VOD 시스템을 위한 효율적인 P2Proxy 캐싱 기법

권 춘 자[†] · 최 치 규[††] · 이 치 훈[†††] · 최 황 규[††††]

요 약

인터넷 상에서 VOD 서비스 보급이 확산되면서 대규모 VOD 서비스 실현을 위한 P2P 환경에서의 VOD 시스템에 대한 관심이 대두되고 있다. 본 논문은 대규모 VOD 시스템을 실현하기 위하여 기존의 프록시를 대체하여 P2P 환경 하에서 다수의 클라이언트들로 구성되는 새로운 프록시 캐싱 방법인 P2Proxy 기법을 제안한다. 제안된 기법은 P2P 환경에서 비디오 스트림을 클라이언트들에 분산 저장하고 이들 클라이언트 그룹을 프록시로 활용하는 방법이다. 즉, 각 클라이언트는 요구한 스트림을 자신이 속한 그룹의 다른 클라이언트들로부터 캐싱된 스트림들을 전송 받으며 캐쉬에 없는 부분만을 서버로부터 전송 받는다. 본 논문은 P2Proxy의 그룹에 포함된 클라이언트와 서버 간의 스트림 캐싱 과정을 통해 각 그룹의 생성과 소멸 과정을 보이고, 클라이언트의 캐싱 정보를 공유하기 위한 디렉토리 구조를 제안한다. 이 디렉토리 정보를 이용하여 그룹에 참여한 다른 클라이언트의 정보를 얻는 과정을 보이며, 이를 활용함으로써 재생과 전송을 위한 메시지 교환을 최소화한다. 또한, P2P 환경에서 클라이언트의 불규칙한 행동으로 인한 이탈에 따른 복구 과정도 제안한다. 본 논문은 성능 평가를 통해 제안된 기법이 기존의 P2P 스트리밍 기법에 비하여 그 성능이 우수함을 보인다.

키워드 : P2P, P2Proxy, VOD, 프록시 캐싱, 패칭, 스트리밍

An Efficient P2Proxy Caching Scheme for VOD Systems

Chun Ja Kwon[†] · Chi Kyu Choi[††] · Chi Hun Lee[†††] · Hwang Kyu Choi[††††]

ABSTRACT

As VOD service over the Internet becomes popular, a large scalable VOD system in P2P streaming environment has become increasingly important. In this paper, we propose a new proxy caching scheme, called P2Proxy, to replace the traditional proxy with a scalable P2P proxy in P2P streaming environment. In the proposed scheme, each client in a group stores a different part of the stream from a server into its local buffer and then uses a group of clients as a proxy. Each client receives the request stream from other clients as long as the parts of the stream are available in the client group. The only missing parts of the stream which are not in the client group are directly received from the server. We represent the caching process between clients in a group and a server and then describe a group creation process. This paper proposes the directory structure to share the caching information among clients. By using the directory information, we minimize message exchange overload for a stream caching and playing. We also propose a recovery method for failures about the irregular behavior of P2P clients. In this paper, we evaluate the performance of our proposed scheme and compare the performance with the existing P2P streaming systems.

Key Words : Peer to Peer, Peer to Peer Proxy, Video on Demand, Proxy Caching, Patching, Streaming

1. 서 론

인터넷 활용의 증가와 디지털 컨텐츠 제작 기술의 발전은 멀티미디어를 기반으로 하는 인터넷 서비스의 확대 보급을 이끌었다. 실시간 멀티미디어 스트리밍 기술은 이러한 멀티미디어 서비스의 대표적인 응용분야로서 원격 교육, 온라인 오락, 디지털 비디오 도서관, 전자상거래 등 모든 인터넷 기반의 응용 분야에 활용되고 있다.

한편, 다수의 인터넷 사용자들이 하나의 네트워크 그룹을 형성하여 협력하는 P2P(Peer-to-Peer)는 인터넷을 정보의 바다 역할을 하는데 힘을 더하였다. 가장 일반적인 P2P 응용 시스템은 파일공유 프로그램을 들 수 있으며, 그 예로는 '냅스터(Napster)'와 '소리바다', 'e-Donkey' 등이 있고, 이외에 AOL과 같은 인스턴트 메시지 서비스 시스템과 게임 등이 다양하게 활용되고 있다[1]. 또한 P2P 시스템은 다수의 피어(peer)들을 연결하여 분산 컴퓨팅 환경을 구축함으로써 시스템의 성능을 높일 수 있다. 즉, P2P 기반의 시스템은 피어들 간에 CPU와 메모리 등의 자원과 서비스를 공유하여 작업을 수행하므로 시스템의 수행 용량을 증가 시키는 효과와 더불

※ 본 논문은 2005년 정보통신기초기술연구지원사업(과제번호 B1220-0401 0203) 연구결과의 일부임.
† 준 회 원 : 강릉영동대학 사이버경찰과 조빙교수
†† 정 회 원 : 바로비젼(주) 뉴미디어연구소 연구원
††† 준 회 원 : 강원대학교 컴퓨터정보통신공학과 석사과정
†††† 정 회 원 : 강원대학교 IT특성화학부(대학) 컴퓨터정보통신공학전공 교수
논문접수 : 2006년 1월 25일, 심사완료 : 2006년 3월 21일

어 작업의 부하를 다수의 피어에게 분산시키는 효과도 얻을 수 있다.

일반적인 VOD 시스템은 클라이언트-서버 방식으로 구성되었으며, 다수의 클라이언트의 요청에 따른 부하가 서버에 집중되어 서비스 용량의 한계를 초래한다. 이에 따라 최근에는 클라이언트의 자원을 활용하여 서버의 부하를 줄이는 P2P 미디어 스트리밍 기법에 대한 연구가 활발히 이루어지고 있다. 이러한 가장 최근의 대표적인 기법으로는 P2VoD[10]가 있으며, 여기에서는 클라이언트 그룹을 선형구조로 구성하여 하위 클라이언트들에게 전송함으로써 서버의 부담을 감소시키고자 하였다. 즉, 클라이언트가 미디어를 요청하여 스트림을 재생하는 동안 이를 버퍼링하였다가 다른 클라이언트에게 재전송함으로써 서버의 부하를 줄일 수 있다. P2VoD는 요청 클라이언트 수가 많아질수록 서버의 부담을 증가시키지 않으면서 수용할 수 있는 클라이언트 수를 증가시킬 수 있는 장점이 있으나, 다수의 클라이언트를 구성하고 제어하기 위한 추가적인 부담이 따른다. 또한 재생시간 동안에만 임시적으로 유지되는 시스템이므로 클라이언트의 불규칙한 행동에 대응하기 위한 방법이 필요하다.

본 논문은 P2P를 기반으로 하여 미디어 스트리밍을 제공하기 위한 새로운 캐싱 기법인 P2Proxy(Peer-to-Peer Proxy) 기법을 제안한다. 제안된 기법은 비디오의 초기 요청에 대하여 기본 채널을 생성하고 이후의 요청은 각 요청 시간에 해당하는 기본 스트림을 클라이언트의 버퍼 크기만큼 저장하여 캐싱을 수행한다. 즉, 재생을 요청한 클라이언트는 자신이 속한 그룹의 클라이언트로부터 미디어 스트림을 전송받아 사용함으로써 서버의 부하를 줄이고자 한다.

본 논문의 나머지 구성은 다음과 같다. 먼저 2장에서는 P2P 미디어 스트리밍 기법과 함께 본 논문과 관련된 연구 결과를 살펴봄으로써 본 논문의 연구 배경을 설명한다. 3장에서는 본 논문에서 제안한 P2Proxy 기법의 기본 동작, 캐싱 및 스트림 전송 과정, 클라이언트의 이탈과 복구 과정 등에 대하여 상세하게 기술한다. 4장에서는 제안된 기법이 기존의 P2VoD 기법보다 성능 향상을 이루었음을 실험 결과를 통해 보인다. 그리고 마지막으로 5장에서는 제안된 기법에 대한 결론을 도출한다.

2. 관련 연구

VOD 시스템은 다수의 클라이언트에게 대용량 멀티미디어 데이터를 연속적으로 제공하여야 하므로 고성능의 서버를 필요로 한다. 그러나 클라이언트의 수가 급속히 늘어나면 고성능의 서버라고 할지라도 성능의 한계를 피할 수 없다. 특히 네트워크 대역폭의 한계는 시스템 성능을 떨어뜨리는 주요 원인이 된다. 이 문제를 해결하기 위하여 VOD 서버에 집중되는 부하를 줄이기 위한 연구가 꾸준히 진행되었다. 이에 대한 연구 분야는 네트워크 대역폭 최적화, 프록시(proxy) 캐싱 서버 활용 방법, 클라이언트를 캐시 서버로 활용하는 P2P 미디어 스트리밍 방법 등으로 나눌 수 있으며, 다음과 같이 요약할 수 있다.

2.1 네트워크 대역폭 최적화 기법

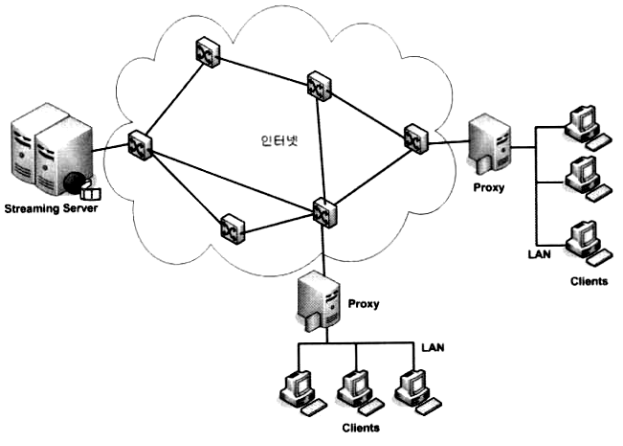
IP 멀티캐스트를 통한 미디어 전송은 다수의 클라이언트에게 연속 미디어를 제공하는 VOD 서버의 부하를 줄이기 위한 효과적인 기법이다. 이를 활용한 기법으로는 Pyramid Broadcasting[2]과 Skyscraper Broadcasting[3]과 같이 일정 시간 단위를 주기로 다수의 사용자에게 브로드캐스트(broadcast) 하는 배칭(batching) 기법이 있다. 또한 일정 분량의 요청을 모아서 한 번에 멀티캐스트(multicast)하는 동적 배칭(dynamic batching) 기법[4]도 연구되었다. Pyramid 방법과 Skyscraper 방법이 정해진 주기로 브로드캐스트 하는 것과 달리 동적 배칭 기법은 각 비디오에 대한 요청을 큐(queue)에 저장한 후, FCFS(First Come First Service)와 MQL(Maximum Queue Length) 등의 방법을 활용하여 큐에 저장된 클라이언트의 요청을 한 번에 멀티캐스트 하는 방법이다. 그러나 이러한 방식은 요청을 수행한 사용자에 대한 초기 지연시간이 존재하며 이러한 지연 시간이 길어지면 클라이언트가 재생을 거부하는 확률(reneging probability)이 높아진다. 이와 같이 사용자의 거부 확률이 높아질 경우 VOD 시스템의 성능이 낮아지는 원인이 된다.

또한 배칭 기법의 초기 지연 시간을 줄이기 위하여 제안된 패칭(Patching) 기법[5]은 초기 요청에 대한 정규 채널을 생성하고, 이후의 요청에 대하여 지나간 프리픽스(prefix)를 전송받는 패칭 채널을 추가로 생성하는 방법이다. 이는 패칭 윈도우의 크기를 나누는 방식에 따라 크게 Greedy Patching, Grace Patching, Optimal Patching[11] 등의 세 가지로 구분할 수 있으며 그 기준은 패칭 윈도우의 크기가 된다. 패칭 기법은 사용자의 요청에 대하여 지나간 부분에 대한 채널만을 추가적으로 요구하므로 서버의 대역폭 소모량을 최소로 줄일 수 있다. 또한 각각의 요청에 대한 초기 지연시간이 존재하지 않아 사용자의 거부 확률이 작다는 장점을 가진다. 반면에 패칭 기법은 재생과 버퍼링을 동시에 수행하므로 배칭에 비하여 높은 클라이언트 시스템의 성능과 버퍼공간과 같은 추가 자원을 요구한다. 따라서 서버의 부하를 분산시키기 위하여 클라이언트와 근거리에 위치한 프록시 서버를 활용하는 방법이 연구되었다.

2.2 프록시 캐싱(Proxy Caching) 기법

프록시(proxy)란 클라이언트와 멀리 떨어져 있는 VOD 서버를 대신하여 근거리에 위치한 소규모의 서버로서, VOD 서버에 저장된 데이터의 캐싱 역할을 수행한다. 즉, 프록시를 활용한 기법은 (그림 1)과 같이 스트리밍 서버의 프록시를 사용자와 가까운 네트워크 내에 두어 미디어 데이터의 앞부분 또는 자주 요청되는 일부분을 저장할 수 있는 캐싱 서버의 역할을 수행하며, 서버의 부하를 감소시킴과 동시에 네트워크 대역폭 요구를 크게 줄일 수 있은 방법으로 인터넷을 기반으로 하는 스트리밍 서비스에서 그 활용도가 매우 높다.

프록시 서버를 활용하는 연구 결과로는 Proxy Prefix Caching [12]과 같이 프록시 서버에 비디오 파일의 프리픽스를 저장



(그림 1) 프록시 기반의 스트리밍 시스템 구성

해 두어 초기 지연시간을 줄이는 기법이 제안되었다. 또한 프록시 캐싱의 최적화를 통하여 네트워크 대역폭의 사용 비용을 최소화 하는 연구도 진행되었다[13, 14]. 기존의 패칭 기법의 관점에서 프록시 서버의 프리픽스와 버퍼 확장을 통해 최적의 패칭 윈도우(optimal patching window)의 크기를 늘리는 기법도 제안되었다[7, 8]. 최근에는 프록시 서버를 Caching Agent로 활용하여 프록시 서버 간의 미디어 스트림의 일부 또는 전체를 전송하여 서버의 역할을 대신하는 기법도 제안되었다[15].

그러나 위와 같은 프록시 캐싱 기법을 사용한다고 하더라도 프록시 서버의 저장용량의 한계로 많은 수의 대용량 미디어 데이터에 대하여 캐시 할 수 있는 용량이 매우 제한적이며, 사용자의 수가 증가할 경우 프록시 서버로 부하가 집중되어 서비스 용량의 한계를 초래한다. 이러한 문제점을 해결하고 보다 확장성 있는 스트리밍 시스템을 구현하고자 시작된 연구가 P2P 스트리밍 시스템으로 그에 대한 연구가 활발히 진행 중이다.

2.3 P2P 미디어 스트리밍(P2P Media Streaming) 기법

P2P 미디어 스트리밍은 사용자 수가 증가함에 따라 서비스 용량이 증가하는 장점이 있어 유니캐스트(unicast)를 기반으로 하는 환경 하에서 활발히 연구되고 있다. 먼저, 별도의 디렉토리 서버를 두어 이 서버를 통해 요청하는 방법이 아닌 P2P 미디어 스트리밍에 참여하는 클라이언트들의 메시지 교환만으로 멀티캐스트 트리를 구성하는 ZIGZAG 기법[17]이 제안되었다. 그런데 N개의 Peer로 이루어진 트리구조를 탐색하기 위하여 평균적으로 $\log N$개의 메시지를 교환하여야 하는 부담이 따른다. 이와 달리 기존의 패칭 기법을 응용하여 패칭 스트림을 클라이언트에 전송하였다가 재전송하는 P2Cast[9]가 제안되기도 하였다. P2Cast 역시 기존의 패칭 기법을 응용 계층에서 수행하기 위하여 기본 스트림의 재전송 트리를 형성하고 스트림을 재전송하는 방식이다. 즉, 기본 채널과 패칭 채널을 모두 먼저 요청한 클라이언트가 이후에 요청한 클라이언트에게 재전송하는 방식이다.

이외에도 IP 멀티캐스트를 활용하는 기존의 프록시 패칭 기법과 P2P 미디어 스트리밍을 활용하여 정규 채널에 대한 캐싱을 수행하는 기법도 제안되었다[20]. 즉, 프록시 서버를 네트워크로 연결된 다수의 클라이언트에 대한 디렉토리 서버와 패칭에 대한 프리픽스 캐싱 서버로 활용하고, 프록시에 연결된 클라이언트를 패칭 윈도우 단위로 묶어 정규 스트림의 캐싱을 수행한다. 이 기법은 패칭을 수행할 때 빈번히 생성되는 정규채널의 수를 줄여 서버의 대역폭 요구량을 줄일 수 있다. 이와 유사한 기법으로 기존의 Chaining 기법[18, 19]을 바탕으로 클라이언트들을 그룹으로 묶어 활용한 P2VoD [10]는 선형 구조를 이용한다. 이 기법은 먼저 요청을 수행한 미디어 스트림을 버퍼링 하였다가 이후에 요청한 클라이언트에게 재전송하는 기법이다. 이러한 P2P 미디어 스트리밍은 서버로부터 하나의 채널을 소모하고 클라이언트들이 연속적으로 재전송을 수행하는 가상의 멀티캐스트 방식이라고 할 수 있다.

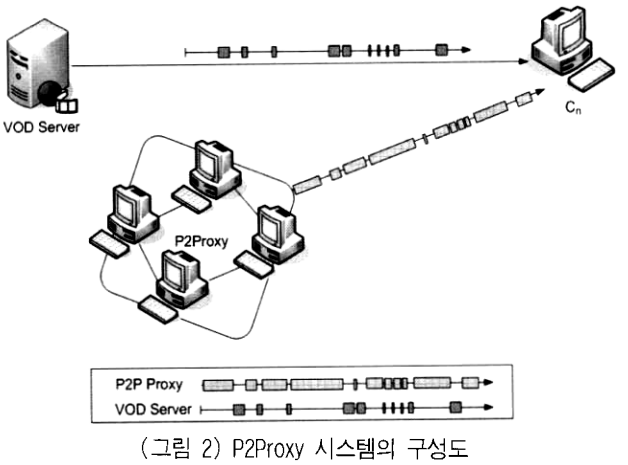
위와 같은 P2P 미디어 스트리밍 기법의 문제점으로는 먼저 미디어 스트림이 다수의 클라이언트를 거쳐 전송되므로 미디어 소스로부터 목적지까지의 전송 지연(End-to-End delay)이 발생한다는 점을 들 수 있다. 또한 다수의 클라이언트들이 불규칙하게 움직이므로 이탈에 따른 복구 방법이 필요하다. 또 각 클라이언트들을 제어하기 위한 오버헤드를 최소화하는 방법도 요구된다.

3. P2Proxy 캐싱(Peer-to-Peer Proxy Caching) 기법

본 장에서는 본 논문에서 제안한 P2P 기반의 환경에서 하나의 기본 채널을 공유하는 클라이언트 그룹을 캐싱 서버로 활용하는 P2Proxy 기법에 대하여 상세히 기술한다.

3.1 P2Proxy 기법의 개요

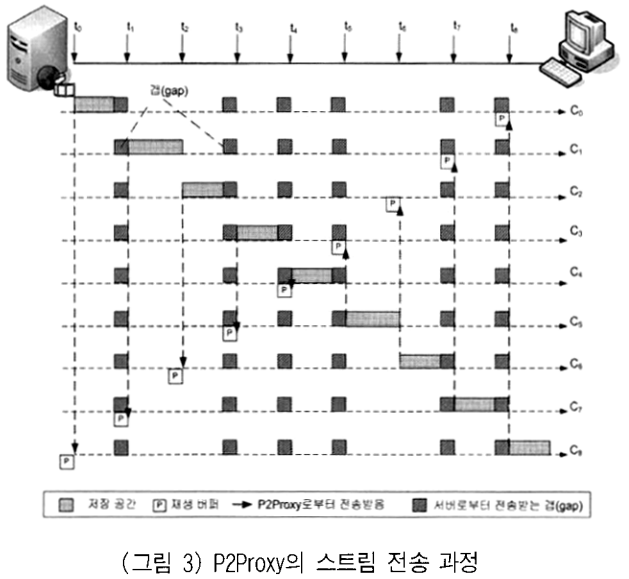
본 논문에서 제안한 P2Proxy 기법은 미디어 스트림이 같은 그룹에 속한 다수의 클라이언트들에게 고르게 분산되어 캐싱되는 방법이다. 즉, 클라이언트의 그룹이 비디오 스트림에 대한 캐시 서버의 역할을 수행하므로 VOD 서버에 요구되는 부하를 줄일 수 있다.



(그림 2) P2Proxy 시스템의 구성도

(그림 2)는 본 논문에서 제안한 P2Proxy 시스템 기본 구성을 나타낸 것으로, VOD 서버와 다수의 클라이언트인 P2Proxy 그룹으로 구성된다. P2Proxy의 각 클라이언트는 캐싱을 수행하기 위한 저장 공간을 가지고 있으며, 자신이 속한 그룹의 클라이언트들로부터 저장되어있는 미디어 스트림을 전송받아 사용하고 없는 부분만을 VOD 서버에서 전달받아 사용한다. P2Proxy 시스템에서 VOD 서버는 비디오 스트림 전체에 대한 서비스를 책임지며, 클라이언트 그룹의 캐싱 정보에 관한 디렉토리를 관리하는 역할을 한다. 각 클라이언트는 요청 시간에 해당하는 기본 스트림을 자신의 버퍼에 저장하며, 자신이 속한 그룹의 디렉토리 정보를 서버로부터 전송받아 재생과 전송에 활용한다. 또한 자신의 상태 변화를 그룹 내의 클라이언트와 VOD 서버에게 메시지로 전송하며 그룹에 속한 모든 클라이언트가 동일한 정보를 유지하도록 한다.

(그림 3)은 각 클라이언트가 저장하고 있는 스트림의 양과 서버에 요구되는 부하의 양을 보여주는 것으로, 가로선은 요청 시간이 다른 각 클라이언트에 대한 미디어 스트림을 나타낸다. (그림 3)의 C_0는 그룹의 초기 요청을 수행하는 클라이언트이다. 각 그룹의 초기 요청은 기본 채널(base channel)을 생성하며 이러한 스트림을 기본 스트림(base stream)이라고 한다. 더불어, (그림 3)에 나타난 어두운 사각형은 각 클라이언트가 저장하고 있는 기본 스트림의 시간적인 위치와 분량을 나타낸다. 또 가로선 아래의 P로 표시된 사각형은 현재 시점에서 각 클라이언트의 재생 위치를 나타낸다. 마지막으로 빗금으로 표시된 사각형은 P2Proxy의 그룹에 속하지 않아 클라이언트 버퍼 사이의 내용이 끊긴 부분이다. 이렇게 클라이언트 버퍼 사이의 끊어진 스트림을 갭(gap)이라고 정의하며, 갭은 모든 클라이언트가 서버로부터 전송받아 재생하여야 한다. P2Proxy에 존재하지 않는 갭들을 전송하기 위한 채널을 패칭 채널(patching channel)이라 하고, 이 채널을 통해 전송되는 스트림을 패칭 스트림(patching stream)이라 한다.



(그림 3) P2Proxy의 스트림 전송 과정

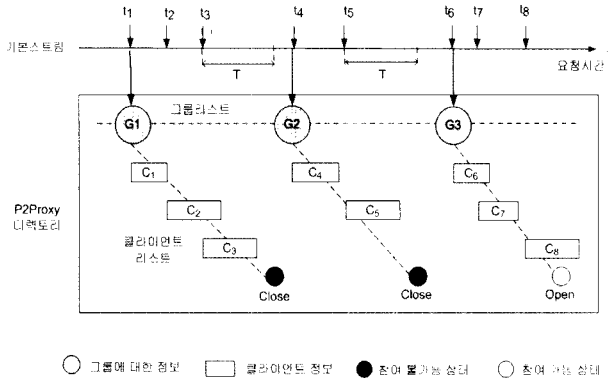
(그림 3)은 P2Proxy의 그룹에 속한 각 클라이언트가 자신의 요청 시간에 대한 기본 스트림을 캐싱하고 있는 상태를 나타낸다. 즉, t_0 시점에 발생한 초기 요청에 대하여 클라이언트 C_0가 서버로부터 기본 스트림을 전송 받아 자신의 버퍼 분량만큼 기본 스트림을 저장한다. 이후 t_1 시점에 클라이언트 C_1이 스트림을 요청하면 C_1은 클라이언트 C_0로부터 스트림을 전송 받아 재생한다. 동시에 C_1은 진행 중인 기본 스트림을 버퍼에 저장한다. 이와 같은 방법으로 각 클라이언트는 자신의 요청 시간과 버퍼 크기에 따라 전체 미디어 스트림을 분할하여 저장한다. 각 클라이언트는 자신이 재생할 부분을 P2Proxy 그룹이나 서버로부터 전송받는다. (그림 3)을 보면, C_0에서 C_7의 요청이 이루어진 후에 클라이언트 C_8의 요청이 t_8의 시점에 이루어졌다. 이때, 각 클라이언트는 자신의 재생 시점에 해당하는 스트림을 저장하고 있는 클라이언트들로부터 스트림을 전송받아 사용한다.

P2VoD와 같은 기존의 P2P 미디어 스트리밍 기법은 다음과 같은 특징을 지닌다. 우선적으로 클라이언트의 요청 순서에 따라 버퍼링과 재전송을 수행하며 클라이언트의 이탈이 일어나지 않는 한 소스(source)가 고정적으로 유지된다. 따라서 P2P 스트리밍 시스템 전체의 구조가 트리와 같은 체인 구조를 형성하며 스트림이 정해진 순서로 전송된다. 또한 서버로부터 정해진 구조를 따라 다수의 클라이언트를 거쳐 전송되며, 항상 요청 순서가 앞선 클라이언트의 내용이 이후의 클라이언트로 전송되는 구조이다. 그리고 전송되는 클라이언트들 간의 버퍼의 내용이 연속적이며 각 클라이언트의 요청 간격이 클라이언트의 버퍼 크기보다 작아야 한다.

이와 비교하여 제안된 P2Proxy 기법은 하나의 기본 스트림에 대하여 이후의 각 클라이언트의 요청 시점에 따른 분할 캐싱을 수행한다. 따라서 다수의 클라이언트에 미디어 스트림이 분할되므로 재생이 진행됨에 따라 소스가 시간에 따라 바뀌는 특성을 지닌다. 더불어 요청 순서에 따라 트리 또는 체인 구조를 형성하는 것이 아니라 하나의 기본 스트림 단위로 P2Proxy의 그룹을 형성하고 그룹 내의 클라이언트들에 저장된 스트림을 필요한 시간에 전송하는 방식이다. 즉, P2Proxy 내의 클라이언트는 요청 순서가 늦더라도 자신이 가진 스트림을 재생을 원하는 클라이언트에게 전송할 수 있다. 특히 기존의 방식이 별도의 디렉토리 서버를 이용하거나 메시지 교환을 이용하는 방식인데 비하여 제안된 기법은 VOD 서버의 디렉토리를 활용한다. 즉, 서버로부터 그룹의 디렉토리 정보를 전송받아 메시지를 통하여 그룹에 참여하고 자신의 상태 변화를 그룹의 구성원에게 알려 동기화를 이루는 방식이다.

3.2 P2Proxy의 디렉토리 구조

제안된 P2Proxy 기법에서는 P2P 미디어 스트리밍의 스트림 캐싱 정보 관리를 위해 VOD 서버에 디렉토리를 구성한다. 각 클라이언트는 비디오 스트림 요청 시 자신이 속한 그룹에 대한 디렉토리 정보를 받아 그룹의 클라이언트들에게



(그림 4) P2Proxy의 디렉토리 구조

참여를 알린다. 즉, 각 클라이언트와 VOD 서버의 디렉토리 정보는 메시지 교환을 통하여 서버와 모든 클라이언트들에게 동일하게 유지된다. 특히 서버는 모든 그룹에 참여하므로 모든 그룹에 대한 정보를 가지며, 각 클라이언트는 자신이 속한 그룹에 대한 정보를 서버로부터 전송받아 사용한다.

일반적으로 미디어 스트리밍을 구현하는 VOD 서버는 각 클라이언트의 세션 정보를 관리한다. 따라서 본 논문에서 제안한 P2Proxy 기법을 적용하기 위하여 별도의 디렉토리 서버를 두기 보다는 VOD 서버에서 관리되는 클라이언트의 세션 정보에 필요한 구성 요소를 추가하여 재구성한다. (그림 4)는 VOD 서버에 유지되는 P2Proxy의 디렉토리 구조를 보여준다. (그림 4)에 나타난 P2Proxy의 디렉토리는 하나의 기본 스트림 단위로 생성되는 그룹의 리스트와 각각의 그룹에 속한 클라이언트의 리스트로 구성된다. 그룹에 대한 정보는 그룹의 ID와 클라이언트 리스트에 대한 HEAD와 TAIL, 그리고 새로운 클라이언트가 참여 가능한지를 나타내는 FLAG로 구성된다. 또한 각 클라이언트 정보는 각 클라이언트의 ID(IP 주소)를 비롯하여 클라이언트의 요청 시점 및 기본 스트림의 저장 위치, 그리고 버퍼의 크기 정보를 포함한다. 또한 이를 이용하여 각 클라이언트가 현재 재생되고 있는 스트림의 위치를 알 수 있다.

(그림 4)는 VOD 서버에 구성된 임의 시점 t_1에서 t_8까지의 클라이언트 요청에 대하여 형성된 디렉토리 정보를 나타낸다. t_1의 시점에 이루어진 요청에 의하여 하나의 기본 채널이 생성되고 이와 함께 그룹 G_1이 참여 가능(open) 상태로 생성된다. 또, t_1의 시점에 요청한 C_1은 G_1의 HEAD로서 참여한다. 이후 t_2와 t_3에 요청한 C_2와 C_3은 C_1의 TAIL로 참여한다. 이후 t_3의 시간부터 정해진 T(threshold) 시간 동안 요청이 들어오지 않아 G_1의 상태가 참여 불가능(close)상태로 바뀌게 된다. 이후 t_4의 시점에 C_4가 요청하여 새로운 그룹 G_2가 참여 가능 상태로 생성된다. 또, C_4는 G_2의 HEAD로서 참여한다. 이와 같은 과정을 거쳐 G_2에 C_5를 참여시킨다. 또한 정해진 T 시간 동안 요청이 없으므로 G_2의 상태는 참여 불가능이 된다. 이후 C_6의 요청에 의하여 t_6의 시점에 G_3을 생성하고 t_7, t_8에 요청한 C_7과 C_8을 참여시킨다.

즉, 클라이언트의 요청 시점에서 기존의 그룹이 참여 가능 상태이면 기존 그룹에 TAIL로서 참여하고, 참여 불가능 상태이면 새로운 그룹을 생성하고 생성된 그룹의 HEAD로서 참여한다. 또한, 클라이언트가 자신의 그룹에 대한 디렉토리만을 관리하는 것과는 달리 VOD 서버가 모든 그룹에 대한 디렉토리를 유지한다. 이는 VOD 서버가 디렉토리 정보에 대한 디렉토리 서버의 역할을 함과 동시에 모든 그룹의 구성원으로 패칭 스트림을 전송하는 특수한 클라이언트로서의 역할을 하기 때문이다.

3.3 클라이언트 참여와 이탈

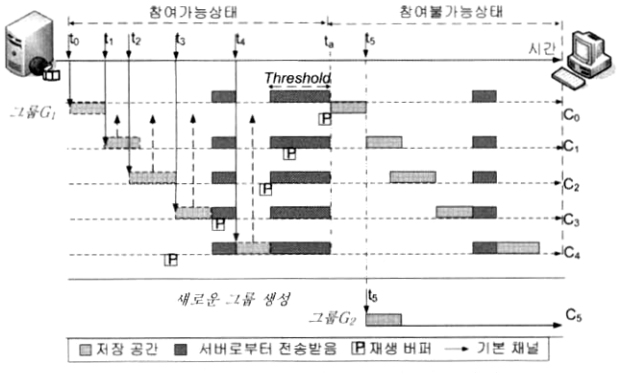
제안된 기법은 새로운 클라이언트 X의 요청에 대하여 진행 중인 기본 스트림을 클라이언트 X가 저장하고 재생 시간 동안 캐싱 서버의 역할을 수행하는 기법이다. 만일 클라이언트 X가 재생을 마치게 되면 캐싱 서버의 역할도 종료하고 P2Proxy 그룹에서 이탈한다. 즉, P2P 스트리밍의 각 클라이언트는 참여와 이탈을 불규칙적으로 수행하며 이에 따라 P2Proxy 그룹의 상태도 변하게 된다. 본 절에서는 제안된 P2Proxy 기법의 미디어 전송 과정에 따른 클라이언트의 참여와 이탈에 대하여 상세히 기술하며, 또한 각 클라이언트의 행동에 따른 P2Proxy 그룹의 상태 변화에 대하여 설명한다.

3.3.1 새로운 클라이언트의 참여

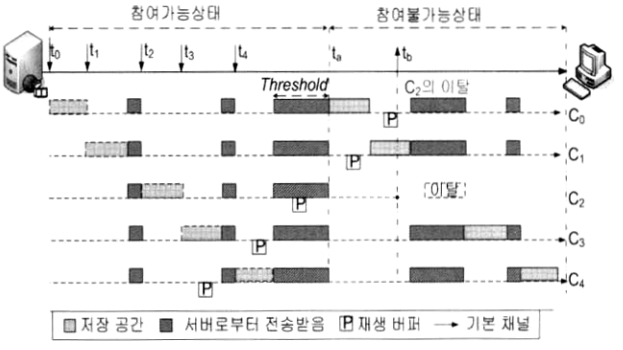
P2Proxy에서는 각 클라이언트가 미디어의 재생을 요청할 때, 자신이 속한 그룹의 정보를 서버로부터 전송받아 그룹의 모든 구성원에게 참여 메시지를 전송한다. 또한 같은 그룹에 속한 클라이언트와 서버는 메시지를 교환하여 동일한 디렉토리 정보를 유지한다. 이러한 참여의 과정이 이루어진 클라이언트는 P2Proxy의 캐싱 서버로서의 역할을 수행한다.

클라이언트의 참여 과정은 기본 스트림의 진행 시간과 그에 따른 클라이언트들의 요청에 의하여 결정된다. 또한 각 그룹을 나누는 기준이 되는 Threshold 적용 방법에 의해서도 영향을 받는다. Threshold는 임계값을 의미하며, 이 절 이후에 자세하게 설명한다. 제안한 P2Proxy 기법에서 갭의 크기의 허용범위를 고정적인 Threshold 값으로 적용한다. 즉, 각 요청사이의 간격이 클라이언트의 버퍼 분량을 초과할 경우 그 차이가 갭의 크기가 되며, 가장 큰 갭의 크기가 Threshold의 범위를 넘을 경우 기존의 그룹이 참여 불가능 상태가 된다. 따라서 이후의 요청은 참여 가능한 새로운 그룹을 생성하고 클라이언트는 HEAD로서 참여하게 된다.

(그림 5)는 클라이언트의 순차적인 요청에 따른 클라이언트의 참여와 그에 따른 캐싱의 수행과정을 나타낸다. 먼저 t_0 시간에 요청한 클라이언트 C_0에 대한 새로운 기본 채널이 생성되고 그에 대한 그룹 G_1이 생성된다. 이후 t_1 시점에 클라이언트 C_1의 요청이 들어오면 C_1은 기본 채널로부터 스트림을 전송받아 이를 C_0에게 재전송함과 동시에 자신의 버퍼에 저장한다. 이후 t_2와 t_3의 시간에 요청한 C_2와 C_3에 대하여도 같은 과정을 반복한다. 이때, 각 요청 사이의 시간차



(그림 5) 새로운 클라이언트의 참여과정



(그림 6) 클라이언트의 이탈에 따른 복구 과정

는 클라이언트의 버퍼 크기이내이므로 C_1, C_2 그리고 C_3의 버퍼에 스트림이 연속적으로 저장된다. 그러나 이후 t_4 시점에 요청이 발생하면, t_3과 t_4의 요청간격이 C_3의 버퍼 공간을 초과하므로 t_3과 t_4의 사이에 버퍼에 저장되지 못하는 갭이 발생한다. 이와 같이 제안된 P2Proxy 기법은 각 클라이언트 사이에 정해진 범위 이내에서 발생된 갭은 허용하지만, 갭의 크기가 큰 경우에는 요청한 클라이언트를 참여시키지 않고 새로운 그룹을 생성한다. 즉, (그림 5)에서 t_4 시간에 요청이 이루어진 후 정해진 Threshold 시간 동안 요청이 들어오지 않아 t_a 시점에 그룹을 참여 불가능 상태로 변경한다. 이후 t_5 시점에 요청이 이루어지면 기존의 그룹이 참여 불가능 상태이므로 새로운 기본 채널을 할당하고 이에 대한 그룹 G_2를 생성한다.

(그림 5)에서 t_a 시점에 그룹 G_1이 참여 불가능 상태로 변경되면 G_1에 속한 TAIL의 재생 지점이 지난 클라이언트들의 버퍼를 회수하여 기본 스트림이 캐싱하는데 재사용한다. 즉, 그룹이 참여 불가능 상태로 변경되면 그룹에서 더 이상 사용하지 않는 버퍼가 발생하므로 이를 활용하여 다시 기본 스트림을 캐싱 할 수 있다. (그림 5)에 나타난 바와 같이 t_a의 시점에 G_1이 참여 불가능 상태로 변경되면 G_1의 TAIL인 C_4의 재생 지점이 지난 t_a 이후 클라이언트 C_0와 C_1의 버퍼를 회수하여 진행 중인 기본 스트림을 저장해 나간다. 또 이러한 과정은 기본 스트림이 진행되면서 계속하여 반복적으로 이루어진다.

3.3.2 클라이언트의 이탈과 복구

P2P 미디어 스트리밍 방법은 불안정한 클라이언트 시스템을 캐싱 서버로 활용하므로 클라이언트의 불규칙한 행동에 의한 이탈이 발생한다. 따라서 P2P를 활용하는 스트리밍 방법은 이러한 이탈에 대한 복구 과정을 필요로 한다. P2Proxy의 그룹에 참여한 클라이언트가 이탈할 경우, 자신이 속한 그룹의 모든 구성원에게 이탈 메시지를 전송한다. 또 이탈 메시지를 받은 모든 구성원은 자신의 디렉토리 정보를 갱신한다. 이러한 이탈 과정이 종료되면 그룹에 새로운 갭이 발생할 수 있는데, 이 갭에 해당하는 스트림은 패칭 채널을 이용하여 VOD 서버로부터 전송받는다. 또한 그룹 내의 모든 클라이언트가 이탈할 경우 그룹도 소멸된다.

(그림 6)은 제안된 P2Proxy 기법에서 클라이언트의 이탈에 따른 복구 과정을 나타낸다. 클라이언트가 재생 중에 이탈한 경우 새롭게 발생한 갭에 해당하는 스트림을 VOD 서버로부터 전송받는 모습을 보여준다. (그림 6)에서 각각 t_0, t_1, t_2, t_3와 t_4의 시간에 C_0, C_1, C_2, C_3 및 C_4의 클라이언트가 P2Proxy의 그룹을 구성하여 진행 중일 때 클라이언트 C_2의 이탈이 발생하였다. 하지만 그 이전에 임의의 순간 t_a에서 단절된 스트림의 크기가 임계시간에 도달하여 그룹이 참여 불가능 상태로 바뀌게 된다. 따라서 P2Proxy 그룹은 자신의 TAIL인 C_4의 재생 지점이 지나 더 이상 스트림을 저장할 필요가 없는 재사용이 가능한 버퍼를 찾아 기본 스트림의 캐싱을 계속 수행한다. 이때, (그림 6)에서 나타난 바와 같이 t_b의 시점에 클라이언트 C_2가 이탈하면 C_2가 가진 버퍼 공간이 사라져 새로운 갭이 발생한다. 이를 위해 각 클라이언트는 VOD 서버로부터 C_2의 이탈에 의하여 발생한 갭만큼의 크기를 추가로 패칭한다.

3.4 클라이언트간의 메시지 교환 최소화

본 논문에서 제안한 P2Proxy 기법은 하나의 클라이언트가 스트림을 재생할 때 시간에 따라 스트림의 소스가 변하는 것이 특징이다. 이렇게 변하는 소스의 위치는 서버로부터 전송받은 디렉토리 정보를 이용하여 재생할 스트림을 저장하고 있는 클라이언트를 찾을 수 있다. 그러나 소스의 위치를 안다 할지라도 분할된 각각의 스트림을 다수의 클라이언트에게 요청하여 전송받는 방식은 전송 지연과 더불어 메시지의 수가 지나치게 증가한다. 즉, 제안된 기법에서 하나의 그룹에 N개의 클라이언트가 존재할 경우 위와 같은 방법을 사용하면 최소 N^2개의 메시지를 필요로 한다. 따라서 본 절에서는 그룹에 참여한 클라이언트가 디렉토리 정보를 이용하여 그룹 내에서 자신이 저장하고 있는 스트림을 원하는 클라이언트를 파악하여 전송하는 방법이다. 이 방법은 클라이언트가 참여와 이탈 과정을 제외한 메시지의 전송을 최소로 만들어 준다.

P2Proxy 기법의 캐싱 알고리즘을 설명하기 위하여 본 논문은 <표 1>과 같은 표기법을 사용한다. <표 1>에 나타난 바와 같이 mB는 P2Proxy를 위한 최소한의 버퍼 공간을 말

〈표 1〉 P2Proxy의 표기법

기 호	의 미
mB	캐시를 위한 최소 버퍼 공간
$B(X)$	클라이언트 X의 버퍼 공간
$p(X)$	비디오 스트림에 대한 클라이언트 X의 재생 위치
$c(X)$	비디오 스트림에 대한 클라이언트 X의 저장 위치
$G(X)$	클라이언트가 X가 속한 그룹
$Head(X)$	클라이언트 X가 속한 그룹의 $HEAD$
$Tail(X)$	클라이언트 X가 속한 그룹의 $TAIL$
$Threshold$	P2Proxy를 위한 임계값
t_X	서버에 대한 클라이언트 X의 요청 시간
t	서버의 현재 시간

하며 각 클라이언트는 mB 크기 이상의 버퍼 공간을 가지고 있다고 가정한다. $B(X)$는 클라이언트 X의 버퍼 크기를 말하며 mB 이상의 공간을 의미한다. 또한 $p(X)$는 클라이언트 X가 재생하고 있는 스트림의 위치를 나타내며, $c(X)$는 클라이언트 X가 저장하고 있는 스트림의 위치를 나타내고, $p(X)$와 $c(X)$ 모두 비디오 전체 길이에 대한 상대적인 값을 의미한다. 또한 $G(X)$는 X가 속한 그룹을 의미하며, $Head(X)$와 $Tail(X)$는 각각 $G(X)$의 $HEAD$와 $TAIL$을 의미한다. 마지막으로 $Threshold$ 값은 그룹을 나누기 위한 임계값을 의미하며, t_X 와 t는 클라이언트 X의 요청시간과 서버의 현재 시간을 나타내는 절대적인 값이며 VOD 서버의 시간을 그 기준으로 한다.

각 클라이언트가 비디오의 시작 지점부터 재생한다고 가정할 때, <표 1>의 기호를 사용하면 다음과 같은 식이 성립한다. 임의의 시간 t에서 그룹 $G(X)$에 대한 기본 스트림의 위치는 $t-t_{Head(X)}$의 값을 가진다. 더불어 임의의 시간 t_X에 비디오를 요청한 새로운 클라이언트 X는 $p(X)=0$을 초기값으로 가진다. 이 때, 클라이언트 X가 저장하여야 할 스트림의 위치는 $c(X)=t_X-t_{Head(X)}$의 값을 가지며, 이는 기본 스트림의 진행 지점과 일치한다. 또한, 시간에 따라 변하는 X의 재생 위치는 $p(X)=t-t_{Head(X)}-c(X)$의 값을 가진다.

동일 그룹 내의 모든 클라이언트는 동일한 디렉토리 정보를 유지하므로 위와 같은 수식을 이용하여 자신이 속한 그룹의 다른 클라이언트들에 대하여 현재 재생중인 위치와 그들이 저장하고 있는 스트림의 위치를 계산할 수 있다. 따라서 이를 이용하여 각 클라이언트는 자신이 저장하고 있는 스트림을 언제 다른 클라이언트가 원하는지 파악할 수 있다. 이러한 방법은 P2Proxy 기법과 같이 다수의 클라이언트에 전체 스트림이 분산된 환경에서 추가적인 메시지의 요구량을 최소로 하면서 다수의 소스에 흩어져있는 원하는 스트림의 전송이 성공적으로 이루어지도록 한다. 즉, 본 논문에서 제안된 P2Proxy 기법은 각 클라이언트와 서버가 자신이 가지고 있는 디렉토리 정보를 이용하여 그룹 내의 스트림을 필요로 하는 클라이언트를 파악하여 별도의 메시지를 요구하지 않고 전송하는 방식이다.

3.5 임계값(Threshold)의 적용 방법

본 절에서는 클라이언트의 참여시 기존 그룹의 참여 여부를 판단하는 기준이 되는 임계값(threshold)의 적용 방법에 대하여 기술한다. 기본적으로 임계값의 적용 방법으로 그룹 내의 최대 갭의 크기가 임계값을 넘지 않도록 한다. 즉, 일정 시간동안 요청이 들어오지 않는 경우 그룹을 참여 불가능 상태로 변경하며 이후의 요청은 새로운 그룹을 생성하여 참여하는 방법이다.

이와 함께 본 논문에서는 그룹 내의 패칭 스트림의 비율을 임계값으로 하여 그룹의 패칭 스트림의 비율이 전체 비디오 스트림의 일정 범위를 넘지 않도록 한다. 즉, 전체 비디오 스트림에 대한 패칭 스트림의 백분율을 P라고 할 때, $P=\frac{(N-1)\times G}{L}\times 100$이다. 이때 N은 그룹의 클라이언트 수를 의미하여, G는 그룹에 존재하는 갭의 크기 합을 나타낸다. 마지막으로 L은 그룹의 시간적인 크기를 나타내며 이는 $t_{Tail(X)}+B(Tail(X))-t_{Head(X)}$로 계산할 수 있다. 본 논문은 위와 같이 그룹을 위한 임계값의 적용 방법을 제안하였으며, 성능 평가를 통하여 임계값의 적용 방법에 따른 비교 결과를 보인다.

4. 실험 및 성능 평가

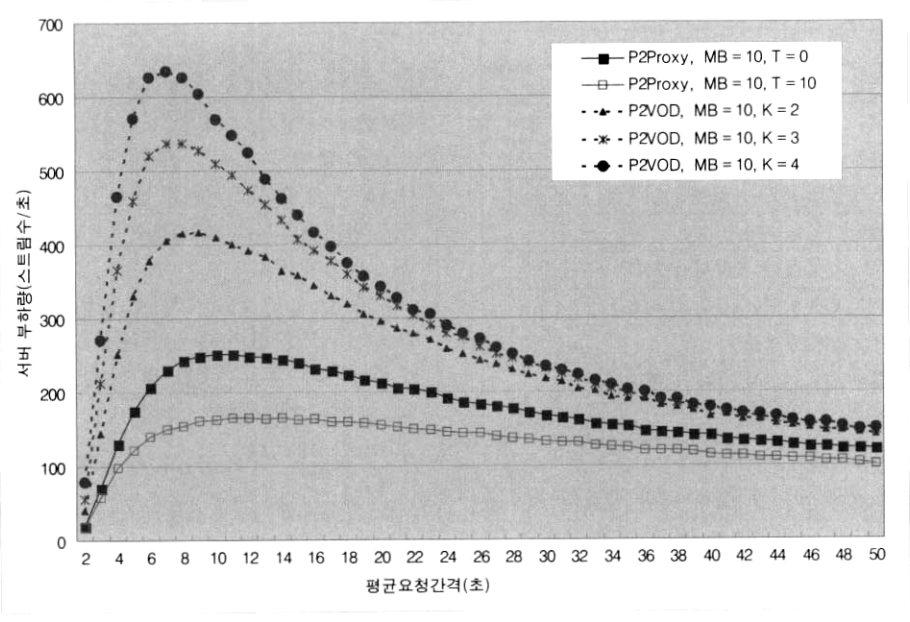
4.1 실험 환경

P2Proxy 기법에 대한 성능 평가를 위하여 <표 2>에 나타낸 파라미터를 이용하여 서버 부하량을 측정하였다. VOD 서버에는 120분 분량의 비디어 스트림이 저장되어 있을 때 비디오를 요청하는 각 클라이언트는 임의 크기의 버퍼 공간을 가지고 있다고 가정하였다. 클라이언트의 비디오 요청은 Poisson 분포를 따르며, 10시간 분량의 요청에 대하여 실험하였다. 또, CBR(Constant Bit Rate)의 비디오 스트림이라 가정할 때 VOD 서버에 요구되는 부하량은 채널 수에 비례한다. 따라서 VOD 서버에 요구되는 부하량을 비교하기 위하여 전체 요청시간 동안 소모되는 스트림 수의 평균을 구하여 비교하였다. 또한, 실험을 10회 반복한 값의 평균을 실험 결과로 얻었다.

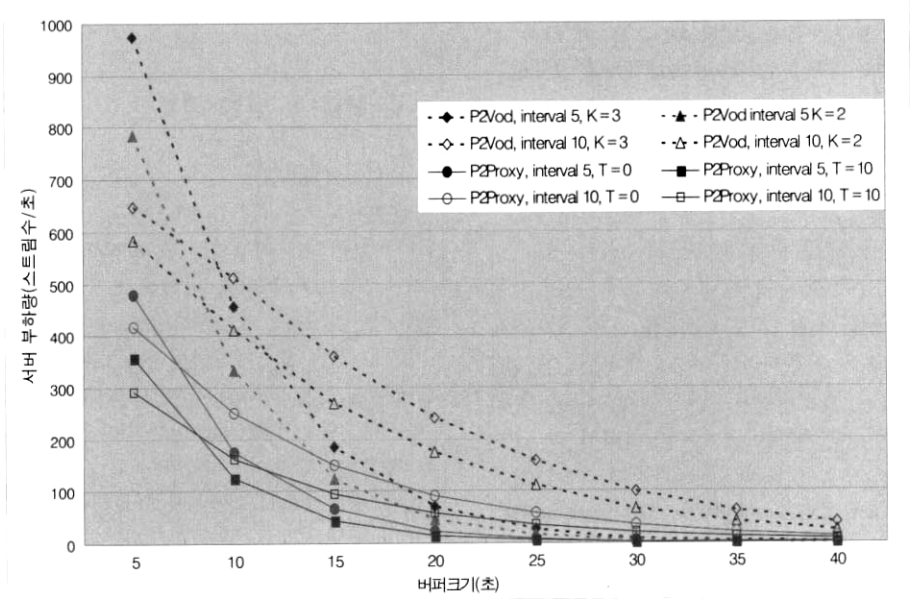
먼저 평균 요청 간격에 따른 서버의 부하 요구량에 대해 제안된 기법과 P2VoD기법을 비교하였으며, 제안된 기법에 대해 평균 요청 간격과 임계값에 따른 서버의 부하 요구량의 변화를 비교하였다. 또한 클라이언트의 버퍼 크기에 따른 서

〈표 2〉 성능 평가 파라미터

파라미터	기본값	변화량
비디오의 수	1	N/A
클라이언트 버퍼 크기 MB (초)	10	5-40
갭의 허용 범위 (초)	10	0-50
평균 요청 간격 $1/\lambda$ (초)	5	50
비디오 길이 $|v|$ (분)	120	N/A
실험 시간 (시간)	10	N/A



(그림 7) 평균 요청 간격에 따른 서버 부하량 비교(MB= 10)



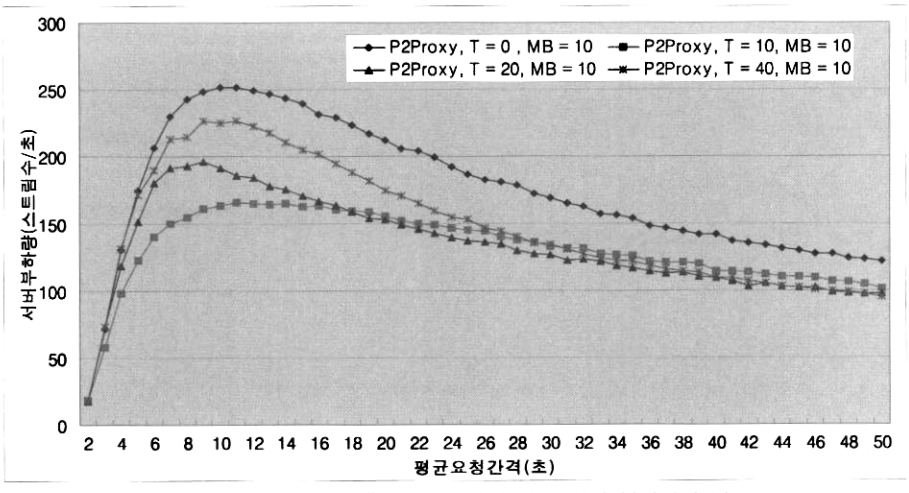
(그림 8) 클라이언트의 버퍼 크기에 따른 부하량 비교

버의 부하 요구량을 비교하였고, 각 임계값에 따른 서버의 부하 요구량의 변화를 살펴보았다.

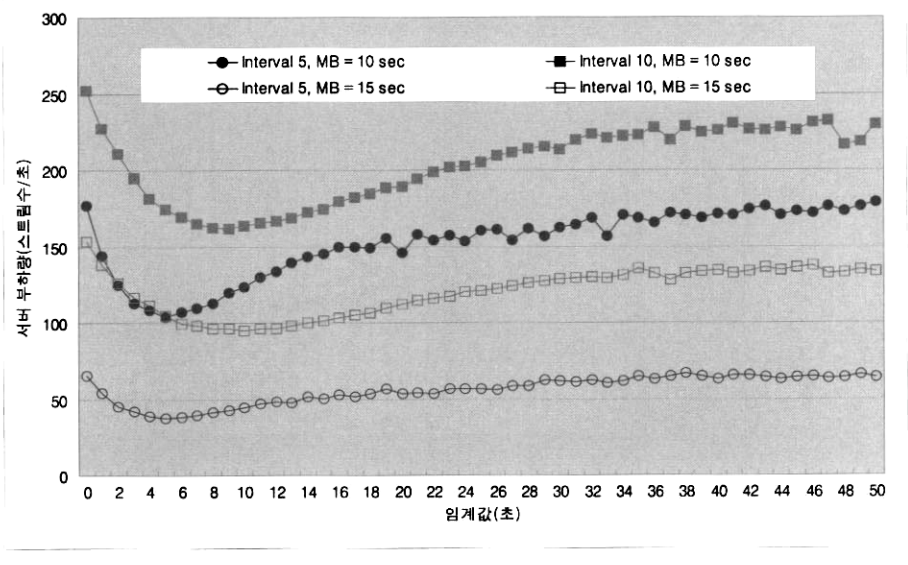
4.2 성능 평가 결과

(그림 7)은 각 클라이언트는 10초 분량의 버퍼를 가지고 있을 때 평균 요청 간격의 변화에 따라 VOD 서버의 부하 요구량을 제안된 P2Proxy 기법과 기존의 P2VoD 기법을 비교한 그래프이다. 그래프에서 기존의 P2VoD 기법에 대한 결과는 점선으로 표시되었으며, 제안된 기법인 P2Proxy의 결과는 실선으로 표시되어 있다. (그림 7)에서 가장 아래쪽의 밝은 사각형으로 표시된 그래프는 제안된 기법에서 임계값(T)을 10초로 주었을 때 얻은 값이며, 바로 위의 어두운 사각형으로 표현된 것은 임계값이 0일 때의 값이다. 또한 위쪽 세 개의 그래프는 각각 K 값이 2, 3, 4일 때 P2VoD 기법에 대한 서버의 부하 요구량을 나타낸다. 여기서 K는 P2VoD 기법에서 갖게 되는 레벨에 해당하는 값이며, K 값이 증가함에 따라 서버의 부하 요구량도 늘어나는 것을 볼 수 있다. (그림 7)에서 제안된 P2Proxy 기법이 서버에 대한 부하 요구량이 기존의 P2VoD에 비하여 적은 것을 볼 수 있다.

(그림 8)은 클라이언트의 버퍼 크기에 따른 P2Proxy와 P2VoD의 서버 부하 요구량을 나타낸다. 그래프에서 점선으로 표시된 것은 P2VoD 기법을 나타내며, 실선으로 표시된 내용은 P2Proxy 기법에 대한 결과를 나타낸다. 또, 그림에서 진한 도형으로 표시된 것은 평균 요청 간격이 5초인 것을 의



(그림 9) 평균 요청 간격과 임계값에 따른 서버 부하량의 비교

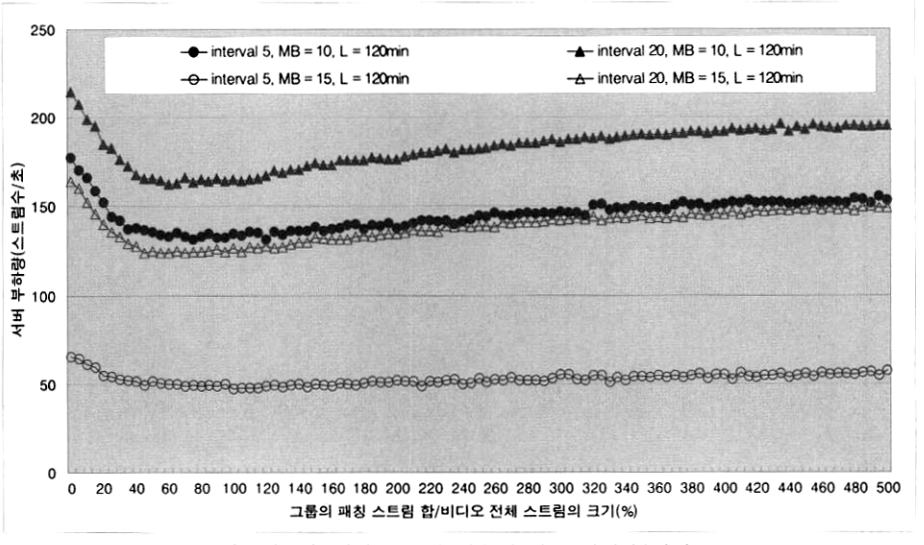


(그림 10) 임계값의 변화에 따른 서버 부하량 비교

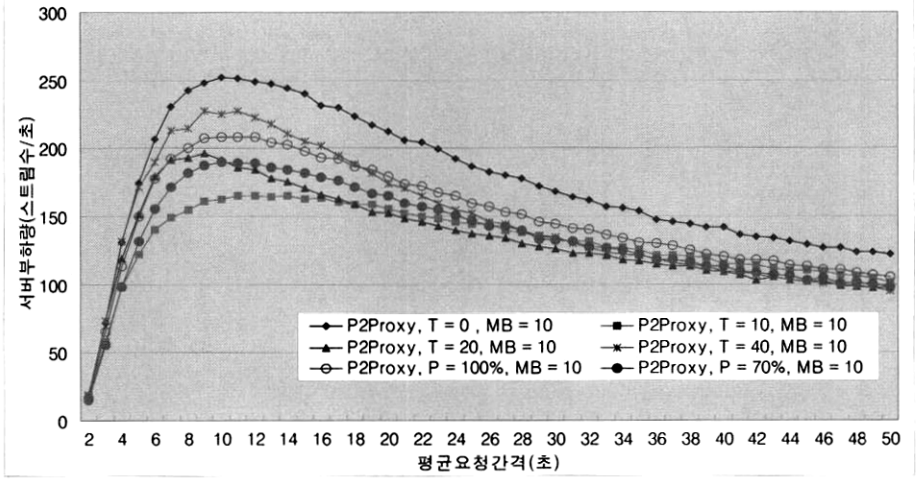
미하며, 밝은 도형으로 표시된 부분은 평균 요청 간격이 10초인 경우를 의미한다. (그림 8)에서 P2VoD의 K 값은 2와 3을 사용하였으며, P2Proxy의 임계값(T)으로는 0과 10을 사용하였다. (그림 8)에서 볼 수 있듯이 버퍼 크기가 증가할수록 서버에 대한 부하 요구량은 작아진다. 특히 동일한 조건에서 P2VoD와 비교할 때 P2Proxy가 성능이 우수한 것을 볼 수 있다. 또한 요청 간격이 10초인 경우가 기본적으로 서버의 부하량이 적게 나타나며, P2VoD나 P2Proxy의 버퍼 크기가 작은 경우는 서버의 부하량이 크게 나타난다. 그러나 클라이언트의 버퍼 양이 늘어나 버퍼 활용 효과가 커지면 평균 요청 간격이 작더라도 서버의 부하 요구량이 오히려 작게 나타난다.

(그림 9)는 그룹을 생성하는 기준이 되는 임계값(T)의 적용 방법에 따른 서버의 부하 요구량의 변화를 비교한 결과이며, 클라이언트의 버퍼 크기는 10초로 하였다. 즉, 평균 요청 간격과 임계값에 따라 서버의 부하 요구량을 나타내었다. 먼저 (그림 9)에서 진한 마름모 모양으로 표현된 것이 임계값을 0으로 하여 버퍼간의 단절된 공간을 허용하지 않는 방법이다. 즉, 갭의 크기를 허용하지 않은 경우를 나타낸다. 어두운 네모로 표시된 것은 임계값을 10초로 하였을 때로, 갭의 크기를 10초 범위에서 허용한 것을 나타낸다. 또한 삼각형과 별 모양은 각각 임계값인 갭의 크기가 20초인 것과 40초인 경우를 나타낸다. 그래프에서 보면 평균 요청 간격이 증가함에 따라 그래프들이 교차하는 현상이 나타난다. 즉, 평균 요청 간격에 따라 최적의 허용 범위의 값이 달라진다. 따라서 각각의 경우에 대하여 적절한 허용 범위의 값을 사용하는 것이 바람직하다. (그림 9)의 결과에서 보듯이 적절한 범위 내에서 갭을 허용하고 요청 간격이 지나치게 긴 경우에는 새로운 그룹을 생성하는 것이 보다 높은 성능을 얻을 수 있다.

(그림 10)은 임계값을 변화시켰을 때 요청 간격과 클라이언트 버퍼 크기에 따른 서버의 부하 요구량을 나타낸다. 그래프에서 색이 채워진 도형은 클라이언트의 버퍼 크기가 10초인 경우이며, 색이 채워지지 않은 도형으로 표시된 것은 버퍼의 양이 15초인 경우이다. 또 평균 요청 간격이 5초인



(그림 11) 패칭 스트림 비율에 따른 서버 부하량



(그림 12) 임계값의 적용 방법에 따른 서버 부하량 비교

경우는 원으로 표시되었고, 평균 요청 간격이 10초인 경우는 사각형으로 표시되었다. (그림 10)의 결과에서 보듯이 각각의 경우에 대하여 임계값을 늘려 가면 서버의 부하 요구량이 최저점에 이르렀다가 다시 증가함을 알 수 있다. 즉, 임계값에 대한 최적점이 존재하며 단절된 스트림의 크기를 그 이상으로 늘리는 것이 좋은 결과를 가져오지 못함을 의미한다. 또한 평균 요청 간격이 짧을수록 임계값의 최적점이 작아진다.

(그림 11)은 생성된 그룹이 전송하여야 하는 패칭 스트림의 크기를 전체 비디오 스트림과의 비율(%)로 계산하였을 때 임계값에 따라 그룹을 정하는 방법을 실험한 결과이다. 즉, 그래프의 X 축은 진행 중인 그룹의 패칭 스트림의 요구량의 합을 비디오 전체 스트림의 양으로 나눈 값에 대한 백분율을 의미한다. (그림 11)에서 클라이언트 버퍼의 크기가 10초인 경우는 색이 채워진 도형으로 표시되었으며, 클라이언트 버퍼의 크기가 15초인 경우는 색이 채워지지 않은 도형으로 표시되었다. 또한 평균 요청간격이 5초인 경우는 원으로 표시되었으며, 평균 요청 간격이 20초인 경우는 삼각형으로 표시되었다. (그림 11)의 결과에서 보듯이 하나의 그룹에 대한 패칭 스트림의 합과 비디오 스트림 크기와의 비율로 그룹의 생성 방법을 정하는 경우에도 최적점이 존재한다.

(그림 12)의 결과는 그룹을 생성하는 기준이 되는 임계값의 적용 방법에 따른 서버의 부하 요구량을 비교한 결과이다. 즉, 요청 간격을 기준으로 임계값을 정하는 방법과 패칭 스트림의 합을 임계값으로 정하여 그 기준으로 그룹을 정하는 경우에 대한 비교이다. (그림 12)에서 색이 채워진 동그라미로 표현한 것은 그룹의 패칭 스트림의 평균 요구량이 전체 비디오 스트림의 70%를 넘지 않도록 하여 구한 결과이다. 또한 색이 채워지지 않은 원으로 표시된 그래프는 패칭 스트림의 요구량이 비디오 스트림에 대하여 100%를 넘지 않도록 하여 구한 결과이다. (그림 12)의 결과에서 보듯이 패칭 스트림의 평균 요구량을 기준으로 요청 간격과 무관하게 그룹을 생성해가는 방법보다는 평균 요청 간격에 따라 임계값으로 설정하고 요청 간격이 지나치게 긴 경우는 새로운 그룹을 생성하는 것이 보다 높은 성능을 얻을 수 있음을 보인다.

5. 결 론

VOD 시스템은 대용량의 연속 미디어를 다수의 클라이언트에게 실시간으로 전송하기 위하여 고성능의 서버를 필요로 하며, 사용자 수의 증가에 따라 서버 부하의 집중 현상이 서비스 용량을 제한한다. 지금까지 이러한 문제를 해결하기 위하여 많은 연구가 진행되었으며, 패칭 기법은 패칭 스트림을 재생하는 동시에 정규 스트림을 버퍼링함으로써 초기 지연시간 없이 서버의 부하량을 크게 줄일 수 있다. 이러한 IP 멀티캐스팅 기법은 서버의 대역폭 요구량을 효과적으로 줄이는 방법이지만, 날로 증가하는 클라이언트를 수용하는 데 한계를 가지고 있으며, IP 멀티캐스트를 기존의 네트워크 환경에서 이용하는데 제약이 따른다. 이러한 문제를 해결하기 위하여 최근 유니캐스트를 이용하여 서비스 용량을 늘리는 P2P(Peer-to-Peer) 미디어 스트리밍에 관한 연구가 활발히 진행 중이다. P2P 미디어 스트리밍 기법은 클라이언트가 증가함에 따라 시스템의 서비스 용량이 증가하는 장점을 가지며, 클라이언트가 미디어를 재생하는 동안 캐싱 서버의 역할을 하므로 서버의 부하 요구량을 효과적으로 줄이는 방법이다.

본 논문에서는 클라이언트의 그룹을 프록시 서버로 활용하여 미디어 스트림 캐싱을 수행하는 새로운 P2Proxy 기법을 제안하였다. P2Proxy에서 각각의 클라이언트는 자신의 요청 시점에 진행 중인 기본 스트림을 저장하고 그룹에 참여한 후, 이 스트림을 원하는 그룹의 다른 클라이언트에게 전송하는 방식이다. VOD 서버는 각 클라이언트의 캐싱 정보에 대한 디렉토리를 관리하는 디렉토리 서버의 역할도 함께 수행한다. 또한 각 클라이언트는 서비스의 요청을 수행할 때, 서버로부터 자신이 속한 그룹의 디렉토리 정보를 가져와 그룹에 참여하도록 한다. 그리고 각 그룹에 참여한 클라이언트들과 서버사이에서 P2Proxy 방법으로 스트림 캐싱이 수행된다. 특히 각 클라이언트는 비디오 요청시 서버의 디렉토리 정보를 전송 받으며, 참여와 이탈 과정을 수행한다.

P2Proxy 기법에서 프록시 서버를 이루는 데 기본이 되는 그룹을 생성하기 위해서는 먼저, 전체 비디오 스트림을 각 클라이언트의 요청 시점에 따라 분산 저장할 때 요청 간격에 대한 임계값을 정하여 이를 초과할 경우 새로운 그룹을 생성하도록 하였다. 또한, 그룹에서 요구되는 패칭 스트림의 합을 구하고 전체 비디오 스트림과의 백분율을 계산하여 정해진 비율을 넘을 경우 새로운 그룹을 생성하도록 제안하였다. 마지막으로, 진행 중인 그룹에 대한 패칭 스트림 요구량의 평균값을 계산하여 이것이 더 이상 줄어들지 않는 경우에 그룹의 참여를 종료하고 새로운 그룹을 생성하도록 하였다.

본 논문은 서버의 부하량을 실험을 통하여 측정함으로써 P2Proxy 기법의 성능을 분석하고, 이를 기존의 P2VoD 기법과 비교하였다. 실험은 우선 평균 요청 간격에 따른 서버 부하량을 측정하였으며, 그 결과 제안된 P2Proxy가 최소 성능을 나타내는 임계치의 경우에도 P2VoD의 어떠한 구성 방법과 비교하였을 때에도 우수한 성능 개선 효과를 보였다. 이는 제안된 P2Proxy의 경우 완전 분산된 캐싱 기법을 활용하여 하나의 그룹이 하나의 기본 스트림만을 서버에 요청하고 클라이언트 요청 시간 간격이 큰 경우만 서버에서 요청하는 반면, P2VoD의 경우 하나의 그룹 내에서도 여러 개의 기본 스트림을 서버에 요구하기 때문이다. 또한, 성능평가는 클라이언트가 활용할 수 있는 버퍼 크기의 변화에 대하여도 서버 부하량을 분석하고 비교하였으며, 임계값의 범위를 분석하기 위하여 그 값을 변화시키면서 서버의 부하 요구량을 평가하여 보았다. 결과적으로 본 논문에서 제안된 P2Proxy 기법은 실험을 통하여 서버의 부하 요구량이 기존의 P2VoD 기법과 비교할 때 우수한 성능을 나타냄을 보였다.

참 고 문 헌

[1] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer Computing," HP Laboratories, Palo Alto, Mar., 2002.

[2] S. Viswanathan and T. Imielinski, "Pyramid Broadcasting for Video on Demand Service," Proc. of ST/SPIE Conference on Multimedia Computing and Networking(MMCN), 1995.

[3] K. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan VoD Systems," ACM SIGCOMM, 1997.

[4] A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic Batching Policies for an On-Demand Video Server," Multimedia Systems, 4(3):112-121, June, 1996.

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

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

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

[8] 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 3rd International Conf. on Computer and Information Science, Los Angeles, CA, Aug., 2004.

[9] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: Peer-to-peer Patching Scheme for VoD Service," Proc. of

the 12th World Wide Web Conference(WWW-03), Budapest, Hungary, May, 2003.

[10] T. Do, K. A. Hua, and M. Tantaoui, "P2VoD: Providing Fault Tolerant Video- on-Demand Streaming in Peer-to-Peer Environment," Technical Report 2003, SEECS, UCF, http://www.cs.ucf.edu/tdo/.

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

[12] S. Sen, J. Rexford, and D. Towsley, "Proxy Prefix Caching for Multimedia Streams," Proc. of the IEEE INFOCOM, Vol. 3, 1998.

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

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

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

[16] D. Xu, M. Hefeeda, S. Hambrusch, B. Bhargava, "On Peer-to-Peer Media Streaming," Proc. of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02), Jul., 02-05, 2002.

[17] D. Tran, K. Hua, and T. Do, "Zigzag: An Efficient Peer-to-Peer Scheme for Media Streaming," Proc. of IEEE INFOCOM'03, San Francisco, CA, Apr., 2003.

[18] S. Sheu, K. A. Hua, and W. Tavanapong, "Chaining: A Generalized Batching Technique for Video-On-Demand Systems," Proc. of IEEE International Conf. on Multimedia Computing and Systems(ICMCS'97), pp.110-117, Ottawa, Canada, Jun., 1997.

[19] K. A. Hua, D. A. Tran, and R. Villafane, "Caching Multicast Protocol for On-Demand Video Delivery," Proc. of the ACM/SPIE Conference on Multimedia Computing and Networking, pp.2-13, San Jose, CA, Jan., 2000.

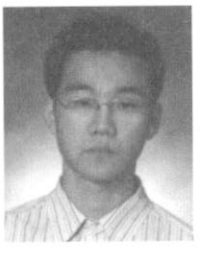
[20] C. J. Kwon, C. K. Choi, and H. K. Choi, "A Peer to Peer Proxy Patching Scheme for VOD Servers," Proc. of 7th Asia Pacific Web Conference, Shanghi, China, Mar., 2005.

권 춘 자

e-mail : kwoncj@mail.kangwon.ac.kr
1986년 2월 한양대학교 전자공학과(학사)
1991년 2월 한양대학교 전자계산학과 (공학석사)
1991년 4월~1994년 4월 한국과학기술정보연구원 데이터베이스운영실 연구원
2002년 3월~2005년 2월 한림성심대학 컴퓨터정보과 초빙교수
2005년 2월 강원대학교 대학원 컴퓨터정보통신공학과(공학박사)
2005년 3월~현재 강릉영동대학 사이버경찰과 초빙교수
관심분야 : 멀티미디어 VOD 시스템, 데이터베이스 시스템, 클러스터 웹 서버 시스템 등

최 치 규

e-mail : ch2k@varovision.com
2003년 2월 강원대학교 컴퓨터공학과 (학사)
2005년 2월 강원대학교 컴퓨터정보통신공학과(공학석사)
2005년 3월~현재 바로비젼(주) 뉴미디어 연구소 연구원
관심분야 : 멀티미디어 시스템, 클러스터 서버 시스템 등



이 치 훈

e-mail : chihun80@hotmail.com
2005년 2월 강원대학교 전기전자정보통신공학부(학사)
2005년 3월~현재 강원대학교 컴퓨터정보통신공학과 석사과정
관심분야 : 멀티미디어 시스템, 클러스터 서버 시스템, 유비쿼터스 시스템 등

최 황 규

e-mail : hkchoi@kangwon.ac.kr
1984년 2월 경북대학교 전자공학과(학사)
1986년 2월 한국과학기술원 전기및전자공학과(공학석사)
1989년 8월 한국과학기술원 전기및전자공학과(공학박사)
1994년 7월~1995년 7월 Univ. of Florida 방문교수
1999년 3월~2001년 2월 강원대학교 전자계산소 소장
2002년 7월~2003년 7월 Univ. of Minnesota 방문교수
1990년 3월~현재 강원대학교 IT특성화학부(대학) 컴퓨터정보통신공학전공 교수
관심분야 : 멀티미디어 시스템, 데이터베이스 시스템, 유비쿼터스 시스템 등