P2P 환경에서의 프록시 캐싱 기법

이치훈*, 최치규*, 최황규*
*강원대학교 컴퓨터정보통신공학과
e-mail: chihun88@hotmail.com, starflower22@hanmail.net,
hkchoi@kangwon.ac.kr

A Proxy Caching Scheme for P2P Environment

Chi Hun Lee*, Chi Kyu Choi*, Hwang Kyu Choi*
*Dept. of Computer and Communication Engineering,
Kangwon National Univdrsity

요 약

본 논문은 P2P 네트워크 환경에서 VOD 시스템의 새로운 프록시 캐싱 기법을 제안한다. P2P 환경에서 비디오 스트림을 그룹에 포함된 다수의 클라이언트에 저장하고, 이러한 클라이언트의 그룹을 Proxy로 활용하는 P2Proxy 기법을 제안한다. 우선 P2Proxy의 그룹에 포함된 클라이언트들과 서버간의 스트림이 전송되는 과정을 보였으며, 둘째 각 그룹의 생성과정과 서멸 과정에 대해 설명한다. 또한 클라이언트의 캐싱 정보를 공유하기 위한 디렉토리 구조를 기술하고 각 클라이언트의 참여 과정과 이탈과정을 보였다. 또 디렉토리 정보를 이용해 그룹에 참여한 다른 클라이언트의 정보를 계산하는 과정을 보였으며 이를 활용하여 재생과 전송 중에 메시지의 교환을 최소로 하도록 하였다. 제안된 P2Proxy 기법은 기존의 P2VoD 기법에 비하여 서버에 대한 부하의 요구량이 훨씬 작은 결과를 보였다.

1. 서론

일반적으로 VOD 시스템은 클라이언트-서버 방식으로 구성되었으며, 다수의 클라이언트의 요청에 따른 부하가 서버에 집중되어 서비스 용량의 한계를 초래한다. 따라서 최근에는 클라이언트 시스템의 자원을 활용하여 서버의 부하를 줄이는 P2P 스트리밍 기법에 대한 연구가 활발히 이루어지고 있다. 즉, 클라이언트가 미디어를 요청하여 스트림을 재생하는 동안 이를 버퍼링했다가 다른 클라이언트에게 재전송함으로써 사용자가 많아질수록 활용도가 높은 클라이언트 시스템의 수가 증가한다는 장점이 있다. 반면에 다수의 클라이언트를 구성하고 제어하기 위한 추가적인 부담이 따른다.

본 논문은 P2P 미디어 스트리밍을 위한 새로운 캐싱 기법인 P2P Proxy를 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 서버에 부과되는 부하의 양을 줄이게 된다.

본 논문의 구성은 2장에서 본 논문과 관련된 연구 결과를 살펴보고, 3장에서 본 논문에서 제안한 P2Proxy 기법에 대해 설명한다. 4장에서는 제안된 기법에 대한 성능을 분석하고, 끝으로 5장에서 본 논문의 결론을 맺는다.

2. 관련연구

VOD 시스템은 다수의 사용자에게 대용량 멀티미디어 데이터를 연속적으로 제공해야 하므로 고성능의 서버를 필요로 한다. 그러나 사용자의 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 1) 네트워크 요구 대역폭의 최적화, 2)프록시(proxy) 캐시 서버 활용 방법, 3) 클라이언트를 캐시 서버로 활용하는 P2P 미디어 스트리밍 방법이 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[1]과 Skyscraper Broadcasting[2]과 같이 일정 시간 단위를 주기로 다수의 사용자에게 브로드캐스팅 하는 배칭(Batching)이 있으며, 일정 분량의 요청을 모아서 한번에 멀티캐스팅하는 동적배칭(Dynamic Batching)기법[3]도 연구되었다. 또한 배칭 기법의 초기 지연 시간을 줄이기 위하여 제안된 패칭 기법[4]은 초기 요청에 대한 정규채널을 생성하고, 이후의 요청에 대하여 지나간 Prefix를 전송받는 패칭 채널을 추가로 생성하는 방법이다. 패칭 기법은 재생과 버퍼링을 동시에 수행하므로 배칭에 비하여 클라이언트 시스템에 높은 성능과 자원(버퍼 공간)을 요구한다. 또, 패칭 윈도우의 크기를 나누는 방식에 따라 크게 Greedy Patching, Grace Patching, Optimal Patching[5]으로 구분할 수 있으며 그 기준은 패칭 윈도우의 크기가 된다.

2.2 프록시 캐싱 기법

프록시란 멀리있는 서버를 대신해 근거리에 위치한 소규모의 서버로서, 서버에 저장된 데이터의 캐싱 역할을 수행한다. 프록시(Proxy)를 활용한 기법은 스트리밍 서버의 캐시를 사용자와 가까운 네트워크 내의 미디어 데이터의 앞부분 또는 자주 요청되는 일부분을 저장할 수 있는 캐시 서버의 역할을 수행하며 그 활용도가

본 논문은 2004년도 정보통신부 지원 기초기술 연구지원 사업 연구결과의 일부임

매우 높다.

프록시 서버를 활용하는 연구 결과로는 Proxy Prefix Caching [6]과 같이 프록시 서버에 비디오 파일의 prefix를 저장해 두어 초기 지연시간을 줄이는 기법이 제안되었다. 또한 Proxy Caching의 최적화를 통하여 네트워크 대역폭의 비용을 최소화 하는 연구도 진행되었다[7][8]. 기존의 패칭 기법의 관점에서 프록시 서버의 perfix와 버퍼 확장을 통해 Optimal Patching Window의 크기를 늘리는 기법도 제안되었다[9][10]. 최근에는 프록시 서버를 Caching Agent로 활용하여 프록시 서버 간의 미디어 스트림의 일부 또는 전체를 전송해 서버의 역할을 대신하는 기법도 제안되었다[11].

2.3 P2P 미디어 스트리밍 기법

P2P 미디어 스트리밍은 사용자 수가 증가함에 따라 서비스 용량이 증가하는 장점이 있어 유니캐스트 기반 환경에서 활발히 연구되고 있다. 먼저, P2P 스트리밍에 참여하는 클라이언트들의 트리를 구성하기 위해 ZIGZAG 기법[12]이 제안되었다. 또한, Chaining 기법[13][14]과 이를 그룹으로 활용한 P2VoD[15]는 선형 구조를 형성한다. 이 기법은 먼저 요청을 수행한 미디어 스트림을 버퍼링했다가 이후에 요청한 클라이언트에게 재전송하는 기법이다. P2P 미디어 스트리밍은 서버로부터 하나의 채널을 소모하고 클라이언트들이 연속적으로 재전송을 수행하는 가상의 멀티캐스트 방식이라고 할 수 있다.

기존의 패칭 기법을 응용해 패칭 스트림을 클라이언트에 전송했다가 재전송하는 P2Cast[16]가 제안되기도 했다. P2Cast 역시 기존의 패칭 기법을 응용 계층에서 수행하기 위해 기본 스트림의 재전송 트리를 형성하고 이를 재전송하는 방식이다. 즉, 기본 채널과 패칭 채널을 모두 먼저 요청한 클라이언트가 이후에 요청한 클라이언트에게 재전송하는 방식이다. 더불어 IP 멀티캐스트를 활용하는 기존의 프록시 패칭 기법과 P2P 스트리밍을 활용하여 정규 채널에 대한 캐싱을 수행하는 기법도 제안되었다[17]. 이 기법은 패칭을 수행할 때 빈번히 생성되는 정규채널의 수를 줄여 서버의 대역폭 요구량을 줄일 수 있다.

3. P2P 기반의 프록시 캐싱 기법

본 장에서는 P2P 기반의 클라이언트 시스템 그룹을 프록시 서버로 활용하는 P2Proxy 기법을 제안한다. P2Proxy 기법에 대한 전반적인 개요와 P2Proxy의 디렉토리 구조를 살펴보고, 클라이언트의 참여,이탈 과정과 그에 따른 스트림의 캐싱 및 복구 과정을 설명한다. 또한 제안된 기법에서 Peer간의 메시지 교환을 최소화 하기 위한 방법에 대해 기술한다. 끝으로 클라이언트의 참여시 P2Proxy의 그룹의 참여 가능 여부의 기준이 되는 임계값(Threshold)의 적용방법에 대해 설명한다.

3.1 P2Proxy 기법의 개요

그림 1은 본 논문에서 제안한 P2Proxy 기법의 VOD 서버와 다수의 클라이언트인 P2Proxy 그룹으로 구성된다. P2Proxy의 각 클라이언트는 캐싱을 수행하기 위한 저장 공간을 갖고 있으며, 각 클라이언트는 자신이 속한 그룹에 저장된 미디어 스트림을 전송받아 사용하고 없는 부분만을 서버에서 전달받아 사용한다.

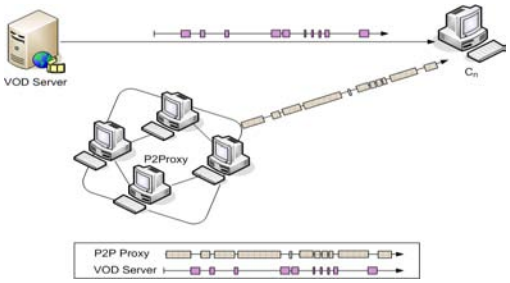

그림 1. P2Proxy 시스템의 구성도

P2Proxy 시스템에서 VOD 서버는 비디오 파일 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 Peer가 동일한 정보를 유지하도록 한다.

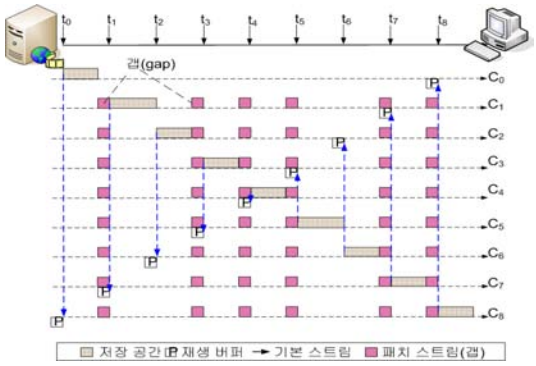

그림 2. P2Proxy의 스트림 전송 과정

그림 2는 각 클라이언트가 저장하고 있는 스트림의 양과 서버에 요구되는 부하의 양을 보여주는 것으로, 가로선은 요청 시간이 다른 각 클라이언트에 대한 미디어 스트림을 나타낸다. 그림 2의 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. 어두운 사각형은 각 클라이언트가 저장하고 있는 기본 스트림의 시간적인 위치와 분량을 나타낸다. 또 사각형 P는 현재 시점에서 각 클라이언트의 재생 위치를 나타낸다. 빗금으로 표시된 사각형은 P2Proxy의 그룹에 속하지 않아 클라이언트 버퍼 사이의 내용이 끊긴 부분이다. 이를 갭(gap)이라고 정의하며, 갭은 모든 클라이언트가 서버로부터 전송받아 재생해야 한다. P2Proxy에 존재하지 않는 갭들을 전송하기 위한 채널을 패칭 채널(Patching Channel)로, 이 스트림을 패칭 스트림(Patching Stream)이라 한다. 그림 2는 P2Proxy의 그룹에 속한 각 클라이언트가 자신의 요청 시간에 대한 기본 스트림을 캐싱하고 있는 상태를 볼 수 있다. 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 P2Proxy그룹이나 서버로부터 전송받는다. 그림 2에서 C_0에서 C_7의 요청이 이루어진 후에 클라이언트 C_8의 요청이 t_8의 시점에 이루어졌다. 이 때, 각 클라이언트는 자신의 재생 시점에 해당하는 스트림을 저장하고 있는 클라이언트들로부터 스트림을 전송받아 사용한다.

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 바뀌는 특성을 지닌다. 더불어 요청 순서에 따라 트리 또는 체인 구조를 형성하는 것이 아니라 하나의 기본 스트림 단위로 P2Proxy의 그룹을 형성하고 그룹 내의 멤버들에 저장된 스트림을 필요한 시간에 전송하는 방식이다. 즉, P2Proxy 내의 멤버는 요청 순서가 늦더라도 자신이 가진 스트림을 재생을 원하는 클라이언트에게 전송할 수 있다. 마지막으로 기존의 방식이 디렉토리 서버를 이용하거나 메시지 교환을 이용하는 방식인데 비하여 제안된 기법은 VOD서버의 디렉토리를 활용한다. 즉, 서버로부터 그룹의 디렉토리 정보를 전송받아 메시지를 통하여 그룹에 참여하고 자신의 상태 변화를 그룹의 구성원에게 알려 동기화를 이루는 방식이다.

3.2 클라이언트의 참여와 스트림의 캐싱

P2Proxy에서는 각 클라이언트가 미디어의 재생을 요청 할 때, 자신이 속한 그룹의 정보를 서버로부터 전송받아 그룹의 모든 구성원에 참여 메시지를 전송한다. 또한 같은 그룹에 속한 클라이언트와 서버는 메시지를 교환하여 동일한 디렉토리를 유지한다. 이 때 클라이언트는 P2Proxy의 캐싱 서버로서의 역할을 수행한다.

그림 3 은 클라이언트의 순차적인 요청에 대한 참여와 그에 따른 캐싱의 수행과정을 나타낸다. 즉, t_0, t_1, t_2,…, t_5의 시점에 들어온 요청에 대하여 각 클라이언트 C_0, C_1, C_2, …, C_5에 대한 그룹의 생성과 그에 따른 캐싱의 수행 과정을 나타낸다. 먼저 t_0 시간에 요청을 수행한 C_0에 대한 새로운 기본 채널이 생성되고 그에 대한 그룹 G_1이 생성된다. 이후 t_1 시점에 클라이언트 C_1의 요청이 있으면 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 대해서도 같은 과정을 반복한다.

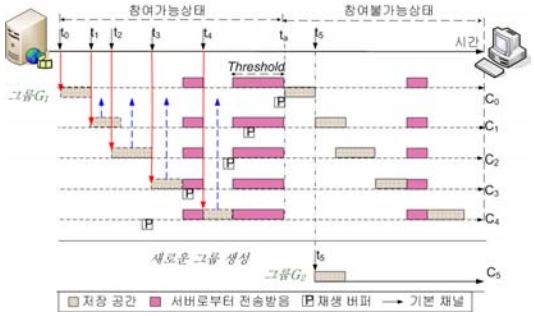

그림 3. 클라이언트의 참여 및 캐싱의 수행

이때, 각 요청 사이의 시간차는 클라이언트의 버퍼 크기내이므로 C_1, C_2, C_3에 스트림이 연속적으로 저장된다. 그러나 이후의 t_4 시점에 요청이 발생하면, t_3과 t_4의 요청간격이 C_3의 버퍼 공간을 초과하므로 C_3과 C_4의 버퍼 사이에 저장되지 못하는 갭이 발생한다. 이와 같이 본 P2Proxy 기법은 각 클라이언트의 버퍼 사이에 정해진 범위 이내에서 스트림이 끊기는 것을 허용하지만. 끊기는 스트림의 크기가 큰 경우에는 요청한 클라이언트를 허용하지 않고 새로운 그룹을 생성한다. 즉, 그림 3에서 t_4 시간에 요청이 이루어진 후 정해진 임계 시간(Threshold) 동안 요청이 들어오지 않아 t_a 시점에 그룹을 참여 불가능 상태로 변경한다. 이후 t_5 시점에 요청이 이루어지면 기존의 그룹이 참여 불가능 상태이므로 새로운 기본채널을 할당하고 이에 대한 그룹 G_2를 생성한다.

그림 3 에서 t_a 시점에 그룹 G_1이 참여 불가능한 상태로 변경되면 G_1의 TAIL의 재생 지점이 지난 클라이언트들의 버퍼를 회수하여 기본 스트림이 캐싱하는 데 재사용한다. 즉, 그룹이 참여 불가능 상태로 변경되면 그룹에서 더 이상 사용하지 않는 버퍼가 발생하므로 이를 활용하여 다시 기본 스트림을 캐싱할 수 있다. 그림 3 에 나타난 바와 같이 t_a의 시점에 G_1이 참여 불가능 상태로 변경되면 G_1의 TAIL인 C_1의 재생 지점이 지난 C_0와 C_1의 버퍼를 다시 회수하여 진행 중인 기본 스트림을 저장할 수 있다. 이 과정은 기본 스트림이 진행되면서 계속하여 반복적으로 이루어진다.

위와 같이 P2Proxy의 클라이언트 참여 과정은 각 클라이언트의 요청에 대한 그룹의 참여 과정과 그에 따른 스트림의 캐싱 과정으로 이루어져 있다. 또, 그룹의 상태가 임계시간 값을 지나게 되면 그룹이 참여 불가능 상태가 되고 이후의 요청은 새로운 그룹을 생성한다. 또한 P2Proxy의 그룹은 그 자신의 크기를 하나의 주기를 가지고 반복적으로 나타난다.

3.3 클라이언트의 이탈과 복구

P2P 스트리밍 방식은 불안정한 클라이언트 시스템을 캐싱 서버로 활용하므로 클라이언트의 불규칙한 행동에 의한 이탈이 발생한다. 따라서 P2P를 활용하는 스트리밍 방식은 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 갭은 패칭 채널을 이용하여 서버로 부터 전송받아야 한다. 또한 그룹 내의 모든 클라이언트가 이탈할 경우 그룹도 소멸된다.

그림 4 은 각각 t_0, t_1, t_2, t_3와 t_4의 시간에 $C_1, C_2 C_3$ 및 C_4의 클라이언트가 P2Proxy의 그룹을 구성하여 진행 중인 그림이다. 이때, 임의의 순간 t_a에서 단절된 스트림의 크기가 임계시간에 도달하여 그룹이 참여 불가능 상태로 바뀌게 된다. 따라서 P2Proxy 그룹은 자신의 TAIL인 C_1의 재생 지점이 지나 재사용이 가능한 버퍼를 찾아 기본 스트림의 캐싱을 계속 수행한다. 이때, 그림 7 에서 나타난 바와 같이 t_b의 시점에 클라이언트 C_2가 이탈하면 C_2가 가진 버퍼 공간이 사라져 새로운 갭이 발생한다. 따라서 각 클라이언트는 서버로부터 C_2의 이탈에 의하여 발생한 갭만큼의 크기를 추가로 패칭한다.

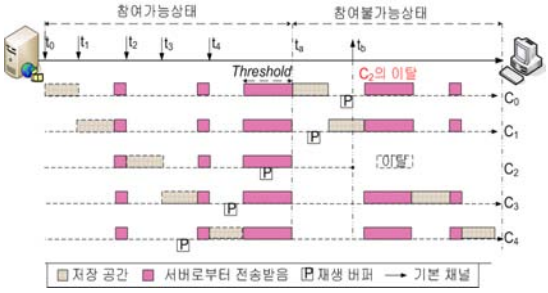

그림 4. 클라이언트의 이탈에 따른 복구 과정

4. 실험 및 성능평가

4.1 실험 환경

P2Proxy 기법에 대한 성능 평가를 위하여 표 2의 파라미터를 이용하여 서버 부하량을 측정하였다. 사용자의 요청은 비디오에 대한 Poisson 분포를 따르며, 10시간 분량의 요청에 대하여 평가했다. 또, CBR(Constant Bit Rate)의 비디오 스트림으로 가정할 때, VOD 서버에 요구되는 부하량은 채널 수에 비례한다. 따라서 VOD 서버에 요구되는 부하량을 비교하기 위하여 전체 시간에 대해 소모되는 스트림 수의 평균을 구하여 비교했다. 또한, 실험을 10회 반복한 값을 평균하여 결과의 정확성을 기했다.

먼저 요청 간격에 따른 부하 요구량에 대한 P2VoD기법과 제안된 기법을 비교했으며, 제안된 요청 간격에 따른 부하 요구량의 변화를 서로 다른 임계시간(Threshold)에 대해 비교했다. 클라이언트의 버퍼 사이즈에 따른 부하요구량을 비교했고, 각 임계값에 따른 부하 요구량의 변화를 살펴보았다.

표 2. 성능평가 파라미터

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

4.2 성능 평가 결과

그림 5 는 평균 요청 간격의 변화에 대한 부하의 요구량은 기존의 P2VoD 기법과 비교한 결과로서. 기존이 P2VoD 기법에 대한 결과는 점선으로 표시되었으며, 제안된 기법인 P2Proxy의 결과는 실선으로 표시되었다. 그림 5의 각 클라이언트는 10초 분량의 버퍼를 가지고 있다. 그래프에서 아래쪽의 밝은 사각형으로 표현된 것이 제안된 기법에서 Threshold값을 10초로 주었을 때의 값이며, 어두운 사각형으로 표현된 것은 Threshold값이 0일 때의 값이다. 또한 그래프의 위쪽은 각각 K 값이 2, 3, 4인 P2VoD의 부하 요구량을 나타낸다. P2VoD기법의 경우는 K 값이 증가함에 따라 그 부하의 요구량도 늘어나는 것을 볼 수 있다. 그림 5에서 제안된 기법의 부하요구량이 기존의 P2VoD에 비하여 낮은 것을 볼 있다.

그림 6 은 버퍼 크기에 따른 P2VoD와 P2Proxy의 부하 요구량을 각각 점선과 실선으로 나타냈다. 그림에서 진한 도형으로 표시된 것은 평균 요청 간격이 5초인 것을 의미하여 밝은 도형으로 표시된 부분은 평균 요청 간격이 10초인 경우를 의미하며, P2VoD의 K 값은 2와 3을, P2Proxy의 임계값으로는 0과 10을 사용하였다. 그림 6 에서 볼 수 있듯이 버퍼 크기가 증가할수록

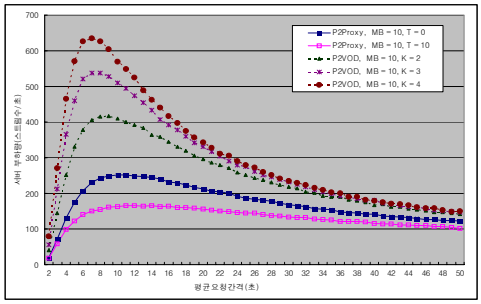

그림 5. P2VoD와 P2Proxy의 서버부하량 비교

서버에 대한 부하 요구량은 작아진다. 특히 동일한 조건에서 P2VoD와 비교할 때 P2Proxy가 높은 성능을 나타냈다. 또한 요청 간격이 10초인 경우가 기본적으로 부하량이 적으므로 P2VoD나 P2Proxy의 버퍼 크기가 작은 경우는 부하요구량이 작게 나타난다. 그러나 버퍼 양이 늘어나 버퍼 활용 효과가 커지면 평균 요청 간격이 작은 경우에 서버의 부하요구량이 오히려 작게 나타난다.

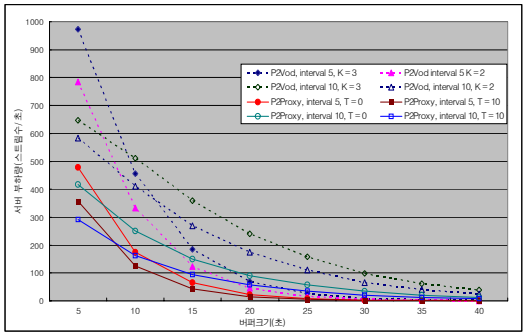

그림 6. 버퍼 크기에 따른 부하량 비교

그림 7 은 그룹을 생성하는 기준이 되는 임계값의 적용 방법에 따른 부하량을 비교한 것으로, 짙은 마름모는 갭의 크기를 허용하지 않은 것을 나타내며, 네모는 갭의 크기를 10초 범위에서 허용한 것을 나타낸다. 또한 삼각형과 별 모양은 갭의 크기가 20초인 것과 40초인 경우를 나타낸다. 그림 7 의 결과에서 보듯이 적절한 범위 내에서 갭을 허용하고 요청간격이 지나치게 긴 경우에는 새로운 그룹을 생성하는 것이 보다 높은 성능을 얻을 수 있다.

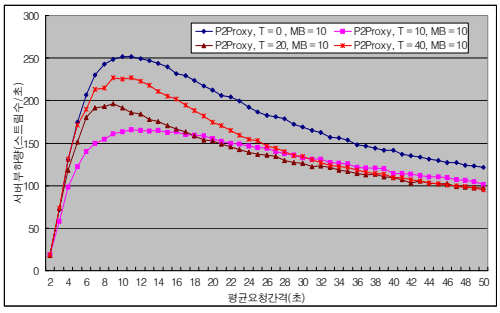

그림 7. Threshold 값에 따른 부하량의 비교

5. 결 론

본 논문은 클라이언트 그룹을 이용하여 미디어 스트림의 캐싱을 수행하는 P2Proxy 기법을 제안하였다. 각각의 클라이언트는 자신의 요청 시점에 진행 중인 기본 스트림을 저장하고, 그룹의 다른 클라이언트에게 전송하는 방식이다. 또한, VOD 서버는 각 클라이언트의 캐싱 정보에 대한 디렉토리를 관리하는 디렉토리 서버의 역할을 함께 수행한다. 각 클라이언트는 요청시 서버의 디렉토리 정보를 전송받으며, 참여와 이탈과정을 수행한다. 본 논문에서 제안된 P2Proxy기법은 실험을 통하여 서버의 부하 요구량을 측정하였으며, 기존의 P2VoD 기법과 비교하여 우수한 성능을 보였다.

참고문헌

[1] S. Viswanathan and T. Imielinski. "Pyramid broadcasting for video on demand service", In Proceedings of ST/SPIE Conference on Multimedia Computing and Networking (MMCN), 1995.

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

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

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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