주문형 비디오 분배를 위한 웹-캐싱 멀티캐스트 전송 기법

김 백 현† · 황 태 준† · 김 익 수††

요 약

본 논문은 동일 비디오를 저장하고 있는 n개의 단말노드에서 생성되는 전송채널들을 병합하여 네트워크 대역폭을 n배 줄일 수 있는 멀티캐스트 전송 기법을 제안한다. 클라이언트는 패칭 기법을 사용하여 전송중인 멀티캐스트 전송 트리에 가입함으로서 지연 없는 서비스를 제공받을 수 있으며 패칭과 멀티캐스트 채널을 통하여 전송되는 데이터들을 동시에 수신하여 재생한다. 비디오에 대한 요청빈도수가 증가할수록 요청된 비디오가 단말노드 사이에 저장될 확률은 증가하기 때문에 이를 이용하여 서비스를 제공함으로서 서버의 필요 전송채널수를 감소시킬 수 있다. 서버는 비디오 재생이 HEN 사이에 저장된 데이터의 양을 초과하여 지속되는 경우에만 새로운 멀티캐스트를 생성하여 전송한다. 제안된 기술은 비디오의 인기도에 따라 멀티캐스트 그룹 간격이 동적으로 변화되기 때문에 서버의 부하 및 네트워크 대역폭을 크게 감소시킬 수 있다. 시뮬레이션 결과로부터 제안된 멀티캐스트 전송 기법은 서버의 부하 및 필요 전송 채널의 수를 감소시켜 시스템 성능을 향상시킬 수 있는 방법을 제시하고 있음을 확인하였다.

키워드 : 웹 캐싱, 멀티캐스트, 프락시, 패칭

Web-Cached Multicast Technique for on-Demand Video Distribution

Backhyun Kim† · Taejune Hwang† · Iksoo Kim††

ABSTRACT

In this paper, we propose multicast technique in order to reduce the required network bandwidth by n times, by merging the adjacent multicasts depending on the number of HENs (Head-End-Nodes) n that request the same video. Allowing new clients to immediately join an existing multicast through patching improves the efficiency of the multicast and offers services without any initial latency. A client might have to download data through two channels simultaneously, one for multicast and the other for patching. The more the frequency of requesting the video is, the higher the probability of caching it among HENs increases. Therefore, the requests for the cached video data can be served by HENs. Multicast from server is generated when the playback time exceeds the amount of cached video data. Since the interval of multicast can be dynamically expanded according to the popularity of videos, it can be reduced the server's workload and the network bandwidth. We perform simulations to compare its performance with that of conventional multicast. From simulation results, we confirm that the proposed multicast technique offers substantially better performance.

Key Words : Web-caching, Multicast, Proxy, Patching

1. 서 론

본 논문은 네트워크 트래픽 및 서버의 부하를 감소시키고, 프락시 사이에 동일 비디오가 중복 저장되는 것을 방지하며 균등부하를 갖도록 하는 프락시 캐싱 기법을 사용하는 새로운 멀티캐스트 전송기법을 제안한다. 초기 서비스 지연을 방지하기 위하여 동시에 최대 2개의 다운로드 스트림을 사용하여 비디오의 재생을 가능하게 하는 패칭을 사용한다[1]. 제안된 기법은 패칭 및 캐싱 기법을 사용하며 실시간 스트리밍 서비스에 적합한 특징을 갖고 있다. 단말노드 (HEN: Head-End-Node)는 프락시로서 네트워크상에서 전송되는 비디오 스트림들을 저장하며, 클라이언트의 요청순서에 따라 논리적으로 정렬된다. 제안된 기법 하에서 하나의 비디오는 멀티캐스트 그룹 간격 I_m과 동일 크기의 블록 단위로 분리되어 요청순서에 따라 단말노드로 전송되며, 각 단말노드는 동일비디오 v를 위하여 단지 I_m 분량의 비디오만을 저장한다. 만약 동일비디오를 저장하고 있는 단말노드의 수가 n 이라면 $n \cdot I_m$ 시간 동안의 비디오 데이터가 저장되었기 때문에 비디오의 재생 구간 $n \cdot I_m$에 대한 요청은 비디오 서버가 아니라 단말노드들에 의하여 생성되는 멀티캐스트 m_t를 사용하여 제공될 수 있다. 임시 멀티캐스트 m_t를 이용한 서비스 개시이후 처음 I_m 시간 동안 동일 비디오를 요청한 클라이언트는 지연 없는 서비스를 제공받기 위하여 패칭 기법을 사용하며 처음 I_m 시간 내에서만 수행된다. 패칭이 수

※ 본 연구는 산업자원부, 한국산업기술평가원 지정 인천대학교 멀티미디어 연구센터의 지원에 의한 것입니다.
† 준 회 원 : 인천대학교 정보통신공학과 대학원 박사과정
†† 정 회 원 : 인천대학교 정보통신공학과 교수
논문접수 : 2005년 7월 21일, 심사완료 : 2005년 10월 20일

행된 후 클라이언트는 단말노드에 저장된 $n \cdot I_m$ 데이터를 초과하여 재생을 지속할 때까지 임시 멀티캐스트 m_t의 가입 상태를 유지한다. $|v|$를 비디오의 재생시간이라고 하면 비디오의 나머지 부분 $|v| - n \cdot I_m$ 은 서버에서 생성되는 동적 멀티캐스트 m_d를 통하여 제공된다. 동적 멀티캐스트 m_d가 생성되면 동일비디오 v를 임시 멀티캐스트 m_t에 가입하여 시청하고 있는 모든 클라이언트들은 생성된 동적 멀티캐스트 m_d에 가입하며 이를 통하여 수신되는 데이터를 저장하고 재생한다. 따라서 동적 멀티캐스트 m_d 의 생성 간격은 $\lfloor n \cdot I_m \rfloor$ 이 되기 때문에 제안된 멀티캐스트 기법은 전송 채널의 수를 1/n으로 줄일 수 있게 된다.

본 논문은 2장에서는 웹-캐싱 기법들을 설명하고 있으며, 3장에서는 분산 웹 캐싱 멀티캐스트 기법을 제안하며, 4장에서는 제안된 기법의 성능을 분석하며, 5장에서는 제안된 기술을 시뮬레이션하고 결과를 분석한다. 마지막으로 6장에서는 결론에 대하여 기술한다.

2. 웹-캐싱 및 멀티캐스트 전송 기술

멀티캐스트 전송기법은 클라이언트에게 단절 없는 스트리밍 서비스를 제공하기 위하여 요구되어지는 전송 대역폭을 최소화할 수 있는 기법 중에 하나이며, 클라이언트들이 하나의 전송채널을 통한 비디오 스트림의 공유를 가능하게 한다[1, 2, 3, 4]. 멀티캐스트를 사용하는 전송기법은 배칭과 패칭의 두 가지로 구분할 수 있다[5, 6]. 배칭 기법은 가능한 한 많은 요청들을 하나의 멀티캐스트 전송채널을 사용하여 제공하기 위하여 멀티캐스트 그룹 간격 I_m 내의요청들을 지연시켜 서비스를 제공하는 방식이다. 만약 I_m 시간동안 동일 비디오에 대한 요청이 발생한다면 매 I_m 시간마다 주기적으로 멀티캐스트 전송 채널이 생성되게 된다. 따라서 일부 클라이언트들은 새로운 전송 채널이 생성될 때까지 $\lceil I_m \rceil$시간 동안 대기하여야 한다. 패칭 기법 하에서 비디오를 요청하는 클라이언트는 해당 비디오를 전송하는 멀티캐스트 채널이 존재하는 경우 해당 전송채널에 가입한다. 그러나 비디오의 시작부분과 가입한 멀티캐스트 채널 사이에는 비디오 재생 시차가 존재하기 때문에 비디오 서버는 두 채널의 차이만큼의 데이터를 유니캐스트를 사용하여 전송한다. 따라서 클라이언트는 요청이후 비디오 재생까지의 지연이 발행하지 않게 된다. 그러나 요청빈도가 높은 경우 서버의 부하가 커지게 되며 대량의 네트워크 대역폭을 필요로 하는 단점이 발생하게 된다.

인접한 멀티캐스트 채널들을 병합함으로서 전송 채널의 수를 줄이기 위하여 광고나 예고편 등과 같은 정보를 비디오 스트림에 삽입하여 앞선 스트림의 재생 지점을 뒤의 스트림과 일치시키는 방법 및 앞선 스트림은 초당 재생프레임 수를 낮추고 뒤의 스트림은 초당 재생 프레임 수를 높임으로서 인접한 두개의 스트림들을 병합하는 방법이 제안되었다[7].

멀티캐스트 트래픽을 분산시키기 위하여 제한된 크기를 갖는 다층 수직구조의 클러스터들을 사용하는 ZIGZAG 방식이 제안되었다[8]. 각각의 클러스터는 하나의 헤드 노드와 다수의 비-헤드 노드들로 구성된다. H를 계층의 수라고 하면, 계층 $j < H$에 있는 클러스터의 헤드노드는 $j < H-1$을 만족하는 경우 계층 j+1의 멤버가 된다. 각 클러스터의 비-헤드 노드들은 그들이 속한 클러스터의 헤드노드로부터가 아니라 다른 클러스터의 헤드노드로부터 데이터를 수신한다. 이 방식은 클러스터에 소속된 노드들의 수를 증가시킴으로서 데이터 전송에 필요한 홉 수를 줄일 수 있으며, 전송 데이터들이 다른 클러스터들을 통하여 전송되기 때문에 네트워크 병목현상의 발생을 억제할 수 있는 장점을 갖고 있다.

프락시는 서버와 클라이언트 사이에 존재하는 일종의 서버로서 네트워크 성능을 개선할 수 있으며 서버의 부하를 감소시킬 수 있다. 일반적인 프락시 기법은 인기 있는 비디오를 저장하고 있는 프락시의 경우 다른 프락시와 비교하여 상대적으로 큰 부하를 갖게 되며 동일한 비디오가 중복적으로 저장되는 문제점을 갖고 있다[9, 10, 11].

다양한 네트워크 환경에서 고품질 및 낮은 서비스 지연을 갖는 비디오 분배 방식이 제안되었다[12]. 부분 캐싱을 기반으로 하는 프락시 기법에서 캐쉬 버퍼의 효율적인 사용을 위하여 비디오 스트림은 블록 단위로 나누어져서 전송된다. 캐쉬 히트인 경우 프락시는 저장된 데이터를 요청된 수준의 QoS에 맞추어 재구성한 후 클라이언트에게 전송한다.

캐쉬는 전체 네트워크 트래픽에 영향을 줄 수 있기 때문에 어떠한 데이터가 저장되어야 하며 어떠한 데이터가 삭제되어야 하는지를 결정하는 캐쉬 알고리듬의 신중한 선택이 필요하다. 현재 대부분의 프락시 시스템은 구현의 단순성 때문에 LRU 캐싱 알고리듬을 사용한다. 그러나 LRU는 아이템 크기, 지연 및 인기도에 따른 요청 빈도수 등을 사용하지 않기 때문에 최상의 캐쉬 히트율을 얻기가 어렵다는 단점이 있다[13].

본 논문은 균등 부하를 갖는 분산 웹 캐싱 기법을 사용하여 사용자들에게 지연 없는 동영상을 제공하는 주문형 시스템을 제안한다. 비디오 동영상의 경우 대용량의 데이터를 장시간 동안 전송하여야 하기 때문에 전송경로 상에 매우 큰 트래픽을 유발하게 된다. 따라서 멀티캐스트 전송기법을 사용하여 발생되는 트래픽의 양을 줄이도록 하였으며, 사용자에게 서비스가 지연되어 제공되는 문제점은 패칭 기법을 사용하여 해결하였다. 제안된 캐싱 기술은 단말노드들에게 캐쉬를 적용하여 서버로부터 전송되는 비디오 스트림을 저장하도록 하였으며, 서버로부터 전송되는 비디오 스트림은 단말노드에 저장된다. 서버는 동일 비디오가 다수의 사용자들에 의하여 요청되는 경우 처음 한번만 전송하게 되며 이후의 서비스 요청은 단말노드에 저장된 데이터를 사용하여 서버와는 무관하게 제공된다. 단말노드들에 저장되는 비디오 데이터는 요청순서 및 인기도에 따라 순차적으로 분산 저장되도록 하여 비디오의 동일 데이터가 다수의 단말노드에 중복 저장되는 것을 방지하였으며, 각 단말노드들에 부

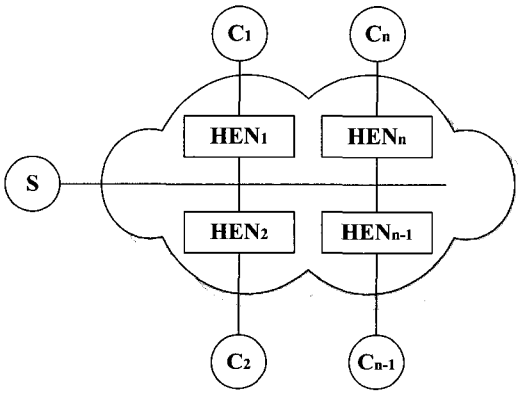
하가 동일하게 발생되어 특정 단말노드가 상대적으로 큰 부하를 갖게 되는 문제를 해결하였다.

3. 동적 멀티캐스트 간격을 갖는 웹 캐슁 시스템

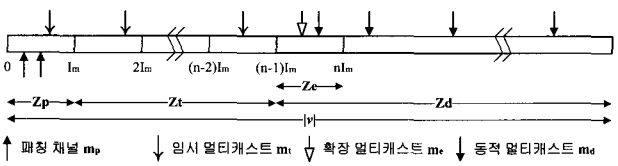
패칭은 하나의 멀티캐스트 채널과 하나의 패칭 유니캐스트 채널 등 동시에 2개의 다운로드 채널들을 사용하여 전송되는 스트림들을 수신하여 사용하는 방식이다. 멀티캐스트 채널은 전체 비디오 데이터 |v|를 전송하며, 유니캐스트 채널은 가장 최근의 멀티캐스트 전송채널의 재생 시간으로부터 요청 시간까지의 시간 차이만큼의 데이터를 전송하는데 사용된다. 클라이언트는 패칭 유니캐스트 채널을 통하여 전송되는 데이터를 재생하는 동안 멀티캐스트를 통하여 전송되는 데이터를 자신의 버퍼에 저장한다. 패칭 채널을 통한 재생이 끝나면 클라이언트는 자신의 버퍼에 저장된 데이터를 사용하여 지속적인 비디오 재생을 수행한다. 그러나 서버가 유일한 비디오 데이터의 공급원이기 때문에 배칭 멀티캐스트 기법과 비교하여 보다 많은 네트워크 대역폭을 요구하게 된다.

본 논문은 단절 없는 서비스 제공을 위하여 요구되어지는 네트워크 대역폭을 줄이기 위하여 균등부하 프락시 및 패칭 기법을 사용한 웹-캐슁 기법을 제안한다. (그림 1)은 제안된 웹-캐슁 네트워크를 보여주고 있으며, S 및 C_n는 각각 서버 및 단말노드 (HEN) n 에 있는 클라이언트를 나타내고 있다 [14]. 단말노드는 요청 빈도로 표시되는 인기도에 따라 전송되는 비디오를 저장하며 클라이언트에게 멀티캐스트 채널을 사용하여 서비스를 제공하는 역할을 수행한다.

각각의 단말노드는 저장하고 있는 데이터에 대한 정보를 공유하기 위하여 캐슁 테이블 CT(비디오 ID, 순차번호, 멀티캐스트 ID)을 생성하며 매 I_m마다 주기적으로 나머지 단말노드들에게 전송한다. 순차번호는 비디오의 i 번째 블록 $v[i \cdot I_m, (i+1)I_m]$을 나타내며, $v[i \cdot I_m, (i+1)I_m]$은 재생시간 $i \cdot I_m$부터 $(i+1)I_m$ 까지의 비디오 v의 세그먼트를 표시한다. 단말노드는 공유된 캐슁 테이블을 사용하여 각각의 비디오에 대한 스위칭 테이블 ST(순차번호, 멀티캐스트 ID, 단말노드 ID)을 생성한다. 생성된 스위칭 테이블은 단말노드가



(그림 1) 웹-캐슁 멀티캐스트 네트워크



(그림 2) 패칭 기법을 사용한 전송 채널의 순서도

관리하는 클라이언트들에게 전송되며, 이 정보는 클라이언트들이 요청한 비디오가 어느 멀티캐스트 채널로 전송되는지를 확인하여 해당 채널로 가입할 수 있도록 한다. 단말노드에 저장된 비디오 데이터에 변경이 발생되면 단말노드 및 클라이언트들이 최신 정보를 유지, 관리할 수 있도록 하기 위하여 변경 내용은 즉시 전송된다.

(그림 2)는 제안된 패칭 멀티캐스트 기법을 사용하는 경우 전송되는 채널들을 보여주고 있다. 비디오는 구성된 단말노드의 수 및 요청 시간에 따라 패칭 구간 z_p, 임시 구간 z_t, 확장 구간 z_e, 동적 구간 z_d 등 총 4개의 구간으로 구분되며, 각각의 구간에서 사용되는 전송 채널은 패칭 채널 m_p, 임시 멀티캐스트 m_t, 확장 멀티캐스트 m_e, 동적 멀티캐스트 m_d라 한다. 시간 t1에서 단말노드 HEN1에 있는 클라이언트 C_1이 비디오 v를 요청하였다고 가정하자. 요청을 수신한 서버는 비디오 v를 전송하기 위하여 새로운 비디오 스트림을 생성하여 HEN1으로 유니캐스트 채널을 사용하여 전송한다. 만약 비디오의 시작 시간을 0이라 하면, HEN1은 서버로부터 수신되는 데이터 $v[0, I_m]$를 저장하며, 멀티캐스트 채널을 생성하여 클라이언트 C_1에게 저장된 데이터를 전송한다. HEN1에 의하여 생성된 멀티캐스트 채널은 멀티캐스트 그룹 간격 I_m 동안만 지속되며, 임시 멀티캐스트 m_t라고 한다. 또한 멀티캐스트 m_t는 동일 비디오 v의 블록들을 저장하고 있는 단말노드들에 의해서만 생성될 수 있다.

$0 < t2 - t1 < I_m$인 시간 t2에서 HEN2에 있는 클라이언트 C_2가 동일 비디오 v를 요청한다면, 스위칭 테이블로부터 HEN1에 의해서 생성된 멀티캐스트 m_t에 가입한 후 데이터를 수신하며 자신의 버퍼에 저장한다. 동시에 HEN1에 의하여 생성된 패칭 채널 m_p를 통하여 $\Delta = t2 - t1$ 시간 동안의 비디오 데이터를 수신하여 재생하며, 이것은 $S(v[0, \Delta], m_p)$로 표시된다. 따라서 클라이언트가 비디오를 요청한 시간이 I_m보다 작은 경우 실시간 서비스를 제공하기 위하여 패칭기법이 사용되면 이 구간을 패칭 구간 z_p이라 한다. 전송 트래픽을 네트워크상에 분산시키기 위하여 서버는 처음 I_m 분량의 비디오를 HEN1에 전송하며, 두 번째 I_m 분량의 비디오를 HEN2로 전송한다. Δ 시간이 지난 후 클라이언트 C_2는 패칭 채널의 접속을 끊고 멀티캐스트 m_t를 통한 데이터만을 수신하여 비디오를 재생한다.

$I_m \leq t3 < nI_m$ 인 시간 t3에 패칭 기법을 지원하지 않는 임시 구간 z_t이라고 하는 두 번째 I_m 구간에 클라이언트 C_1이 진입하면, HEN2는 클라이언트 C_1을 위하여 새로운 임시 멀티캐스트 m_t를 생성한다. 클라이언트 C_2는 클라이언트 C_1과 동일한 멀티캐스트 그룹 멤버이므로 HEN2에 의하여 생성된 멀티캐스트 m_t 에 가입하며 비디오 데이터 $v[I_m, 2I_m]$

을 수신하여 저장한다. 따라서 두 번째 I_m 이후의 구간에서는 늦게 멀티캐스트 그룹에 가입한 클라이언트를 위하여 새로운 전송 채널을 할당할 필요가 없게 된다.

$t3 < t4 < nI_m$ 이며 $t4 - t1 > I_m$인 시간 t4에서 HEN3의 클라이언트 C_3가 비디오 v를 요청하면 서버로부터 전송되는 비디오는 I_m 단위로 HEN1, HEN2, HEN3에 순서대로 저장된다. 이 경우 C_3는 HEN1으로부터 비디오를 수신하여야 하지만 현재 HEN1으로부터 전송중인 멀티캐스트 채널이 없기 때문에 HEN1은 새로운 임시 멀티캐스트 m_t를 생성한다. 따라서 클라이언트 C_3는 HEN1의 $v[0, I_m]$, HEN2의 $v[I_m, 2I_m]$, HEN3의 $v[2I_m, 3I_m]$ 순서로 비디오 v를 재생하게 된다. t4 이후에 HEN1, HEN2, HEN3를 제외한 다른 HEN으로부터 비디오 v에 대한 요청이 발생하지 않을 경우 HEN들에 저장된 비디오 v의 양은 $3I_m$이 된다.

비디오의 나머지 부분 $v[3I_m, |v|]$은 단말노드들 사이에 저장되어 있지 않기 때문에 서버는 새로운 멀티캐스트 전송 채널인 동적 멀티캐스트 m_d를 생성한다. 단말노드들 사이에 비디오 데이터가 저장되지 않은 구간을 동적 구간 z_d라고 한다. 동적 멀티캐스트 m_d들의 간격은 동일 비디오를 저장하고 있는 단말노드들의 수 n에 의하여 $\lfloor n \cdot I_m \rfloor$의 값을 갖게 되며, 위의 경우는 $\lfloor 3I_m \rfloor$이 된다. 동적 멀티캐스트 m_d가 생성되면 구간 z_p 및 구간 z_t에 있는 동일비디오를 요청한 클라이언트들은 동적 멀티캐스트 m_d에 가입한다. 따라서 구간 z_t에 있는 클라이언트의 재생 순서는 $S(v[I_m, 3I_m, m_t), S(v[3I_m, |v|], m_d)$ 가 되며, 구간 z_p에 있는 클라이언트는 $S(v[0, t_t], m_p)$, $S(v[t_t, 3I_m], m_t)$, $S(v[3I_m, 3I_m + \delta], m_e)$, $S(v[3I_m + \delta, |v|], m_d)$ 가 되며, t_t는 구간 z_p에서 임시 멀티캐스트 m_t에 의하여 비디오 데이터가 저장되는 시점을 표시한다.

패칭 기법 하에서 클라이언트는 동시에 2개의 다운로드 채널만을 사용할 수 있으므로 구간 z_p에 있는 클라이언트는 패칭 유니캐스트 및 멀티캐스트 m_t를 사용하기 때문에 서버로부터 생성된 멀티캐스트 m_d를 수신할 수 없게 된다. 따라서 생성된 멀티캐스트 m_d로의 가입은 패칭이 완료된 후 가능하기 때문에 멀티캐스트 m_d의 시작 시간 t_{bd} 으로부터 패칭의 완료 시간 t_{fp}의 시간 차이 $\delta = t_{bd} - t_{fp}$ 분량의 데이터를 필요로 하게 된다. 이를 위하여 서버는 확장 멀티캐스트 m_e 라는 채널을 사용하여 $\delta < I_m$ 시간의 데이터를 전송하며 이 구간을 확장 구간 z_e라고 한다. 시간 t4 이후 서버는 I_m 시간이 아니라 $\lfloor 3I_m \rfloor$ 마다 멀티캐스트 채널을 사용하여 비디오 데이터를 전송하기 때문에 기존의 배칭 멀티캐스트 기법보다 3배 이상의 네트워크 사용 효율의 증대를 기대할 수 있게 된다.

본 논문에서 사용하고 있는 캐싱 기법은 단말노드에 저장된 비디오 데이터를 다음의 4가지 경우로 나누어 처리하며, 각 경우는 (C_{first}, C_{last})로 표시되는 플래그 값에 의하여 정의된다.

-. 비디오의 첫블록 $(C_{first}, C_{last}) = (1, 0)$
-. 비디오의 마지막 블록 $(C_{first}, C_{last}) = (0, 1)$
-. 비디오의 처음이자 마지막 블록 $(C_{first}, C_{last}) = (1, 1)$
-. 나머지 블록 $(C_{first}, C_{last}) = (0, 0)$

수신되는 비디오 블록을 저장할 공간이 없다면 단말노드에 저장된 비디오 블록들은 제한된 저장 용량 때문에 삭제되어야 한다. 멀티캐스트 m_d의 생성 간격을 최대로 유지하기 위하여 HEN은 $C_{last} = 1$인 비디오의 마지막 블록들을 우선적으로 삭제한다. 클라이언트는 시청 여부를 결정하기 위하여 일반적으로 비디오의 시작 부분은 시청을 하기 때문에 서비스 지연을 줄이기 위하여 비디오의 시작 블록 $C_{first} = 1$은 마지막에 삭제한다[15, 16]. 인기도가 높은 비디오는 다른 비디오와 비교하여 상대적으로 높은 요청 빈도수를 갖기 때문에 인기도의 반영이 가능한 LFU를 캐쉬 제거 알고리듬으로 사용한다. 그러나 cut-off Zipf's 분포[17]에 따르면 인기도가 낮은 비디오들은 요청 빈도수는 매우 낮으며 그들의 값이 거의 비슷하기 때문에 LFU를 사용하는 것은 효율적인 방법이 아니다. 따라서 낮은 요청 빈도수를 갖는 비디오들에 대해서는 LRU 캐쉬 제거 정책을 사용한다. 또한 낮은 요청 빈도수를 갖는 비디오들은 단말노드의 캐쉬에 저장될 확률이 낮으며, 저장되어 있는 양 또한 적다. LRU가 적용되는 비디오 블록은 $(C_{first}, C_{last}) = (1, 1)$ 의 값을 갖는다. 이 경우는단말노드들 내부에 저장된 블록이 하나이기 때문에 시작블록이면서 동시에 마지막블록으로 표시되는 것이다.

4. 성능 분석

3장에서 설명한 것처럼 서버는 1) 최초 요청으로 인한 해당 비디오 전송, 2) 패칭 멀티캐스트로 인한 확장 구간 z_e내에서의 전송, 3) 동적 구간 z_d 에 해당하는 비디오의 나머지 부분을 전송하는 3가지 경우에 새로운 채널을 생성하여 비디오 데이터를 전송한다. 따라서 하나의 비디오를 재생하기 위하여 전송되는 데이터의 총량은 D로 계산될 수 있다. D는 초기요청에 따른 전송량 D_i, 구간 z_e내의 데이터를 멀티캐스트 m_e에 의하여 전송되는 양 D_e, 구간 z_d 구간을 멀티캐스트 m_d를 통하여 전송하는 양 D_d의 합으로 표시된다.

초기 요청에 따라 서버로부터 전송되는 양은 HEN들에게 저장되는 양 $n \cdot I_m$과 동일하며, 구간 z_p 및 구간 z_t의 합과 동일하다. 만약 다른 HEN들로부터 새로운 요청이 발생하지 않는다면 HEN들에게 저장된 $n \cdot I_m$ 분량의 비디오는 서버로부터 전송될 필요가 없다. 구간 z_e에 해당하는 비디오 데이터는 멀티캐스트 m_e를 사용하여 전송되기 때문에 전송되는 데이터의 총량은 구간 z_p 내에 있는 클라이언트의 수와는 상관없이 I_m 이 된다. 또한 생성된 멀티캐스트는 해당 비디오를 저장하고 있는 HEN의 수 n의 값이 2이상인 경우 $\lfloor n \cdot I_m \rfloor$ 시간마다 생성될 수 있다. 멀티캐스트 그룹 간격이 I_m 시간동안 패칭이 발생할 확률을 $P(p, I_m)$이라고 하면, I_m 시간동안 멀티캐스트 m_e를 통하여 전송되는 데이터양 D_e는 식 (1)과 같다.

$$D_e = \frac{I_m}{n} \cdot P(p, I_m)$$
$$P(p, I_m) = \begin{cases} 1, & \lambda \geq 2 \\ \lambda - 1, & 1 < \lambda < 2 \\ 0, & \lambda \leq 1 \end{cases} \quad (1)$$

λ는 서비스 요청률이며 분당 요청수 (requests/minute)로 표시된다. 구간 z_d 내에서 멀티캐스트 m_d에 의하여 전송되는 데이터량은 요청률에 따라 식 (2)와 같다.

$$D_d = \begin{cases} \sum_{i=1}^{\left\lceil \frac{|v| - nI_m}{nI_m} \right\rceil} i \cdot n \cdot I_m, & \lambda \geq \frac{1}{n} \\ \sum_{i=1}^{\left\lceil \frac{|v| - nI_m}{\tau} \right\rceil} i \cdot n \cdot I_m, & \lambda < \frac{1}{n} \end{cases} \quad (2)$$

연속된 멀티캐스트 m_d의 평균 간격은 $\tau = 1/\lambda$라 하면 $n \cdot I_m + \tau$가 되기 때문에 N개의 비디오를 전송하기 위하여 서버에서 필요로 하는 네트워크 대역폭 B_{server} 는 식 (3)과 같다.

$$B_{server} = \sum_{v=1}^{N} \frac{D_{e,v} + D_{d,v}}{|v_v| - n_v I_m} \cdot r_v \quad (3)$$

식 (3)에서 r_v는 비디오 v의 재생률이며 $D_{e,v}$ 및 $D_{d,v}$는 멀티캐스트 m_e 및 멀티캐스트 m_d를 통하여 전송되는 비디오 v의 데이터양이다. HEN은 서버로부터 수신되는 비디오 데이터를 저장하며 클라이언트에게 패칭 및 멀티캐스트 m_t를 통하여 전송하는 역할을 수행한다. 생성되는 멀티캐스트 m_t는 평균 간격은 $I_m + \tau$이기 때문에 구간 z_p 및 구간 z_t에서 전송되는 멀티캐스트 m_t의 수는 $\lceil (n \cdot I_m)/(I_m + \tau) \rceil$이다. 따라서 멀티캐스트 m_t에 의하여 전송되는 데이터의 양 D_t은 식 (4)와 같이 된다.

$$D_t = \sum_{i=1}^{\left\lceil \frac{nI_m}{I_m + \tau} \right\rceil} i(I_m + \tau) \quad (4)$$

멀티캐스트 m_t는 I_m 시간동안만 지속되기 때문에 각 HEN에서 멀티캐스트 m_t를 전송하기위하여 필요로 하는 대역폭 B_t는 식 (5)와 같다.

$$B_t = \begin{cases} r, & \lambda \geq 1 \\ \lambda \cdot r, & \lambda < 1 \end{cases} \quad (5)$$

패칭 채널은 요청률 $\lambda \geq 2$인 경우 구간 z_p에서 생성되며 총 $\lambda - 1$개가 존재한다. 따라서 I_m 시간동안 패칭 채널을 통해 전송되는 데이터의 양은 식 (6)의 값을 갖는다.

$$B_p = \sum_{k=1}^{\lambda - 1} k \cdot \frac{I_m}{\lambda} \cdot r \quad (6)$$

그러므로 각 HEN에서 N개의 비디오를 전송하기 위하여 사용되는 네트워크 대역폭은 식 (7)로 계산될 수 있다.

$$B_{HEN} = \sum_{v=1}^{N} (f_v \cdot B_{p,v} + c_v \cdot B_{t,v}) \quad (7)$$

$B_{p,v}$ 및 $B_{t,v}$는 비디오 v를 전송하기 위하여 패칭 채널 및 멀티캐스트 채널을 통하여 전송되는 데이터양이다. c_v는 HEN에 비디오 v의 저장 유무를 나타내며, 저장된 경우 1 저장되지 않은 경우 0의 값을 갖는다. f_v는 HEN에 저장된 블록이 비디오의 첫 I_m 블록인지를 표시하며 $f_v = C_{first}$이다.

5. 시뮬레이션 및 결과분석

각 HEN에 하나의 비디오를 저장하기 위하여 할당된 공간은 멀티캐스트 그룹 간격 I_m이며 1분으로 설정하였다. 비디오 서버는 최대 1,000개의 비디오를 제공할 수 있으며 각 비디오의 재생 시간은 100분 ($100I_m$)의 값을 갖는다. 서버는 최대 10,000개의 전송 채널을 사용할 수 있으며, 사용자들의 요청은 100분 동안 최대 10,000개가 발생하도록 제한하였다. 비디오 서버가 제공하는 비디오의 수를 N이라 하면, 비디오 v를 요청하는 조건부 확률은 $P_N(v)$로 표시할 수 있다. 각 비디오는 요청수에 따라 인기도가 설정되며 비디오 v는 v번째 인기가 높은 비디오를 나타내도록 하였다. v = 1, 2, 3, ..., N 인 경우 비디오 v를 요청할 조건부 확률 $P_N(v)$는 cut-off zipf 분포를 갖는다고 가정하였으며 식 (8)과 같다.

$$P_N(v) = \frac{\Omega}{v^s}, \quad where\ \Omega = \left(\sum_{v=1}^{N} \frac{1}{v^s} \right)^{-1} \quad (8)$$

식 (8)에서 사용된 skew factor s 는 큰 값을 갖을수록 사용자들의 요청이 더 적은 수의 비디오에 집중되는 현상을 갖게 되며, 시뮬레이션에 사용된 값은 0.7 이다[13, 17, 18, 19]. Zipf 분포에 따르면 모든 클라이언트들의 서비스 요청률은 λ 로 표시되며, N개의 비디오 중 비디오 v에 대한 서비스 요청률은 $\lambda v = \lambda P_N(v)$로 구할 수 있다. 인기도에 기초한 요청률은 각 비디오들을 전송하는데 필요로 하는 대역폭을 계산하는데 사용된다. 인기도가 높은 비디오는 인기도가 낮은 비디오와 비교하여 상대적으로 높은 서비스 요청률을 갖기 때문에 가장 높은 인기도를 갖는 비디오는 (v=1) 다른 비디오들과 비교하여 높은 가중치를 가져야 한다. 본 시뮬레이션에서는 비디오를 선택하기 위한 가중치 파라미터로서 $P_N(v)$를 사용하였다. 따라서 인기도가 높은 비디오는 $P_N(v)$의 값이 크기 때문에 높은 서비스 요청률을 갖게 된다[20]. 표1은 시뮬레이션에서 사용된 변수들을 나타내고 있다.

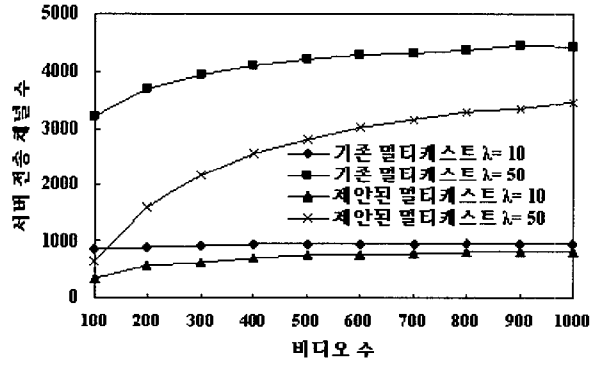
(그림 3)은 VOD 서버에서 제공하는 비디오의 수가 100편에서 1000편까지 100편씩 증가하는 경우 서버에서 사용된 평균 전송 채널의 수를 보여주고 있다. 이 경우 단말노드는 10개 이며 서버는 최대 10000 채널을 사용할 수 있으며 각 단말노드는 100개의 채널을 사용할 수 있다. 시뮬레이션은 클라이언트로부터 발생되는 요청률 λ 를 분당 10과 50으로 설정하여 수행하였다. 각 단말노드는 100분($100I_m$) 크기의 캐쉬를 갖고 있기 때문에 단말노드에서 저장할 수 있는 비

〈표 1〉 시뮬레이션 환경

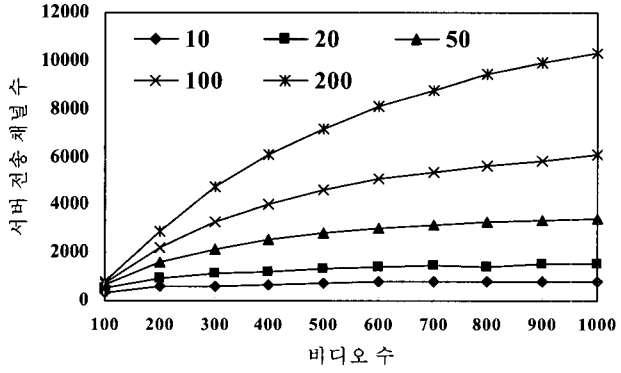
변수	기본값	범위
비디오 수	1000	100~1000
비디오 재생시간(분)	100	N/A
서버 대역폭(채널 수)	10000	N/A
HEN 대역폭(채널 수)	100	N/A
요청률 (요청수/분)	50	10~200
HEN의 캐쉬 크기(분)	100	N/A

디오 데이터의 최대값은 1000분(1000I_m)이 된다. VOD 서버로부터 생성되는 전송채널은 식 (3)처럼 멀티캐스트 m_e를 통하여 $n \cdot I_m$마다 $\lceil I_m \rceil$ 크기의 비디오 데이터를 전송하거나, 멀티캐스트 m_d를 통하여 $|v| - n \cdot I_m$ 크기의 데이터를 전송하기 위하여 사용된다. 이러한 전송 채널들은 HEN들에 저장된 비디오 데이터 전부를 재생한 후 생성된다. 따라서 n_v는 비디오 v에 대한 블록들을 저장하고 있는 단말노드의 수라 하면 $n_v \geq 2$ 인 경우 각 비디오 v에 대하여 전송채널의 수를 $1/n_v$로 줄일 수 있게 된다. 따라서 제안된 멀티캐스트 기법 하에서 멀티캐스트 그룹 간격 I_m은 단말노드에 저장된 블록 수 n_v에 따라 I_m부터 $n_v \cdot I_m$까지 다양한 값을 갖는다.

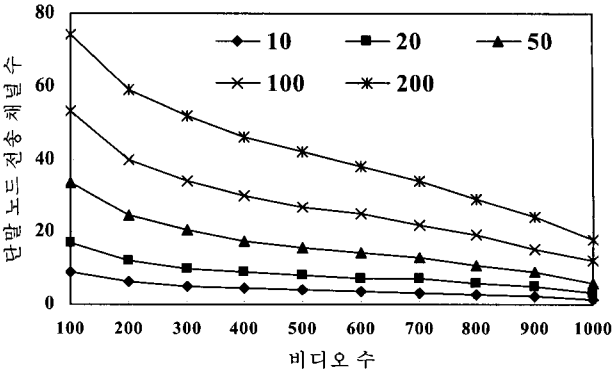
(그림 3)의 결과로부터 멀티캐스트 기법만을 사용하는 경우와 비교하여 제안된 기법은 요청률 $\lambda = 10$ 인 경우 100편의 비디오에 대하여 약 59%, 1000편의 비디오에 대하여 약 15% 정도 전송 채널의 수를 감소시킬 수 있다. 요청률 $\lambda = 50$ 인 경우는 100편의 비디오에 대하여 약 80%, 1000편의 비디오에 대하여 약 22% 정도 전송 채널의 수가 감소되었다. 서버가 제공하는 비디오의 수가 100편 또는 1000편인 경우 단말노드들은 전체 비디오의 1%와 0.1% 정도의 데이터만을 저장할 수 있다. 따라서 1000편의 경우 100편과 비교하여 상대적으로 낮은 캐쉬-히트 율을 갖기 때문에 서버는 캐쉬-미스된 데이터들을 전송하기 위하여 더 많은 전송 채널을 사용하게 된다.



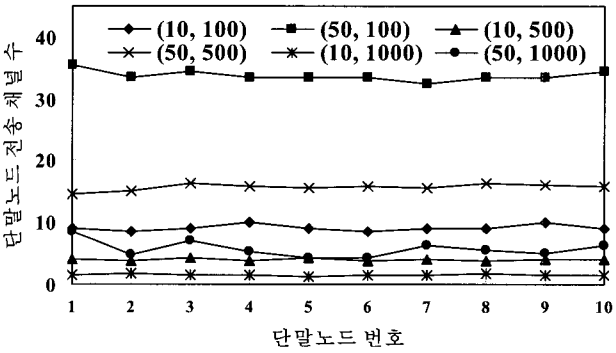
(그림 3) 서비스 요청률 λ=10과 50에 대한 평균 서버 전송 채널 수



(그림 4) 요청률 λ=10, 20, 50, 100, 200 및 비디오 수 N에 따른 평균 서버 전송 채널 수



(그림 5) 요청률 λ=10, 20, 50, 100, 200 및 비디오 수 N에 따른 단말노드의 평균 전송 채널 수



(그림 6) (요청률 λ, 비디오 수 N)에 따른 각 단말노드에서의 평균 전송 채널 수

(그림 4)는 요청률 $\lambda = 10, 20, 50, 100, 200$ 인 경우에 대하여 제공하는 비디오 수가 100편에서 1000편으로 100편씩 증가하는 경우 서버에서 생성되는 평균 전송채널의 수를 보여주고 있다. 사용된 단말노드는 10개 이며 서버는 최대 10000 채널을 사용할 수 있으며 각 단말노드는 100개의 채널을 사용할 수 있다. 시뮬레이션 결과는 서비스 요청률의 증가량에 따라 서버로부터의 평균 전송 채널수는 선형적으로 증가하고 있음을 보여주고 있다. 동일한 요청률에서 비디오 수 N이 증가하는 경우 클라이언트들의 요청들은 많은 비디오에 분산되기 때문에 전송 채널의 수는 선형적으로 증가하게 된다. 그러나 비디오 수의 증가는 비인기 비디오의 수를 증가시키게 되며, 이러한 비디오들은 매우 낮은 요청률들을 갖으며 그 값이 거의 비슷하기 때문에 평균 전송 채널수가 지속적으로 증가할 수 없게 되기 때문에 일정이상에서 수렴하는 결과를 나타내고 있다.

(그림 5)는 요청률 $\lambda = 10, 20, 50, 100, 200$ 인 경우에 대하여 제공하는 비디오 수가 100편에서 1000편으로 100편씩 증가하는 경우 각 단말노드들에서의 평균 전송 채널수를 보여주고 있다. 사용된 단말노드는 10개 이며 서버는 최대 10000 채널을 사용할 수 있으며 각 단말노드는 100개의 채널을 사용할 수 있다. Zipf 분포로부터 서버가 적은 수의 비디오를 제공할수록 클라이언트들의 서비스 요청은 소수의 비디오들에게 집중되며, 제공되는 비디오 수가 커질수록 요청들은 넓게 분산 분포하는 특성을 갖고 있다. 서비스 요청

이 집중되면 단말노드는 지연 없는 서비스를 제공하기 위하여 다수의 패칭 채널들을 생성하기 때문에, 서버가 제공하는 비디오의 수가 적을수록 단말노드에서 생성되는 전송채널의 수는 증가하게 된다. 또한 비디오 편수가 증가할수록 멀티캐스트들의 병합이 적게 발생되기 때문에 평균전송 채널의 수는 선형적으로 증가함을 보여주고 있다.

(그림 6)은 (요청률 λ, 비디오 수 N) = (10, 100), (10, 500), (10, 1000), (50, 100), (50, 500), (50, 1000) 인 경우, 각 단말노드에서의 평균 전송 채널수를 보여주고 있다. 사용된 단말노드는 10개 이며 서버는 최대 10000 채널을 사용할 수 있으며 각 단말노드는 100개의 채널을 사용할 수 있다. 시뮬레이션 결과로부터 각 단말노드에서 생성되는 전송 채널의 수가 거의 비슷하게 생성되고 있음을 알 수 있다. 요청률 λ가 증가하거나 서버에서 제공되는 비디오 수 N이 커질수록 각 단말노드에서의 평균 전송 채널수가 큰 편차를 갖는다. 이것은 멀티캐스트들의 낮은 병합으로 인하여 패칭 채널을 생성하여야 하는 비디오의 첫 블록 (C_{first} = 1)을 갖고 있는 단말노드는 첫 블록을 갖고 있지 않은 단말노드와 비교하여 상대적으로 많은 전송 채널을 필요로 하기 때문이다. 만약 단말노드들이 비디오의 첫 블록들을 인기도에 따라 균등하게 저장하게 되면 (그림 6)에서 보이는 편차는 거의 발생하지 않게 될 것이다.

6. 결 론

프락시를 사용하는 웹-캐싱 기법은 낮은 서버 대역폭을 필요로 하지만 인기 있는 비디오를 저장하고 있는 프락시는 다른 프락시와 비교하여 상대적으로 큰 부하를 갖게 되며 동일한 비디오 컨텐츠가 중복적으로 저장될 수 있는 문제점들을 갖고 있다. 본 논문에서는 서버로부터 전송되는 비디오 데이터를 인기도에 따라 분산 저장하는 단말노드를 사용한 멀티캐스트 전송 기법을 제안하였다. 서버는 비디오를 멀티캐스트 그룹 간격 I_m 단위로 블록화하여 요청한 단말노드에게 전송한다. 단말노드들은 각 비디오를 위하여 멀티캐스트 그룹 간격 I_m과 동일한 크기의 캐쉬를 할당하여 단지 하나의 비디오 블록 I_m 만을 저장한다. 클라이언트들의 요청은 단말노드에 해당 데이터가 저장되어 있는 경우 서버와는 무관하게 전송되며, 저장된 데이터를 초과하여 지속되는 경우 서버는 새로운 멀티캐스트 전송 채널을 사용하여 제공한다. 이때 생성되는 멀티캐스트 채널은 그룹 간격은 동일 비디오를 저장하고 있는 단말노드의 수 n에 따라 I_m으로부터 $n \cdot I_m$ 까지 다양하게 설정되기 때문에 각 비디오를 위한 전송 채널의 수를 1/n로 줄일 수 있다. 비디오 블록은 요청순서에 따라 단말노드에 분산 저장되기 때문에 각 단말노드의 부하가 균등하게 발생된다. 시뮬레이션 결과로부터 서버의 전송 채널수를 요청률 λ = 10 인 경우 100 비디오에 대하여 약 59%, 1000 비디오에 대하여 약 15% 정도 감소시킬 수 있다. 또한 요청률 λ = 50 인 경우는 100비디오에 대하여 약 80%, 1000 비디오에 대하여 약 22% 정도의 전송 채널의 수를 감소시킬 수 있다. 따라서 제안된 기법은 서버의 부하 및 네트워크상에 전송되는 채널수를 크게 감소시킬 수 있으며 구성된 각 단말노드사이에 데이터가 중복 저장되는 것을 방지하며 균등 부하를 발생시키기 때문에 병목현상과 같은 트래픽 집중화 현상을 개선할 수 있다.

참 고 문 헌

[1] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A Multicast Technique for True Video-on-Demand Services," ACM Mutimedia'98, pp.191-200, Bristol, UK, 1998.

[2] J. Pasquale, G. Polyzos, and G. Xylomenos, "The Multimedia Multicasting Problem," ACM Multimedia Systems, Vol.6, No.1, 1998.

[3] K. Almeroth and M. Ammar, "Providing a scalable, interactive Video-on-Demand Service using multicast communication," In ICCCN'94, San Francisco, CA, Sept., 1994.

[4] W. Liao and V.O.K. Li, "The Split and Merge Protocol for Interactive Video-on-Demand," IEEE Multimedia, pp.51-62, 1997.

[5] C. Griwodz, M. Zink, M. Lieport, G. On, and R. Steinmetz, "Multicast for Savings in Cache-Based Video Distribution," Multimedia Computing and Networking, San Jose, CA, Jan., 2000.

[6] K. A. Hua, D. A. Tran, and R. Villafane, "Caching Multicast Protocol for On-Demand Video Delivery," Proc. of SPIE Multimedia Computing and Networking 2000, Vol.3969, pp.2-13, Dec., 1999.

[7] P. Basu, A. Narayanan, R. Krishnan, and T.D.C. Little, "An Implementation of Dynamic Service Aggregation for Interactive Video Delivery," Proc. of SPIE Multimedia Computing and Networking, Vol.3310, pp.110-112, San Jose, CA, Jan., 1998.

[8] D.A. Tran, K.A. Hua, and T.T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming," Proc. Of IEEE INFOCOM 2003, San Francisco, CA, Mar., 2003.

[9] A. Mahanti and C. Williamson, "Web Proxy Workload Characterization," Technical Report Univ. of Saskatchewan, Feb., 1999.

[10] R. Rajaie, M. Handley, H. Yu, and D. Estrin, "Proxy caching mechanism for multimedia playback streams in Internet," 4th Int'l WWW Caching Workshop, Mar., 1999.

[11] R. Rajaie, H. Yu, M. Handley, D. Estrin, "Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet," Proc. of IEEE INFOCOM, Tel-Aviv, Israel, 2000.

[12] M. Sasabe, Y. Taniguchi, N. Wakamiya, M. Murata, and H. Miyahara, "Proxy Caching Mechanisms with Quality Adjustment for Video Streaming Services," Proc. Of SPIE Vol.4519, pp.276-284, July, 2001.

[13] P. Cao and S. Irani, "Cost-aware WWW Proxy Caching Algorithms," Proc. Of the 1997 USENIX Symposium on Internet Technology and Systems, pp.193-206, Dec., 1997.

[14] B. Kim and I. Kim, "Web Proxy Caching Mechanism to evenly Distribute Transmission Channel in VOD System," 2nd Int'l Workshop on Grid and Cooperative Computing, LNCS 3032/2004, pp.1099-1102, Dec., 2003.

[15] S. Pakinikar, M.Kankanhali and K.R. Ramakrishnan, "A Caching and Streaming Framework for Multimedia," Proc. of the 8th ACM Int'l Conf. on Multimedia, pp.13-20, 2000.

[16] S. Acharrya and B.Smith, "Middleman: A Video Caching Proxy Server," Tech. Report Cornell, 1999.

[17] L. Breslau, P.Cao, L, Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," Proc. of the Conf. on Computer Commun (IEEE Infocom), New York, Mar., 1999.

[18] C. Williamson, "On Filter Effects in Web Caching Hierarchies," ACM Trans. on Internet Technology, Vol.2, No.1, pp.47-77, Feb., 2002.

[19] M. Arlitt and C. Williamson, "Trace-driven simulation of document caching strategies for Internet Web Server," Simulation Journal 68, pp.23-33, Jan., 1997.

[20] S.Gribble and E. Brewer, "System Design Issues for Internet Middleware Services: Deductions form a Large Client Trace," Proc. of the 1997 USENIX Symposium on Internet Technology and Systems, December, 1997.

김 백 현

e-mail : hidesky24@incheon.ac.kr

1993년 2월 인천대학교 정보통신공학과 (공학사)

1993년 8월~1997년 12월 삼성전자 전임연구원

2001년 2월 인천대학교 정보통신공학과 (공학석사)

2000년 12월~2002년 12월 PNP네트워크 선임연구원

2003년 3월~현재 인천대학교 정보통신공학과 대학원 박사과정

관심분야 : QoS, MANET, 분산처리, 멀티미디어

황 태 준

e-mail : tjhwang@incheon.ac.kr

1997년 2월 인천대학교 전자계산학과 (공학사)

1999년 2월 인천대학교 전자계산학과 (공학석사)

1999년 3월~현재 인천대학교 정보통신 공학과 박사과정

관심분야 : Multicast, VOD, Webcaching, 데이터베이스, 멀티미디어

김 익 수

e-mail : iskim@incheon.ac.kr

1977년 동국대학교 전자공학과(공학사)

1981년 동국대학교 전자공학과(공학석사)

1985년 동국대학교 전자공학과(공학박사)

1988년 3월~현재 인천대학교 정보통신 공학과 교수

1993년 3월~1994년 2월 North Carolina State Univ. 객원교수

2004년 3월~2005년 2월 California State University Sacramento 객원교수

관심분야 : Multicast, Network, VOD, Webcaching