인터넷상에서 NOD 서비스를 위한 연속미디어 전송 및 푸쉬-캐싱 기법

박 성 호† · 임 은 지† · 최 태 욱† · 정 기 동††

요 약

멀티미디어 뉴스 정보를 인터넷을 통해 서비스하기 위해서는 서버의 과부하, 네트워크의 혼잡, 클라이언트에 대한 응답 지연 등의 문제를 해결하여야 한다. 본 논문에서는 연속미디어 데이터의 일부분 또는 전체를 푸시 또는 캐싱시키는 프락시 푸시-캐싱기법과 이를 지원하기 위한 연속미디어 전송기법인 RTP-RR과 RTP-*n*R을 제안하였다. 프락시 푸시-캐싱기법은 연속미디어 데이터가 생성되었을 때 데이터의 일정부분을 프락시로 푸시시키고, 데이터의 캐싱 효율값에 따라 캐싱되는 길이를 유동적으로 변화시킴으로써 클라이언트의 서비스 초기지연시간을 감소시키는 동시에 캐쉬에서 서비스되는 데이터량을 증가시킨다. 또한, 연속미디어 스트림이 캐쉬에서 차지하는 디스크 공간과 스트림에 대한 클라이언트의 요구량 사이의 상관관계를 고려하여 스트림의 캐싱효율(caching utility)을 측정하고, 그 값을 이용하여 세그먼트 재배치 기법을 제안하며, 프락시 서버의 relay buffer의 대기 시간을 이용하여 연속미디어의 실시간성을 보장하면서 패킷손실을 줄이는 방법을 제시하였다. 그리고 모의 실험을 통하여 제안하는 기법과 다른 캐싱 기법들과의 성능을 비교 측정하였다. 프락시 푸시-캐싱기법의 성능을 LRU와 비교하면 BHR(Byte Hite Rate)은 3%~15% 증가하였으며, 서비스 초기지연시간은 약 20% 감소하였다. 그 외 재배치 횟수, 패킷손실률 측면에서도 성능이 향상됨을 보였다.

A Push-Caching and a Transmission Scheme of Continuous Media for NOD Service on the Internet

Seong-Ho Park† · Eun-Ji Lim† · Tea-Uk Choi† · Ki-Dong Chung††

ABSTRACT

In multimedia news service on the internet, there are problems such as server overload, network congestion and initial latency. To overcome these problems, we propose a proxy push-caching scheme that stores a portion of continuous media stream or entire stream, and a transmission scheme of NOD continuous media, RTP-RR and RTP-*n*R to exploit push-caching scheme. With the proposed push-caching scheme, NOD server pushes fixed portion of stream to a proxy when news data is generated, and the cached size of each stream changes dynamically according to the caching utility value of each stream. As a result, the initial latency of client side could be reduced and the amount of data transmitted from a proxy server to client could be increased.

Moreover, we estimate a caching utility value of each stream using correlation between disk space occupied by the stream and the amount of data stream requested by client. And we applied the caching utility value to replacement policies.

The performance of the proxy push-caching and continuous media transmission schemes proposed were compared with other schemes using simulations. In the simulation, these schemes show better results than other schemes in terms of BHR (Byte Hit Rate), initial latency, the number of replacement and packet loss rate.

※ 이 논문은 (1998)년 한국학술진흥재단의 학술연구비에 의하여 지원되었음.

† 준 회 원 : 부산대학교 대학원 전자계산학과
†† 종신회원 : 부산대학교 전자계산학과 교수

논문접수 : 2000년 4월 26일 심사완료 : 2000년 5월 25일

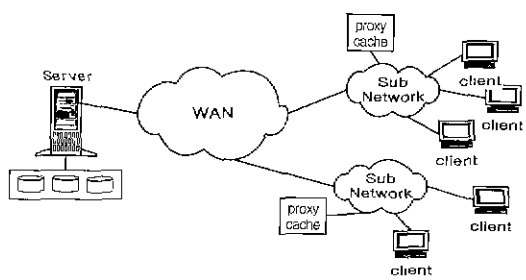
1. 서 론

컴퓨터와 통신기술, 그리고 디지털 처리 기술의 급격한 발전으로 기존에 종이와 잉크로 표현되던 신문의 뉴스보다 멀티미디어를 통한 뉴스에 대한 요구가 증가하고 있다. 따라서 인터넷을 통하여 멀티미디어 뉴스 정보를 제공하는 NOD(News On Demand) 시스템에 대한 연구가 진행되고 있다. 그런데, 최근 들어 인터넷의 사용자가 증가함에 따라 서버의 과부하, 네트워크의 혼잡, 클라이언트에 대한 응답 지연 등의 문제가 많은 응용에서 대두되고 있다. 동시에 수많은 사용자들에게 비디오나 오디오와 같은 대용량의 실시간 데이터를 제공해야 하는 NOD 시스템에서도 역시 이러한 문제는 간과될 수 없다.

그 해결책의 하나로 웹 캐싱(caching) 기법의 사용을 들 수 있다[1, 2]. 이 기법은 인터넷상에서 자주 접근되는 데이터를 지역적으로 클라이언트에서 가까운 위치에 있는 프락시 서버의 디스크 캐쉬에 저장해 둠으로써 클라이언트가 겪게 되는 초기 지연시간(initial latency)과 서버로의 접근 횟수를 줄여서 서버의 부하와 네트워크 트래픽을 감소시킨다. 그러나 기존의 웹 캐싱 기법을 NOD 시스템에 그대로 적용시키기에 다음과 같은 여러 가지 문제점이 따른다. 첫째, 기존의 웹 캐싱 기법들은 텍스트나 이미지와 같은 전통적인 미디어를 위해 고안되었기 때문에, 상대적으로 데이터의 크기가 크고, 실시간성을 요구하는 연속미디어 서비스에 적용하기에는 적합하지 않다. 둘째, 기존의 웹 캐싱은 텍스트나 이미지 데이터를 객체 단위로 캐싱하므로 연속미디어와 같은 대용량 데이터를 캐싱하는 정책으로 적합하지 않다. 즉, 캐싱된 연속미디어 데이터가 접근 빈도(hit rate)가 낮을 경우, 디스크 캐쉬 공간의 낭비가 발생한다. 셋째, 현재의 프락시 서버는 텍스트 또는 이미지와 같은 비연속미디어를 기반으로 설계되었기 때문에 실시간 특성을 가지는 연속미디어를 전송하기에는 적합하지 못하다. 또한 인터넷의 전송 프로토콜인 TCP는 신뢰성 있는 전송을 보장하지만, 이를 위한 에러복구나 재전송 메커니즘은 실시간성을 보장할 수 없으며, UDP는 일정 수준의 실시간성을 제공할 수 있으나 네트웍의 과부하가 발생하는 경우 인터넷상에서 데이터를 전송할 때 많은 패킷손실이 발생한다. 그리고 최근에 많이 사용되는 RTP는 연속미디어의 실시간 전송과 피드백을 이용한 흐름제어 메커니즘을 제공하지만 본질적으로 하위 전송프로토콜을 UDP를 사용하기 때문에 네트웍 과부하상태에서 패킷손실이 일어날 경우 복구할 수 있는 메커니즘이 없다.

따라서 본 논문에서는 (그림 1)과 같은 NOD 서비스 환경에서 연속미디어에 적합한 프락시 캐싱 방법과 서버에서 기사가 생성되었을 때 기사 데이터의 일정부분을 멀티캐스트를 통해 프락시 서버로 푸쉬하여 캐쉬의 성능을 향상시키고 서버 및 네트워크의 부하를 줄이는 프락시 푸쉬-캐싱기법을 제시한다. 프락시 푸쉬-캐싱 기법은 연속미디어의 특성을 고려하여 선택된 스트림 전체를 캐싱하기보다는 스트림의 앞부분을 캐싱함으로써 클라이언트의 초기지연시간을 줄이고, 캐싱하는 양을 유동적으로 변화시켜 스트림의 인기에 따른 디스크 사용공간을 차별화한다. 그리고 스트림의 디스크 사용량과 클라이언트에서 재생되어지는 스트림 데이터의 총합을 고려한 재배치 정책을 적용하여 NOD 서비스 시스템의 성능을 향상시킨다

또한 프락시 서버에서 실시간성을 가진 연속미디어 데이터 푸쉬와 캐싱을 위한 전송기법인 RTP-RR과 캐싱되지 않는 데이터를 서버에서 프락시 서버를 통해 클라이언트에게 실시간으로 전송하는 기법인 RTP-*n*R 전송기법을 제시한다. RTP-*n*R과 RTP-RR 전송기법은 인터넷 표준프로토콜 RTP를 확장하여 신뢰성과 실시간성을 제공한다. 그리고 프락시 서버는 서버와 클라이언트간에 원활한 서비스를 위해 흐름제어나 에러복구기능을 수행한다.



(그림 1) NOD 서비스 환경

본 논문의 구성은 다음과 같다. 2장에서 현재까지 진행되어 온 관련 연구에 대하여 고찰하고, 3장에서는 연속미디어 데이터를 위한 프락시 푸쉬-캐싱 기법을 제시하며, 4장에서는 프락시 서버를 통해 클라이언트에게 실시간으로 데이터를 전송하는 연속미디어 전송기법을 제시한다. 그리고 5장에서는 모의 실험을 통해

본 논문에서 제시한 기법의 성능을 평가하고, 마지막 6장에서는 결론과 향후 연구 방향에 대하여 살펴본다.

2. 관련연구

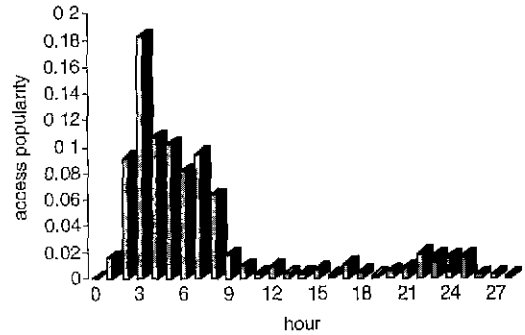
2.1 연속미디어 데이터의 캐싱

파일시스템이나 데이터베이스 시스템에서의 메모리 캐싱 정책은 대부분 데이터 접근의 최신성(recency)과 빈도수(frequency)를 기반으로 한다[3, 4]. 이러한 정책은 텍스트 위주의 데이터를 다루기 때문에 대용량, 고대역폭을 요구하는 NOD 연속미디어 데이터의 캐싱 정책으로 적합하지 못하다. 또한 비디오나 오디오와 같은 연속미디어 데이터를 위한 버퍼 재배치기법인 Distance Caching 방법[5]과 접근의 연속성을 이용하여 요구 시간 간격이 일정한 값 이하의 경우에는 그 데이터를 연속적으로 캐싱하고 일정한 값을 초과하면 캐싱하지 않는 Interval Caching 방법[6] 등이 제시되었다. 이러한 기법들은 메모리 기반의 캐싱 기법이므로 인터넷 환경의 프락시 캐쉬에 직용시키기는 부적합하다.

그리고 인터넷상에서 객체 단위로 데이터를 캐싱하는 웹 문서 캐싱에 대한 연구는 주로 웹 캐쉬의 재배치 알고리즘에 초점을 맞추고 있다 보편적으로 널리 사용되는 알고리즘은 LRU, LFU, SIZE, LRU-SIZE, LRU-MIN, LRFU 등을 대표적인 예로 들 수 있다[7] 그러나 웹 문서는 대부분 text나 image와 같이 비교적 크기가 작은 데이터에 적용하기 위해 연구되었으며 대용량인 연속미디어 적용하기에 적합하지 않다 특히 [7]에서는 사용자 접근 정보와 네트워크 위상(topology)과 같은 서버가 가진 전역적인 정보를 이용하여 캐싱하는 geographical push-caching을 제안했고, [8]은 향후 데이터의 접근을 예측하여 미리 캐쉬에 Preload 또는 Prefetching하는 방법으로 캐쉬 성능을 향상시키는 방법을 제안하였다.

2.2 NOD 데이터의 생명주기와 인기도

NOD 기사는 수시로 생성되고 일정 시간이 지나면 접근 빈도가 급격히 감소 등의 생명 주기를 가진다. 또한 사용자는 대부분은 새로운 기사를 더 선호하므로 생명주기에 따른 인기도 변화 패턴이 뚜렷하게 나타난다. 즉, 생성 직후에 접근 빈도가 급격하게 증가했다가 시간이 흐를수록 접근 빈도는 서서히 감소한다. (그림 2)는 실제 전자신문의 로그파일을 분석하여, 시간에 따른 기사의 인기 변화를 조사한 것이다[9].



(그림 2) 생명주기에 따른 접근확률 분포

2.3 연속미디어 전송 기법

네트워크를 통해 동화상이나 음성과 같은 연속미디어 데이터의 원활한 서비스를 제공하기 위한 많은 연구가 수행되었다 특히 VBR 데이타의 특성을 이용하여 네트웍의 대역폭을 효율적으로 사용하는 대역폭완화(Bandwidth Smoothing)기법[11, 12], Layered Coding에 기반하여 네트웍의 상태에 따라서 전송품질을 조절하는 기법[14] 등이 연구되었다 그러나 위에서 제시한 방법은 대역폭 예약이 어려운 인터넷 환경에서 좋은 성능을 보이지 못한다. 또한 인기 있는 연속미디어의 경우 Boradcast, Multicast, Batching 등을 통한 전송 방법들이 있다[15].

2.4 프락시 서버에 관한 연구

서버의 부하를 줄이고, 네트웍 통신비용을 줄이기 위한 Proxy를 이용하는 연구가 많이 진행되어왔다 [16-18] 그러나 대부분 Proxy 서버에 관한 연구는 이미지나 텍스트와 같은 웹 파일을 대상으로 하고있어, 연속미디어의 경우는 그 연구가 미비한 상태이다.

3. 프락시 푸쉬-캐싱

일반적으로 인터넷상의 프락시 서버는 (그림 1)과 같이 서버에서 클라이언트로의 경로 상에서 클라이언트에 근접하여 위치한다 NOD 서버와 프락시 서버는 아래의 시나리오에 따라 NOD 데이터를 클라이언트에게 서비스한다

• 시나리오

① NOD 서버는 새로 생성된 연속미디어 스트림 객체의 일정부분을 멀티캐스트 전송방법으로 각 프락시 서버로 푸쉬한다.

② 클라이언트의 요청에 따라 프락시 서버는 NOD 서버를 대신하여 스트림을 사용자에게 전송한다.

③ 프락시 서버는 일정시간 동안에 발생하는 스트림 객체에 대한 클라이언트의 요구량을 기준으로 각 스트림에 대한 캐싱 데이터 양을 최적화 한다. 그러므로 프락시 서버의 캐쉬는 스트림 데이터의 일부분 또는 전체를 저장한다.

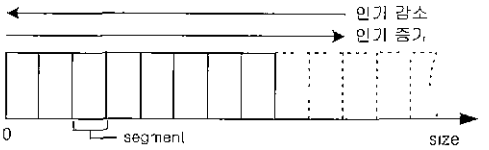
④ 프락시 서버는 클라이언트 요구한 스트림 중, 프락시 서버의 캐쉬에 없는 데이터에 대하여서는 Relay 서비스를 실시한다.

⑤ 클라이언트는 스트림 데이터 전체를 요청하거나 시작부터 일정부분까지를 요청한다. 즉 클라이언트는 재생 후 임의의 시간에 연속미디어의 재생을 정지시킬 수 있다.

3.1 NOD 연속미디어 데이터를 위한 캐싱 기법

멀티미디어 데이터는 대용량이기 때문에 전통적인 데이터와 같이 객체단위로 캐싱할 경우 적은 수의 객체만을 저장할 수 있어서 캐쉬 공간의 사용 효율이 감소하고 재배치 비용이 커진다. 따라서 캐쉬에는 스트림의 일부분을 저장하되 앞부분부터 점진적으로 캐싱하는 방식을 사용하여 클라이언트에 대한 서비스 초기 지연시간을 줄이고, 스트림에 대한 클라이언트의 요구량의 변화에 따라 캐싱되는 데이터 양을 유동적으로 변화시켜 프락시 캐쉬의 사용 효율을 증가시킨다. 즉, 각 스트림에 대한 클라이언트들의 요구량이 증가하면 캐싱되는 데이터 양을 점차적으로 늘려주고 반대로 감소하면 캐싱되는 데이터의 양을 점차로 줄인다. 이러한 방법으로 생명주기에 따라 인기 변화가 뚜렷한 NOD 데이터의 특성을 수용하고, 각 스트림에 대한 클라이언트의 요구량에 따라 캐쉬에서 차지하는 공간을 차별화한다. 이 캐싱 정책은 클라이언트의 서비스 초기지연을 줄이고, 프락시 캐쉬에 캐싱되는 스트림의 데이터 양을 스트림에 대한 클라이언트의 요구량에 비례하는 상태로 유지시킴으로써 캐싱된 데이터에 대한 Hit Rate를 증가시키고 재배치 횟수를 줄인다. 그러나 Hit Rate와 재배치 횟수는 Trade-off 관계에 있으므로 본 논문에서는 최적의 Hit Rate와 재배치 횟수를 고려한 캐싱 기법을 제시한다.

본 논문에서는 일정 시간 동안에 특정 스트림 데이터에 대한 클라이언트의 데이터 요구량를 인기도라 정하며, 인기도를 기준으로 스트림 데이터를 세그먼트 단위로 캐싱하거나 재배치하는 정책을 사용한다. 즉, 인기도에 큰 변화가 없을 경우에는 재배치 횟수가 적고 캐쉬는 안정화된 상태로 머물게 되는 반면 인기도에 급격한 변화가 있을 경우에는 재배치 오버헤드는 증가한다. (그림 3)은 이러한 캐싱 정책에 따른 스트림의 캐싱 패턴을 보여준다.



(그림 3) 스트림의 인기도 변화에 따른 캐싱 패턴

클라이언트의 요구가 들어오면 프락시는 캐쉬에 저장되어 있는 앞부분의 데이터를 클라이언트에게 바로 전송해주고, 저장되어 있지 않은 뒷부분의 데이터를 서버로부터 패치(fetch)하여 클라이언트에게 전송한다. 스트림이 요구될 때마다 프락시 서버는 요구된 데이터 스트림의 인기도와 캐싱되어 있는 데이터의 양을 고려하여 추가의 세그먼트를 캐싱할 것인지를 판단한다. 즉, 스트림의 인기도에 비해 캐싱되어 있는 데이터의 양이 작다고 판단될 경우 추가의 세그먼트를 캐싱하여 캐싱된 스트림의 길이를 늘려준다. 그러나 새로운 세그먼트를 캐싱하기 위한 여유 공간이 캐쉬에 없다면 캐쉬 재배치를 수행한다.

3.2 재배치 정책

프락시 캐쉬에 새로운 세그먼트를 저장하려고 할 때 여유 공간이 없으면 현재 캐싱되어 있는 스트림들의 세그먼트를 재배치하여야 한다. 본 논문에서는 일정시간 동안에 프락시 서버를 통해 서비스되어진 스트림 데이터 양과 그 스트림에 할당된 캐쉬 공간에 기반한 *캐싱 효율(caching utility)*을 적용하여 희생(victim) 스트림을 선택하고, 그 스트림의 마지막 세그먼트를 재배치 세그먼트로 선택한다.

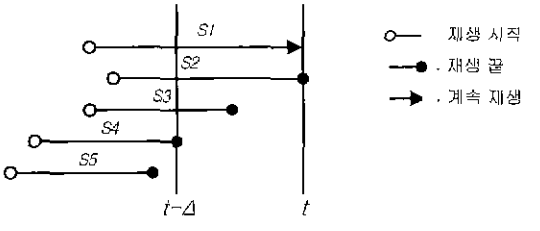
3.2.1 캐싱 효율(Caching Utility)

캐싱 효율은 스트림을 캐싱함으로써 얻을 수 있는

이익과 자원의 사용량에 의해 결정된다. 스트림의 비용에 대한 캐쉬 이익의 비율을 스트림의 캐쉬 효율이라고 정의한다.

$$Caching\ Utility = \frac{Caching\ Benefit}{Cost} \quad (1)$$

- **비용(cost)** : 스트림이 캐쉬에서 차지하는 저장공간, 즉, 스트림의 캐쉬되어 있는 양으로 볼 수 있다. 세그먼트 단위로 계산되며, 스트림 i의 캐쉬되어 있는 세그먼트 수를 C_i라 한다.
- **캐쉬 이익(Caching benefit)** : 어떤 연속미디어 데이터를 얼마만큼 캐쉬하는 것이 전체 시스템을 위해 가장 효율적인가를 결정하는 **캐쉬 이익**은 일정 시간 t를 기준으로 하여 프락시 서버를 통해 클라이언트에서 재생되어진 스트림 데이터의 총량으로 나타낸다. (그림 4)는 특정 스트림에 대한 클라이언트의 재생 총량을 나타낸다. 즉, 스트림 *S2*, *S3*, *S4*와 같이 $[t-\Delta, t]$에 종료하거나 *S1*과 같이 t시간에 재속적인 재생을 요구하는 클라이언트의 재생 총량이다. 그러나 스트림 *S5*는 특정 시간 t를 기준으로 한 캐쉬 이익을 산출하는 데이터에서는 제외된다 이는 클라이언트가 임의의 위치에서 연속미디어의 재생을 정지시킬 수 있으므로 본 논문에서는 캐쉬이익을 접근 빈도수보다는 클라이언트에서 재생되는 스트림 데이터의 총량(*S1*+*S2*+*S3*+*S4*)으로 정의한다.



(그림 4) 클라이언트 재생 총량

캐쉬 이익은 스트림에 대한 클라이언트의 접근빈도와 재생지속시간에 의해 결정되며, 다음과 같이 정의된다. 스트림 i의 총 재생량을 P_i라 하고, 클라이언트 j가 스트림 i를 재생한 량을 $P_{i,j}$, 클라이언트 j가 스트림 i를 캐쉬에서 재생한 량을 $CS_{i,j}$, 클라이언트 j가 스트림 i를 Relay를 통해 재생한 량을 $RS_{i,j}$ 하며, 재배치가 발생하는 시간 t, 시간 윈도우(time windows)의 크기 Δ, 일정 시간간격 $t-\Delta$ 동안에 스트림 i를 재생한 클라이언트 수를 k, 일정 시간간격 $t-\Delta$ 동안에 재생이 종료된 클라이언트를 TF_j, 특정 시간 t에서 계속적으로 재생을 요구하는 클라이언트를 TC_j라고 하면, 총 재생량 P_i는 식 (2)와 같다. 계산 단위는 세그먼트 개수이며, 클라이언트의 재생이 끝날 때마다 재생량을 기록하고, 프락시는 그 정보를 유지한다

$$\begin{aligned} P_i &= \sum_{j=0,\ TF_j \in [t-\Delta,\ t)}^{k} (CS_{i,j} + RS_{i,j}) \\ &\quad + \sum_{j=0,\ TF_j \in [t]}^{k} (CS_{i,j} + RS_{i,j}) \\ &= \sum_{j=0,\ TF_j \in [t-\Delta,\ t)}^{k} P_{i,j} + \sum_{j=0,\ TC_j \in [t]}^{k} P_{i,j} \end{aligned} \quad (2)$$

여기서 Δ는 시간 윈도우(time windows)의 크기를 나타내며, 시간 윈도우를 사용함으로써 최근에 많이 재생이 된 스트림일수록 P_i 값은 크게 나타나고, 최근에 적게 재생된 스트림은 P_i 값이 작게 나타난다. 그러므로 P_i 접근의 최근성(recency)를 반영한다. 또한, 스트림에 대한 인기도의 변화에 따라 P_i 값도 변화한다.

그러므로 스트림 i의 캐쉬효율 CU_i는 식 (3)과 같다.

$$\begin{aligned} CU_i &= \frac{P_i}{C_i} \\ &= \frac{\sum_{j=0,\ TF_j \in [t-\Delta,\ t]}^{k} P_{i,j} + \sum_{j=0,\ TC_j \in (t-\Delta,\ t]}^{k} P_{i,j}}{C_i} \end{aligned} \quad (3)$$

3.2.2 재배치 알고리즘

제안하는 캐쉬 재배치 정책의 주된 목적은 스트림의 캐쉬된 길이를 스트림의 인기도에 비례하는 상태로 유지하는 것이다. 즉, 모든 스트림들의 캐쉬 효율 값이 비슷한 상태로 되게 하는 것이다. 따라서 재배치를 수행할 때는 캐쉬 효율 값이 가장 작은 스트림의 맨 뒷 세그먼트를 해제하여 그 스트림의 캐쉬된 길이를 줄임으로써 캐쉬 효율 값을 증가시킨다.

s : 세그먼트의 크기

L_i : 스트림 i의 전체 길이

C_i : 스트림 i의 캐쉬된 세그먼트 수

P_i : 스트림 i의 총 재생량

$CU_i(C_i)$: 스트림 i의 캐쉬 된 세그먼트 수 C_i에 대한 캐쉬 효율 값

```
save_segment_in_cache(stream i, segment k) {
        // cache에 여유공간이 있을 경우
        if ( free space in cache >= s ) {
                ε = L_i - C_i·s, // 캐싱되지 않은 데이터 양
                if( free space in cache >= ε )
                        stream i의 뒷부분 ε를 캐싱;
                else
                        stream i의 뒷부분을 free space 만큼 캐싱;
                return;
        }
        // cache에 여유공간이 없을 경우
        // stream i의 segment k를 제거했을 때의 캐싱효율 계산.
        CU_i(C_i+1) = P_i/(C_i+1).
        while(캐쉬 상의 모든 stream에 대하여)
                CU_i 계산.
        CU_i 값이 최소인 것을 victim stream v로 선택;
        // i의 인기가 적으면 segment k를 캐싱하지 않음
        if ( CU_v(C_v-1) > CU_i(C_i+1) ) return;

        stream v의 캐싱된 맨 뒷 segment 하나를 해제.
        stream i의 segment k를 캐싱.
        return;
}
```

(그림 5) 재배치 알고리즘

3.3 NOD 서버에서의 푸시

NOD 데이터는 매일 수시로 생성되고, 새로운 데이터에 대한 선호도가 높다는 특징을 가진다. 그러므로 서버에서 NOD 데이터가 생성되었을 때 연속미디어 데이터의 일정부분을 멀티캐스트를 통해 프락시 서버로 푸쉬하여 캐쉬의 성능을 향상시키고 서버 및 네트워크의 부하를 줄이는 프락시 푸쉬 정책을 제시한다. 즉, 단순한 캐싱 기법만을 사용하면 최근에 생성된 데이터에 대한 사용자 요구가 캐쉬에 캐싱될 때까지는 일정한 시간이 경과하여야 함으로 캐쉬의 효율을 떨어뜨린다. 이러한 문제를 해결하기 위하여 새로운 데이터에 대한 요구가 발생하기 전에 미리 데이터를 프락시에 캐싱하는 방법을 사용한다.

NOD 서버에서 새로 생성된 NOD 데이터는 오래된 NOD 데이터보다 접근될 확률이 높을 것이라고 예측하고, 연속미디어 데이터가 생성되면 데이터의 앞부분 일정량을 프락시로 푸쉬시켜서 연속미디어 데이터에 대한 요구의 초기지연시간과 히트률을 향상시킨다. 그리고 푸쉬된 데이터의 실제 인기도가 예측된 인기도보다 낮게 나타나면 기사는 캐쉬에서 삭제하고, 반대로 인기가 증가하면 추가로 캐싱한다. NOD 서버는 연속미디어 데이터의 앞부분을 세그먼트 단위로 프락시에게 푸쉬하며, 프락시의 환경에 따라 푸시하는 세그먼트 개수 n을 결정한다. 즉, n이 너무 작으면 푸쉬를 사용함으로써 얻을 수 있는 효과가 작고, n이 너무 크면 재배치 오버헤드가 크게 나타난다.

4. 연속미디어 전송기법

NOD 서비스 시스템의 성능을 향상시키기 위하여 인기도가 높은 스트림 데이터를 프락시 서버에 푸싱이나 캐싱 정책을 적용하기 위해서는 전송되는 데이터에 손실이 없는 전송기법이 요구되며, 프락시 서버에 캐싱 되지 않은 데이터를 서버에서 프락시 서버를 통해 클라이언트로 전송할 때는 실시간성을 보장하는 전송기법이 필요하다. 후자의 경우, 프락시 서버가 인터넷상에 leaky bucket역할을 함으로써 클라이언트에게 원활한 서비스를 제공할 수 있다. 또한 연속미디어의 실시간성을 보장하고 멀티캐스트 환경을 지원하기 위하여 인터넷 환경에서의 실시간 전송 프로토콜인 RTP를 확장하여 사용한다. 그러나 RTP는 전송과정에서 손실된 데이터에 대한 복구 기능이 없으므로 푸싱이나 캐싱을 위한 전송 프로토콜로 적합하지 않다.

본 논문에서는 프락시 서버를 통해 데이터를 전송함으로써 전송 데이터가 프락시 버퍼에 일정시간동안 대기하는 특성을 이용하여, 이 시간동안에 데이터의 실시간성을 손상시키지 않는 범위에서 서버와 프락시간의 전송오류를 복구하는 RTP-*n*R(with *n*th Retransmission)전송 기법을 제시한다. 그리고 프락시 푸싱-캐싱을 위해 RTP 프로토콜에 신뢰성을 추가한 RTP-RR (with Reliable Retransmission) 전송 기법을 제시한다.

4.1 RTP에 기반한 신뢰성 있는 재전송 기법

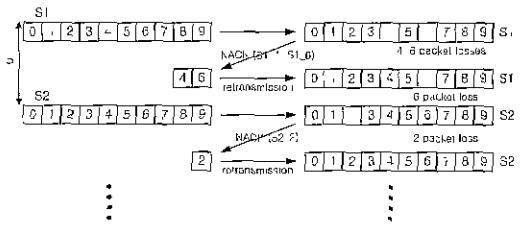
4.1.1 RTP-*n*R 전송 기법

RTP-*n*R 기법의 기본적인 헤더구조와 실시간 전송 방법은 RTP에 기반하며, 손실된 데이터에 대하여서는 프락시 버퍼에서의 대기 시간을 활용하여 재전송을 요청함으로써 패킷손실에 대한 복구기능을 추가한 RTP 확장 프로토콜이다. RTP-*n*R의 재전송 요청 횟수 *n*는 신뢰성과 프락시 버퍼에서의 대기시간을 고려하여 결정된다.

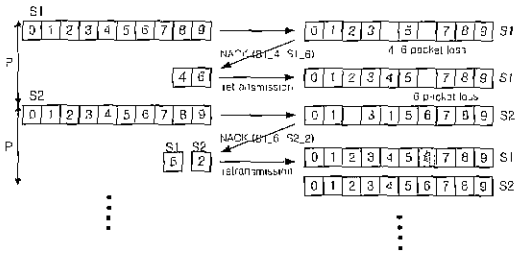
서버는 파일을 일정한 세그먼트의 크기로 나누고 이를 RTP-*n*R 프로토콜을 사용하여 주기적으로 전송한다. 그리고 서버와 프락시 사이의 전송 패킷의 크기가 클 경우 패킷손실의 가능성이 높으므로, 한 주기에서 각 세그먼트를 더 작은 여러 개의 패킷으로 나눠서 전

송한다. 프락시 서버는 일련의 패킷을 저장할 세그먼트 크기의 공간을 유지하고 주기별로 패킷들을 전송 받는다. 이 때 손실된 패킷이 발생할 경우 해당 패킷이 차지하는 세그먼트의 공간을 버퍼에 유지하며, 손실된 패킷들을 검사하여 재전송 요구를 서버에게 보낸다 서버가 프락시 서버의 재전송 요구를 받을 때 서비스 주기의 남는 대역폭을 이용하여 재전송하며, 서비스 주기에 남는 전송 대역폭이 없을 경우 재전송요구를 무시한다. 재 전송된 패킷은 세그먼트정보와 sequence number를 바탕으로 프락시 서버의 버퍼에서 유지되고 있는 세그먼트의 공간에 손실된 패킷을 적재된다.

(그림 6)은 한번의 재전송만을 허용하는 RTP-1R(n=1)이며, (그림 7)은 두 번의 재전송을 허용하는 RTP-2R(n=2) 기법을 보여준다. 전자의 경우, 한 주기 안에 한번의 재전송이 실패할 경우 다시 재전송 요구를 보내지 않으나 후자의 경우 다음 주기에 다시 재전송을 허락한다. 이와 마찬가지로 신뢰성을 높이기 위하여 n이 3이상인 RTP-nR기법이 사용될 수 있다. 재전송 요구 횟수가 많을수록 보다 신뢰성있는 데이터 전송을 할 수 있으나 재전송 비용으로 인해 실시간성이 저해될 수 있으며, 재전송을 기다리는 세그먼트가 프락시 버퍼 내에 유지되어야 함으로 버퍼의 효율을 떨어드린다. 그러므로 이 기법은 실시간성, 신뢰성과 프락시 내의 버퍼 효율을 고려하여 최적의 재전송 횟수를 결정해야 한다.



(그림 6) RTP-1R(n = 1)

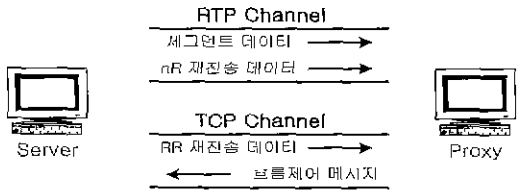


(그림 7) RTP-2R(n = 2)

4.1.2 RTP-RR 전송 기법

NOD 서비스 시스템의 성능을 향상시키기 위하여 인기도가 높은 스트림 데이터를 프락시 서버로 푸싱이나 캐싱하는 기법을 사용한다. 이때 푸싱이나 캐싱되는 데이터는 TCP/IP 수준의 신뢰성을 가진 전송 프로토콜을 요구한다 또한, 서버에서 연속미디어를 여러 프락시 서버로 동시에 전송할 경우 네트워 대역폭 및 서버의 부하를 줄이기 위해서 멀티캐스트 전송기법을 사용하는 것이 효과적이다. 그러나 멀티캐스터 전송 방식은 각 프락시 서버마다 달리 발생하는 패킷 손실을 효과적으로 복구할 수 있는 방법을 제시하지 못한다. 본 논문에서는 서버와 프락시간의 데이터 전송의 주 채널로 멀티캐스터 채널을 이용하며, 손실된 패킷에 대하여서는 제어인 TCP/IP 채널을 통해 재전송 받아 복구하는 RTP-RR 기법을 사용한다. 이 방법은 손실된 데이터의 복구를 위해 기본적으로 TCP를 사용하므로 한번의 재전송요구로 오류가 발생한 데이터의 복구를 보장받을 수 있다.

RTP-RR과 RTP-nR은 (그림 8)과 같이 프락시의 세그먼트 데이터 요구는 RTP 채널을 통해 데이터를 전송한다. 그리고 신뢰성있는 복구를 요청하는 RR 재전송 요구는 TCP 채널을 통해 데이터를 전송하며, 실시간으로 복구를 요청하는 nR 재전송 요구는 RTP 채널을 통해 데이터를 전송한다.



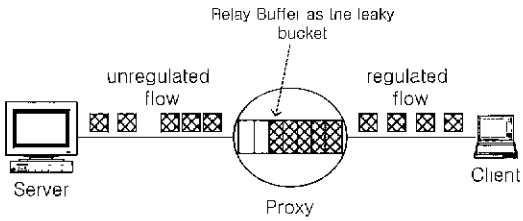
(그림 8) RTP-RR과 RTP-nR의 채널 활용도

4.2 Relay 서비스

프락시에서 캐쉬되지 않는 데이터를 서비스하고자 할 때는 서버에서 프락시 서버를 통해 실시간으로 클라이언트에게 전송해야 한다. 이러한 상황에는 신뢰성보다는 실시간성에 초점을 맞추는 RTP-nR 기법을 적용할 수 있다. 이 기법은 프락시 서버가 실시간으로 인터넷 환경에 위치하는 서버에서 클라이언트에게 연속미디어를 전송함에 있어 전송품질의 저하를 막기 위한 흐름제어와 에러복구를 수행할 수 있는 기능을 제공한다.

● Proxy를 이용한 흐름제어

인터넷상에서의 불규칙한 데이터 전송으로 인한 품질저하를 막기 위해 프락시에 일정량의 relay buffer를 할당하고 이를 네트웍 leaky bucket로써 활용하여 기본적인 흐름제어를 수행한다. 즉, 서버와 인터넷 캐쉬 사이는 relay buffer수준을 보고 가변적으로 전송률을 조정하고 프락시와 클라이언트 사이는 일정한 전송률로 실시간 전송을 하게된다. (그림 9)는 프락시 서버의 relay buffer를 이용한 흐름제어를 보여준다.



(그림 9) Proxy의 Leaky bucket 기능

● RTP-*n*R를 이용한 에러복구

Relay buffer에서 각 세그먼트들은 자신의 전송순서가 될 때까지 큐안에서 일정시간을 대기하게 되는데 이 시간 내에 손실된 패킷을 RTP-*n*R을 이용하여 재전송 받을 경우 보다 손실이 적은 데이터를 클라이언트에게 서비스를 할 수 있다. 이 때 서버는 주기적인 전송 스케줄링을 하면서 대역폭이 남을 경우만 재전송 요구를 허락함으로써 실시간성을 해치지 않고 신뢰성을 추가할 수 있다.

5. 모의 실험

5.1 프락시 푸시-캐싱의 성능 평가

본 논문에서 제안하는 푸시-캐쉬 기법은 연속미디어 스트림의 일부분 또는 전체를 프락시 캐쉬에 캐쉬함으로써 클라이언트에 대한 평균 서비스 초기지연시간을 감소시키고 NOD 서버와 네트워크의 부하를 감소시키는 방법을 제시하였다. 본 절에서는 앞의 3절에서 제시한 NOD 연속미디어 데이터에 대한 푸시-캐쉬 기법의 성능을 모의 실험을 통해 평가한다.

5.1.1 실험 환경

텍스트나 이미지와 같은 전통적인 데이터에 대한 캐쉬 기법의 성능을 평가하기 위하여 HR(Hit Ratio)를 많이 사용하고 있으나 연속미디어 데이터에 대한 캐쉬 기버의 성능을 평기하기 위해서는 적합하지 않다. 그러므로 본 실험에서는 연속미디어의 캐쉬기법의 성능을 더 정확히 측정할 수 있는 BHR(Byte Hit Ratio)을 사용한다 또한, 클라이언트 요구에 대한 서비스의 평균 초기지연시간과 재배치 횟수를 측정하여 다른 재배치 알고리즘들과 비교한다

〈표 1〉 푸시-캐쉬 기법 실험환경

데이터 크기	12MB~60MB
요구 분포	θ=0.27의 Zipf 분포
평균 요구 도착 시간 간격	1초
초기 기사 데이터 개수	200개
기사 생성	1개/분
실험측정시간	8000개의 사용자요구를 처리하는 시간
프락시서버와 클라이언트 사이의 전송 delay	100ms
서버와 프락시 서버 사이의 전송 delay	200ms
segment 크기	7MB

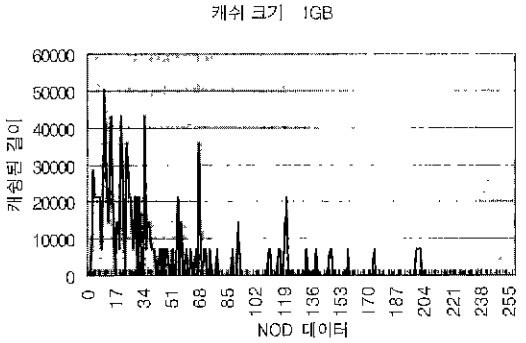
실험환경은 <표 1>과 같이 연속미디어 데이터 크기는 NOD 데이터라는 점을 감안하여 12M~60M(1분~5분의 MPEG1) 범위에서 균일하게 분포한다고 가정하였다. 사용자 요구는 지수분포에 따라 평균 1초마다 한 번씩 발생하며, 인기에 대한 요구 분포는 Zipf 분포를 따른다. 실험 초기에는 서버에 200개의 기사가 존재하고, 1분에 하나씩의 기사가 생성된다고 가정하고, 실험은 총 8000개의 사용자 요구를 처리하는 시간동안 진행하였다.

5.1.2 실험 결과

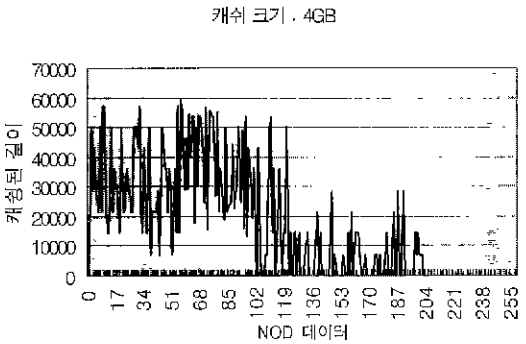
● 생성 시기에 따른 프락시 서버의 데이터 캐쉬 량의 변화

제안된 푸시-캐쉬기법은 NOD 연속미디어 데이터를 대상으로 수행한다. 그러므로 데이터는 생성 시기에 따라 다른 인기도를 가진다. 즉, NOD 시스템에서 사용자는 최근에 생성된 기사를 선호하며, 생성된 지 오래된 기사는 거의 요구하지 않는다 (그림 10)과 (그림 11)은 실험을 통하여 그 결과를 나타낸 것으로, 실험 시작 1시간 후에 NOD 연속미디어 데이터들의 캐쉬 된 길이를 프락시 서버에서 측정한 것이다. NOD 데이터 번호는 각 기사가 생성된 후 경과한 시간을 나타내며, 0번이 가장 최근에 생성된 데이터이고 번호가 클수록 생성된 지 오래된 데이터이다. 이 그림들은 2.2절에서 설명한 NOD

데이터의 생명 주기에 따른 접근 확률 분포와 유사하다.



(a) 캐쉬 Size : 1GB



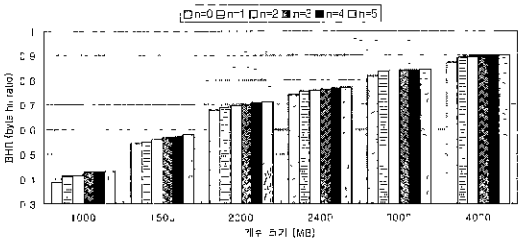
(b) 캐쉬 Size : 4GB

(그림 10) 생성 시기에 따른 인기도 변화

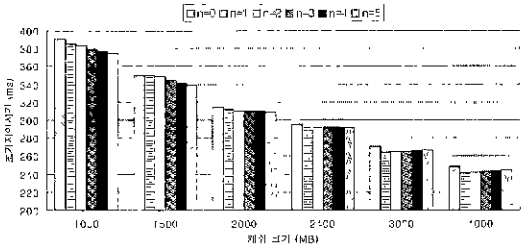
● 푸시(Push)의 성능 분석

NOD 서버에서 연속미디어 데이터가 생성되면 데이터의 앞부분 n개의 세그먼트를 밀티캐스트로 프락시 서버에게 푸쉬한다 새로운 데이터에 대한 사용자 요구가 발생하기 전에 프락시 서버로 멀티캐스트로 전송함으로써 서버의 부하를 줄이고 캐쉬의 성능을 향상시킨다.

(그림 11)은 캐쉬 크기를 증가시키면서 푸시되는 세그먼트의 수에 따른 성능을 비교한 것이다 (그림 11)의 (a)에서 나타나는 것과 같이 BHR은 푸시되는 세그먼트가 많을 수록 좋은 성능을 보이며, 특히 푸시를 하지 않았을 경우에 비해 4%~12%의 성능 향상을 보인다. 그리고 (그림 11)의 (b)는 초기지연시간을 측정한 것으로 캐쉬 크기가 작을 경우 푸시되는 세그먼트 수가 많을수록 초기지연시간이 짧으며, 캐쉬 크기가 클 경우 푸시되는 세그먼트 수가 작을수록 초기지연시간이 짧다. 그러나 전반적으로는 캐쉬 크기가 증가할수록 초기지연시간은 줄어들며, 푸시를 하지 않았을 경우에 비해 평균 5ms~50ms 초기지연 시간이 감소한다.



(a) BHR(Byte Hit Rate)



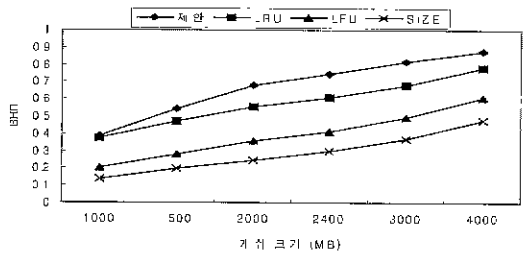
(b) 초기지연시간

(그림 11) 푸시 데이터 크기에 따른 성능 평가

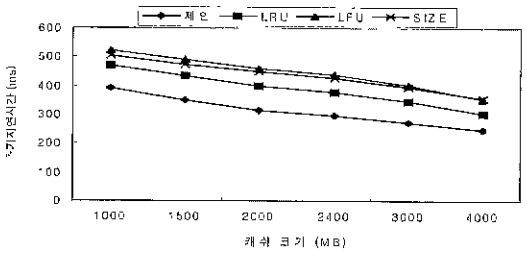
● 제베치 알고리즘에 따른 성능 평가

(그림 12)는 기존의 웹 캐쉬에서 많이 사용되는 알고리즘과의 성능을 비교한 것이다. 즉, LRU(least recently used), LFU(least frequently used), SIZE를 비교 대상으로 하여 성능을 평가하였다. 먼저, 캐쉬 크기에 따른 BHR의 변화를 측정한 결과는 (그림 12)의 (a)에서 나타나는 것과 같이 제안하는 캐쉬기법을 사용했을 경우 LRU 알고리즘에 비해 약 3%~15%, SIZE 알고리즘에 비해 30%~50%의 성능이 향상됨을 보여준다 (그림 12)의 (b)는 프락시와 클라이언트 사이의 네트워크 지연시간을 100ms, 프락시와 서버 사이의 네트워크 지연시간을 200ms로 가정하고 클라이언트의 평균 서비스 초기지연시간을 측정한 것이다. 제안하는 푸시-캐쉬기법을 사용했을 경우에 서비스 초기지연시간은 LFU 알고리즘에 비해 평균 25%, LRU 알고리즘에 비해 평균 20%가 감소한다. (그림 12)의 (c)는 재배치 횟수를 비교한 것이다 제안하는 푸시-캐쉬 기법을 사용할 경우, 캐쉬의 크기가 작을 경우 가장 좋은 성능을 보인다 그러나 캐쉬의 크기가 증가할수록 일정 크기까지는 다른 알고리즘과 달리 재배치 횟수가 증가한다 이 이유는 3.2절의 식 (3)에서 보듯이 각 스트림에 할당되는 캐쉬 세그먼트 수(C_i)가 작으면 새로운

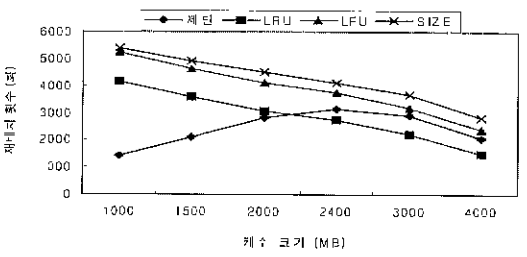
세그먼트 추가에 따른 캐쉬효율(*CUi*) 값이 크게 감소하기 때문에 재배치가 발생하기 위해서는 캐쉬 이익(*Pi*)가 크게 증가하여야 한다. 그러나 캐쉬의 크기가 증가한다면 각 스트림에 할당되는 캐쉬 세그먼트 수가 증가하여 캐쉬 이익(*Pi*)가 크게 증가하지 않더라도 재배치가 발생한다. 또한 일정 크기 이상으로 캐쉬 크기가 증가한다면 충분한 캐쉬이 이루어지므로 재배치 횟수는 다시 감소한다. 이와 같이 제안된 푸쉬-캐쉬 기법은 재배치 횟수에 있어 위와 같은 Trade-off가 발생하나 전체적으로 LFU나 SIZE 알고리즘보다는 좋은 성능을 보이고 캐쉬 크기가 일정 크기 이상에서 증가할 경우, LRU 알고리즘보다 많은 재배치를 수행하나 비슷한 비율로 감소함을 볼 수 있다



(a) BHR(Byte Hite Rate)



(b) 초기지연시간



(c) 재배치 횟수

(그림 12) 재배치 알고리즘에 따른 성능 평가

5.2 연속미디어 전송기법의 성능 평가

본 절에서는 릴레이 서비스를 위해 제안된 RTP-nR 전송 기법의 성능을 파악하기 위해서 실제 인터넷환경에서 프락시에서의 패킷 손실량과 마감시간 실패율을 측정하였다.

5.2.1 실험 환경

서버와 프락시는 UNIX상에서 gnu-C를 이용하여 코딩하였고 클라이언트는 Window98위에서 Visual C++와 Direct Show 6.0 SDK를 이용하여 프로그램 하였으며, 실험 데이터는 <표 2>에서와 같이 MPEG-1으로 압축된 뉴스기사 샘플 데이터를 사용하였다. 비교할 기법으로 RTP, 재전송을 한번만 수행하는 RTP-1R, 재전송을 두 번까지 허락하는 RTP-2R, 그리고 TCP를 선택하였으며 릴레이 전송 기법 자체의 특성만을 측정하였다

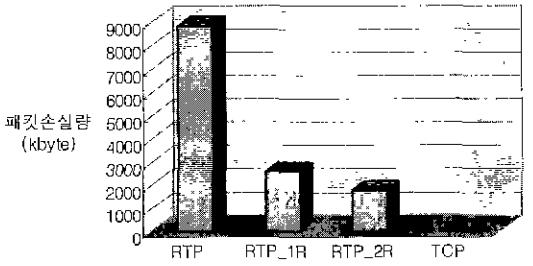
〈표 2〉 실험 데이터의 내용

인자 이름	내 용
데이터의 종류	MPEG-1 System
데이터의 크기	45.691Mbyte
총 상영 시간	4분 28초
평균 전송률	185 ms
총 세그먼트의 개수	1428
세그먼트 크기	32 Kbyte
패킷 크기	1 Kbye

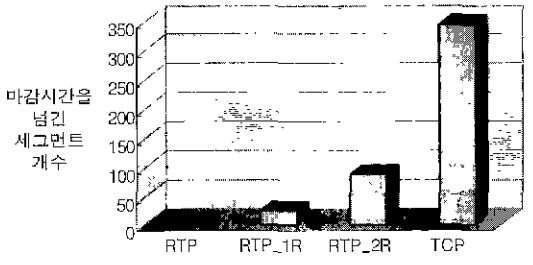
5.2.2 성능평가

실험을 위해 부산경상대학에 서버를 두고 부산대학교에 프락시를 설치하고 클라이언트에서 각 전송기법의 성능을 측정하였다. 재전송을 하지 않는 RTP의 경우 많은 패킷 손실이 있는 반면 마감시간 실패의 세그먼트는 없었으며, TCP를 사용하여 전송하는 경우는 반대로 패킷 손실량은 없었으나 마감시간을 초과하는 세그먼트가 많아진다 그리고 RTP-1R, RTP-2R기법은 재전송 요구에 따라 패킷 손실량은 줄어들지만 재전송 요구를 처리하기 위해 relay buffer에서의 대기시간 증가와 재전송 overhead로 인해 마감시간 실패 횟수가 증가한다 (그림 13)의 (a)에 나타나듯이 인터넷상에서 프락시 서버가 RTP-*n*R을 사용함으로써 RTP에 비해 약 3배의 패킷손실량이 감소하며, (그림 13)의 (b)에 나타나는 RTP-1R과 RTP-2R의 마감시간을 넘기 세그먼트의 수는 재전송 횟수에 따라 증가한다. 그

리므로 네트워크 환경과 프락시 relay buffer의 비용에 따라 패킷손실량과 마감시간실패 횟수를 고려하여 적절한 RTP-nR의 재전송횟수는 결정하여야 한다.



(a) 패킷손실량



(b) 마감시간을 넘긴 세그먼트의 갯수

(그림 13) 인터넷상에서의 성능 측정

6. 결론 및 향후 연구과제

본 논문에서는 연속미디어 데이터의 일부분 또는 전체를 푸시 또는 캐싱하는 프락시 푸시-캐싱기법과 프락시 푸시-캐싱기법에 적용하기 위한 연속미디어 전송기법인 RTP-RR과 RTP-*n*R을 제안하였다. 프락시 푸시-캐싱기법은 연속미디어 데이터 생성되었을 때 데이터의 일정부분을 프락시로 푸시시키고, 데이터의 캐싱 효율값에 따라 캐싱되는 길이를 유동적으로 변화시킴으로써 클라이언트의 서비스 초기지연시간을 감소시키는 동시에 캐쉬에서 서비스되는 데이터량을 증가시킨다. 또한, 연속미디어 스트림이 캐쉬에서 차지하는 디스크 공간과 스트림에 대한 클라이언트의 요구량 사이의 상관관계를 고려하여 캐싱효율(caching utility)을 측정하고, 그 값을 이용하여 세그먼트를 재배치 기법을 제안하였으며, 프락시 서버의 relay buffer의 대기시간을 이용한 연속미디어의 실시간성 보장하면서 패킷손실을 줄이는 방법을 제시하였다. 그리고 모의 실험을 통하여 제안하는 기법과 다른 캐싱 기법들과의 성능을 비교 측정하고, BHR과 서비스 초기지연시간, 재배치 횟수, 패킷손실률 측면에서 성능이 향상됨을 보였다.

향후 연구로써는 RTP-nR과 RTP-RR기법을 이용하여 멀티캐스트 하였을 경우, 시스템 전체 성능에 대한 연구가 수행되어야하며, 재전송 패킷에 관한 오버헤드를 줄이는 기법과 프락시 서버의 relay buffer의 비용을 줄이는 연구를 수행하여야 한다.

참 고 문 헌

[1] 김영주, 최태욱, 정기동, "주문형 전자신문 시스템에서 사용자 접근 패턴을 이용한 기사 프리페칭 기법", 한국정보처리학회 정보처리논문집, 제6권 제5호, 1999.

[2] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, K. J. Worrell, "A Hierarchical Internet Object Cache." In Proc. of 1996 Usenix Technical Conference, January 1996.

[3] A Luotonen, K Alus, "World Wide Web Proxies," In Proceedings of the First International Conference on the WWW, May 1994.

[4] J. T. Robinson, N. V. Devarakonda, "Data Cache Management Using Frequency-based Replacement," in Proc of ACM SIGMETRICS Conference, 1990.

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

[6] A Dan, D. Dias, R Mukherjee, D Sitaram, R. Tewari, "Buffering and Caching in Large-Scale Video Servers," In Proc. of IEEE COMPCON, March 1995

[7] B. Ozden, R. Rastogi, A. Silberschatz, "Buffer replacement algorithms for multimedia storage systems." In Proc. of the International Conference on Multimedia Computing and Systems, June 1996.

[8] J. Gwertzman, M Seltzer, "The Case for Geographical Push-Caching," In Proc. of the 1995 Workshop on Hot Operating Systems, 1995.

[9] E. P. Markatos and C E. Chronaki, "A TOP-10 approach to prefetching on Web", Proc. of INET'98.

[10] H Schulzrinne, S. Casner, R. Frederick, V Jacobson, "RTP : A Transport Protocol for Real-time Applications." RFC-1889, 1996

[11] W. Feng and S. Sechrest, "Smoothing and buffering for delivery of prerecorded compressed video," in IS&T/SPIE Symp. on Multimedia Comp. and Networking, pp.234-242, February, 1995

[12] W. Feng and S. Sechrest, "Critical bandwidth allocation for delivery of compressed video," Comp. Comm., pp.709-717, October, 1995.

[13] Reza Rejaie, Haobo Yu, Mark Handely, Deborah Estrin. "Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet," To Appear in Proceedings of IEEE Infocom'2000 , Tel-Aviv, Israel, March 2000.

[14] Dan. A., P. Shahabuddin, D Sitaram and D. Towsley, "Channel allocation under batching and VCR control in video-on-demand systems." Journal of Parallel and Distributed Computing, 30(2) : 168-179, Nov. 1994.

[15] L. Gao, D. Towsley. "Supplying Instantaneous Video-on-Demand Services Using Controlled Multicast," Proceedings of IEEE Multimedia Computing and Systems, Florence, Italy, June, 1999.

[16] S. Sen, J Rexford and D. Towsley, "Proxy prefix caching for multimedia streams," In Proc. IEEE Infocom, March 1999

[17] S. Sahu, P Shenoy and D. Towsley, "Design considerations for Integrated Proxy Servers." In Proc. IEEE NOSSDAV'99, June 1998.

[18] Y Wang, Z.-L. Zhang, D Du, and D Su, "A Network- Conscious Approach to End-to-End video Delivery over Wide Area Networks Using Proxy Servers," In proc IEEE Infocom, April 1998.

박 성 호

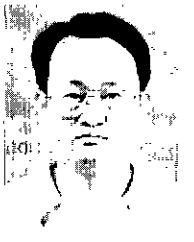
e-mail : shpark@melon.cs pusan.ac.kr
1996년 부산대학교 전자계산학과 졸업(학사)
1998년 부산대학교 전자계산학과 졸업(석사)
2000년 부산대학교 전자계산학과 박사 수료
관심분야 : 멀티미디어 통신, 인터넷 캐싱, IP Telephony

임 은 지

e-mail melon.cs.pusan.ac.kr
1999년 부산대학교 전자계산학과 졸업(이학사)
1999년~현재 부산대학교 전자계산학과 재학중(석사과정)
관심분야 : 컴퓨터 네트워크, 캐쉬 서버, 멀티미디어

최 태 욱

e-mail tuchoi@melon.cs pusan.ac.kr
1997년 동의대학교 전산통계학과 졸업(이학사)
1999년 부산대학교 대학원 전자계산학과 졸업(이학석사)
2000년~현재 부산대학교 대학원 전자계산학과 재학중 (박사과정)
관심분야 : 멀티미디어, 병렬처리, 분산처리



정 기 동

e-mail : kdchung@melon.cs pusan ac.kr
1973년 서울대학교 졸업(학사)
1975년 서울대학교 대학원 졸업 (석사)
1986년 서울대학교 대학원 계산통계학과 졸업(이학박사)
1990년~1991년 MIT, South Carolina 대학 교환 교수
1995년~1997년 부산대학교 전자계산소 소장
1978년~현재 부산대학교 전자계산학과 교수
관심분야 : 병렬처리, 멀티미디어