중에 버퍼링을 통한 재전송 기법을 제안하였다. 이를 위하여 본 논문은 프록시 서버

를 활용한 인덱스의 구성과 새로운 클라이언트의 참여 방법, 재생중의 버퍼링과 재전송 방법, 그리고 불안정한 클라이언트의 이탈과 그에 따른 복구 방법에 대한 세부적인 내용을 제시하였다. 마지막으로 본 논문은 성능 평가를 통하여 제안된 기법과 기존의 패칭 기법의 비교 결과를 보여주고자 한다.

4. 성능평가

4.1 성능 평가 모델

제안된 기법의 성능 평가를 위하여 표1과 같은 파라미터를 사용하여 기존의 패칭 기법과 제안된 기법의 성능을 비교하고자 하였다.

표 2 성능 평가를 위한 파라미터

parameter	default	variation
비디오 수	100	N/A
비디오의 길이(minutes)	90분	N/A
Mean-Inter Arrival Time $1/\lambda$ (seconds)	10	5-100
Prefix 크기(minutes)	10분	0-20
Normal Playback Rate b(Mbps)	1.5	N/A
Skew Factor	0.271	0.0-1.0
프록시 서버의 수	7	N/A
시뮬레이션 시간(hours)	10	N/A

표 1에 나타난 바와 같이 90분 분량의 비디오 파일 100개를 서비스 하는 VOD 서버와 7개의 프록시 서버를 포함하는 네트워크 모델을 가정하였다. 또한 각각의 프록시는 인기도가 높은 상위 10%의 비디오의 prefix 부분을 저장한다. 또 프록시에 저장된 비디오에 대해 패칭 기법이 적용되며 패칭 윈도우 크기는 prefix의 크기와 일치한다고 가정한다. 또, VOD 서버에 대한 클라이언트의 요청은 λ의 발생 빈도를 가지는 포아송 분포를 따른다고 가정할 때, 서버에 대한 평균 요청 간격은 $\frac{1}{\lambda}$ 은 10초를 기본 값으로 하였다. 그리고 프록시에 저장되는 비디오 파일에 대한 prefix의 크기는 10분을 기본 값으로 설정하였다. 마지막으로, 전체 비디오에 대한 요청 패턴이 Θ를 skew factor로 하는 Zipf-like 분포를 따른다고 가정할 때, 비디오 i에 대한 요청 확률 $P_i = \frac{c}{i^{(1-\theta)}}$ 의 확률을 가진다[12]. 이때 skew factor 인 Θ가 작을수록 특정 비디오에 대한 요청이 집중적으로 이루진다. 또, skew factor에 대한 기본 값은 0.271로 하였다.

성능평가는 위와 같은 조건으로 다음과 같이 세 가지 항목에 대하여 제안된 기법과 기존의 패칭 기법에 대한 서버의 대역폭 요구량을 비교하였다. 서버의 대역폭 요구량은 10시간 분량의 요청에 대하여 서버가 사용한 대역폭의 전체 소모량을 수행 시간에 대한 평균값을 구하였으며, 5회 반복한 결과의 평균으로 최종 수치를 구하였다.

4.2 성능평가결과

4.2.1 Prefix 크기에 따른 성능 비교

그림 6은 프록시 서버에 저장된 prefix의 크기에 따른 서버의 대역폭 요구량을 나타낸다. 앞의 가정에서 성능 평가를 위하여 전체 100개의 비디오들 중에서 인기도가 높은 10%의 비디오만이 프록시에서 서비스 된다고 가정하였고, 평균 요청 간격은 10초이다. 또한 프록시에 저장된 perfix의 크기는 패칭 윈도우의 크기와 일치한다. 그림 6에서 볼 수 있듯이 prefix의 크기가 증가할수록 서버의 대역폭 요구량이 감소한다. 또, 기존의 패칭 기법과 비교하여 볼 때 제안된 클라이언트 시스템을 이용한 정규 스트림의 재전송을 통한 캐싱 기법이 prefix 크기의 증가에 따른 대역폭의 감소량이 훨씬 적게 나타났다.

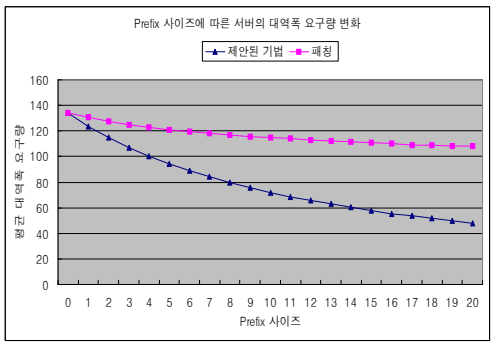


그림 6 Prefix의 크기에 따른 서버 대역폭 요구량

4.2.2 평균 요청 간격에 따른 성능 비교

그림 7은 서버에 대한 클라이언트의 평균 요청 간격 $\frac{1}{\lambda}$ 의 변화에 따른 서버의 대역폭 요구량 변화를 보인다. 이때 인기도가 높은 10%의 비디오에 대한 프록시 서버의 prefix 크기는 10분으로 하였다. 그림 7은 평균 요청 간격이 늘어남에 따라 서버의 대역폭 요구량이 줄어드는 것을 보여준다. 또한 기존의

패칭 기법보다 본 논문에서 제안한 기법이 동일한 평균 요청 간격이 같을 때 서버에 대한 대역폭 요구량이 작게 나타나며 특히 요청의 빈도가 높을수록 제안된 기법의 효과가 크게 나타남을 볼 수 있다.

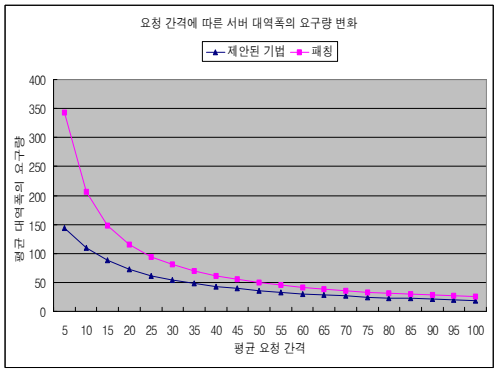


그림 7 평균 요청 간격에 따른 서버 대역폭 요구량

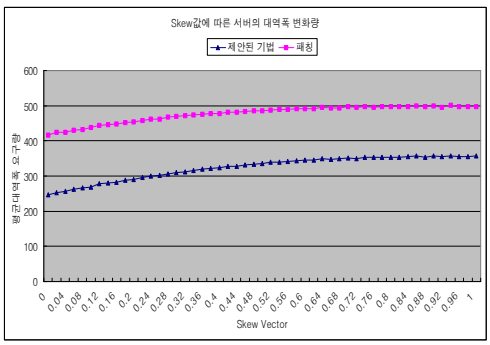


그림 8 Skew factor 값에 따른 서버 대역폭 요구량

4.2.3 요청 패턴에 따른 성능 비교

그림 8은 전체 비디오에 대한 요청 패턴이 θ를 skew factor로 하는 Zipf 분포를 따를 때 θ의 변화에 따른 대역폭 요구량의 변화를 나타낸다. 이때 θ의 범위는 0.0에서 1.0이며 θ의 값이 작을수록 특정 비디오에 대한 요청비율이 높은 것을 나타낸다. 또 θ의 값이 1인 경우는 모든 비디오에 대한 요청 확률이 같다.

그림 8에 나타난 결과에 의하면 θ의 값이 작을수록 패칭과 제안된 기법 모두에서 서버의 대역폭 요구량이 적게 나타난다. 이는 인기도가 높은 10%의 비디오만이 프록시 서버에 의하여 패칭되므로 θ값이 작아 프록시 서버에 대한 접근이 많아질수록 서버의 부하가 적어지기 때문이다. 또한 그림 8에서 제안된 기법이 기존의 패칭 기법보다 동일한 θ값에서 서버의 대역폭을 훨씬 작게 요구하는 것을 볼 수 있다.

5. 결 론

최근 급속히 증가하는 VOD 서비스는 대용량의 비디오 파일에 대한 연속적인 재생을 요구하므로 서버 측의 자원 소모량이 매우 크다. 따라서 이를 줄이기 위한 연구가 활발히 진행되었으며 멀티캐스팅과 클라이언트 시스템의 버퍼링을 활용하는 패칭 기법은 서버의 부하를 줄이는 대표적인 방법이다. 그러나 중앙 집중 방식의 서버-클라이언트 시스템의 서버는 패칭 기법을 사용한다 할지라도 패칭 윈도우에 해당하는 주기마다 새로운 정규 채널을 생성하여 비디오 스트림을 전송하여야 하므로 패칭을 통하여 서버의 부하를 줄이는 데는 한계가 있다.

위와 같은 한계를 극복하기 위하여 본 논문은 기존의 중앙 집중 방식의 서버-클라이언트 시스템과는 달리 클라이언트에 저장된 정규스트림의 캐싱을 사용하는 P2P 프록시 패칭 기법을 제안하였다. 제안된 기법은 패칭 윈도우 크기의 시간을 벗어나는 클라이언트의 요청에 대하여 서버에 새로운 정규 채널을 요청하는 대신 이전의 패칭 윈도우에 속한 클라이언트 시스템에 버퍼링된 스트림을 재전송 받아 사용하는 방식이다.

이를 위하여 본 논문에서는 프록시 서버를 prefix를 제공하는 패칭 서버로 활용하는 동시에 클라이언트의 상태를 기록하고 탐색하기 위한 인덱스 서버로 활용하였다. 또 클라이언트의 상태 정보를 유지하기 위하여 프록시에 저장되는 인덱스의 구조를 제안하였으며, 이러한 인덱스는 새로운 클라이언트의 참여와 재생중의 불규칙한 행동으로 인한 이탈에 의하여 갱신되어야 함을 보였다. 또 불안정한 클라이언트 시스템이 정규스트림을 재전송 중에 이탈하였을 때 전송중인 정규 스트림을 복구하는 방법을 보였다.

마지막으로 본 논문은 시뮬레이션을 통하여 제안된 기법의 성능을 평가하여 이를 기존의 패칭 기법과 비교하여 제안된 기법이 우수함을 보였다. 실험은 동일한 조건하에서 prefix 크기의 변화에 따른 대역폭 요구량과 평균 요청 간격의 변화에 따른 대역폭 요구량, 그리고 비디오 요청 패턴에 따른 대역폭의 요구량에 대하여 이루어 졌다. 또한, 실험 결과 모든 경우에 대하여 제안된 기법이 기존의 패칭 기법에 비하여 서버의 대역폭 요구량이 크게 줄었음을 보였다.

참 고 문 헌

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

[4] L. Zhu, Z. Sahinoglu, G. Cheng, A. Vetro, N. Ansari, H. Sun, "Proxy Caching for Video on Demand Systems in Multicast Networks", *The John Hopkins University Conference on Information Sciences and Systems (CISS)*, March 2003 (CISS 2003).

[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] K.A. Hua, S. Sheu, D.A. Tran, , "A New Caching Architecture for Efficient Video Services on the Internet", *Proceedings of IEEE Symposium on Applications and the Internet (SAINT 2003)*, Orlando, FL, Jan. 27-31, 2003.

[8] S. Sheu, K. A. Hua, and W. Tavanapong, "Chaining: A Generalized Batching Technique for Video-On-Demand Systems", *In Proc. Of IEEE Int'l Conf. On Multimedia Computing and Systems (ICMCS'97)*, pp. 110-117, Ottawa, Canada, June 1997.

[9] K. A. Hua, D. A. Tran, and R. Villafane. "Caching multicast protocol for on-demand video delivery", *In Proc. of the ACM/SPIE Conference on Multimedia Computing and Networking*, pages 2-13, San Jose, CA, January 2000.

[10] Y. Guo, K. Suh, J. Kurose, D. Towsley "P2Cast: Peer-to-peer Patching Scheme for VoD Service", *Proceedings of the 12th World Wide Web Conference (WWW-03)*, Budapest, Hungary, May 2003.

[11] D. Tran, K. Hua, and T. Do. "Zigzag: An efficient peer-to-peer scheme for media streaming", *In Proc. of IEEE INFOCOM'03*, San Francisco, CA, USA, April 2003.

[12] A. Dan, D. Sitaram, and P. Shahabuddin. "Dynamic batching policies for an on-demand video server", *Multimedia Systems*, 4(3):112-121, June 1996.

[13] H. J. Kim, and Y. Zhu, "Channel Allocation Problem in VOD System Using Both Batching and Adaptive Piggybacking", *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 969-976, 1998.

[14] D. Ghose, and H. J. Kim, "Scheduling Video Streams in Video-on-Demand: A Survey", *Multimedia Tools and Applications*, vol. 11, no. 2, pp.167-195, 2000.

[15] C. J. Kwon, C. K. Choi, and H. K. "An Improved Patching Scheme for Video-On-Demand Servers", *Proc. of the International Conf. on Parallel and Distributed Processing Techniques and Applications*, June. 2004.

[16] C. J. Kwon, C. K. Choi, and H. K. Choi. "An Efficient Patching Scheme Based on Proxy Prefix Caching and Buffer Expanding for Video-On-Demand Services", *Proc. of the ACIS 3rd ACIS International Conf. on Computer and Information Science*, Aug. 2004.