# 네트워크 동적 참여 기반의 효율적인 피어-투-피어 웹 캐싱 모델

## (An Efficient Peer-to-Peer Web Caching Model with the Dynamic Participation of Peers)

류 영 석 † 양 성 봉 ††
(Young-Suk Ryu) (Sung-Bong Yang)

**요 약** P2P(peer-to-peer) 웹 캐싱 모델은 서버 쪽에 집중되는 트래픽을 완화시킴으로써 전통적인 웹 캐싱 모델을 보완할 수 있다는 측면에서 최근에 활발히 연구되어 왔다. P2P 웹 캐싱은 클라이언트들의 로컬 캐쉬를 활용하여 부가적인 인프라의 추가없이 캐쉬 공간이 확대되는 효과를 얻을 수 있지만, 각 클라이언트들은 독립된 피어로서의 자율성(autonomy)을 가지므로 이러한 자율성의 제한을 최소화해야한다. 본 논문에서는 피어의 자율적인 동적 참여와 로컬 캐싱 전략을 보장하여 시스템의 실행 가능성(feasibility)을 높인 환경에서 효율적인 디렉토리 기반 P2P 웹 캐싱 시스템을 제안하였다. 제안하는 P2P 웹 캐싱 시스템은 동적인 P2P 네트워크 상에서의 오브젝트의 lifetime을 예상하여 이를 이웃 선택(neighbor selection)과 저장 공간 관리(storage management)에 적용하였다. 시스템의 성능 평가를 위하여 클라이언트의 http 요청 로그 데이터셋을 이용하여 트레이스 기반(trace-driven) 시뮬레이션을 수행하였다. 시뮬레이션 결과를 통하여 제안하는 시스템이 기존의 시스템에 비하여 주어진 동일한 환경에서 더 높은 정확성과 더 적은 리디렉션 실패(redirection failure)를 가짐을 확인하였다.

**키워드** : 피어-투-피어 시스템, 웹 캐싱, 인터넷 시스템, 월드 와이드 웹

***Abstract*** A peer-to-peer web caching has been studied recently as it can reduce the traffic converged on the server side and can support the traditional web caching model. Although the peer-to-peer web caching has the merit of having additional cache space from the local caches of peers without additional infrastructure, several constraints such as dynamic participation and local caching strategy caused by the autonomy of peers in peer-to-peer networks may limit the performance of the peer-to-peer web caching. To overcome these limitations, we propose an efficient directory-based peer-to-peer web caching system under dynamic participation of peers. In the proposed caching system, we present new peer selection and replica management schemes by introducing the concept of the object lifetime in P2P networks. We evaluate the effectiveness of the proposed system through trace-driven simulations with a web log dataset. Simulation results show that the proposed system has higher accuracy and fewer redirection failures than the conventional directory-based P2P web caching system in feasible peer-to-peer networks.

**Key words** : peer-to-peer system, web caching, internet system, world wide web

## 1. 서 론

P2P(peer-to-peer) 웹 캐싱에서는 클라이언트 각각이 피어로서 P2P 네트워크를 형성한다. 전통적인 웹 캐싱에서는 클라이언트가 웹 오브젝트를 요청(request)했을 때 클라이언트 자신의 로컬 캐쉬를 살펴본 뒤 해당 오브젝트가 없다면 원본 서버나 프락시 서버에 오브젝트를 요청하게 되지만, P2P 웹 캐싱에서는 클라이언트가 웹 오브젝트를 요청했을 때 자신의 로컬 캐쉬에 해당 오브젝트가 없다면 해당 오브젝트는 P2P 네트워크를 검색하여 찾은 이웃(neighbor) - 해당 오브젝트를 과거에 이미 전송받은 클라이언트 -에게 오브젝트를 요청하여 해당 오브젝트를 전송받으며, 이웃을 찾지 못했다면 원

† 학생회원 : 연세대학교 컴퓨터과학과
ryu@cs.yonsei.ac.kr
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
yang@cs.yonsei.ac.kr
논문접수 : 2004년 7월 15일
심사완료 : 2005년 9월 15일

본 서버나 프락시 서버에 오브젝트를 요청하게 된다. 따라서 P2P 웹 캐싱은 부가적인 인프라의 추가 없이도 피어들의 로컬 캐쉬들을 활용함으로써 캐쉬 공간이 확대되는 효과를 얻을 수 있다. 또한, 서버 쪽에 집중되던 트래픽을 다수의 피어들이 나누어 담당함으로써 트래픽의 분산 효과를 얻을 수 있다. 그러나 피어의 로컬 캐쉬는 피어 자신을 위해 우선적으로 쓰여져야하기 때문에 기존의 협력 캐싱(cooperative caching) 모델에서의 전용(dedicated) 캐쉬처럼 사용되기는 힘들다. 또한, 피어의 자율성(autonomy)으로 인하여 P2P 웹 캐싱은 여러가지 고려해야할 문제들이 있게 된다. 첫째, 각 피어는 그 자신의 정책(local policy)를 가지므로, 피어에게 부가적인 부담(load)를 지우는 것을 최소화해야한다. 부가적인 부담으로는 오브젝트를 다른 피어들에게 제공(serve)하는 의무, P2P 네트워크에 일정기간 머물러야 하는 의무 등이 있을 수 있다. 둘째, 피어가 직접 요청하지 않은 오브젝트를 강제적으로 피어의 로컬 캐쉬에 저장하도록 하는 등의 피어의 로컬 캐싱 정책을 거스르는 경우는 최소화해야 한다. 셋째, 피어의 네트워크로의 동적인 참여 특성을 고려해야한다. 네트워크 상에 항상 상주해 있는 전용 캐쉬와는 달리, 피어는 자신의 의지를 가지고 동적으로 P2P 네트워크로의 참여와 탈퇴(dynamic join and leave)를 반복하게 된다. 그러므로 이러한 동적 참여/탈퇴로 인한 시스템 운영 오버헤드나 성능 저하를 최소화해야한다.

이러한 P2P 웹 캐싱 모델의 종류는 이웃 탐색(neighbor discovery) 방식에 따라 디렉토리 기반, 플러딩(flooding) 기반, DHT(distributed hash table) 기반 모델의 크게 3가지로 나눌 수 있다. 디렉토리 기반 P2P 웹 캐싱 시스템 [1-3]에서는 중앙의 서버나 수퍼피어(superpeeer)가 각 오브젝트를 기존에 전송받아간 피어의 정보 목록인 리디렉션 디렉토리(redirection directory)를 운영하며, 클라이언트의 오브젝트 요청을 받으면 이 요청을 리디렉션 디렉토리를 참조하여 해당 오브젝트를 보유하고 있는 이웃에게 재전달한다. 이 방식은 개념이 간단하고 복잡한 P2P 오버레이(overlay)가 필요 없기 때문에 구현이 비교적 쉽다는 장점이 있다. 그리고 기본적으로 단 두 홉(hop) 만에 원하는 피어를 찾을 수 있기 때문에 룩업 비용(lookup cost)이 매우 적다. 그러나 중앙 집중 방식으로 인해 시스템의 안정성(reliability)이 떨어지는 문제점이 있다. 또한, 피어가 매우 동적으로 참여하는 환경일 경우 최신의(up-to-date) 정보를 가지도록 디렉토리를 운영하려면 부가적인 오버헤드가 생길 수 있다. 디렉토리가 최신의 정보를 유지하지 못하면 리디렉션 실패(redirection failure)의 빈도가 증가하게 되며, 이는 룩업 비용을 높이는 요인이 될 수 있다.

플러딩 기반 P2P 웹 캐싱 시스템 [4-6] 은 순수 P2P 탐색 방식과 유사하게 이웃을 탐색한다. 즉, 룩업 요청 메시지를 인접한 피어에게 계속 전파시켜 나가는 방식으로 피어 탐색을 수행한다. 따라서 임의의 오버레이를 가지는 P2P 네트워크에서도 잘 동작하며, 탐색 시 소수의 피어에게만 의존하지 않으므로 안정성이 뛰어나다. 그러나 탐색을 위해 전파되는 룩업 메시지가 네트워크 혼잡을 야기할 수 있으며 특히 P2P 네트워크 상에 희귀하게 존재하는 레어(rare) 오브젝트를 검색할 경우 과도한 트래픽과 지연(latency) 오버헤드(overhead)가 생기게 된다.

DHT 기반 P2P 웹 캐싱 시스템 [7,8]의 경우 찾고자 하는 오브젝트를 가지는 이웃을 해쉬 테이블을 이용해 비교적 소수의 유한한 홉 수만에 찾을 수 있다. 그러나 DHT 기반 탐색 방법은 웹 캐싱에 적용하는데 있어서 몇 가지 문제점들이 있다. 첫째, 각 오브젝트는 기본적으로 하나의 피어가 전담하므로, 많은 요청이 한꺼번에 몰리는 hot spot 오브젝트를 맡은 피어는 과중한 오브젝트 제공 부담을 가지게 된다. 둘째, DHT 기반 탐색은 매우 구조화된 P2P 오버레이 위에서 이루어지므로, 피어가 동적으로 참여/탈퇴하는 웹 환경에서는 P2P 오버레이를 유지하기 위한 운영 오버헤드가 매우 크게 된다[4]. 셋째, 피어가 직접 요청하지 않은 오브젝트를 강제적으로 피어의 로컬 캐쉬에 저장하도록 하기 때문에, 피어의 로컬 캐싱 정책을 거스르게 된다.

본 논문에서는 피어의 자율적인 동적 참여와 로컬 캐싱 전략을 보장하여 시스템의 실행 가능성(feasibility)을 높인 환경에서 효율적인 디렉토리 기반 피어-투-피어 웹 캐싱 시스템을 제안하였다. 제안하는 시스템은 디렉토리 기반 P2P 웹 캐싱 모델을 취하고 있으며, P2P 웹 캐싱을 하고자 하는 웹 사이트는 자신이 소유하고 있는 오브젝트들을 요청해오는 클라이언트 피어들을 관리하는 수퍼피어 역할을 하게 된다. 즉, 웹 사이트는 자신이 소유한 각 오브젝트와 그 오브젝트를 과거에 전송받아간 이웃들의 쌍들로 이루어진 리디렉션 디렉토리를 운영하며, 클라이언트의 오브젝트 요청을 받으면 이 요청을 디렉토리를 참조하여 해당 오브젝트를 보유하고 있는 이웃중의 한명에게 재전달한다. 이러한 과정을 거쳐 해당 오브젝트는 결국 이웃으로부터 클라이언트로 전송되게 된다.

제안하는 웹 캐싱 시스템은 높은 정확성과 적은 리디렉션 실패를 통하여 웹 캐싱의 성능을 높이는 것을 목표로 한다. 이를 위하여, 우리는 P2P 네트워크 상에 존재하는 오브젝트의 라이프타임(lifetime)을 예상하고, 이를 제안하는 시스템 프레임워크의 이웃 선택(neighbor selection) 단계와 저장 공간 관리(storage management)

단계에서 활용하여 시스템의 성능 향상을 꾀하였으며, 제안하는 시스템을 OLP(Object Lifetime-based P2P web caching system)로 명명하였다. 오브젝트의 라이프타임을 예상할 때는 피어의 로컬 캐쉬 내에서의 오브젝트 라이프타임만 고려하는 일반적인 웹 캐싱 시스템과 달리, OLP는 해당 오브젝트를 보유한 피어 자체의 P2P 네트워크 상에서의 라이프타임도 고려한다. 예상된 오브젝트 라이프타임은 이웃 선택 시 해당 오브젝트를 여전히 보유하면서 P2P 네트워크 상에 머물고 있을 가능성이 큰 이웃을 선택하는데 활용된다. 또한 OLP는 효율적인 전체 P2P 네트워크의 저장 공간 관리를 위하여 오브젝트의 라이프타임과 요청 빈도(request rate)를 고려하여, 요청한 피어의 로컬 캐쉬 내에 오브젝트를 선택적으로 저장한다.

시스템의 성능 평가를 위하여 클라이언트의 http 요청 로그 데이터셋을 이용하여 트레이스 기반(trace-driven) 시뮬레이션을 수행하였다. 시뮬레이션의 결과를 통하여 제안하는 시스템이 기존의 디렉토리 기반 P2P 웹 캐싱 시스템에 비하여 주어진 동일한 환경에서 더 높은 정확성과 더 적은 리디렉션 실패를 가짐을 확인한다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하며, 3장에서는 본 연구에서 제안하는 P2P 웹 캐싱 시스템을 기술한다. 4장에서는 시뮬레이션을 통한 성능 평가 결과를 설명하고, 5장에서는 고려해야할 문제점에 대하여 토의하며, 6장에서는 결론 및 향후 연구 진행 방향을 제시한다.

## 2. 관련 연구

Pseudoserving[1]은 선구적인 피어-투-피어 웹 캐싱 시스템의 하나이며 서버의 트래픽 병목 현상을 완화시키는데 초점을 맞추고 있다. Pseudoserving에서는 피어의 동적인 참여에 따른 P2P 네트워크 관리의 어려움을 해결하기 위하여 계약(contract)이라는 개념을 제시하고 있는데 이는 피어에게 의무적으로 네트워크에 머물러야 하는 시간을 부여하는 것이다. 이러한 계약은 그때 그때의 오브젝트 요청 빈도에 맞추어 해당 오브젝트를 가진 피어가 필요한 수만큼 존재하지 않을 때 그 오브젝트를 새로이 전송받아가는 피어에게 요구되어지게 되는데, 피어에게 인위적인 의무를 부여한다는 측면에서 바람직하지 않다고 할 수 있다.

CoopNet[2,9]은 디렉토리 기반 P2P 웹 캐싱 시스템으로서 Pseudoserving과 유사하게 서버의 트래픽 병목 현상을 완화시키는데 초점을 맞추고 있으며, 미국의 9.11 테러 사태 때의 MSNBC 사이트로의 접속 로그 데이터를 가지고 시뮬레이션을 수행하였다. 또한 서버에 오브젝트를 요청한 피어에게 이웃을 연결해주기 위한 서버로부터의 리디렉션 메시지 전송 자체의 트래픽까지 고려하여 이를 최소화하려고 하였다. 그러나 CoopNet은 개개의 피어들의 로컬 캐쉬 운영은 고려하지 않았으며, 리디렉션 실패의 위험도 고려하지 않았다. Browsers-aware proxy server[3]의 경우, 마찬가지로 디렉토리 기반 방식을 취하고 있으며, 상위의 프락시 서버가 자신에게 속한 클라이언트 피어들의 온라인 상황과 각 피어의 로컬 캐쉬 내의 모든 오브젝트들의 인덱스(index)까지 전부 관리하여, 이를 통해 원하는 오브젝트를 가진 피어를 탐색할 때 리디렉션 실패 없이 찾을 수 있게 했다. 그러나 인덱스를 최신의 정보로 관리하기 위해서는 빈번한 주기적인 갱신이 필요하며, 특히 피어의 수가 많을 경우 피어와 서버간의 통신 오버헤드가 커지게 된다.

Squirrel[7]과 Backslash[8]는 DHT 기반 P2P 웹 캐싱 시스템으로서, Squirrel의 경우 Pastry[10]의 탐색 방법에, Backslash의 경우 CAN[11]의 탐색 방법에 기반하였다. DHT 기반 방식은 찾고자 하는 오브젝트를 비교적 소수의 유한한 홉 수만에 찾을 수 있다는 장점을 가지지만, 전술한 것처럼 P2P 웹 캐싱 분야에는 적합하지 않은 면이 있다.

PROOF[4]는 플러딩 기반 P2P 웹 캐싱 시스템으로, 서버로 클라이언트들의 요청들이 한꺼번에 몰릴 경우에도 오브젝트를 효과적으로 전송하는데 초점을 맞추고 있다. PROOF는 플러딩 기반 탐색을 통하여 서버 대신 오브젝트를 전송해 줄 이웃을 찾게 되며, 피어가 동적으로 참여 탈퇴하는 경우에도 P2P 오버레이 운영에 큰 영향을 받지 않도록 시스템을 설계하였다. 그러나 P2P 네트워크 상에 흔하지 않은 오브젝트를 탐색할 경우에는 네트워크 상으로 전파되어 나가는 룩업 메시지들이 과도한 네트워크 트래픽을 야기하게 된다. BuddyWeb[6,12]의 경우 역시 플러딩 기반 탐색 방식을 취하고 있으며, 룩업 비용을 줄이기 위하여 유사도 함수 (similarity function)을 사용해 룩업 메시지를 찾고자하는 오브젝트를 가지고 있을 가능성이 높은 피어 쪽으로 전파시키는 방법을 취하였다. 그러나 각 피어의 쌍들에 대해 그들이 소유한 모든 오브젝트의 정보를 인자로 하여 유사도 함수를 계산하는 과정에서 야기되는 과도한 계산량이 시스템의 오버헤드가 될 수 있다.

본 논문에서는 피어의 자율적인 동적 참여가 존중되고 오브젝트 탐색 비용이 과도하지 않은 실행 가능한 환경에서 효과적인 P2P 웹 캐싱의 제안에 초점을 맞추고 있다. 이러한 차별화된 목표 환경에서는 기존의 DHT 기반 방식은 피어의 동적 참여로 인한 P2P 오버레이의 유지 비용이 아주 크기 때문에 적합하지 않다. 기존의 플러딩 기반 방식 또한 레어 오브젝트를 탐색할 시 과도한 룩업 비용이 필요할 수 있다는 점에서 목표 환경

에 적합하지 않다고 할 수 있다. 따라서, 제안하는 시스템은 디렉토리 기반 P2P 웹 캐싱에 기반하고 있으며, 일반적인 디렉토리 기반 웹 캐싱 프레임워크 내에서 오브젝트 라이프타임 개념을 활용한 접근 방식을 통하여 그 성능을 높이는 것을 목표로 한다.

## 3. 제안하는 P2P 웹 캐싱 시스템

### 3.1 기본 오퍼레이션

OLP는 웹 서버와 웹 서버를 방문하는 클라이언트들로 P2P 네트워크를 구성한다. 클라이언트들은 웹 서버를 떠나 P2P 네트워크에서 탈퇴하기 전까지 피어로서 P2P 네트워크에 존재하게 된다.

OLP에서 서버는 리디렉션 디렉토리 파일을 관리한다. 클라이언트로부터 서버로 웹문서 오브젝트 요청이 오면, 서버는 이웃 선택 단계(neighbor selection process)에서 리디렉션 디렉토리를 참조하여 해당 오브젝트를 이미 전송받아간 수 명의 이웃을 선택한 후 이 이웃 목록(neighbor list)을 포함하는 리디렉션 메시지를 클라이언트에게 보낸다. 클라이언트는 이웃 목록을 참조하여 이웃들에게 오브젝트 요청을 하게 되며, 만약 그 중 한 이웃이 해당 오브젝트를 여전히 소유하고 있고 온라인 상태 - P2P 네트워크에 존재하고 있는 상태 - 이면 해당 오브젝트를 클라이언트에게 전송하게 된다. 만약 그러한 이웃이 없을 때는 서버가 직접 해당 오브젝트를 클라이언트에게 전송한다. 저장 공간 관리 단계(storage management process)에서는 클라이언트가 오브젝트를 이웃으로부터 전송 받았을 경우 오브젝트가 클라이언트에게 보여진 후, 클라이언트의 로컬 에이전트가 해당 오브젝트를 자신의 로컬 캐쉬에 저장할지 여부를 결정하게 된다. 클라이언트가 서버로 특정 오브젝트를 요청했을 때의 OLP의 기본 동작 과정을 정리하면 다음과 같다. 그림 1에서도 이러한 기본 과정을 나타내고 있다.

i. 만약 오브젝트 요청이 클라이언트의 로컬 캐쉬에서 히트 (hit) 한다면, 오브젝트를 클라이언트에게 보인 후 모든 과정을 종료한다.

ii. 서버는 리디렉션 디렉토리를 참조하고 **이웃 선택 단계**를 통해 오브젝트를 이미 전송받아간 $N$ 명의 이웃을 선택하여, 선택한 $N$ 명의 이웃들의 주소 목록을 포함하는 리디렉션 메시지를 클라이언트에게 보낸다. 만약 리디렉션 디렉토리에서 오브젝트의 이웃을 찾을 수 없다면, 서버는 오브젝트를 클라이언트에게 직접 전송하고 리디렉션 디렉토리를 갱신 한 후 모든 과정을 종료한다.

iii. 클라이언트는 서버로부터의 리디렉션 응답을 받은 후, 이웃 목록의 각 이웃에게 오브젝트를 요청한다.

iv. 만약 오브젝트 요청이 특정 이웃의 로컬 캐쉬에서 히트한다면, 오브젝트는 해당 이웃으로부터 클라이언트로 제공된다. $N$ 명의 이웃 모두의 로컬 캐쉬에서 미스 (miss) 인 경우, 클라이언트는 서버에게 오브젝트를 재요청하게 되며, 서버는 오브젝트를 클라이언트에게 직접 전송하고 리디렉션 디렉토리를 갱신한 후 모든 과정을 종료한다.

v. 오브젝트가 이웃으로부터 전송 완료되고 클라이언트에게 보여진 후, 클라이언트의 로컬 에이전트는 오브젝트를 자신의 로컬 캐쉬에 저장할지 여부를 **저장 공간 관리 단계**를 통해 결정한다.

vi. 클라이언트는 서버로 갱신 메시지를 보낸다. 서버는 메시지를 받은 후 필요하다면 리디렉션 디렉토리를 갱신한다.

상기 과정 중 효율적인 이웃 선택과 저장 공간 관리 단계를 위하여 우리는 P2P 네트워크에서의 오브젝트 라이프타임을 예측하고 이를 활용하였다. 이후의 장들에서 이러한 주요 요소들을 세부적으로 기술한다.

### 3.2 주요 인자

3.2.1 P2P 네트워크 환경에서의 오브젝트 라이프타임

일반적인 웹 캐싱에서 클라이언트의 로컬 캐쉬 내 오브젝트의 라이프타임은 클라이언트의 로컬 replacement 정책으로 인해 해당 오브젝트가 로컬 캐쉬에서 삭제됨으로써 끝나게 된다[13]. 그러나 P2P 웹 캐싱에서의 오브젝트 라이프타임의 경우에는, 피어 자체도 P2P 네트워크를 동적으로 참여/탈퇴하기 때문에, 피어의 로컬 캐쉬 내 오브젝트의 라이프타임 뿐만 아니라 해당 오브젝

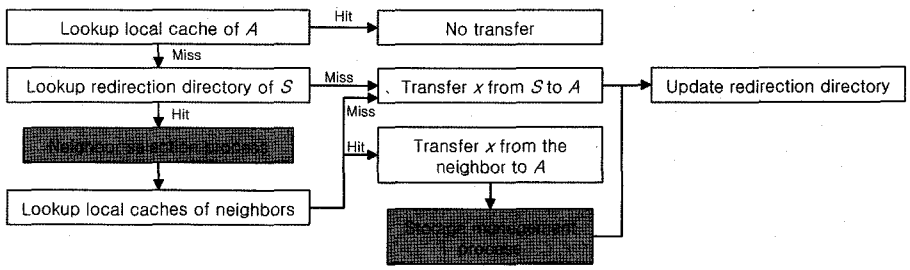



그림 1 OLP의 기본 동작 과정 (피어 $A$가 서버 $S$에게 오브젝트 $x$를 요청했을 경우)

트를 소유하고 있는 피어 자체의 라이프타임도 고려하여야 한다. 그러므로 우리는 P2P 네트워크에서 오브젝트 라이프타임을 고려하고 이를 OLP 프레임워크에 활용하였다.

$LT_A$를 피어 $A$에 저장되어 있는 오브젝트의 라이프타임이라고 하면, $LT_A$는 식 (1)과 같이 피어 $A$의 평균 활동 기간과 피어 $A$의 평균 오브젝트 replacement 시간 중 최소값으로 예측될 수 있다. 식 (1)에서 $p_A$는 피어 $A$의 총 탈퇴 횟수이고, 피어 $A$의 $i$ 번째 참여 시각은 $T_j^i$ 이며 $i$ 번째 탈퇴 시각은 $T_l^i$ 이다. $q_A$는 피어 $A$의 로컬 캐쉬에서 오브젝트가 삭제된 총 횟수이며, 피어 $A$가 자신의 로컬 캐쉬에 저장한 $k$ 번째 오브젝트의 저장 시각은 $T_s^k$ 이고 삭제 시각은 $T_r^k$ 이다.

$$LT_A = \min\left(\frac{\sum_{i=1}^{p_A}(T_l^i - T_j^i)}{p_A}, \frac{\sum_{k=1}^{q_A}(T_r^k - T_s^k)}{q_A}\right) \quad (1)$$

따라서, 우리는 $T_j^{p_A+1}$ 이후 - 피어 A가 $p_A$ +1 번째 참여한 시간 이후 - 의 시간 $t$에 피어 $A$의 로컬 캐쉬에 저장되는 오브젝트가 P2P 네트워크 상에서 소멸되는 시각을 예측하였다. $DT_A^t$를 시간 $t$에 피어 $A$에 저장된 오브젝트의 소멸 시간이라고 하면, $DT_A^t$는 식 (2)와 같이 예측될 수 있다.

$$DT_A^t = \min\left(T_j^{p_A+1} + \frac{\sum_{i=1}^{p_A}(T_l^i - T_j^i)}{p_A}, t + \frac{\sum_{k=1}^{q_A}(T_r^k - T_s^k)}{q_A}\right) \quad (2)$$

3.2.2 오브젝트 요청 빈도

서버 $S$는 오브젝트 별 요청 빈도를 관리하며 OLP의 저장 공간 관리를 위하여 요청 빈도를 이용한다. 요청 빈도를 통하여 해당 오브젝트가 다음 번에 요청될 시간을 예상할 수 있는데, 현재 시간 $t'$에 웹 문서 $x$에 대한 다음 요청 예상 시간(predicted next request time) $RT_x^{t'}$는 다음 식 (3)과 같다.

$$RT_x^{t'} = t' + \frac{1}{the\ request\ rate\ of\ x} \quad (3)$$

### 3.3 이웃 선택(neighbor selection)

리디렉션 디렉토리는 기록할 당시에는 정확한 정보지만 이를 참조한 리디렉션 작업이 실패할 경우가 있는데, 크게 해당 피어의 로컬 캐쉬 정책으로 인해 이미 해당 웹 문서의 replacement가 이루어진 경우와 해당 피어 자체가 P2P 네트워크에서 탈퇴하여 오프라인인 상태일 경우의 두 가지로 들 수 있다. 이러한 경우를 방지하기 위해서 각 피어의 소유 오브젝트의 인덱스나 온라인/오프라인 여부를 서버가 모두 관리할 경우 관리 비용이 지나치게 커지므로 현실적인 대안이 될 수 없다.

OLP는 기본적으로 피어들의 온라인/오프라인 여부만을 관리하며, 더 나아가서 온라인/오프라인 여부조차 실시간으로 관리하지 않는 가벼운(lightweight) 시스템이다. 그러므로 OLP는 이로 인한 리디렉션 실패의 위험을 최소화해야 한다. OLP에서는 특정 오브젝트 요청에 대하여 리디렉션 디렉토리에 등록된 이웃이 둘 이상일 경우 리디렉션 작업 전에 있게 되는 이웃 선택 단계 때, 리디렉션 실패를 방지하기 위해 해당 오브젝트를 여전히 보유하면서 P2P 네트워크 상에 머물고 있을 가능성이 큰 이웃을 선택하는데 초점을 맞추었다. 이를 위하여 OLP는 오브젝트 라이프타임을 이웃 선택 단계에 활용한다. 즉, 사이즈 $N$의 이웃 목록을 확정할 때, OLP는 $DT$가 가장 긴 상위 $N$명의 피어를 순서대로 이웃으로 선택한다.

### 3.4 저장 공간 관리(storage management)

산재된 피어들이 협력하는 환경에서 오브젝트의 동일한 복제본 즉 replica 관리와 효율적인 저장 공간 관리는 중요한 이슈 중의 하나이다. 일반적인 협력적 캐싱 분야에는 효율적인 저장 공간 활용을 위하여 개개의 로컬 캐쉬 정책을 희생하여 협력의 긴밀도를 높여서 웹 캐싱 환경 전체의 성능향상을 추구하는 방식이 많은데, 이는 개개의 캐쉬 자체가 전용 캐쉬이기 때문에 가능한 것이다. 반면에 P2P 웹 캐싱의 경우 개개의 캐쉬는 클라이언트 피어의 로컬 캐쉬이며 로컬의 의지의 반하는 정책은 피어의 자율성을 침해하는 결과를 가지므로 피어가 직접 요청하지 않은 오브젝트를 그 피어의 로컬 캐쉬에 강제적으로 저장하게 하는 방법은 바람직하지 않다. 효율적인 저장 공간 관리를 통한 캐싱의 성능 향상을 위하여, 우리는 피어가 직접 요청한 오브젝트에 대해, '가치 있는(valuable)' 오브젝트만 그 피어의 로컬 캐쉬에 저장하게 하여, '가치 없는(valueless)' 오브젝트를 저장함으로 인하여 일어나는 로컬 캐쉬의 오브젝트 replacement를 최소화하려고 하였다. 여기서 가치있는 오브젝트란 해당 오브젝트가 저장될 경우, 충분한 시간 동안 계속 존재하여 다른 피어로부터 요청받아 그 피어에게 제공되어질 가능성이 높은 오브젝트를 의미한다.

OLP에서는 피어가 자신이 요청한 오브젝트를 이웃으로부터 전송받았을 경우, 전송받은 오브젝트를 클라이언트가 사용한 후, 저장 공간 관리 단계에서 오브젝트를 자신의 로컬 캐쉬에 저장할지 여부를 결정하게 된다. 만약 로컬 캐쉬에 해당 오브젝트를 저장할 수 있을 만큼 비어 있는 사용되지 않은 여분의 공간이 있다면 해당 오브젝트는 저장되게 된다. 그러한 공간이 없을 경우에도, 피어의 $DT$가 이웃의 $DT$와 오브젝트의 $RT$보다 더 나중이라면 가치있는 오브젝트로 간주하여 해당 오브젝트는 저장되게 된다. 그림 2는 피어 $A$가 오브젝트 $x$를 시간 $t'$에 요청하여 시간 $t$에 전송완료 받았을 때, 이러한 저장 공간 관리 단계를 pseudo 코드로 나타낸 것이다.

```
IF (A receives x from the server)
   x is stored in the local cache of A;
ELSE IF (A receives x from B who is one of the neighbors in the neighbor list)
   IF (there is enough space for x in the local cache)
      x is stored in the local cache of A;
   ELSE
      A calculates DT_A^t;
      A reads DT_B^t'' from the neighbor list;
            // t'' is the time when the x was stored in B
      A reads the request rate for x from the neighbor list and calculates RT_x^t';
   IF (DT_A^t >= DT_B^t'' AND DT_A^t >= RT_x^t' )
      x is stored in the local cache of A;
   ELSE
      After shown to A, x is not stored in the local cache of A;
```

그림 2 저장 공간 관리 단계

### 3.5 리디렉션 디렉토리 파일 및 리디렉션 메시지

OLP에서 서버는 다음과 같이 구성된 리디렉션 디렉토리 파일을 관리한다.

*List of < Object_File_ID, Request_Rate, List of <Neighbor_IP_Address, DT_of_Neighbor> >*

리디렉션 디렉토리에서 한 오브젝트 당 등록되는 이웃의 수는 수십 개 정도를 한도로 한다. 서버가 피어로부터 오브젝트 요청을 받으면 리디렉션 디렉토리를 참조해 $N$명의 이웃을 선택하고 요청한 오브젝트에 대한 요청 빈도 값 및 이웃 IP주소와 $DT$로 이루어진 $N$명의 이웃 목록을 리디렉션 메시지로 하여 피어에게 응답하게 된다.

### 3.6 각 피어의 로컬 에이전트

OLP에서 각 피어의 에이전트는 최근 참여/탈퇴 시간, 평균 활동 기간, 평균 오브젝트 replacement 시간 등의 로컬 정보를 관리한다. 이러한 정보를 이용하여 에이전트는 오브젝트 라이프타임을 예측하여 활용하게 된다. 즉, 피어가 오브젝트를 요청한 후 해당 오브젝트를 이웃으로부터 전송받았다면, 오브젝트가 클라이언트에게 보여진 후 에이전트는 저장 공간 관리 단계를 통하여 선택적으로 해당 오브젝트를 로컬 캐쉬에 저장한다. 오브젝트를 로컬 캐쉬에 저장하는 경우에, 에이전트는 갱신 메시지를 서버에게 보내어 해당 오브젝트에 대해 자신을 리디렉션 디렉토리에 추가시킨다. 또한 이웃의 상황이 변하여 리디렉션을 통한 검색 과정에 실패(failure)가 생겼을 경우에도 갱신 메시지를 서버에 보내어 리디렉션 디렉토리에 등록된 이웃 정보를 최신정보로 갱신한다.

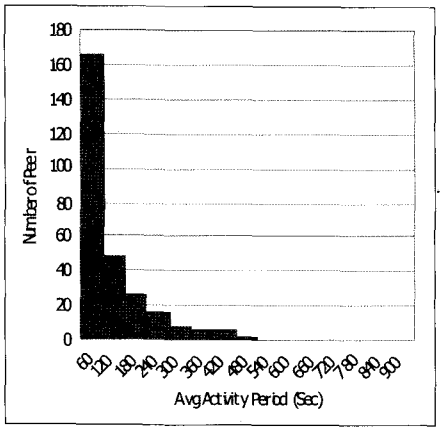
## 4. 성능 평가

### 4.1 시뮬레이션 환경

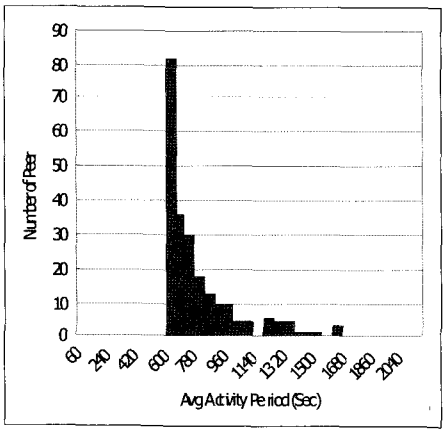
제안한 OLP의 성능을 평가하기 위하여 클라이언트의 웹 문서 요청 로그를 사용해 trace-driven 시뮬레이션을 수행하였다. 사용한 웹 로그는 1998년 월드컵 데이터셋으로 1998년 4월 30일부터 7월 26일까지 클라이언트들의 월드컵 웹 사이트로의 1,352,804,107개 요청 정보들로 이루어져 있으며, 각 요청 정보는 요청 시간, 클라이언트 아이디, 오브젝트 아이디, 오브젝트 사이즈 등으로 구성된다. 각 일자의 데이터셋은 수만명의 클라이언트들의 수백만 개의 요청들로 구성되어 있다. 본 시뮬레이션은 로그 파일 중 임의의 10일의 데이터를 선택하고 각각의 데이터에 대해 임의의 300명의 클라이언트를 선정한 후 이들의 웹 문서 요청을 로그 파일에 따라 모사하여 수행하였다. 이는 수만 명의 클라이언트들 중 300명만이 P2P 웹 캐싱에 자발적으로 참여하여 동작하는 상황을 가정한 것이다. 시뮬레이션에서 클라이언트들은 각자 자신의 로컬 캐쉬 공간을 가지며, 시뮬레이션 결과 값은 각 경우를 평균 내어 계산한 값이다.

네트워크 모델은 클라이언트 피어와 서버간에 1.5Mbps의 연결 속도를 가지며, 클라이언트 피어끼리는 일괄적으로 1.5Mbps의 연결 속도를 가진다. 오브젝트의 일관성(consistency) 문제에 관해서는 같은 오브젝트라고 해도 사이즈가 다를 경우 갱신된 오브젝트로 간주하였다. 클라이언트 피어와 서버 사이에 프록시 서버가 존재하지 않는 경우를 가정하였는데, 이는 P2P 웹 캐싱 자체만의 영향을 살펴보기 위해서이다.

피어가 부하 한계를 동일하게 가지는 환경 하에서 시스템 간의 공정한 비교를 위해서 각 피어는 동시에 한 개의 다른 피어에게만 오브젝트를 제공할 수 있게 부하를 한정하였다. 피어 로컬 캐쉬의 replacement 전략은 LRU(Least Recently Used) 이다. 피어의 동적인 참여를 모사하기 위하여 피어 자신의 각 요청의 요청 시간




(a) 활동 공백 임계값 $\delta$ = 60




(b) 활동 공백 임계값 $\delta$ = 600

그림 3 피어의 평균 활동 기간 분포의 예

사이의 공백을 기준으로 해당 피어의 P2P 네트워크로의 참여와 탈퇴 동작을 설정하였다. 즉, 피어가 최근의 요청을 한 후 활동 공백 임계값 $\delta$ 초만큼의 시간이 지났는데도 다음 요청을 하지 않는다면 해당피어는 P2P 네트워크에서 탈퇴한 것이며, 그 다음 요청이 일어나는 순간 다시 P2P 네트워크에 참여해 활동을 시작하는 것으로 하였다. 따라서, 임계값 $\delta$가 적을수록 피어의 참여/탈퇴가 더 활발해지게 된다. 활동 공백 임계값 $\delta$는 60초부터 600초까지 60초 간격으로 변경해 가며 모든 시뮬레이션을 수행하였다. 그림 3은 이렇게 모사된 피어의 평균적인 활동 기간의 예를 보여주고 있다.

피어의 동적 참여에 대한 서버의 관리는 tight 모드와 loose 모드의 두 가지 모드의 경우로 구분하였다. tight 모드일 경우는 서버가 피어의 참여/탈퇴 상황을 실시간 모니터링하며, loose 모드일 경우는 서버가 각 피어의 동적 참여를 실시간으로 인지할 의무는 없다. 리디렉션 메시지에 포함되는 이웃 목록의 크기는 $N$은 1, 3, 5로 변경하며 시뮬레이션을 수행하였다. 리디렉션 메시지를 받은 피어는 오브젝트가 발견될 때까지 이웃 목록의 $N$명의 이웃에게 순서대로 요청을 보내며, $N$명의 이웃 모두에게 리디렉션 실패가 난다면 오브젝트는 서버로부터 피어로 직접 전송되게 된다.

성능 비교를 위해서, 이웃 선택 단계에서 가장 최근 등록된 이웃을 선택하는 일반적인 중앙 집중형 디렉토리 기반 P2P 웹 캐싱 - 즉 이웃 선택 단계 시 가장 최근에 오브젝트를 받아간 이웃을 선택하며 저장 공간 관리 단계 시에는 받은 오브젝트를 항상 로컬 캐쉬 공간 내에 저장하는- 시스템(GCD), 제안한 오브젝트 라이프타임 기반 이웃 선택을 통해 GCD를 개선한 시스템(OLP_PS), 제안한 오브젝트 라이프타임 기반 저장 공간 관리를 통해 GCD를 개선한 시스템(OLP_SM), 제안한 오브젝트 라이프타임 기반 이웃 선택과 저장 공간 관리 모두를 수행하는 완전한 OLP 시스템(OLP_PS+SM)을 모사하여 그 성능을 비교 분석하였다.

### 4.2 성능 평가 기준(Performance Metrics)

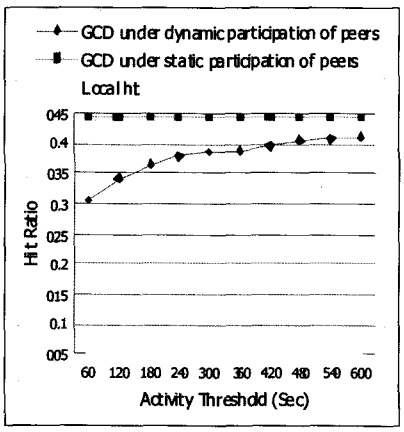
캐싱의 성능을 평가하기위해 사용한 기준은 *hit ratio*와 *byte hit ratio* 의 두 가지이다. hit ratio 는 전체 요청 회수에서 로컬 캐쉬에서의 탐색 성공이거나 이웃 피어 캐쉬에서의 탐색 성공인 회수가 차지하는 비율이며, byte hit ratio는 전체 요청의 대상 오브젝트들의 총 사이즈에서 로컬 캐쉬에서의 탐색 성공이거나 이웃 피어 캐쉬에서의 탐색 성공인 오브젝트들의 사이즈가 차지하는 비율이다. 룩업 비용을 평가하기 위한 기준으로는 캐쉬 히트일 경우 사용된 *the number of lookup messages*을 사용하였다. 이는 오브젝트 탐색 성공 시 이를 위해 몇 회의 룩업 메시지가 보내어 졌는지를 의미한다. 즉, 이웃 목록에 있는 각 이웃에게 순서대로 룩업 메시지를 보낼 경우 몇 번째 이웃에게서 탐색이 성공하였는지를 뜻하므로, 이 값이 적을수록 탐색이 빨리 성공했다는 것을 의미한다.

### 4.3 시뮬레이션 결과

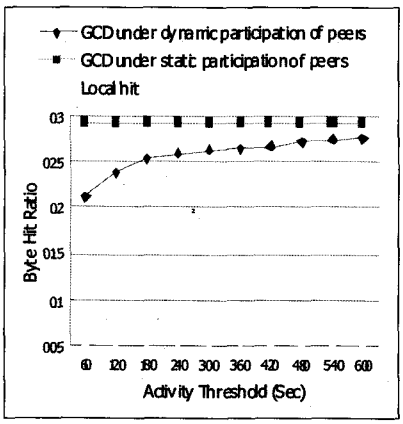
#### 4.3.1 피어의 동적 참여의 영향

그림 4는 피어가 정적으로 참여하는 환경과, 활동 공백 임계값을 변경해가며 동적으로 참여하는 환경에서 GCD의 hit ratio와 byte hit ratio를 비교한 그래프이다. 그림 4를 통하여 피어가 동적으로 참여할 경우의 캐싱의 성능은 정적으로 참여할 경우의 성능보다 떨어지며, 활동 공백 임계값이 적어 참여/탈퇴가 빈번할수록 그 성능의 차이는 큼을 알 수 있다. 이는 기본적으로 동적 참여가 심할수록 특정 시점에 P2P 네트워크에 머무르고 있는 피어의 절대 수가 적어지기 때문이다. 또한 빈번한 참여/탈퇴로 인해 피어의 상태를 실시간으로 모

니터링하지 않는다면 리디렉션 실패가 빈번하게 발생하게 되어 성능이 떨어지게 된다.




(a) Hit ratio

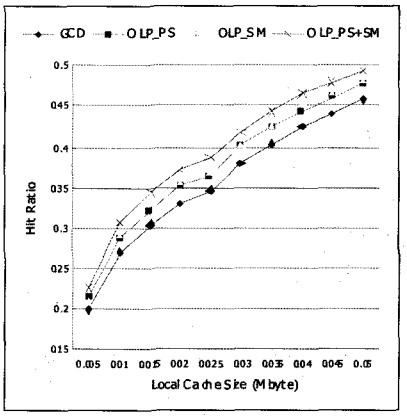



(b) Byte hit ratio

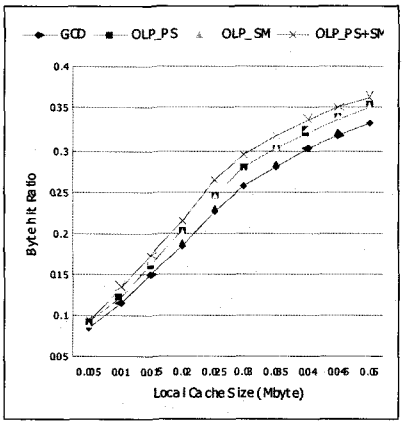
그림 4 GCD의 활동 공백 임계값에 따른 캐싱의 정확도 비교 ($N$=3, the loose mode)

4.3.2 캐싱의 정확도

먼저 캐싱의 hit ratio와 byte hit ratio 측면에서의 성능을 로컬 캐쉬의 크기를 변경해가며 살펴보면 그림 5와 같다. 각 측정값은 활동 공백 임계값 $\delta$을 달리하며 계산한 성능의 평균값이다. 그림 5에서 모든 시스템에서 로컬 캐쉬의 크기가 커질수록 캐싱 성능이 향상되며, 그 중에서 OLP_PS+SM은 향상된 이웃 선택과 저장 공간 관리를 통해 다른 시스템들에 비해 일관적으로 가장 좋은 성능을 보이고, GCD는 가장 낮은 성능을 보인다. OLP_PS와 OLP_SM은 비슷한 정도로 GCD보다 향상된 성능을 보인다.




(a) Hit ratio

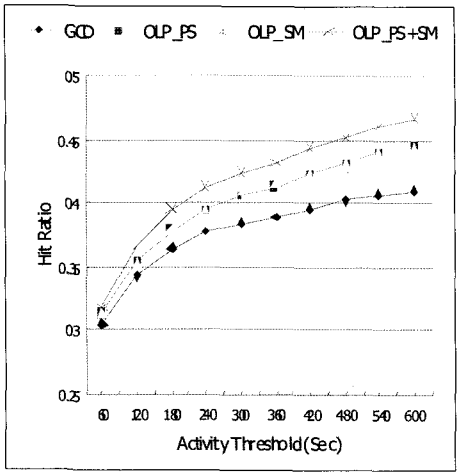



(b) Byte hit ratio

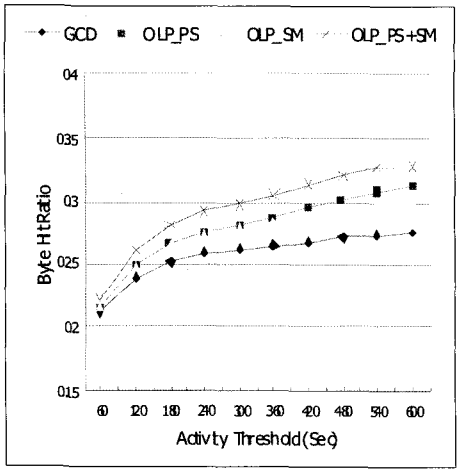
그림 5 캐쉬 사이즈에 따른 캐싱의 정확도 비교 ($N$=3, the loose mode)

그림 6은 로컬 캐쉬의 크기 $S$를 고정시킨 상태에서 시스템들의 캐싱 성능을 활동 공백 임계값 $\delta$ 각각에 대하여 평가하여 비교한 것이다. OLP_PS+SM이 피어의 다양한 동적 참여 환경에 상관없이 일관적으로 가장 좋은 성능을 보이며 OLP_PS와 OLP_SM은 서로 비교적 유사한 성능을 보인다. 그러므로 OLP_PS+SM이 피어가 동적으로 참여/탈퇴하는 P2P 네크워크에 비교대상 시스템들 중 가장 적합하다고 할 수 있다.

다음으로 그림 7은 캐싱의 성능을 이웃 목록의 크기 $N$을 변경해 가며 비교한 것이다. 각 측정값은 활동 공백 임계값 $\delta$을 달리하며 계산한 성능의 평균값이다. 룩업 시도가 많을수록 요청한 오브젝트를 찾을 가능성이 높아지므로 그림 7에서 $N$이 증가할수록 캐싱의 성능이 향상되는 것을 알 수 있다. $N$=1일 경우, OLP_PS와 GCD의 성능 차이가 OLP_SM과 GCD의 차이보다 훨

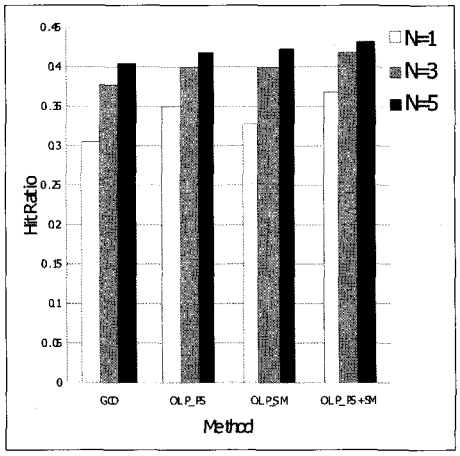



(a) Hit ratio

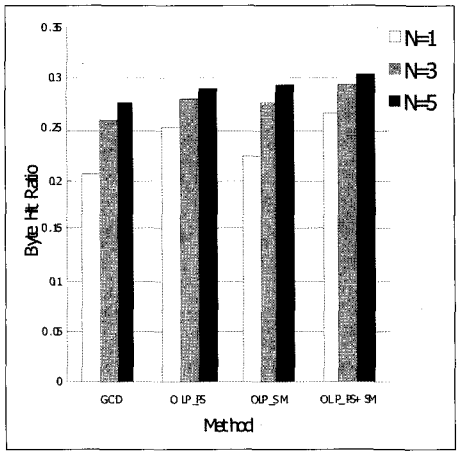



(b) Byte hit ratio

그림 6 활동 공백 임계값에 따른 캐싱의 정확도 비교 ($S$=0.03, $N$=3, the loose mode)




(a) Hit ratio




(b) Byte hit ratio

그림 7 이웃 목록 사이즈에 따른 캐싱의 정확도 비교 ($S$=0.03, the loose mode)

씬 더 큰데, 이는 룩업 시도가 1회뿐이기 때문에 정확한 이웃 선택이 가장 중요한 성능 향상의 요인이기 때문이다. 또한 $N$이 커질수록 GCD와 다른 시스템들 간의 성능 차이의 정도가 줄어드는데 이는 $N$이 커질수록 늘어나는 룩업 시도로 인한 캐쉬 히트의 가능성이 OLP의 순전한(pure) 성능 향상의 효과를 희석시키기 때문이다.

4.3.3 룩업 비용

OLP에서는 각 피어의 동적 참여와 로컬 replacement 정책으로 인해, 원하는 오브젝트를 항상 2회의 홉수 만에 찾을 수 있는 것은 아니다. 또한 원하는 오브젝트를 찾으려는 룩업 시도 횟수가 많을수록 캐싱의 정확도는 향상되겠지만 그만큼의 룩업 비용이 더 들게 되는 trade-off가 있게 된다. 룩업 비용을 평가하기 위해 우리는 그림 8에서 이웃 목록 크기 $N$을 가변해가며 캐쉬 히트의 경우에 사용된 룩업 메시지의 개수를 표시하였다. 이러한 룩업 메시지들을 한번에 보낼 경우는 대역폭이 소모되는 룩업 비용이 들게 되고, 순차적으로 보낼 경우에는 시간이 지연되는 룩업 비용이 들게 되므로, 사용된 룩업 메시지의 수가 많을수록 룩업 비용이 더 드는 것을 의미한다. 그림 8에서 각 측정값은 활동 공백 임계값 $\delta$을 달리하며 계산한 성능의 평균값이다. OLP_PS는 오브젝트 라이프타임을 활용하여 더 정확한 이웃 목록을 생성하여 룩업 메시지의 개수를 감소시킨다. OLP_SM은 제안한 저장 공간 관리 과정을 통해 각 피어가 오브젝트 라이프타임이 비교적 긴 오브젝트만을 저장하는 경향이 있으므로 로컬 replacement로 인한 리디렉션 실패의 빈도를 줄여 역시 룩업 메시지의 개수를 감소시킨다.

OLP_PS+SM의 경우, 이웃 선택과 저장 공간 관리의 두 단계의 시너지 효과를 통해 비교 대상 시스템 중 가장 적은 룩업 비용을 보였다.

$N$이 커질수록 캐싱의 정확도 측면에서 GCD와 OLP의 성능 격차가 그림 7과 같이 줄어든 반면에, 그림 8에서 룩업 비용 측면에서의 성능 격차는 점차 늘어나고 있음을 알 수 있다. 이러한 결과는 GCD가 이웃 목록 사이즈를 늘려 OLP의 성능에 근접하게 리디렉션 성공 횟수를 늘리더라도 그만큼의 룩업 비용이 OLP보다 더 들어간다는 것을 의미한다.

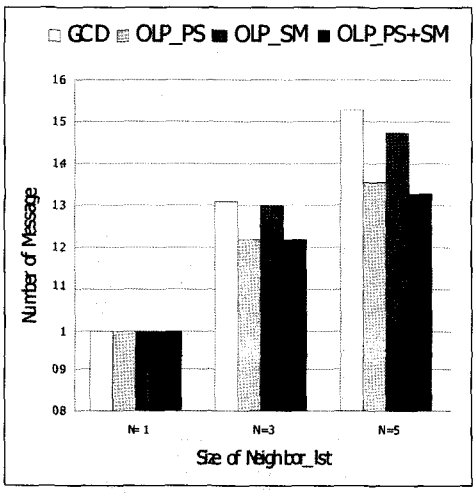
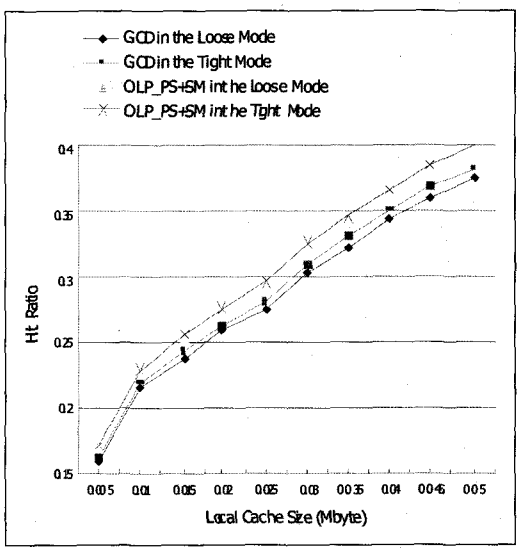


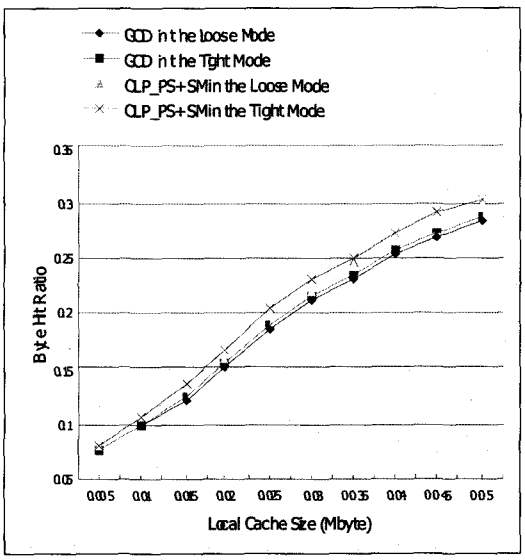

그림 8 이웃 목록 크기에 따른 캐쉬 히트의 경우 사용된 룩업 메시지의 개수 비교($S$=0.03, the loose mode)

#### 4.3.4 피어의 동적 참여에 대한 서버의 실시간 모니터링의 효과

Tight 모드일 경우의 각 시스템의 실험 결과도 loose 모드의 결과와 비슷한 경향성을 가지므로, 지면 관계상 전술한 시뮬레이션 결과에서 tight 모드의 결과는 기술하지 않았다. 그림 9에서는 피어의 동적 참여에 대한 서버의 실시간 모니터링의 효과를 확인하기 위하여 GCD와 OLP_PS+SM의 캐싱 정확도를 tight 모드와 lose 모드에서 비교하였다. 각 측정값은 활동 공백 임계값 $\delta$을 달리하며 계산한 성능의 평균값이다. 그림 9에서 tight 모드일 때 각 시스템의 성능이 loose 모드일 때보다 일관적으로 더 좋음을 확인할 수 있으며, 서버의 실시간 모니터링이 좀더 정확한 리디렉션을 야기했음을 알 수 있다. 또한 loose 모드의 OLP_PS+SM이 tight 모드의 GCD보다 약간 더 좋은 성능을 보이는데, 제안한 오브젝트 라이프타임 기반 이웃 선택 및 저장 공간 관리가 피어의 참여/탈퇴 상태의 실시간 정보 없이도 P2P 웹 캐싱 성능 향상에 기여했음을 분석할 수 있다.




(a) Hit ratio




(b) Byte hit ratio

그림 9 Tight 모드와 loose 모드의 캐쉬 사이즈에 따른 캐싱 정확도 비교 ($\delta$ = 60, N=3)

## 5. 토의(Discussion)

각 피어간의 전송 속도가 실제로는 모두 다르기 때문에 이웃 선택 단계에서 요청한 피어와 가까이 있는 피어를 선택하는 여러 가지 기존의 방법들이 있다. 그러나 우리는 이웃 피어에서의 리디렉션 실패나 캐쉬 미스를 최소화하는데 초점을 맞추어 이웃 선택을 수행하였다. 따라서 OLP에서는 요청된 오브젝트를 여전히 보유하면서 P2P 네트워크 상에 머물고 있을 가능성이 큰 피어

를 이웃으로 선택하였다. OLP에서 더 효율적인 이웃 선택을 위해서는 피어의 지역성(locality)을 고려하는 것이 중요한 문제일 것이다.

저장 공간 관리 단계에서 OLP의 선별적인 오브젝트 저장 방식은 전체 P2P 네트워크에서 더 많은 캐쉬 히트를 이끌어 내었음을 시뮬레이션 결과를 통해 확인하였다. 그러나 만약 이러한 결과가 각 피어의 로컬 캐쉬에서의 캐쉬 히트를 줄이고 이웃 피어의 로컬 캐쉬에서의 캐쉬 히트를 늘림으로써 이루어진 것이라면, 실제 룩업 시간은 증가되어 룩업 비용은 늘어나게 된다. 확인 결과, 우리의 선별적인 오브젝트 저장 방식의 로컬 캐쉬 히트는 기존의 비선별적 저장 방식의 로컬 캐쉬 히트와 0.001 단위에서 차이를 보여 거의 유사함을 확인하였다. 이는 선별적인 오브젝트 저장으로 인해 replacement 되지 않고 로컬 캐쉬에 계속 존재하게 된 오브젝트에 대한 로컬 히트가, 비선별적인 오브젝트 저장 방식이었다면 새로 저장되었을 오브젝트에 대한 로컬 히트를 상쇄시켰기 때문이라고 분석된다.

## 6. 결 론

본 연구에서는 클라이언트들의 다양한 참여/탈퇴 패턴을 보장하고 각 클라이언트로의 강제적인 오브젝트 저장을 배제한 실행 가능성이 높은 환경에서 효율적인 디렉토리 기반 P2P 웹 캐싱 시스템을 제안하였다. 제안한 OLP 시스템은 P2P 네트워크의 오브젝트 라이프타임을 활용하여 이웃 선택과 저장 공간 관리 방안을 제안하였다. trace-driven 시뮬레이션을 통해 오브젝트 라이프타임 기반 접근 방식이 효율적이며 기존의 디렉토리 기반 P2P 웹 캐싱 시스템에 비해 높은 정확도와 적은 룩업 비용을 가짐을 확인할 수 있었다. 향후에는 오브젝트 라이프타임을 활용한 접근 방식을 플러딩 기반 탐색 모델 등으로 확장해 적용하여 연구할 계획이다.

## 참 고 문 헌


[1] K. Kong and D. Ghosal, "Mitigating Server-Side Congestion in the Internet Through Pseudoserving," IEEE/ACM Transactions on Networking, Vol. 7, No. 4, pp. 530-544, 1999.

[2] V. N. Padmanabhan and K. Sripanidkulchai, "The Case for Cooperative Networking," Proc. International Workshop on Peer-to-Peer Systems(IPTPS), pp. 178-190, 2002.

[3] L. Xiao, X. Zhang, A. Andrzejak, and S. Chen, "Building a Large and Efficient Hybrid Peer-to-Peer Internet Caching System," IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 6, pp. 754-769, 2004.

[4] A. Stavrou, D. Rubenstein, and S. Sahu, "A Lightweight, Robust P2P System to Handle Flash Crowds," IEEE Journal on Selected Areas in Communications, Vol. 22, No. 1, pp. 6-16, 2004.

[5] T. Tay, Y. Feng, and M. Wijeysundera, "A Distributed Internet Caching System," Proc. IEEE Conference on Local Computer Networks(LCN), pp. 624-633, 2000.

[6] X. Wang, W. Ng, B. Ooi, K. Tan, and A. Zhou, "BuddyWeb: A P2P-based Collaborative Web Caching System," Proc. Peer-to-Peer Computing Workshop on Networking, 2002.

[7] S. Iyer, A. Rowstron, and P. Druschel, "Squirrel: A Decentralized Peer-to-Peer Web Cache," Proc. Symposium on Principles of Distributed Computing, pp. 213-222, 2002.

[8] T. Stading, P. Maniatis, and M. Baker, "Peer-to-Peer Caching Schemes to Address Flash Crowds," Proc. International Workshop on Peer-to-Peer Systems(IPTPS), pp. 203-213, 2002.

[9] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributed Streaming Media Content Using Cooperative Networking," Proc. ACM International Workshop on Network and Operating System Support for Digital Audio and Video(NOSSDAV), pp. 177-186, 2002.

[10] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," Proc. International Conference on Distributed Systems Platforms, pp. 329-350, 2001.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S.Shenker, "A Scalable Content-Addressable Network," Proc. ACM SIGCOMM, pp. 161-172, 2001.

[12] W. Ng, B. Ooi, and K. Tan, "BestPeer: A Self-Configurable Peer-to-Peer System," ICDE, 2002.

[13] L. Ramaswamy and L. Liu, "A New Document Placement Scheme for Cooperative Caching on the Internet," ICDCS, pp. 95-103, 2002.



류 영 석

1999년 연세대학교 컴퓨터과학과 학사 2001년 연세대학교 컴퓨터과학과 석사 2001년~현재 연세대학교 컴퓨터과학과 박사 과정. 관심분야는 P2P 컴퓨팅, 전자상거래 등



양 성 봉

1981년 연세대학교 공학사. 1984년 Univ. of Oklahoma 컴퓨터과학과 석사 1992년 Univ. of Oklahoma 컴퓨터과학과 박사. 1993년~1994년 전주대학교 전자계산학과 전임강사. 1994년~현재 연세대학교 컴퓨터과학과 교수. 관심분야는 P2P 컴퓨팅, 3D 그래픽스 등