論文
4-12-21

인터넷 상에서 비디오 스트림을 위한 효과적인 캐싱기법

A efficient caching scheme of Continuous stream service on the internet

김종경(Jong-gyung Kim)[1] 남현우(Hyun-woo Nam)[2]

요 약

본 논문에서는 인터넷 상에서 비디오 스트림을 서비스할 때 인기도에 기반으로하는 서비스 지연시간을 최소화하는 방법을 제안하였다. 접근 빈도와 각 서버들의 총 캐싱 공간의 상관관계를 고려하여 제안한 알고리즘의 성능을 측정한 결과는 에이전트 캐싱 방법보다는 서비스 지연에 대한 캐쉬 활용율이 15%의 향상 그리고 인터벌 캐싱보다는 3배의 정도의 성능 향상을 보여주었다.

Abstract

This paper propose method to minimize service delay time based in popularity when service video stream in internet Result that measure performance of algorithm that propose considering correlation of cache space all of access frequency and each servers showed performance elevation of degree of triple than elevation of cache practical use rate 15% and interval caching about service laytency than agent caching scheme.

1) 정회원 : 인천전문대학 컴퓨터정보과 겸임교수

2) 정회원 : 인천전문대학 강사

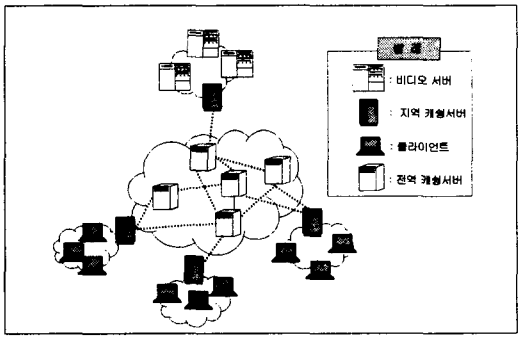
논문접수 : 2003. 12. 11.
심사완료 : 2003. 12. 19.

1. 서론

최근 초고속 인터넷의 계속적인 폭발적인 성장 때문에 더욱더 "World Wide Wait" 문제로 알려진 개체들의 검색 지연현상과 망 대역폭의 증가 그리고 "Hot spots"와 같은 경험적인 몇 가지의 문제를 지니고 있다. 이러한 문제를 해결하기 위한 방법중의 하나는 캐싱 프락시를 사용하는 것이다[1,2,3]. 지난 몇 년에 캐싱은 이러한 문제를 해결하기 위한 기법으로서 적절히 활용하게 되었다.

많은 논문에서 인터넷에서 혼잡과 지연 그리고 대역폭의 감소를 위하여 프락시 캐쉬를 기반으로 하는 기법들을 제안하였다[13,14,15,16,7,8]. 대부분의 이러한 기술은하나 이상의 프락시 서버를 사용하여 주로 지역 네트워크에서 비디오를 캐싱하는 역할을 한다. 최근의 연구에서는 같은 지역 네트워크에서 다수의 비디오 프락시들을 협력하는 기법을 소개하였다[1,6]. [1]에서는 여러 프락시들 중에서 요청된 비디오 스트림을 서비스를 하기 위하여 프락시를 선정하는 중앙 조정자(centralized coordinator)라는 것를 사용하는데, 이는 캐쉬 교체정책을 수행하는 캐싱된 정보를 관리하는 역할을 한다. 반면에 [9]에서는 두 레벨의 계층 캐쉬를 가진 프락시의 집단을 구성하여 협력관계를 수행한다. 중앙 협력 비디오 캐싱 기법은 전체 네트워크에 대하여 중앙 조정자가 제어함으로써 다음과 같은 문제점이 발생하여 확장성과 신뢰성이 떨어진다. 첫째, 중앙 조정자가 캐싱 프락시와 클라이언트를 연결하는 역할을 하여 높은 지연(Latency)이 발생하는 것이고 둘째, 중앙 조정자의 역할로 인하여 병목 현상이 발생하거나 이용할 수 없는 단일장애점(single point of failure)이 발생할 수 있다. 특히 클라이언트들이 증가하면 네트워크의 부하나 지연은 더욱더 증가할 것이다. 계층형 협력캐싱은 각 협력 프락시들이 할 수 있는 역할 즉, 요청 비디오에 대하여 저장할 수 있는 용량 그리고 지역 네트워크의 클라이언트들과 협력관계를 확대해야 한다.

많은 용량의 비디오를 지원하기 위하여 협력 연속 미디오 캐쉬 프락시들은 원본 비디오가 저장되어있는 서버로 비디오 스트림을 가져오는 것보다 가까운 프락시 서버에서 비디오 스트림을 가져오는 것이 효과적이나 유연성있는 협력 캐싱 프로토콜의 지원이 필요하다. 이와 같이 네트워크의 대역폭과 서버들의 자원들을 고려한 여러 클라이언트에게 비디오를 전송하기 위한 Native-IP 기반의 멀티캐스트 연구가 활발하게 진행되고 있으나 기술상의 문제로 인터넷상에서 활용하는데 많은 제약이 따른다[17]. 최근에 응용-레벨의 멀티케스트를 Native-IP 기반의 멀티캐스트를 오버레이(overlay) 네트워크에 대안으로 연구되고 있다. 오버레이의 초기 연구는 Overcast[18], Scattercast[19] 등이 있으며 이러한 기법들은 멀티케스트 기능을 할 수 있는 라우터 없이 멀티케스트를 구현하였다. 그 이후에 많은 연구들이 활발히 진행되고 있는 실정이다.



[그림.1] 협력 캐싱 구조

본 논문은 광역 통신망에서 동적이고 효과적인 협력 캐쉬를 실현할 수 있는 오버레이 구조 위에서 접근빈도를 고려하고 협력 서버들 사이에서 메모리의 효율적인 캐싱기법을 제안하여 서비스 지연을 감소시키는 유니캐스트 기반의 멀티캐스트 기법을 제안한다.

본 논문의 구성은 2장에서 웹 캐싱, 연속 미디어의 캐싱 기법, 오버레이, 오버레이 캐싱 기법에 대하여 언급하고 3장에서는 본 논문에서 제안한 기법을 설명하고며, 4장에서는 실험결과를 보여주고 5장에서는 결론을 기술한다.

2. 관련 연구

2.1 웹 캐싱

전통적으로 프로그램의 수행 능력 향상을 위해서는 디스크의 엑세스를 줄이고 가장 많이 사용되어질 페이지의 집합을 캐쉬에 저장하여 사용한다[4]. 캐쉬를 참조를 극대화하기 위해서 캐쉬의 용량을 초과하게 되는데, 이러한 것을 해결하기 위하여 접근의 빈도성와 최근성을 고려하여 앞으로 사용되어질 객체들을 교체 알고리즘의 연구들이 활발이 진행되었다. 최근성(recency) 기반의 알고리즘[14]은 프로그램의 특성인 참조 지역성(Locality)을 활용한 방식인 반면에 접근의 빈도(frequency)기반의 알고리즘[15]은 skewed access pattern에 적합한 방법으로 알려졌고, 그 밖에 LRU-k[13], LRU-MIN[16] 등이 있다.

이러한 캐싱 효과를 증진하기 위한 강력한 기법은 캐쉬의 협력이고 그 이유는 다음과 같다. 첫 번째, 바쁜 캐쉬가 있으면 이웃하는 한가한 캐쉬를 이용할 수 있고, 또 하나의 이유는 하나만 존재하는 오브젝트들을 적절한 복제를 통하여 히트율과 히트 시간을 향상시킬 수 있다[11].

그러나 이와 같이 전통적인 캐싱 기법에서는 연속 미디어에 대한 캐싱이 적용하기에 미흡하다.

2.2. 연속 미디어 데이터의 캐싱

최근 들어 연속 미디어 데이터에 대한 캐싱에 대해 연구가 활발히 진행되고 있다[5]. 이 연구들에서 나타난 연속미디어 스트림의 특징을 보면 주기적과 순차적인 성질이 있으며, 대용량, 고대역폭을 요구하고 있다. 이러한 것을 해결하기 위한 연속 미디어 스트림을 효과적으로 서비스를 하기 위한 기술들을 보면 다음과 같다.

논문 [7]에서는 인기있는 비디오 스트림들의 첫부분을 프락시에 저장하고 클라이언트에서 특정 비디오 스트림을 요청할 때 프락시 캐쉬에서 해당 비디오 첫 부분을 재생하는 동안에 서버에 남아있는 프레임들을 동시에 전송하는 방식으로서 Workahead Smoothing을 수행한다.

Workahead Smoothing은 클라이언트 재생 지연없이 실제적으로 프락시와 클라이언트 사이에 발생하는 네트워크 자원 요구량에 적절히 대응하는 기법으로서 프리픽스(prefix) 크기, 유연한 전송 스케쥴, 재생 버퍼 등을 다룬다.

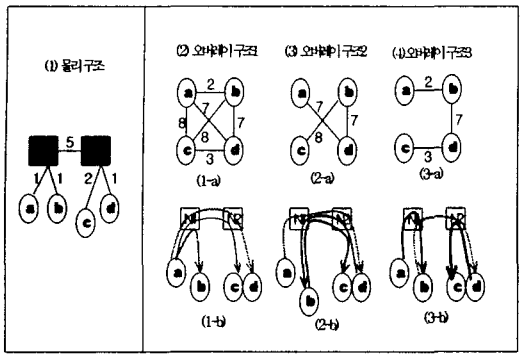
논문 [8]은 동일한 연속 미디어형 오브젝트를 선행 스트림과 후행 스트림과 사이에 consecutive pair 개념으로 연구된 새로운 버퍼링 기법이다. 버퍼의 요구량은 두 스트림 간의 시간간격(time interval)에 의한 함수로 정의한다. 프락시 캐쉬 공간과 대역폭을 프락시 자원으로 으로서 각 요청들이 이를 요구하는 기법이다.

논문 [14]는 클라이언트들에게 인기있는 비디오 스트림을 최대로 전송하기 위한 레이어(layer) 엔코딩 멀티미디어 스트림 기법을 제안하였는데 현재의 대역폭에 대한 변화를 고려하여 효과적으로 대응하는 기법이다.

이러한 기법들은 많은 네트워크의 대역폭과 서버들의 자원들을 요구하고 있기 때문에 비디오 스트림을 제공할 수 있는 기반 즉, 네트워크의 환경을 고려하지 않을 수 없다. 따라서 이러한 요구를 전송하기 위한 Native-IP 기반의 멀티캐스트 연구가 활발하게 진행되고 있지만, 많은 기술상의 문제로 인터넷상에서 활용하는데 많은 제약이 따른다[17].

2.3 오버레이

인터넷상에서 네트워크의 대역폭 증가, 다량의 서버들의 자원 요구들의 문제점을 개선하기 위한 방법중에 하나가 오버레이를 이용한 연속 미디어 캐싱 방법이다. IP 인프라구조보다 오버레이 노드 사이에는 대역폭의 비용에서 보면 최적의 지연을 보여준다[23]. 따라서 노드간의 협력 캐싱은 가능한 대역폭을 가지고 짧은 경로를 선택함으로서 대역폭의 절약과 서비스의 지연 등을 줄일 수 있다. [그림.2]에서 보면 물리적 네트워크의 구조 (1)는 노드 "a"가 노드"b"와 노드 "c" 그리고 노드 "d"에게 메시지를 보내기 위한 방법 3가지의 가능한 오버레이를 보여주고 있다.



[그림.2] 오버레이 구조의 경로

(1-a)는 완전 연결된 오버레이 구조를 보여주고 있으며 이것은 중복 메시지를 보내는 멀티캐스트 방식으로 대역폭 소비를 유발시킨다. (2-a)는 물리적 연결 경로는 하나인데 반면에 N1과 N2 사이를 연결하는 물리적 경로에 대한 지연시간인 22는 최악을 보여주고 있다. (3-a)는 이상적인 오베레이 구조로서 물리적 연결 경로는 2이고 연결지연시간 12를 보여주고 있다.

2.4 오버레이 캐싱 기법

오버레이 캐싱 기법은 클라이언트가 요청한 비디오를 오버레이 노드들에게 복제하기 위한 기법이다. 그러나 이러한 오버레이 캐싱 기법은 오버레이 노드들 사이에 지연(laytency)과 같은 네트워크 속성들을 고려한 구축이 바람직하다. 다양한 형태의 오버레이 구조상에서 멀티캐스트 서비스 알고리즘의 연구는 [20, 9, 21, 22] 등이 있다.

[9]는 확장성이 유연한 오버레이 구조 위에서 클라이언트들의 모든 서비스를 지원하는 에이전트들의 캐싱 공간을 고정크기의 청크로 나누고 모든 에이전트들에 대한 복사본을 가지게 함으로서 서비스 지연을 줄이며 RAM의 FIFO 버퍼를 사용하여 빠른 수행시간을 보이고 있다.

[20]은 노드 당 작은 수의 터널을 유지하면서 노드 사이에 지연시간을 최소화하는 오버레이 위상을 제안하였다. 터널은 초기 세트에 의해 구성되고 주기적으로 터널들을 추가시키고 삭제시키면서 하나의 터널을 관리하기 위해 프로세스를 추가와 삭제시킨다.

[22]는 멀티캐스트의 세션에 있는 서비스들은 가상 멀티캐스트 트리(virtual multicast tree)로 연결한다. 예를 들면 유니캐스트들로 구성된 트리는 end-host들과 연결을 한다. 트리는 MST(Minium Spanning Tree)로 형성되며 각 링크의 비용은 응용 명세 측정 기준(application specific metric)에 따른다.

위에서 언급한 오버레이에 관한 대부분의 연구들은 경제적이고 짧은 경로의 오버레이를 구축함으로서 사용자 서비스의 향상뿐만 아니라 네트워크의 확장에 유연하게 대처할 수 있는 장점을 보이고 있다. 그러나 이러한 방식을 효과적으로 구현하기 위하여서는 추가로 몇몇의 고려해야할 사항이 있다. 즉 서비스 성능을 높이기 위한 노드간 컨텐츠의 빈번한 복사는 오히려 메모리 활용의 비효율성을 초래할 수가 있으며, 오버레이의 수평적 동일성은 계층적 구조 등에 비하여 네트워크 내 서비스 정보의 수집이나 배분이 다소 복잡해 질 수가 있으므로 이를 위한 효과적인 전략을 고려할 필요가 있다. 본 논문에서는 이러한 목적을 달성하기 위한 효과적인 오버레이 캐싱 정책을 제안한다.

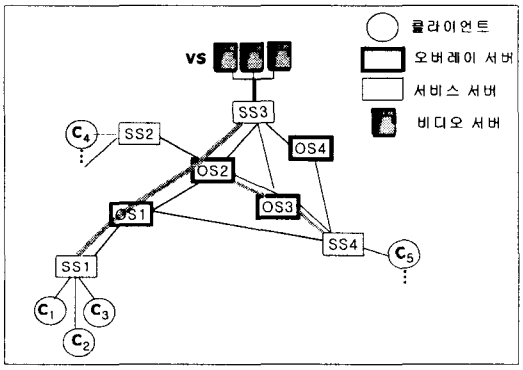
3. 제안기법

통상 오버레이 구조에서 서비스의 히트율을 높이기 위한 캐싱 스케쥴링은 한 개의 노드가 자신의 캐쉬에 원하는 비디오 스트림을 가지고 있지 않을 때, 이웃하는 노드에게 이를 방송함으로서 복사본이 어디에 있는지를 찾아 서비스하는 방식을 사용하고 있다. 그리고 비디오 원본이 저장되어있는 루트 서버에서 가져오게 되는 경우는, 그 경로에 속해 있는 모든 중간 노드들에게 복사 본을 저장하게 된다. 이러한 전략은 메모리 낭비가 심할 뿐만 아니라 상당이 큰 용량의 RAM이 필요하게 된다. 또한 최악의 경우는 인기 비디오 스트림들이 전 노드들의 메모리에 상주하게되어 빈번한 비디오 스트림의 교체로 인하여 서비스 지연이 두드러지게 된다. 따라서 이러한 문제점을 개선하여 적절한 자원 이용율에 대한 정책을 수행함으로서 네트워크의 대역폭 감소와 자원 이용률 향상을

높일 수 있는 구조를 설계한다.

3.1 제안한 오버레이 캐싱 구조

본 논문의 구조는 종전의 프락시 서버의 개념을 확장하고 인기도 기반을 고려하여 비디오 스트림 복사본의 적정한 양을 RAM을 이용하는 오버레이 서버와 비디오 원본을 가지고 있는 비디오 서버로 구성을 하였다. 어떤 클라이언트가 어떤 비디오 스트림을 요청할 때 서비스 서버에게 요청 메시지를 전송하고 서비스 서버는 자신의 캐쉬에 저장되어 있는 복사본이 있는지를 확인하고 복사본이 있으면 비디오 스트림을 클라이언트에게 서비스하고 아니면 오버레이 서버에게 비디오 스트림을 요청한다. 오버레이 서버들은 이러한 복사본을 가지고 있으면 요청한 서비스 서버에게 그 복사본을 전송하고 아니면 비디오 서버에서 원본을 요청한 서비스 서버에게 보내면서 그 경로에 있는 오버레이 서버의 메모리에 캐싱할 것인지를 [그림.4]의 알고리듬을 이용하여 결정한다.



[그림.3] 캐싱 오버레이의 예

예를 들면, 시간 t_0 에서 클라이언트 C_1은 비디오 스트림 k를 자기 자신의 라우터 서버 SS_1 에게 요청한다고 가정하자. 이 때에 라우터 서버 SS_1는 비디오 스트림 k가 자신의 캐쉬에 있으면 그 비디오 스트림을 클라이언트 C_1에게 서비스하고 k가 없으면 가까운 거리에 있는 오버레이 서버들에게 비디오 k가 있는지를 방송한다. 만약에 오버레이 서버들 모두가 비디오 스트림 k를 가지고 있지 않으면 최종적으로 비디오 서버 VS에서 가져오게 되는데, 그때 비디오 서버 VS와 서비스 서버 SS_1 사이의 최단 경로를 조사하여 그 경로내의 서버 즉, SS_3, OS_2, OS_1, SS_1 에 비디오 k의 복사 본을 저장하게 된다. 또한 복사본의 저장은 메모리의 효율성을 높이기 위하여 제안된 캐싱 정책에 의하여 선별 저장된다.

3.2 각 서버의 역할

본 논문에서 제안한 효율적인 캐싱을 하기 위한 오버레이 구조의 노드들은 비디오 서버와 오버레이 서버 그리고 서비스 서버로 구성되며 그 역할은 다음과 같다.

- 비디오 서버

이 서버는 비디오 스트림의 원본을 저장하고 있는 서버이다.

- 오버레이 서버

라우터의 기능을 가지고 있으면서 접근 속도를 높이기 위해 메모리(RAM)를 이용하여 비디오 스트림을 저장한다. 오버레이 서버는 메모리의 효율적인 관리를 위해서 메모리 맵(Memory Map)이라는 구조체를 운영하여 항상 이용 가능한 메모리에 비디오 스트림의 세그먼트들이 적재할 수 있도록 정책을 수립하고 수행한다.

- 서비스 서버

다수의 사용자와 직접 연결되는 네트워크의 최종단에 위치한 서버로서 프락시의 개념을 확장한 서버이다. 다양한 비디오 세그먼트들의 복사본을 저정함으로서 서비스의 질을 향상시킨다. 또한 클라이언트들의 요청하는 비디오의 인기도를 관리함으로서 오버레이 서버의 메모리에 비디오 스트림 유지를 하기위한 정보를 제공한다.

또한 서비스 지연 시간을 줄이기 위한 방법의 하나로서 페이징이 가능한 가상기억장치를 활용한다.

3.3 비디오 스트림 인기도와 캐싱 용량

비디오 스트림을 적절하게 복사본을 오버레이 서버에 두고 이들간의 협력을 함으로서 최소화할 수 있다. 이러한 방법을 구현하기 위해 오버레이 서버에 어떤 데이터를 얼마나 캐싱을 할지는 각 데이터들의 과거의 접근 정보를 활용하는 방법을 활용한다. 비디오 오브젝트의 인기도에서 Zipf 분포를 갖는다고 몇 가지의 경험적 연구에서 보고된 적이 있다[12]. 따라서 본 논문에서 비디오 스트림을 이를 기반으로 다음과 같이 정의한다.

라우터 서버 그룹 x 에서 비디오 오브젝트 i 의 접근빈도를 $f(i)$와 같이 정의하고, 총 비디오 오브젝트 수를 nn이라 하면, 시간 t 에서 비디오 오브젝트 i 의 인기도는 식(1)과 같이 계산할 수가 있다.

$$P_i^t = \frac{f(i)^t}{\sum_{n=1}^{nn} f(n)^t}$$

$$(n \in [1, 2, \ldots nn]) \qquad (1)$$

i 를 비디오 번호, j를 세그먼트 번호라 하였을 때 시간 t에서 비디오 i 에 대한 세그먼트 j의 크기를 S_{ij}라 하고 비디오 스트림의 전송경로에 있는 어떤 라우터 서버에 비디오 세그먼트들이 캐싱 되어 있는 용량을 T 라 하면 식(2)와 같이 정의 할 수가 있다.

$$T = \sum_{i=1}^{n} \sum_{j=1}^{jj} S_{ij}($$

$n = \ total\ video\ count$

$jj = \ total\ segment\ count\ of\ video\ i$)

(2)

또한 어떤 라우터 서버에서 특정한 비디오 오브젝트 i 의 세그먼트들이 캐싱 되어 있는 용량을 T_i 라 하면 역시 식(3)으로 표현할 수 있다.

$$T_i = \sum_{j=1}^{jj} S_j$$

($j = \ total\ segment\ count\ of\ video\ i$)

(3)

여기에서 총 캐싱 총량에서 비디오 오브젝트 i 가 차지하고 있는 비율과 인기도를 참조하여 오버레이 서버에 캐싱 할 것인지를 다음과 같이 결정한다.

```
/* i_j : j segent of Videold i
/* num-mem-seg : the number of memory
                 segment of current router server
/* free_space : free space in cache of current
                 router
/* pop_current : popularity of element in
                  cache of current router server
out-counter=0
temp_space= free_space
for out-counter = 1 to num-mem-seg
   if (i_j is mem-seg(out-counter) then
      if (temp_space ≧ size of i_j ) then
         insert i_j into cache
         free_space= temp_space - size of i_j
         update memory map
        end
      else
        perform cache-update
       end
next
perform cache-update
```

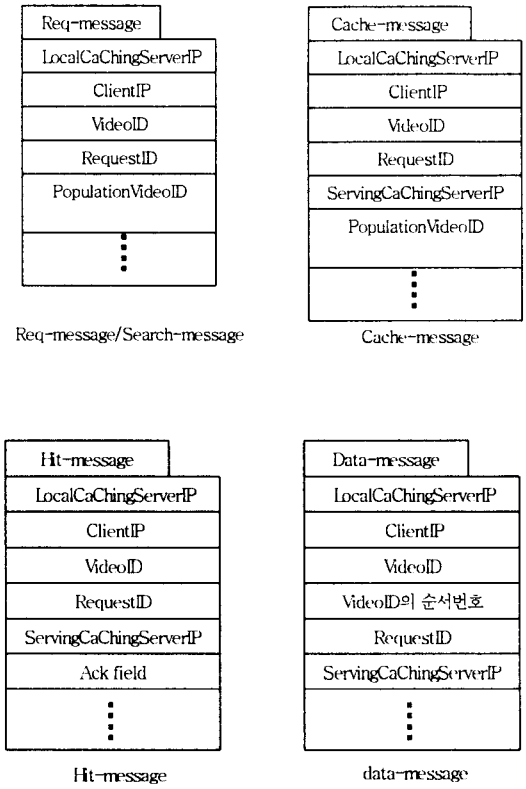
```
/*
/* cache-update routine
/*
in-count=0
for in-counter = 1 to num-mem-seg
   if (pop_current(in-count) ≦ P_i^t )or( P_i^t ≧ T_i / T )
      then
       evict element in cache
       temp_space += size of i_j
       if temp_space ≧ size of i_j ) then
          exit for
       end if
   end if
next
Insert i_j into cache
free_space= temp_space - size of i_j
update memory map
```

[그림.4] 오버레이 서버의 캐싱 배치 및 교체 알고리즘

3.4 비디오 스트림의 캐싱 동작 및 자료구조

클라이언트는 원하는 비디오 스트림을 Req-message
를 만들어 서비스 서버에게 보내면 서비스 서버는 자신의 캐쉬에 클라이언트가 원하는 비디오 스트림이 저장되어 있는지를 확인하고, 있으면 해당 클라이언트에게 비디오 스트림을 보내고, 없으면 인근의 라우터 서버에게 Search-message를 보낸다. 이때, 해당 비디오 스트림을 가지고 있는 라우터 서버들은 Search-message를 보낸 서비스서버에게 자신의 메모리 공간에 해당 비디오 스트림이 캐싱되어 있다는 Cache-message를 보낸다. 그러면 서비스 서버는 그 중에서 가장 가까운 거리에 있는 라우터서버(메시지를 보낼 때 타임 스탬프를 기록한다)를 선택한다. 그러나 모든 라우터 서버가 원하는 비디오 스트림을 가지고 있지 않을 경우는 비디오 서버에 저장되어 있는 비디오를 가지고 온다. 위에서 언급한 요청 프로세스를 수행하기 위한 메시지의 구조는 [그림.5]와 같으며 그 역할은 다음과 같다.

클라이언트가 비디오를 요청하기 위한 Req-message는 클라이언트 그룹에 속해있는 서비스 서버에게 보내는 메시지로서 서비스 서버 주소, 요청한 클라이언트 주소, 비디오 번호, 요청 번호 및 인기도 정보로 구성이 된다. Sarch-message는 서비스 서버가 클라이언트에게 서비스를 할 수 없을 때 이웃하는 라우터 서버들에게 비디오 스트림를 요청내는 메시지로서 Req-message의 구조와 비슷한다. cache-message는 Search-message를 받은 라우터 서버가 해당 비디오 스트림이 자신의 캐쉬에 저장되어 있을 때 보내는 메시지로서 Req-message에다 자신의 주소가 포함된다. Hit-message Search-message를 보낸 라우터 서버중에서 가장 경로가 짧은 라우터 서버에게 해당 비디오 스트림을 요청하고 그 나머지 라우터 서버에게는 무효라는 내용을 보내는 메시지이다. Data-message는 실제로 비디오 데이터를 보내기 위한 메시지로서 Ack-message에서 ack=1인 라우터서버가 그 역할을 담당한다.



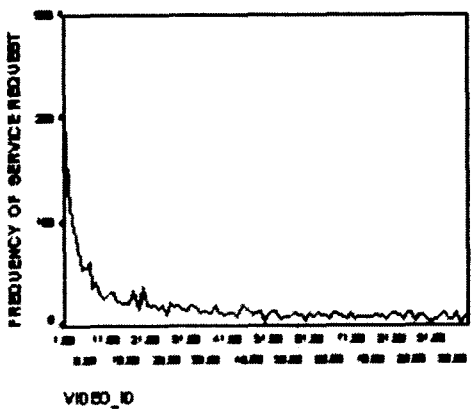
[그림.5] 메시지의 구조

또한, 라우터서버들의 캐싱되어있는 내용과 캐쉬 상태들을 관리하기 위한 캐쉬 맵(cache map table) 정보와 지역 캐싱 서버가 클라이언트들의 서비스 요청들을 관리하기 위한 로그 테이블(log table), 그리고 각 캐싱 서버가 어떤 클라이언트에게 어떤 비디오 스트림을 제공하고 있는 정보를 관리하기 위한 서비스 테이블(service table)들이 있다.

4. 실험 결과

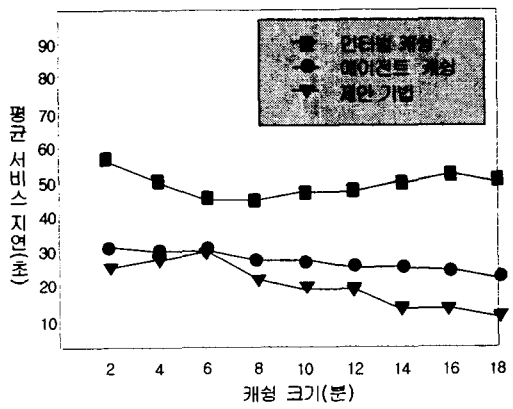
먼저 제안된 캐싱 정책의 효과를 테스트하기 위하여 연속 비디오 스트림 서비스의 표준적인 정책이라고 할 수 있는 인터벌 캐싱[8]및 오버레이를 활용한 에이전트 캐싱[9]에 대하여, 서비스 요청에 대한 지연율 및 캐쉬의 활용율 등을 비교 분석하여 보았다. 시뮬레이션에 사용되는 비디오 스트림은 90분 정도의 50개를 선정하고 세그먼트의 길이는 10분정도의 크기

로 하였으며, 비디오 당 9개의 세그먼트로 나누었다. Zipf 분포의 스큐 팩타는 $Z=0.7$에 따른다. 이를 도표로 나타내면 [그림. 6]와 같다.

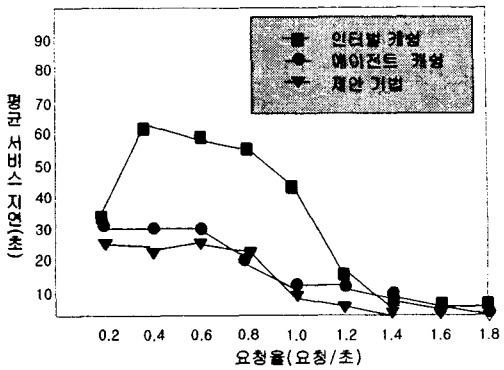


[그림.6] 인기도의 Zipf 분포

[그림.7]는 서비스 지연을 도해하였다. 인터벌 캐싱에서는 상당히 서비스 지연시간이 높은 반면 에이전트 캐싱에서는 상대적으로 매우 낮았다. 그러나 본 논문에서 제안한 방법은 인기 비디오 스트림에 대한 캐싱의 분포가 접근 빈도에 비례한 관계로 캐싱 에이전트에서보다 서비스 지연 시간이 평균이 약 15% 정도 낮은 결과를 보여준다.



[그림.7] 서비스 지연/캐쉬 크기 관계



[그림.8] 서비스 지연/요청율 관계

또한 [그림.8]에서는 서비스 요청율(요청/초)을 보여준다. 인터벌 캐싱은 평균 서비스 시간이 높은 이유는 대부분의 비디오 오브젝트들을 비디오 서버에서 가지고 오기 때문이고 에이전트 캐싱은 가까운 서버로부터 원하는 비디오 스트림을 가져오기 때문에 상대적으로 평균 서비스지연이 낮은 것으로 나타났다. 또한 본 논문에서 제안한 방법은 인기있는 비디오나 비인기의 비디오 스트림의 편차가 원만한 곡선을 보여줌으로써 3가지 방법 중 가장 우수한 결과를 보여주었다.

5. 결론

본 논문에서는 인터넷 상에서 비디오 스트림을 서비스할 때 인기도에 기반으로하는 서비스 지연시간을 최소화하는 방법을 제안하였다. 접근 빈도와 각 서버들의 총 캐싱 공간의 상관관계를 고려하여 제안한 알고리즘의 성능을 측정한 결과는 에이전트 캐싱 방법보다는 15%의 향상 그리고 인터벌 캐싱보다는 3배 정도의 성능 향상을 보여주었다. 그러나 비디오 스트림의 특성을 고려하여 프리패칭의 기법을 연구함으로써 서버들의 부하를 줄이고 클라이언트들에게 서비스의 질을 향상시킬 수 있는 방법이 요구된다.

참고 문헌

[1] S.Acharya and B.Smith, Middleman: "A video

caching proxy server" *In Proc. of The 10th International Workshop on Network and Operating System Support for Digital Audio and Video IEEE NOSSDAV*, 2000.

[2] Peter B Danzig, Richard S. Hall, and Kurt J.Worrell. "A case for caching file objects inside internetwork", *In. ACM SIGCOMM*, September 1993.

[3] Marc Abrams, Charles R.Standridge, Ghaleb Abdulla, Stephen Williams, and Edward A.FOX, "Caching proxies:Limitation and potentials", *In 4th International World Wide Web Conference, Boston, USA, www.w3.prg/conference/www4/papers/155*, 1995

[4] R.H.Patterson, G.A.Gibson, E.Ginting, D.Stodolsky, and J.Zelenka, "Informed Prefeching and Caching" *in Proceedings of the 15th ACM Symposium on Operating Systems Principles*, December 1995

[5] A.Dan, D.Dias, R.Mukherjee, D.Sitaram, and R.Tewari,"Buffering and caching in large scale multimedia servers", *in Proceedings of IEEE COMPCON, pp.217-224*, March 1995

[6] Y.W.Park, K.H.Baek, and K.D.Chung. "Reducing network traffic using two-layered cache servers for continuous media on the internet" *In Proc.of the IEEE Int'l Conf.on Computer Software and applications, pages 389-394*, 2000.

[7] Subhabrata.Sen, Jennifer,Rexford, and Donald F. Towsley, "Proxy Prefix Caching for Multimedia Streams," in *Proc. of IEEE INFOCOM, pp*= "1310-1319", 1999

[8] A.Dan and D.Sitaram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Environments", *in IS&T SPIE Multimedia Computing and Networking Conference,(San Jose, CA)*, January 1996.

[8] S.Gruber, J.Rexford, and A.Basso, "Protocol considerations for a prefix-caching proxy for multimedia streams", *in Proc. of the 9th International World Wide Web Conference, 2000*

[9] Duc.A.Tran, Kien.A.Hua, and Simon Sheu, "A new caching architecture for efficient Video-on-Demand services on the internet", *Proceedings of IEEE on Applications and the Internet, pp.172-181,* January 2003

[10] Machael.D.Dahlin, Randolph Y Wang, and Thomas E.Anderson. "Cooperative caching:Using remote client memory to improve file system performance", in 1st *Symposium on Operating Systems and Implementation,* November 1994.

[11] M.R.Korupolu and M.Dahlin Coordinated Placement and Replacement for Large-scale distributed caches", in *Proc. of the IEEE Workshop on internet applications,* Julj 1999

[12] M.Busari and C.Williamson, "On the sensitivity of web proxy cache performance to workload characteristics," in *Proc. of IEEE INFOCOM, April 2001.*

[13] E.J.O''Neil, P.E.O''Neil, and G. wEIKUM, "The LRU-k page replacement algorithms for database disk buffering," in Proc. of *International Conference on Management of Data,* 1993

[14] Reza,Rejaie, Haobo.Yu, Mark Handley, and Deborah.Estrin, "Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet", in *Proc. of IEEE INFOCOM, pp*= 980-989, 2000

[15] Craig E. Wills and Mikhail Mikhailov, "Towards a better understanding of {Web} resources and server responses for improved caching", *Computer Networks (Amsterdam, Netherlands: 1999),* vol=31, number = 11-16, pages = 1231-1243, 1999

[16] M.Abrams, C.R.Standrige, G.Abdulla, S.Williams, and E.A.Fox, "Caching proxies:Limirations and potentials", *Proceedings of the 4th International WWW Conference*, Boston,MA,Dec, 1995
November 1994.

[17] C.Doit, B.M.Levine, B.Lyles, H.Kassem, and D.Balensiefen, "Deployment issues for the IP multicast service and architecture", *IEEE Network magazine special isuue on multicasting, 14(1):78-88, January/February 2000*

[18] J.Jannotti, D.Gifford, K.Johnson, .Kaashoek, and Jr.J.O'Toole, "Overcast:Reliable multicasting with an overlay network", in Proc. of 4st *Symposium on Operating Systems and Implementation(OSDI), pp=197-212,* October 2000

[19] Y. Chawathe, "Scattercast:an architecture for internet broadcast distribution as an infrastructure service", In *phD Thesis, University of California at Berkeley,2000*

[20] S.Banerjee, and B.Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast", *In ACM SIGCOMM*, 2002

[21] Yang-Hua.Chu, Sanjay.G.Rao, and Hui Zhang, "A case for end system multicast", *ACM SIGMETRICS*, pp=1-12, 2000

[22] D.Pendarakis, Shi.Sherlia, Verma.Dinesh, and Waldvogel, Marcel, ALMI: "An Application Level Multicast Infrastructure," *Proceedings of the 3rd USNIX Symposium on Internet Technologies and Systems (USITS '01), pp=49-60,* Mar, 2001

[23] Stefan.Savage, Tom Anderson, Amit Aggarwal, David.Becker, Neal.Cardwell, Andy.Collins, Eric.Hoffman, John.Snell, Amin.Vahdat, Geoff.Voelker, and John.Zahorjan Dour:"A case for informed Internet routing and transport:, *IEEE Micro, 19(1):50-59,* January 1999

남 현우

1986. 3~1993. 2 : 아주대학교 공과대학 컴퓨터공학과 학사졸업
1994. 3~1996. 2 : 아주대학교 대학원 컴퓨터공학과 석사졸업
1996. 3~1999. 2 : 아주대학교 대학원 컴퓨터공학과 박사수료
2000. 3~현재 : 강남대학교 및 시립인천전문대학 강사
2003. 5~현재 : (주)이테크 기술이사
관심분야 : 컴퓨터그래픽스, 이미지프로세싱, 멀티미디어

김 종경

1980 .3 - 1984. 4 : 광운대학교 전자계산소 주임
1984. 4 - 1999. 7 : 경희대학교 정보처리처 팀장
1999. 3 - 2003. 7 : 인천대학교, 경기대학교 강사
1999. 12 - 현재 : (주)원포텍 이사
2003 . 3 - 현재 : 시립 인천 전문대학 컴퓨터정보과 겸임교수
1990. 2 : 경희대학교 산업정보대학원 전자계산공학 (공학석사)
1999. 7 : 아주 대학교 일반대학원 컴퓨터공학과 (박사수료)
관심분야 : 멀티미디어 응용 및 시스템구조, 소프트웨어 시스템 디자인, 운영체제