

# 이동환경에서 연속미디어 서비스를 위한 협력적인 프록시 캐싱의 성능평가

이 화 세<sup>†</sup>

## 요 약

본 연구는 이동환경에서 연속 미디어 객체에 대한 사용자 요구들에 대하여 효율적으로 자원을 관리하기 위하여, 사용자의 이동성 정보를 반영하는 협력적 프록시 캐싱 정책을 제안한다. 제안된 정책은 담당영역 내에서 사용자의 요구들을 참조하여 캐싱하는 기존의 프록시 캐싱 정책과는 다르게 인접한 영역의 많은 사용자 요구 정보들을 이용하여 캐싱 객체를 선택함으로써 이동환경에서 발생하는 사용자들의 임의접근 요구들을 충분히 반영할 수 있도록 한다. 그리고 제안된 캐싱 정책을 위해서 각 프록시에서 발생한 요구 정보의 가중치와 재생량을 기반으로 하는 재배치 기법을 제안한다. 그래서 객체단위와 세그먼트 단위로 재배치 기법을 수행하여 캐싱 정책의 성능을 평가하였다. 그 결과, 이동환경에서 적절한 세그먼트의 크기, 사용자의 이동성 비율 그리고 가중치가 협력적 캐싱의 성능에 영향을 미치는 중요한 요인이 된다는 것을 알았다.

## A Performance Evaluation of Cooperative Proxy Caching for Continuous Media Services in Mobile Environments

Haw-Sei Lee<sup>†</sup>

## ABSTRACT

This paper proposes a user's mobility based cooperative proxy caching policy for effective resource management of continuous media objects in mobile environments. This policy is different from the existing proxy caching policies in terms of how to exploit users' mobility. In other words, existing caching policies work based on the information about objects by referring to users' requests within a specified domain whereas the proposed caching policy runs by utilizing a number of users' requests across several domains. So, the proposed policy is applicable to random requests in mobile environments. Moreover, we also propose a replacement policy based on weights and playback time. To check the efficiency of the proposed caching policy, the proposed replacement policy is run with different size of caching unit: object or segment. The result of performance evaluation tells that segment sizes, ratio of users' mobility and weights are major factors for the efficient operation of the cooperative caching.

**Key words:** Cooperative Proxy Caching(협력적 프록시 캐싱), Replacement(재배치), Weight(가중치)

※ 교신저자(Corresponding Author): 이화세, 주소: 경남 밀양시 삼랑진읍 청학리 50번지(627-702), 전화: 055)350-5442, E-mail: hslee@mnu.ac.kr

접수일: 2004년 12월 8일, 완료일: 2005년 2월 21일

<sup>†</sup> 정회원, 밀양대학교 컴퓨터공학부 교수

※ 본 연구는 밀양대학교 교내학술연구비의 지원으로 수행되었음.

## 1. 서 론

최근 인터넷에서 스트리밍 서비스는 데이터의 속성상 많은 양을 처리하고 전송하기 위하여 시스템의 높은 계산 능력과 네트워크의 고속 전송 능력을 요구하게 된다. 그러나 멀티미디어 데이터를 위한 스트리

밍 서비스의 지원은 코어 네트워크의 교통량에 심각한 문제를 발생시킨다. 그래서 네트워크상의 교통량과 서버의 오버헤드를 줄여 전체 시스템의 성능을 향상시키고 네트워크의 자원을 효율적으로 관리하는 방법으로서 프록시 서버를 사용한다.

이동환경을 지원하는 프록시는 원격의 중앙서버 자료를 저장하는 기능, 이동 노드의 이동성에 관한 정보, 이동 노드의 요구를 서비스하기 위해 제공된 네트워크의 대역폭, 이동 노드의 사용가능한 자원 등을 고려하여야 한다. 이에 대한 연구로, 프록시가 서버의 데이터를 변환하여 전달해주는 프록시 트랜스코딩 기법에 대한 연구들과 이동 노드의 요구에 대한 지연시간을 줄이기 위한 버퍼링, 선반입(Prefetching) 등에 관한 연구들이 주를 이루고 있다[1-4]. 그리고 [5-7]는 이동환경에서 단말 노드나 하나의 프록시에서 재배치 정책에 대한 연구들로 단말 노드가 여러 셀로 이동하는 경우에 발생하는 문제는 고려하고 있지 않으며, 특히 연속미디어 서비스를 위해서 사용자의 이동성 정보를 반영한 캐싱 정책에 대한 연구들은 아니다. 이동환경에서의 프록시 캐싱은 유선망에서 캐싱 기법과 마찬가지로 객체(Object)의 서로 다른 접근 빈도인 인기도와 같은 미디어 데이터의 특성을 고려하여야 하며 사용자들의 이동에 따른 이동성 정보를 적응성 있게 적용할 수 있는 캐싱 구조와 캐싱 정책의 연구가 필요하다.

본 연구에서는 이동환경에서 사용자의 이동에 따른 객체의 요구 정보를 프록시들 간에 공유하여 캐싱에 반영하는 협력적 프록시 캐싱(Cooperative Proxy Caching) 정책을 제안한다. 제안된 정책은 하나의 프록시의 위치를 기준으로 인접한 주위 프록시들에서 발생한 사용자들의 요구 정보를 서로 공유하여 캐싱 정책에 반영하는 구조이다. 그래서 프록시는 현재 사용자들의 서비스 요구에 대한 정보뿐만 아니라 향후 이동해 올 사용자들의 서비스 요구에 대한 정보도 반영한다. 그리고 객체단위와 세그먼트(Segment) 단위의 재배치(Replacement) 기법으로 캐싱 정책을 수행하여 사용자의 이동성 비율에 따른 성능과 세그먼트 크기와 가중치 적용에 따른 성능을 평가하였다.

본 연구의 구성은 다음과 같다. 2장에서는 관련연구를 살펴보고, 3장에서는 이동환경에서 연속미디어 서비스를 위한 협력적 프록시 캐싱 구조와 재배치 기법에 대해서 설명한다. 그리고 4장에서는 제안된 기법의 성능을 평가하고 5장에서는 결론 및 향후과제를 기술한다.

## 2. 관련연구

### 2.1 이동환경에서 연속미디어 서비스

무선 접속 방식의 패킷 데이터 서비스를 기반으로 하는 이동환경은 케이블이나 초고속 인터넷 등과 같은 유선망 환경과는 다르게 연속미디어 서비스를 위해서는 고려해야 할 많은 문제점들이 존재한다. 연속미디어 서비스는 특성상 많은 양의 멀티미디어 데이터를 처리하기 위해서 시스템의 처리능력이 뛰어나야 하고 또한 처리된 미디어 데이터를 전송하기 위해서는 네트워크 상의 전송 대역폭이 커야 한다. 그래서 이동환경에서 미디어 서비스는 유선망에서와 마찬가지로 미디어 데이터의 특성과 사용자의 이동성에 따른 지터나 끊김없는 서비스, 빠른 응답시간 등을 고려하여야 한다.

### 2.2 이동환경에서 네트워크 캐싱

이동환경에서는 사용자가 이동하는 동안에도 데이터 전송과 필요한 자원에 대한 접근이 가능해야 한다. 그리고 네트워크 전체의 오버헤드를 줄이고 성능을 향상시키면서 이러한 기능을 지원하기 위해서는 이동 노드에 필요한 정보 또는 객체들을 캐싱하는 것이 필요하다. 그래서 게이트웨이나 베이스 스테이션에 무선 인터넷 멀티미디어 프록시를 두어 네트워크 자원 비용의 감소, 미디어의 품질을 향상, 단말 노드의 초기 지연시간 감소와 같은 이점을 가질 수 있다[8]. 그러나 이동환경에서 사용자의 이동성 정보를 반영하는 네트워크 캐싱은 유선망 환경에서와는 다르게 사용자의 작업 패턴이나 공간적 시간적 이동성, 캐쉬 관리 기법, 적응성 기법, 이동성관리 기법, 효율적인 경로 설정, 낮은 전송 속도 등에 대한 고려가 필요하다. 그래서 주로 인터넷상의 웹 서버 또는 HTTP 프록시 서버와 이동 노드(클라이언트)사이의 MSS에 주로 프록시 서버를 두어 이동 노드에 적합한 형태로 객체를 전달한다. 이러한 연구들은 웹 서버나 기존 프록시 캐쉬의 부하를 완화시키는 목적으로 하는 트랜스코딩 프록시에 관한 것들이 많으며, 주로 기존의 유선망 환경에서 구현되어 있다[1].

### 2.3 이동환경에서 캐쉬 재배치 기법

다른 관점의 연구로[5]는 사용자의 로밍(Roam-

ming)을 예측하여 가장 이동가능성이 높은 셀을 선택하여 현재의 캐쉬 내용을 재배치(Relocation)하는 방법을 제공하고 있다. [6]은 단말 노드의 이동성이 발생할 때, 분산으로 연결되어 서로 협력하는 캐쉬들 간에 연결된 동적인 위상을 면밀히 추적하여 객체가 위치한 가까운 연결부분의 캐쉬와 단말 노드를 연결시킨다. 그래서 QoS를 향상시키고 오버헤드를 줄여, 서버와 네트워크 자원의 소모를 줄일 수 있는 방법을 제안하고 있다. [7]은 데이터 검색 지연, 데이터 크기, 액세스 확률, 갱신 빈도를 고려한 SAIU(Stretch: Access-rate Inverse Update-frequency)라는 이득(Gain) 함수를 사용한 캐쉬 재배치 정책을 제안 하였다. [8]은 무선망 환경에서 멀티미디어 프록시를 베이스 스테이션이나 게이트웨이에 두고 비디오 스트리밍을 지원하기 위한 Cost-Based 재배치 기법을 제안하였다. 이 기법은 네트워크 비용, 지연(Latency) 비용, 그리고 미디어 왜곡(Distort)비용을 결합한 비용 측도(Cost Metric)를 재배치 측도로 사용하였다. 그러나 이러한 연구들은 하나의 셀 환경에서 성능을 평가한 것으로 단말 노드가 여러 셀로 이동하는 경우에 발생하는 사용자 이동성에 관한 문제는 고려하고 있지 않다.

### 3. 이동환경에서 협력적 프록시 캐싱

본 연구는 이동환경에서 프록시가 자신의 담당 영역에서 발생하는 객체 요구에 관한 정보뿐만 아니라 이웃하는 영역에서 발생하는 잠재적인 객체 요구에 관한 정보도 고려하여 캐싱하는 협력적 프록시 캐싱 구조를 성능 측정을 위한 모델로 제안한다. 이 구조에서 프록시 간에 서로 협력하는 클러스터 프록시와 클러스터 그룹 내의 프록시들 간의 가중치를 정의한다. 그리고 각 프록시들에서 발생하는 사용자 요구들의 가중치를 부여하고, 이 가중치와 객체의 데이터 재생량(Amount of Playback)을 이용하여 **가중치를 적용한 객체 데이터의 재생량**을 계산한다. 그래서 계산된 값을 협력적 프록시 캐싱에서 객체의 재배치를 위한 척도로 사용하고 이를 기준으로 하는 재배치 객체의 선택 기법을 제안한다.

### 3.1 협력적 프록시 캐싱의 구조

그림 1은 본 연구에서 고려하는 네트워크의 전체 구조로 기존 유선망의 코어 네트워크와 무선망을 지원하는 지역 네트워크로 구성되며, 각 프록시는 한 영역 내에서 이동 호스트와 고정 호스트가 혼재되어 있는 지역 네트워크를 담당한다.

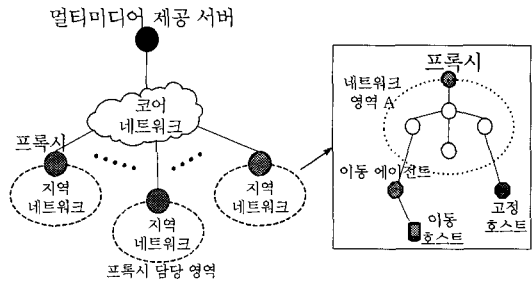


그림 1. 전체 네트워크 구조

협력적 프록시 캐싱 구조에서 협력관계에 있는 각 프록시들의 논리적 연결은 그림 2와 같은  $n * n$  구조로 간주하며, 이들 간의 관계는 정의 1과 같다.

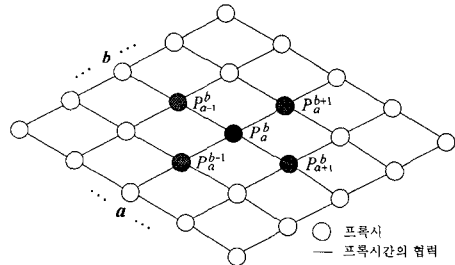


그림 2. 협력적 프록시 캐싱의 구조

#### 정의 1 협력적 프록시 캐쉬 구조

$P_a^b$ 는 임의의 한 프록시이고  $C(P_a^b)$ 는  $P_a^b$ 의 인접한 프록시라 할 때, 그림 2와 같이 프록시들 간에 서로 협력하는 전체 연결 구조는 다음과 같다.

$$1) \exists P_a^b, \forall a, b \in \mathbb{Z}, 1 \leq a, b \leq n \quad (1)$$

$$2) P_a^b \text{를 중심으로 바로 인접한 프록시는}$$

$$P_{a-1}^b, P_{a+1}^b, P_a^{b-1}, P_a^{b+1} \text{로}$$

$$C(P_a^b) = \{ P_a^{b-p} \mid |p| + |q| = 1, \forall p, q \in \mathbb{Z}, -1 \leq p, q \leq 1 \} \quad (2)$$

가 되는 동적인 구조이다.

1) 본 연구에서는 '가중치\_적용\_재생량'이라고 간단하게 표현한다.

3.2 클러스터 프록시

논리적으로 연결된 프록시 구조에서 한 프록시에서 공간적으로 인접한 위치에 있는 프록시들을 같은 클러스터 프록시에 속한다고 보며 각 프록시마다 동적으로 구성할 수 있다.

정의 2 프록시  $P_a^b$ 의 클러스터

$CP(P_a^b)$ 는  $P_a^b$ 의 클러스터 프록시로 자신을 포함하여 바로 인접한 프록시들이라 정의하고

$$CP(P_a^b) = \{P_a^b\} \cup \{P_{a-1}^b, P_{a+1}^b, P_a^{b-1}, P_a^{b+1}\} \quad (3)$$

로 나타낸다.

클러스터 프록시 내의 각 프록시는 자신의 영역 내의 객체 요구에 관한 정보를 클러스터 내의 프록시들에게 전파하여 서로 유지하도록 한다. 그러나 인접한 프록시에서 발생한 사용자 요구는 해당 프록시에 대한 서비스 요청이 아닌 단지 가능성이 있는 잠재적인 서비스 요청이기 때문에 가중치를 차등적으로 적용한다.

정의 3 프록시에서 발생하는 요구들의 가중치 값

1)  $C(P_a^b)$ 에 속하는 프록시들의 개수를  $Num(C(P_a^b))$ 라 하면,  $P_a^b$ 의 가중치는 1로 두고,  $C(P_a^b)$ 에 속하는 각 프록시  $P_{a-p}^{b-q}$ 의 가중치는

$$u(P_{a-p}^{b-q}) = \frac{1}{Num(C(P_a^b))}, \text{ where } \forall p, q \in Z, -1 \leq p, q \leq 1 \quad (4)$$

라 한다.

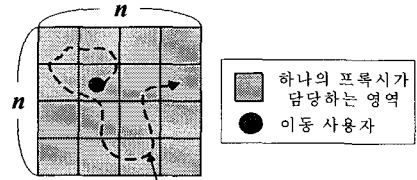
2) 각 프록시에서 발생하는 사용자 요구들의 가중치는 담당 프록시의 가중치를 따른다.

3.3 이동 사용자 모델

본 연구에서는 사용자의 이동 영역과 이동 방향을 정의된 범위 내에서 랜덤하게 선택하고 과거의 사용자 위치와 현재의 위치는 독립적인 환경으로 가정한다. 이때 모델을 단순화시키기 위하여 공간적(Spatial) 이동성 정보<sup>2)</sup>를 반영하는 이 노드의 방향은 이동할 인접 영역을 선택하는 확률 값을 사용하고, 이동 노드의 가변적인 속도는 고려하지 않는다[9]. 그

리고 시간적(Temporal) 이동성 정보<sup>3)</sup>를 반영하는 것으로 이동 노드들의 초기 위치, 한 영역에서 상주 시간(Residence Time), 단위 시간당 사용자들의 서비스 요구 수, 연속미디어들의 특성을 다음과 같이 가정한다.

① 사용자가 이동하는 전체 영역들을 그림 3과 같이  $n \times n$ 형태의 정방형 구조로 간주한다.



사용자의 이동 패턴

그림 3. 사용자 이동 영역들

② 사용자들은 ①과 같은 환경에서 연속미디어를 하나의 프록시 서버에서 서비스 받고, 각 이동 사용자들의 초기 위치는 임의의 한 영역으로 설정된다. 일반적으로 연속미디어 응용에서 단위 시간당 사용자들의 서비스 요구 수는 포아송 분포(Poisson Distribution)를 따르고, 서비스 요구 시간 간격(Inter-Arrival Time)은 포아송 과정을 따른다[8-10]. 그리고 일반적인 연속미디어 응용에서 연속미디어들의 인기도에 따른 접근 확률은 Zipf 분포를 따르고, 연속미디어를 서비스 받고 있는 사용자가 각 영역에서 머무는 시간은 지수 분포(Exponential Distribution)를 따른다. 그래서 사용자가 한 영역에서 상주하는 시간을 추출해 낼 수 있다[10-12].

3.4 협력적 프록시 캐싱

클러스터 프록시 그룹 내에서 각 프록시들은 객체 요구에 관한 정보를 공유하면서 서로 협력하고 있다. 객체 요구에 관한 정보는 사용자의 요구 인덱스, 해당 객체의 인덱스, 객체의 시작 위치, 영역 진입 시간, 영역 진출 시간, 해당영역 가중치의 값을 나타내는 자료 구조이다. 그림 2와 같은 구조에서 임의의 한 프록시에서 객체에 대한 사용자 요구가 발생할 때, 해당 프록시에 존재하지 않으면 원격지 서버로부터

2) 사용자가 이동하는 패턴에 대한 지식정보나 예측이 고려되어 이동할 방향을 추측하는데 사용된다.

3) 사용자가 한 영역(셀)을 방문하여 다른 영역(셀)로 이동할 때까지의 시간으로, 영역(셀)의 상주시간을 구하는데 사용된다.

객체를 서비스 받는다. 이때 서비스되는 객체를 해당 캐쉬에 캐싱하기 위해서 캐쉬의 잔여공간(Remained Cache Space)을 검사하여 재배포 여부를 결정한다. 이와 같은 절차로 이루어지는 협력적 프록시 캐싱은 그림 4와 같다.  $f_{LFU}(O_k, t_n)$ 는  $t_n$ 시간에서의 재배포 척도 값이고  $Size(O_k)$ 는 객체  $O_k$ 의 크기를 나타낸다.

```

1) 요구  $i$ 의 시작에 따른  $P_a^b$ 의 객체 정보 DB의 서비스 정보 수정.
2) 협력 프록시에게 객체  $O_k$ 에 대한 요구  $i$  시작을 알림.
3) while(!요구  $i$ 에 대한 서비스가 종료) {
    if ( $\exists O_k \in \{P_a^b \text{캐쉬}\}$ ) {
         $P_a^b$ 의 캐쉬로부터  $O_k$ 를 서비스
    } else {
        서버로부터  $O_k$ 를 서비스;
    }
    if ( $Size(O_k) < Size(P_a^b \text{의 Remained Cache Space})$ ) {
         $O_k$ 를  $P_a^b$ 의 캐쉬에 캐싱
    } else {
         $P_a^b$ 의 객체 정보 DB로부터 협력 프록시들의  $O_k$ 에 대한
        정보를 검색;
        재배포 척도 값  $f_{LFU}(O_k, t_n)$ 를 계산;
         $P_a^b$ 에서 재배포 척도 값으로 재배포 객체 선택;
         $O_k$ 를  $P_a^b$ 의 캐쉬에 캐싱
    }
}
4) 요구  $i$ 의 종료에 따른  $P_a^b$ 의 객체 정보 DB의 서비스 정보 수정.
5) 협력 프록시에게 객체  $O_k$ 에 대한 요구  $i$ 의 종료를 알림.
    
```

그림 4. 협력적 프록시 캐싱

### 3.5 재배포 객체 선택 기법

각 프록시에서 발생하는 연속미디어 객체에 대한 사용자 요구 객체를 재배포하기 위해서는 클러스터 그룹 내의 프록시들로부터 전달받은 객체 요구 정보 리스트를 검색한다. 그래서 현재 클러스터 그룹 내의 프록시들에서 서비스 중인 객체 스트림의 재생량과 객체의 가중치 값을 적용하여 재배포 객체를 선택한다. 재배포 객체 선택 기법으로서 기존의 캐싱 기법들 중에 연속미디어 서비스에서 객체의 인기도를 잘 반영할 수 있고, 다른 기법에 비해 일반적으로 좋은 성능을 보이며, 비교적 알고리즘이 간단한 LFU 기법을 사용한다.

(1) **가중치\_적용\_재생량을 이용한 재배포 척도값**  
 현재시간  $t_n$ 에서  $\Delta t_n (= |t_n - t_{n-1}|)$  시간 동안 발생한 요구에 대한 가중치 값과 객체의 재생량을 곱

한 값을 **가중치\_적용\_재생량(Weight&Amount)**이라 하고, 이를  $W\_A$ 로 표현한다. 그래서  $\Delta t_n$  동안 발생한  $r$ 개의 요구들의  $W\_A$ 값들을 더한 값을 LFU 알고리즘의 재배포 척도값으로 사용한다.

$$f_{LFU}(O_k, t_n) = Sum\_W\_A_r(S_i(O_k), \Delta t_n) \quad (5)$$

$f_{LFU}(O_k, t_n)$ 는  $\Delta t_n$  시간 동안 연속미디어 객체  $O_k$ 를 참조하는 모든 스트림들의 **가중치\_적용\_재생량의 총합**으로  $t_n$  시간에서 LFU 알고리즘의 재배포 척도를 표시하며,  $S_i(O_k)$ 는 요구  $i$ 에 대한 객체  $O_k$ 를 전송하는 스트림이고,  $Sum\_W\_A_r(S_i(O_k), \Delta t_n)$ 는  $\Delta t_n$  동안  $O_k$ 를 참조 중인  $r$ 개의  $W\_A$ 들의 총합이다. **가중치\_적용\_재생량**을 구하기 위해서 스트림의 가중치를 다음과 같이 정의한다.

#### 정의 4 스트림의 가중치와 협력비

각 프록시에서 발생한 요구  $i$ 의 스트림이  $S_i(O_k)$ 일 때, 정의 3에 의해서  $P_a^b$ 에서 발생한 스트림의 가중치는 1이고,  $P_a^b$ 의 인접한 프록시( $P_{a-p}^b$ )에서 발생한 스트림의 가중치는 인접한 프록시의 가중치에 협력비( $\alpha$ )를 적용한 값으로

$$w(P_{a-p}^b(S_i(O_k))) = \frac{1}{Num(C(P_a^b))} \cdot \alpha \quad (6)$$

와 같이 표현한다. 이때  $\alpha$ 는 한 프록시가 인접한 프록시들과의 협력하는 정도를 나타내는 값으로, 인접한 프록시에서 발생한 요구 스트림의 가중치를 적용성 있게 조절하는 값이다.

수식 (5)에서  $Sum\_W\_A_r(S_i(O_k), \Delta t_n)$ 은 다음과 같은 ①, ② 단계로 구한다.

① 그림 5에서 요구  $i$ 가  $\Delta t_n$  동안  $S_i(O_k)$ 를 재생한 데이터 량을  $Data_i^k$ 라 하고, 수식 (6)에 의해서 요구  $i$ 의  $W\_A(S_i(O_k), \Delta t_n)$ 는 수식 (7)과 같이 표현된다.

$$W\_A(S_i(O_k), \Delta t_n) = Data_i^k \times w(P_{a-p}^b(S_i(O_k))) \quad (7)$$

② 객체의 재생량이 그림 5와 같을 때  $\Delta t_n$  동안 클러스터 프록시에서  $O_k$ 를 참조하는 스트림  $S_i(O_k)$ 에 대한 사용자들의 요구수를  $r$ 이라 가정하면, 수식

(7)에 의해서  $O_k$ 에 대한 전체 가중치\_적용\_재생량으로 수식 (8)과 같이 표현된다.

$$\begin{aligned}
 & Sum\_W\_A_r(S_i(O_k), \Delta t_n) \\
 &= W\_A(S_1(O_k), \Delta t_n) + \dots + (S_r(O_k), \Delta t_n) \\
 &= \sum_i W\_A(S_i(O_k), \Delta t_n), \\
 & \text{where } r = |i|, i = \text{사용자 요구 인덱스}
 \end{aligned} \tag{8}$$

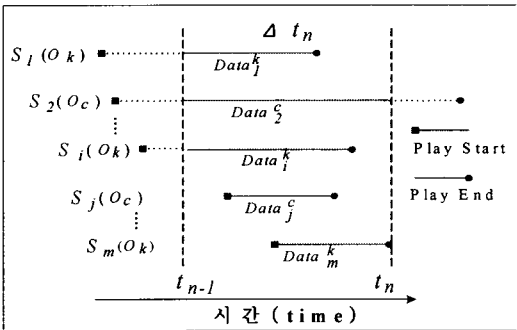


그림 5.  $\Delta t_n$ 에서 객체의 재생량

여기서 최근에 데이터 재생량이 크고 가중치가 높은 프록시에서 요구된 객체일수록  $Sum\_W\_A_r(S_i(O_k), \Delta t_n)$  값이 크게 나타나고, 최근에 재생량이 작고 가중치가 낮은 프록시에서 요구된 객체는 작게 나타나서 접근의 최근성을 반영한다.

**예) 재배포 척도 값( $f_{LFU}$ )의 계산**

그림 6과 같은 클러스터 프록시 구조에서 P1이 담당하는 객체의 스트림 전송은  $S_1$ (P1에서 요구),  $S_2$ (P2에서 요구),  $S_3$ (P3에서 요구),  $S_4$ (P4에서 요구),  $S_5$ (P5에서 요구),  $S_6$ (P1에서 요구),  $S_7$ (P4에서 요구) 이고, 객체의 스트림 전송이 그림 7과 같다. 그리고

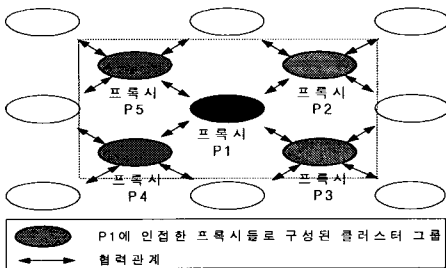


그림 6. P1에 인접한 프록시 그룹

시간 윈도우  $\Delta$ 시간 동안 객체의 재생량을 10으로 가정하고, 시간 윈도우 중간에서 시작하여 재생하거나 끝날 때에는 별도로 재생량을 표시한다.

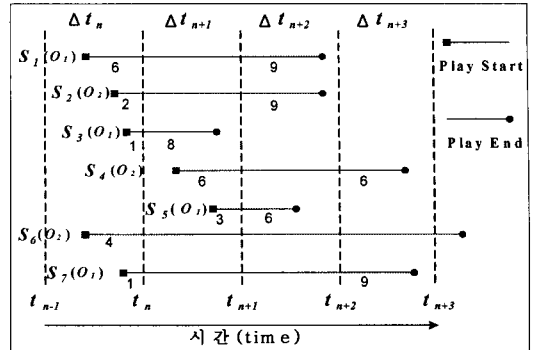


그림 7. 연속미디어 객체의 스트림 전송

수식 (7)에 의해서  $S_i$ 는 가중치가 1이 되고,  $S_3$ 와  $S_7$ 은 인접한 프록시 P3 와 P4 에서 발생한 요구이므로 프록시 가중치 1/4 에 협력비( $\alpha$ )를 적용하여 가중치는  $(1/4) \cdot \alpha$  이 된다. 그러므로  $t_n$ 시간에 요청한  $O_1$ 과  $O_2$ 를 재배포하기 위한  $f_{LFU}(O_1, t_n)$ 과  $f_{LFU}(O_2, t_n)$ 은 수식 (8)으로부터

$$\begin{aligned}
 & Sum\_W\_A_r(S_i(O_1), \Delta t_n) \\
 &= W\_A(S_1(O_1), \Delta t_n) + W\_A(S_3(O_1), \Delta t_n) + W\_A(S_7(O_1), \Delta t_n) \\
 &= 6 + 0.5 \cdot \alpha
 \end{aligned}$$

$$\begin{aligned}
 & Sum\_W\_A_r(S_i(O_2), \Delta t_n) \\
 &= W\_A(S_2(O_2), \Delta t_n) + W\_A(S_6(O_2), \Delta t_n) \\
 &= 0.5 \cdot \alpha + 4
 \end{aligned}$$

가 된다.

이와 같은 방법으로 계산된  $f_{LFU}$ 는 다음과 같다.

$f_{LFU}$ 값 \ 시 점	$t_n$	$t_{n+1}$	$t_{n+2}$	$t_{n+3}$
$f_{LFU}(O_1, t)$	$6 + 0.5\alpha$	$10 + 5.25\alpha$	$9 + 4\alpha$	$2.25\alpha$
$f_{LFU}(O_2, t)$	$4 + 0.5\alpha$	$10 + 4\alpha$	$10 + 4.75\alpha$	$10 + 1.5\alpha$

**(2) LFU 알고리즘**

LFU 알고리즘은 다음과 같은 2단계 구조로 수행된다.

① 1 단계 : 가중치와 재생량을 고려한  $f_{LFU}(O_k)$  값 계산

$f_{LFU}(O_k)$ 를 계산하기 위해서 클러스터 프록시 그룹 내의 프록시에서  $O_k$ 의 요구 정보가 추가될 때,  $O_k$ 의 가중치와 재생량을 고려하여 수식 (6)과 (7)로부터  $W\_A(S_i(O_k), \Delta t_n)$ 을 계산한다. 그리고 이 값을 현재 값과 누적하는 모듈과  $O_k$ 요구의 서비스가

끝났을 때 지원된 스트림의  $WA(S_i(O_k), \Delta t_n)$  값만큼 감소시키는 모듈, 그리고 지원되고 있는 스트림의  $WA(S_i(O_k), \Delta t_n)$  값을 반환하는 모듈로 구성한다.

② 2단계 : 재배치 객체 선택

입력값  $R_{CS}$ 는 한 프록시에서 객체에 할당되지 않은 캐쉬의 잔여 공간을 표시하며,  $C_0$ 는 한 프록시에서 담당하는 캐싱된 객체들의 집합을 나타낸다. 캐싱된 객체들 중 객체  $j$ 를 선택하여  $f_{LFU}(O_j)$  값이 요구한 객체의  $f_{LFU}(O_k)$ 보다 작은 객체를 검색하여 없으면 재배치하지 않고, 있으면 객체  $j$ 를 해당 캐쉬의 잔여 공간에 더한다. 이러한 과정을  $Size(O_k) < R_{CS}$ 가 될 때까지 반복하여  $Size(O_k)$ 를 재배치할 잔여공간을 확보하여 재배치한다. 재배치 객체 선택 알고리즘은 그림 8과 같다.

```

Procedure 재배치 객체 선택
입력 :  $f_{LFU}(O_k), O_k, R_{CS}$ 
출력 : 재배치 여부
Begin
 $V \leftarrow \phi$ 
while(1) {
  if ( $Size(O_k) < R_{CS}$ ) return 재배치함.
  if ( $\{j \mid \forall j \in C_0, f_{LFU}(O_j) < f_{LFU}(O_k), j \notin V \text{인 } j \text{를 선택}\} = \phi$ )
    return 재배치하지 않음.
   $R_{CS} \leftarrow R_{CS} + Size(O_j)$ 
   $V \leftarrow V \cup \{j\}$ 
}
End
    
```

그림 8. 재배치 객체 선택 알고리즘

4. 실험 및 성능 평가

4.1 실험 환경

협력적 프록시 캐싱을 다음과 같은 환경을 기본적으로 가정하며 실험의 구현에 사용된 환경 변수는 표 1, 표 2, 표 3과 같다.

① 각 프록시의 CPU와 메모리 자원의 대역폭은 충분하다.

② 원격지 서버와 각 프록시간과 각 프록시들 간, 그리고 각 프록시와 이동 노드들 간의 네트워크 대역폭은 충분하다고 본다.

③ 이웃하는 프록시들과의 거리는 일정한 거리로 논리적으로 연결되어 있다.

④ 이동 노드가 요구한 미디어 객체는 처음부터 끝까지 본다고 가정한다.

표 1. 실험 환경 변수

환경 변수	고정 값
모의실험 시간	5 시간
연속미디어 개수	1000 개
CBR 연속미디어 길이	20 Mbyte
미디어의 요구 비트율	128 Kbps
시간당 전체 서비스 요구 도착율	1000 (명/초)
프록시 영역의 대역폭	제한 없음
프록시 영역의 개수	64 (8 * 8)

표 2. 캐싱 정책 환경 변수

환경 변수	고정 값
재배치 기법	LFU
프록시 메모리 크기	512 Mbyte
Time Window Size	5 분

표 3. 사용자 이동 환경 변수

환경 변수	값의 범위
이동성향의 구분점 <sup>4)</sup>	10 분
이동성이 큰 사용자 비율 <sup>5)</sup>	0%, 50%, 100%
이동할 영역의 선택 확률	1/3, 1/5, 1/8

4.2 성능평가

제한한 협력적 프록시 캐싱 구조에서 협력적 캐싱을 객체 단위와 세그먼트 단위의 재배치 기법으로 나누어 이동성 비율에 따라 수행하고, 세그먼트의 크기와 스트림의 가중치별로 수행하여 그 결과에 대한 성능을 평가한다. 성능평가를 위해서는 프록시 캐싱에서의 ① BHR(Byte Hit Ratio), ② BRR(Byte Replacement Ratio), ③ 광대역 요구 대역폭<sup>6)</sup>을 기준으로 한다.

4.2.1 이동성 비율에 따른 성능평가

세그먼트 크기는 20 sec 동안 재생되는 데이터 량으로 간주하고, 이동성 비율별로 캐싱을 세그먼트 단위와 객체 단위의 재배치 기법으로 나누어 수행한다.

4) 이동노드가 한 영역에서 10분 이내에 다른 영역으로 이동할 경우 이동성향이 크다고 본다.  
 5) 간단하게 '이동성 비율'이라고 표현한다.  
 6) 프록시가 서버에 요구한 스트림의 량

**(1) BHR 비교**

그림 9에서 전체적으로 세그먼트 단위의 기법이 객체 단위의 기법보다 BHR 값이 크다. 그리고 이동성 비율이 50%일 경우가 다른 경우보다 전체적으로 1~2% 정도 크며, 특히 세그먼트 단위로 협력적 캐싱하는 것이 81.2%로 가장 크다. 그런데 세그먼트 단위 기법에서 이동성 비율이 100%인 경우는 0%나 50%인 경우보다 BHR 값이 3~4% 작다.

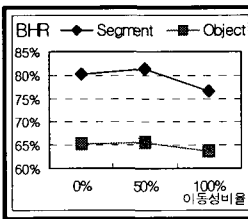


그림 9. BHR

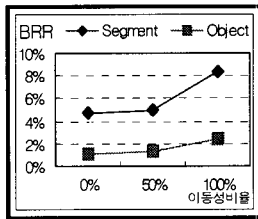


그림 10. BRR

**(2) BRR 비교**

그림 10은 전체적으로 세그먼트 단위의 기법이 객체 단위의 기법보다 평균 2.6%와 4.1% 정도로 BRR 값이 크다. 이동성 비율(100%)이 높을수록 세그먼트 단위 기법의 BRR 값(8.3%)은 증가하여 조금 더 자주 재배치가 발생한다는 것을 의미하며, 이것은 재배치 하는데 부하가 큼을 알 수 있다.

**(3) 광대역 요구 대역폭 비교**

그림 11은 이동성 비율이 0%, 50%, 100% 인 경우의 광대역 요구 대역폭을 나타낸다.

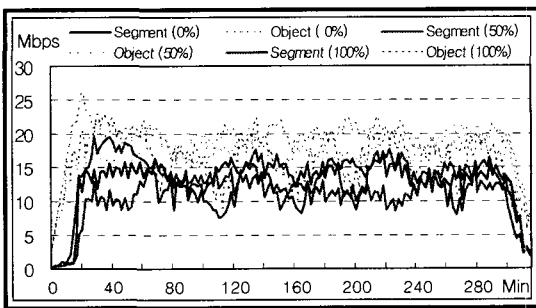


그림 11. 광대역 요구 대역폭

전체적으로는 세그먼트 단위의 재배치 기법이 객체 단위의 재배치 기법보다 4~6 Mbps 정도 작게 나타난다. 이것은 세그먼트 단위가 객체단위보다 효율적인 것을 보여준다. 그리고 세그먼트 단위에서 사용

자 이동성 비율이 50%일 때 광대역 요구 대역폭이 평균 11.2 Mbps 로 가장 작게 나타난다. 이는 이동성 비율이 적당할 경우에 캐싱의 효과가 좋음을 보여준다.

**4.2.2 세그먼트 크기에 따른 성능평가**

협력적 프록시 캐싱의 세그먼트 단위 재배치 기법에서 이동성 비율별로 세그먼트 크기에 따라 수행한다.

**(1) BHR 비교**

그림 12는 이동성 비율 별로 세그먼트의 크기가 커짐에 따라서 BHR의 값이 증가하여 20~25 sec에서 가장 높은 값을 보이다가 크기가 30 sec 이상으로 증가하면 BHR 값은 감소할 나타낸다. 이는 적절한 크기의 세그먼트 크기를 적용하는 것이 BHR 값에 영향을 미침을 나타낸다.

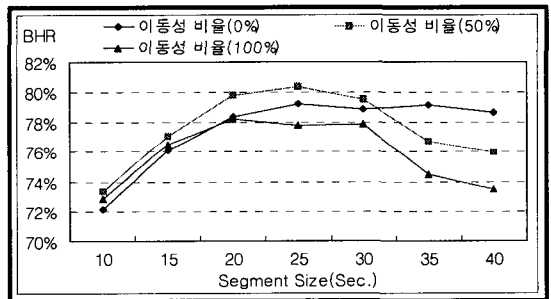


그림 12. BHR

**(2) BRR 비교**

그림 13은 이동성 비율별로 세그먼트의 크기가 커짐에 따라서 BRR 값이 감소하는데, 이동성 비율이 0%와 50%인 경우에서 세그먼트 크기가 15 sec와 20 sec 일 때까지는 1~2%정도 감소하다가 크기가 더 클 경우에는 거의 일정하게 유지됨을 나타낸다. 그리고 이동성 비율이 적당할 경우(50%)가 클 경우(100%)보다 BRR 값이 작게 나타난다.

**(3) 광대역 요구 대역폭 비교**

그림 14는 이동성 비율이 50%일 경우에서, 세그먼트 크기가 15 sec와 35 sec일 경우는 광대역 요구 대역폭이 평균 13 Mbps이고, 세그먼트 크기가 25 sec일 경우가 평균 11 Mbps로 성능이 가장 좋음을 나타낸다. 이는 그림 12에서 이동성 비율이 50%이고



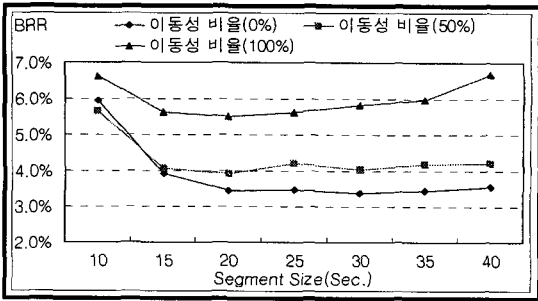


그림 13. BRR

세그먼트 크기가 25 sec일 경우에 BHR 값이 가장 높게 나타난 것과 마찬가지로, 적당한 세그먼트의 크기가 캐쉬에서 히트율을 조금 더 증가시켜 서버로의 광대역 요구 대역폭이 감소됨을 알 수 있다.

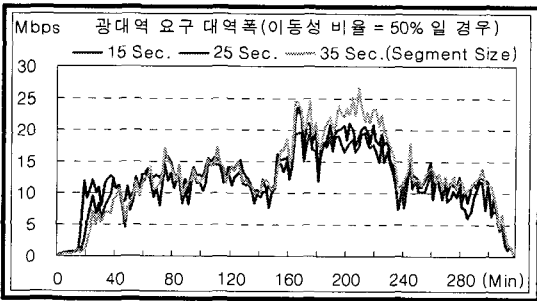


그림 14. 광대역 요구 대역폭

#### 4.2.3 가중치에 따른 세그먼트 단위 재배치 기법의 성능평가

세그먼트 크기를 20 sec로 하고, 이동성 비율에 따라 인접하는 프록시들 간의 협력비를 경험적으로 조정하면서 가중치<sup>7)</sup>에 따른 캐싱의 성능을 평가한다.

##### (1) BHR 비교

그림 15는 이동성 비율이 50%일 때가 가장 좋으며, 각 이동성 비율에서 스트림의 가중치 적용이 1/16 일 때 전반적으로 좋음을 나타낸다. 그리고 가중치 값을 적용하지 않았을 때(No로 표시)는 BHR

값이 매우 작게 나타나서 가중치 적용이 필요함을 보여준다.

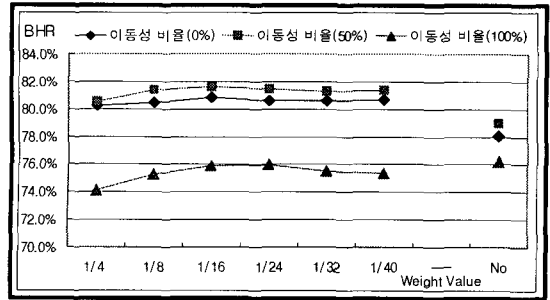


그림 15. BHR

따라서 세그먼트 단위 재배치 기법으로 수행할 때는 이동성이 적절한 경우(50%)에 효과적이고, 이동성이 너무 많으면(100%) 이동성으로 인한 불필요한 Garbage<sup>8)</sup>가 많이 발생하여 효과적이지 못하다.

##### (2) BRR 비교

그림 16은 이동성 비율이 50%일 때가 가장 낮은 BRR 값을 가짐을 나타낸다. 그리고 스트림의 가중치 값을 적용하지 않았을 때(No로 표시) BRR 값은 1~2% 정도 작게 나타난다.

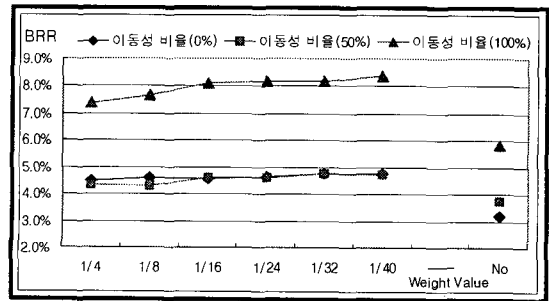


그림 16. BRR

## 5. 결론 및 향후과제

본 연구는 기존의 프록시 캐싱 정책과 달리 이동 환경에서 인접한 영역에서 발생한 많은 사용자 요구 정보들도 활용하여 캐싱하는 협력적 프록시 캐싱 정

7) 그림 6에서 프록시 P1에서 발생하는 스트림의 가중치 값은 1이 되고, 인접하는 프록시 P2, P3, P4, P5에서 발생하는 스트림의 가중치는 프록시의 가중치 1/4 에 협력비  $\alpha$  값을 1, 1/2, 1/4, 1/6, 1/8, 1/10과 같이 적용 하면 스트림의 가중치는 수식 (6)에 의해서 1/4, 1/8, 1/16, 1/24, 1/32이 된다.

8) 사용자가 다른 프록시 영역으로 이동하는 경우, 이전의 영역에서 캐싱한 필요 없게 되는 정보를 의미한다.

책을 제안하여 사용자의 임의접근 요구에 충분히 대응할 수 있는 방법을 제시하였고, 이동환경에서 프록시 캐싱을 적용할 때 고려해야할 요인들을 평가하였다. 세그먼트 단위와 객체 단위로 재배치 기법을 적용하여 실험한 결과, 세그먼트 단위가 객체 단위보다 성능이 우수함을 알 수 있었다. 그리고 그림 9와 그림 11에서 이동성 비율이 적당할 때(50%)에 협력적 프록시 캐싱의 BHR값이 더 크고, 또한 광대역 요구 대역폭이 더 작아 더 좋은 캐싱의 효과를 나타내고 있다. 따라서 본 연구에서는 객체 단위보다는 세그먼트 단위의 재배치 수행이 더 좋으며, 적절한 사용자의 이동성 비율(50%)과 세그먼트의 크기(20~25 sec), 그리고 협력비(1/4)로 조정한 가중치(1/16) 적용이 협력적 프록시 캐싱의 성능에 영향을 미치는 중요한 요인이 됨을 알 수 있었다.

향후 연구되어야 할 부분으로는 이동성이 많은 도로나 거리의 경우와 이동성이 작은 사무실이나 교내, 가정 등의 경우와 같은 사용자 환경을 구분하여 사용자의 이동성향에 따른 이동성 예측 모델의 연구가 필요하며, 이 연구를 기반으로 하여 프록시에서의 선반입, 재배치 기법 등의 성능 향상이 필요하다.

### 참 고 문 헌

[1] J. Jing, et al, "Client-Server Computing in Mobile Environments," *ACM Computing Surveys*, Vol. 31, No. 2, pp. 117-157, 1999.

[2] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," *5th ACM Symposium on Principles of Distributed Computing*, pp. 1-7, 1996.

[3] A. Wolisz, "Mobility in Multimedia Communication," <http://www.tkn.tu-berlin.de/~wolisz/wolisz.html>. 2004.

[4] 박재원, 김문정, 엄영익, "이동 컴퓨팅 환경의 분산 파일 시스템," *한국 정보 과학회지*, 제16권, 제1호, pp. 30-37, 1998년 1월.

[5] S. Hadjiefthymiades and L. Merakos, "Proxy+ Path Prediction : Improving Web Service Provision in Wireless-Mobile Communications,"

*Mobile Networks and Applications*, Kluwer Academic Publishers, Vol. 8, pp. 389-399, 2003.

[6] W. H. O. Lau, M. Kumar, and S. Venkatesh, "A Cooperative Cache Architecture in Support of Caching Multimedia Objects in MANETs," *WoWMoM'02*, pp. 56-63, 2002.

[7] J. Xu, Q. Hu, W. -C. Lee, and D. L. Lee, "Performance Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination," *IEEE TKDE*, Vol. 16, No. 1, pp. 125-139, 2004.

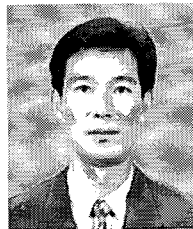
[8] Z. Xiang, et al. "Cost-based Replacement Policy for Multimedia Proxy across Wireless Internet," *IEEE Globecom'01*, Vol. 3, pp. 2009-2013, 2001.

[9] 이화세, 이승원, 박성호, 정기동, "이동환경에서 영역기반의 네트워크 캐싱 효율성 분석," *한국 멀티미디어학회 논문지*, 제7권, 제5호, pp. 668-679, 2004년 5월.

[10] L. Breslau, et al, "Web Caching and Zipf-like Distributions: Evidence and Implications," *IEEE INFOCOM'99*, Vol. 1, pp. 126-134, 1999.

[11] C. Cunha, et al, "Characteristics of WWW Client-based Traces," *Technical Report BU-CS-95-010*, Boston University, July 1995.

[12] 박성호, "프록시 캐쉬에서의 연속미디어 데이터 재배치 기법," 박사학위 논문, 부산대학교, 2002년 2월.



### 이 화 세

1985년 부산대학교 계산통계학과 (학사)  
 1987년 부산대학교 대학원 계산통계학과(이학석사)  
 2004년 부산대학교 대학원 전자계산학과(이학박사)  
 1995년~1997년 밀양대학교 전자계산소 소장

1991년~현재 밀양대학교 컴퓨터공학부 교수  
 관심분야 : 인터넷 캐싱, 멀티미디어