비구조적인 피어-투-피어 네트워크상에서 효율적인 복제기법

최 우 락[†] · 한 세 영[††] · 박 성 용[†††]

요 약

비구조적인 피어-투-피어 시스템에서 효율적인 검색을 위하여 임의 경로 검색 방식이 제안되었는데, 이 방식의 낮은 검색 성공률을 보완하기 위하여 여러 가지 복제 기법이 연구되고 있다. 본 논문에서는 복제본의 생성을 최소화 하고, 캐시를 효과적으로 사용함으로써 검색의 정확성과 속도를 높이고 비용을 줄이는 효율적인 복제 기법을 제안한다. 이 기법에서는 질의가 많이 도착하는 허브 노드에 캐시를 저장하고 그 이웃 노드 중 하나에 복제본을 저장하는데, 제한적이고 지역적인 정보만을 이용하여 허브를 판별하는 알고리듬을 제시하여 동적인 네트워크 내에서 적응적으로 캐시와 복제본을 생성할 수 있게 하였다.

키워드 : 비구조적인 피어-투-피어 네트워크, 복제 기법, 파워-로 토폴로지

An Efficient Replication Scheme in Unstructured Peer-to-Peer Networks

Wurak Choi[†] · Saeyoung Han[††] · Sungyong Park[†††]

ABSTRACT

For efficient searching in unstructured peer-to-peer systems, random walk was proposed and several replication methods have been studied to compensate for the random walk's low query success rate. This paper proposes an efficient replication scheme that improves the accuracy and speed of queries and reduces the cost by minimizing the number of replicas and by utilizing caches. In this scheme, hub nodes store only content's caches, and one of their neighbors stores the replica. By determining hubs with only limited and local information, we can adaptively generate caches and replicas in dynamic peer-to-peer networks.

Key Words : Unstructured Peer-to-peer Network, Replication Scheme, Power-law Topology

1. 서 론

최근 피어-투-피어 서비스는 웹 서비스와 더불어 인터넷 트래픽의 대부분을 차지하고 있는 중요한 인터넷 어플리케이션 중에 하나이다[1]. 시작된 지 불과 5년여 만에 이런 큰 성장을 한 것은 그 확장성 있는 구조 때문이라 할 수 있다.

그 중 비구조적인 방식의 피어-투-피어 시스템의 경우 특정 위상에 기반 하지 않기 때문에 네트워크를 유지하기 위한 비용이 적게 드는 가장 확장성 있는 구조라 할 수 있다. 하지만, 비구조적인 네트워크에서는 질의에 대하여 맹목적인 검색 방식을 사용하기 때문에 검색이 느리고, 해당 파일이 존재하더라도 검색되지 않는 경우가 발생한다. 따라서 비구조적인 방식의 피어-투-피어 네트워크에서 검색 속도와 효율을 높이기 위한 방법으로 임의 경로 검색(Random Walk Searching) 방법이 제안되었다[2].

※ 본 연구는 서강대학교 산업기술연구소의 지원으로 수행되었음.
† 정 회 원 : 퓨처시스템 VPN팀 연구원
†† 정 회 원 : 서강대학교 컴퓨터학과 박사과정
††† 정 회 원 : 서강대학교 컴퓨터학과 부교수
논문접수 : 2005년 8월 23일, 심사완료 : 2005년 11월 1일

임의 경로 검색 방법에서는 검색의 성능 향상을 위해 복제 기법을 사용한다. [3]에서는 위상 적응과 한 단계 복제방법을 이용하여, 용량이 큰 노드가 많은 이웃을 갖게 하고, 많은 파일들에 대한 복제본을 가지게 하였다. 검색 시 용량이 큰 노드들에게 질의가 많이 전달되도록 하여 검색 속도와 그 정확도를 높였다. 하지만 이 방법은 복제에 드는 비용도 문제이거니와 아무리 용량이 큰 노드라 할지라도 복제본이 집중되므로 적정 한계를 넘어버리기 쉬운 문제점을 가지고 있다. 한편, [4]에서는 복제 기법으로 균등 기법과 비례 기법에 대해 분석하고, 최적에 가까운 제곱근 복제 기법을 제시하였으며, 이를 구현하기 위하여 경로 복제방법을 제시하였다. 하지만 이 논문에서는 질의율과 파일의 개수를 고정된 값으로 모델링하고 있으므로, 동적 특성이 큰 피어-투-피어 네트워크에 적용하기 위한 연구가 더 필요하다. 따라서 본 논문에서는 검색의 정확성과 속도를 높이고 비용을 줄이는 효율적인 복제 기법을 제안하고자 한다.

비구조적인 피어-투-피어 네트워크는 일반적으로 파워-로 토폴로지(Power-Law Topology)를 형성하는데, 이는 많은 이웃 노드를 갖는 소수의 허브와 적은 이웃 노드를 갖는

다수의 일반 노드들로 이루어진다[5]. 임의 경로 검색에서는 질의 메시지가 허브에 집중되는 현상을 보이므로 콘텐츠를 허브에 복제하는 것이 검색 효율을 높일 수 있으나, 허브는 많은 질의 메시지를 처리해야 하기 때문에 과부하 상태가 되기 쉽다. 따라서 본 논문에서 허브의 부하를 줄이면서 효율적으로 검색 성능을 향상시킬 수 있도록 허브에는 콘텐츠의 캐시를 저장하고, 허브의 이웃 노드 중 하나에 콘텐츠의 복사본을 저장하는 기법을 제안한다. 또한 이를 위하여 네트워크상에서 제한적이고 지역적인 정보만을 가지고 허브와 일반 노드를 판별하는 알고리즘을 제시함으로써, 동적으로 변화하는 네트워크 내에서 적응적으로 캐시와 복제본을 생성할 수 있도록 한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 관련 연구로 비구조적 피어-투-피어 네트워크에서 제안된 대표적인 복제 기법들을 소개한다. 3장에서는 제안하고자 하는 효율적인 복제 기법을 설명하고, 4장에서 시뮬레이션 결과를 통해 그 효율성을 보인다. 마지막으로 5장에서는 논문의 결론과 향후 연구 방향에 대하여 기술하도록 하겠다.

2. 관련 연구

비구조적 피어-투-피어 시스템에서는 검색의 효율을 높이기 위하여 복제 방법을 사용한다. 기본적으로 Gnutella에서는 콘텐츠를 요청한 노드에 해당 콘텐츠에 대한 복제를 생성하는 수동적인 복제 방법을 사용한다. 반면에, Freenet [9]과 같은 피어-투-피어 시스템은 요청된 콘텐츠에 대해 능동적(Proactive)으로 다른 노드에 복제본을 생성한다. 효율적인 복제를 위해서는 콘텐츠 검색을 위한 비용을 적게 소요하게 만들면서 동시에 복제에 사용되는 비용을 줄여야 한다. 하지만 일반적으로 복제를 많이 생성할수록, 해당 콘텐츠에 대한 검색 비용이 줄어들기 때문에 이 두 가지의 요소는 서로 상충된다.

이 장에서는 기존 연구에서 제시된 한 단계 복제 방법[3], 경로 복제 방법[2, 4], 그리고 가볍고 적응적인 LAR 복제 방법[10]에 대하여 살펴보도록 하겠다.

2.1 한 단계 복제(One-hop Replication)

한 단계 복제 방법은 바로 이웃하는 노드들에게 자신의 모든 콘텐츠의 인덱스를 복제하는 방법이다[3]. 이 방법은 고용량의 노드가 많은 이웃과 연결을 맺게 하는 알고리즘과 함께 사용된다. 복제 방법의 특성 상 이웃이 많으면 복제된 데이터도 많기 때문에, 많은 이웃을 가진 노드들은 복제된 데이터를 저장할 만한 충분한 용량을 소유해야 한다. 파워-로 네트워크의 특성 상 허브들은 매우 많은 숫자의 이웃들을 가지게 되므로, 허브 노드로 데이터의 복제가 많이 일어나게 된다. 또한 허브에 모든 부하가 집중되므로, 허브는 쉽게 과부하 상태가 될 수 있으며 이는 네트워크의 성능을 저하시키는 요인이 된다. 따라서 [3]에서는 허브 노드에 고용량의 노드를 위치할 수 있도록 토폴로지를 적응적으로 변환시켜 주는 알고리즘을 한 단계 복제 방법과 함께 사용하고 있다. 한편 파일 다운로드의 성능을 향상시키기 위하여 인덱스가 아닌 파일 자체를 복제하는 능동적 복제 방법(active replication)이 제안되어 성능 향상을 보였다.

2.2 경로 복제(Path Replication)

능동적인 복제를 위한 방법으로 균등 복제 기법과 비례 복제 기법, 그리고 제곱근 기법 등이 수학적 모델링을 제시하면서 연구되어 왔다[2, 4].

네트워크상에는 n개의 노드가 참여하고 있으며, 총 m개의 구별되는 콘텐츠가 있다 가정하자. 콘텐츠 i가 r_i개의 구별되는 노드들에게 복제되어 있다고 할 때, $R = \sum r_i$는 네트워크 내에 존재하는 모든 콘텐츠의 개수를 의미한다. 그리고 각 콘텐츠들은 q_i의 비율에 따라 요청된다고 하고, $\sum q_i = 1$이라 하자.

이때 균등 복제 기법이란 $r_i = R / m$이 되도록 네트워크 내에 복제본을 생성하는 기법을 말하는데, 각 콘텐츠의 요청 비율에 상관없이 균등한 개수만큼 복제본을 생성한다. 이 방법은 특정 콘텐츠의 요청 비율이 다른 콘텐츠에 비해 매우 높을 때 비효율적이다.

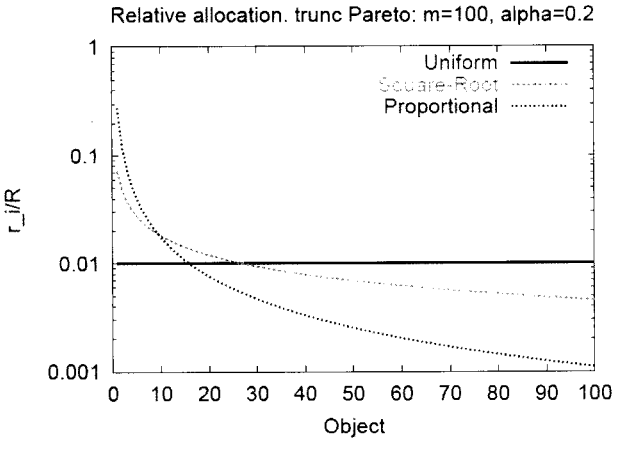
비례 복제 기법에서는 $r_i = Rq_i$가 되도록 네트워크 내에 복제본을 생성하는데, 인기가 많은 콘텐츠는 네트워크 내에 더 많이 복제를 생성하고, 인기가 적은 콘텐츠는 네트워크 내에 적게 복제를 생성하게 한다. 이 복제 기법은 좀 더 인기 있는 콘텐츠에 대한 검색을 빠르게 해 주나, 인기가 없는 콘텐츠에 대한 검색을 매우 느리게 만든다. 그러나 균등 복제 기법과 비례 복제 기법의 성공적인 질의에 대하여 평균 검색 길이(Average Searching Distance)는 같음이 증명되었다.

마지막으로 제곱근 복제 기법에서는 λ가 다음의 (식 1)과 같을 때 $r_i = \lambda\sqrt{q_i}$가 되도록 네트워크 내에 복제본을 생성한다.

$$\lambda = \frac{R}{\sum_i \sqrt{q_i}} \qquad \text{(식 1)}$$

이 방법은 (그림 1)과 같이 낮은 요청 비율을 갖는 콘텐츠에 대해서도 어느 정도의 복제본을 생성함으로써, 인기가 없는 콘텐츠에 대한 검색이 느려지는 현상을 막는다. 또한 제곱근 복제 기법이 균등 복제 기법이나 비례 복제 기법과 비교해 성공적인 질의에 대하여 평균 검색 길이가 작음이 증명되었다.

경로 복제 기법은 이러한 제곱근 복제 기법을 분산 네트워크상에 구현하기 위한 방법으로 제시되었는데, 검색에 걸린 홉 수만큼 복제본을 생성한다. 인기가 있는 콘텐츠는 그 질의 비율이 높지만 네트워크 내에 비교적으로 많이 존재하기 때문에 적은 홉 수만에 검색을 완료할 수 있고, 반면에 비인기 콘텐츠는 질의 비율은 낮지만, 검색에 많은 홉 수를 필요로 한다. 따라서 경로 복제 방법은 질의 비율의 변화에



(그림 1) 균등, 비례, 제곱근 복제 기법의 비교[11]

따라 적응적으로 콘텐츠에 대한 복제를 생성할 수 있다. 그러나 경로 복제 방법은 크기가 큰 네트워크 내에서 여전히 복제에 많은 전송 비용을 초래한다는 한계점이 있다.

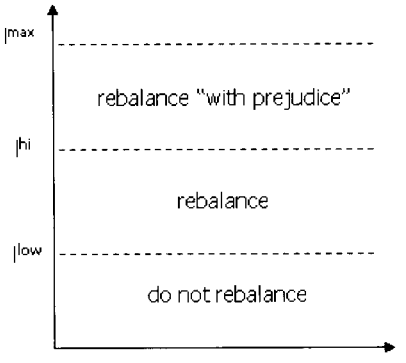
2.3 LAR(Lightweight Adaptive Replication)

앞서 살펴 본 복제 방법들은 복제에 많은 양의 데이터 통신을 필요로 하기 때문에, 실제로 피어-투-피어 네트워크 내에서 사용하기에는 무리가 있다. 따라서 피어-투-피어 네트워크상에서는 좀 더 효율적인 복제 기법을 필요로 한다. LAR 프로토콜은 시스템 독립적인 복제 프로토콜로, 이러한 피어-투-피어 네트워크를 위해 가볍고(Lightweight) 적응적인(Adaptive) 복제(Replication) 방법을 제안하였다[10]. LAR에서는 기본적으로 많은 양의 직접적인 복제를 캐시로 대체하기 때문에 앞서 제시한 복제 기법보다 훨씬 적은 양의 데이터 통신을 필요로 한다. LAR 프로토콜의 작동 방식은 다음과 같다.

LAR 프로토콜은 부하 분산을 목적으로 하기 때문에, 부하에 대한 정의를 제시하고 있다. 부하는 여러 가지 측정 기준이 있을 수 있으며, LAR 프로토콜에서는 전달받은 메시지의 수를 부하로 측정하고 있다. 전달 받은 메시지의 수에 따라 노드의 부하 상태는 (그림 2)와 같이 l^{max}, l^{hi}, l^{low}의 세 가지 상태를 갖는다.

l^{max}란 현재 노드의 상태가 시급히 부하 분산을 수행해야 하는 상태이다. l^{hi} 상태에서는 다른 노드들의 상태에 따라 부하 분산을 수행해야 한다. l^{low} 상태는 현재 노드의 부하가 적절한 상태로 부하 분산을 수행하지는 않고, 다른 부하가 많은 노드의 요청에 의해 해당 노드의 부하를 넘겨받을 수 있는 상태이다.

각 상태에 따른 복제 생성 방법을 보면 다음과 같다. 만일 노드 S_i 가 S_j로부터 생성된 메시지를 전달 받은 경우, 현재의 부하량을 살핀다. 만약 현재가 과부하 상태라면, 노드 S_i는 S_j 에게 부하 분산을 시도한다. 이는 S_j 의 부하에 대한 정보를 메시지에 피기-배킹(Piggy-backing) 함으로써 전달하기가 쉽기 때문에 지역적인 정보만을 이용하여 부하를 분산할 수 있고, 또한 현재의 부하 상태에 S_j 가 일조하



(그림 2) 부하에 따른 분산 방침[14]

기 때문이다.

만일, $l_i > l_i^{hi}$이라면 현재 노드는 과부하 상태이므로, $l_i > l_j + K$인 경우에 두 노드의 부하의 차에 해당하는 콘텐츠들을 S_j 에 복제한다. 여기서 K는 미리 정해진 상수이다. 이로써, 부하 분산의 효과도 거둘 수 있고, 현재 인기가 있는 콘텐츠들을 분산하기 때문에, 질의에 적응적인 복제 생성이 가능하다. 만일 $l_i^{low} < l_i \leq l_i^{hi}$인 경우에는 과부하의 위험이 있으므로, $l_i - l_j > l_i^{low}$인 경우에 과부하 상태와 마찬가지로 두 노드의 부하의 차에 해당하는 콘텐츠들을 S_j에 복제한다.

LAR 프로토콜에서 생성된 복제본에는 캐시와 실제 복제본 두 가지가 있다. 캐시는 실제 콘텐츠나 복제본에 대한 포인터 역할을 하며, 해당 콘텐츠를 가지고 있는 노드의 주소, 콘텐츠의 이름 등의 정보를 가지고 있다. 하나의 콘텐츠에 대하여 여러 개의 캐시가 존재할 수 있으며, 캐시가 많을 경우 LRU(Least Recently Used) 알고리즘에 의해 교체된다. 복제본은 복제된 실제 데이터를 의미하며 다른 노드의 질의에 응답하고 데이터를 전송해 줄 수 있다.

LAR에서는 질의를 라우팅 할 때 현재 노드의 캐시 정보를 참고하여, 캐시 정보가 존재하면 해당 콘텐츠의 캐시들 중 하나를 임의로 선택한다. 그리고 선택된 캐시가 가리키는 노드로 질의를 라우팅한다. 하지만 LAR에서는 잘못된 캐시에 대한 처리를 하지 않기 때문에, 잘못된 캐시가 선택된 경우, 해당 질의는 현재 네트워크상에 존재하지 않는 노드로 라우팅 될 수 있다. 동적인 네트워크상에서는 노드의 출입이 잦기 때문에, 잘못된 캐시가 발생할 확률이 높으므로 이 방법은 동적인 네트워크상에서 적용하기 힘들다.

복제본이 S_j에 생성이 되면, LAR 프로토콜은 S_i와 S_j 사이의 경로에 캐시를 생성하고, 복제본을 가리키도록 한다. 따라서 해당 경로를 지나는 질의는 S_j로 라우팅 되고, 결과적으로 S_i의 부하를 분산시킬 수 있다.

노드에 캐시가 복제되면, 해당 노드는 메시지가 생성될 때마다 노드가 소유하고 있는 여러 캐시들 중 하나를 임의로 선택하여 네트워크상에 전송한다. 메시지는 다른 노드들 사이에 전달되면서, 자신이 피기-배킹하고 있는 캐시를 그 노드에 추가하게 되며, 캐시가 많을 경우에는 위에서 언급한 LRU 방법으로 교체하게 된다.

앞에서 제시한 바와 같이 LAR 프로토콜은 잘못된 캐시

에 대한 처리를 하지 않기 때문에 동적인 네트워크 상황에서 적용하기 힘들다. 또한 전송되는 캐시는 네트워크상에 임의의 노드에서 캐시로 쓰이기 때문에, 잘못된 캐시의 수는 더욱 많아질 수밖에 없고, 만약 이를 다 처리해야 한다면, 매우 큰 비용이 요구된다. 또한, 캐시를 질의율을 고려하지 않고 임의로 네트워크상에 여러 곳에 전파하기 때문에 복제의 효율성이 떨어진다.

3. 효율적 복제 기법

본 장에서는 제안하는 효율적 복제 기법에 대해 설명하고자 한다. 먼저 효율적 복제 기법의 설계 목표를 살펴보고, 효율적 복제 기법을 적용할 네트워크 모델을 제시한 뒤, 효율적 복제 알고리즘에 대해 설명하도록 하겠다.

3.1 설계 목표

본 논문에서 제안하는 복제 기법은 크게 3가지의 설계 목표를 지닌다.

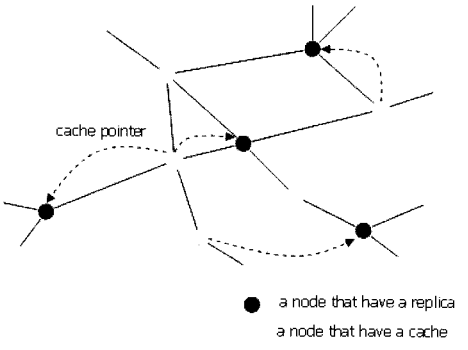
첫째, 복제 기법은 복제에 소요되는 비용을 최소화 하여야 한다. 모든 콘텐츠를 모든 노드에 복제하는 것이 가장 이상적인 상태이나, 이는 현실적으로 불가능하다. 또한 많은 통신비용을 유발시키는 복제 방법 또한 현실적으로 적용하기 힘들다. 따라서 이 효율적 복제 기법은 복제를 최소화하여, 실제적으로 적용될 수 있도록 해야 한다.

둘째, 복제 기법은 효율적이어야 한다. 기본적으로 복제에는 많은 통신비용이 든다. 그리고 이러한 통신비용은 앞서 언급한 바와 같이 사용자를 피어-투-피어 시스템에서 빨리 나가게 만드는 요인이 되고 있다. 따라서 복제 기법에서는 쓸모없는 복제를 최소화하여, 통신비용을 줄여야 한다. 복제를 최소화 하면서 빠른 검색 시간을 유지하려면 복제본을 사용자의 질의가 있는 곳에 배치하여 효율성을 높여야 한다.

셋째, 복제 기법은 지역 정보만을 이용하여 복제 기법에 필요한 각종 결정을 내려야 한다. 분산 알고리즘의 확장성을 높이기 위해서는 되도록 전체 정보를 다루는 일이 없어야 한다. 전체 정보를 수집하는 것은 많은 비용이 소요될 뿐 아니라, 오랜 시간을 필요로 하기 때문에 수집된 정보는 이미 현재 네트워크상의 실제 정보와 큰 차이를 보이게 된다.

3.2 모델

본 논문에서는 피어-투-피어 네트워크의 토폴로지를 파워-로 토폴로지로 가정하였다. 이는 앞서 언급한 바와 같이 비구조적인 피어-투-피어 네트워크가 대부분 파워-로 토폴로지를 형성하기 때문이다. 본 알고리즘에서는 각 노드는 다른 노드와 연결을 맺음으로써 메시지를 전달 할 수 있다. 연결은 해당 노드에 대한 주소 정보를 가지고 있고, 서로 연결을 허용하여 메시지를 주고받을 수 있는 상태를 의미한다. 연결을 맺은 노드를 이웃이라 하고, 질의에 대한 검색은 이웃 노드를 통해 전달된다.



(그림 3) 캐시와 복제본

아울러, 복제된 데이터는 (그림 3)과 같이 캐시와 복제본의 두 가지로 나뉜다. 캐시는 실제 데이터에 대한 포인터의 역할을 한다. 즉 데이터를 직접 복제하는 것이 아니고, 실제 데이터에 대한 주소, 크기 등을 저장함으로써 질의 메시지가 쉽게 실제 데이터로 찾아갈 수 있도록 해준다. 반면에 복제본은 실제 데이터를 나타낸다. 따라서 사용자의 전송 요청이 있으면 해당 데이터를 전송해 줄 수 있다.

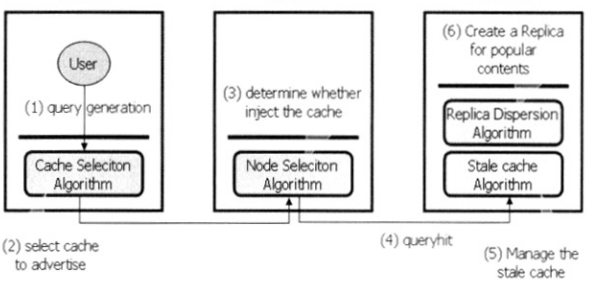
3.3 알고리즘

3.3.1 개요

앞서 제시한 설계 목표를 달성하기 위한 효율적 복제 알고리즘은 (그림 4)와 같이 크게 다음의 세 가지 알고리즘으로 구성된다.

- 캐싱을 위한 알고리즘
- 잘못된 캐시(stale cache)를 처리하기 위한 알고리즘
- 복제 분산 알고리즘

효율적 복제 알고리즘에서는 기본적으로 데이터에 대한 캐시를 이웃 노드의 수가 많은 노드에 전파한다. 이 방법의 이점은 다음과 같다. 먼저 직접 복제본을 옮기지 않고 캐시를 옮김으로써 복제에 대한 비용을 줄일 수 있다. 또한 캐시를 이웃 노드의 수가 많은 노드에 전파함으로써 검색 시 검색율을 높인다. 또한 캐시들을 소수의 장소에 모아둠으로써, 잘못된 캐시에 대한 처리 비용을 줄이고, 처리를 쉽게 만들어서 네트워크상에 존재하는 잘못된 캐시의 수를 줄일 수 있다.



(그림 4) 효율적 복제 알고리즘의 구성

캐싱을 위한 알고리즘은 크게 캐시 선택 알고리즘과 노드 선택 알고리즘으로 나눌 수 있다. 본 알고리즘에서는 전파 비용을 줄이기 위해 캐시를 이미 생성된 메시지에 피기-배킹하여 전파한다. 여기에서, 캐시 선택 알고리즘은 질의 메시지가 생성되었을 때 전파할 캐시를 선택하는 방법을 제시한다. 노드 선택 알고리즘은 메시지가 다른 노드로 전달되었을 때, 지역적인 정보만으로 해당 노드에 피기-배킹한 캐시를 전파할 것인지 결정한다.

잘못된 캐시를 처리하기 위한 알고리즘은 자신이 소유하고 있는 캐시가 잘못된 정보를 가지고 있지 않도록 캐시들을 관리한다. 동적인 네트워크 상황에서는 노드들의 출입이 잦기 때문에, 캐시가 존재하지 않은 노드를 가리킬 수 있다. 이런 경우를 잘못된 캐시라고 하는데, 이는 캐시를 통한 질의 메시지의 라우팅 시 검색율을 낮추는 요인이 된다. 따라서 잘못된 캐시를 처리하기 위한 알고리즘을 통해 질의에 대한 검색율을 높일 수 있다.

복제 분산 알고리즘은 인기 있는 로컬 캐시에 대하여 실제 복제본을 생성한다. 이 때 실제 복제본은 캐시를 소유하고 있는 노드의 요청에 의하여 생성되며, 해당 노드의 이웃 노드에 위치하게 된다. 이 방법이 얻는 이점은 다음과 같다. 보통 연결수가 많은 노드일수록 캐시를 많이 소유할 것이고, 질의 메시지도 많이 전달 받음으로써 인기가 많은 캐시가 발생할 확률이 높다. 따라서 캐시에 대한 복제본을 해당 노드가 직접 소유하는 것은 과부하를 유발하기가 쉽다. 하지만 복제본을 해당 노드의 이웃에 위치시키면, 부하를 분산시키며, 잘못된 캐시를 처리하기 위한 비용도 적게 소요되는 효과가 있다. 또한, 캐시를 소유하고 있는 노드의 요청에 의해 복제본이 생성됨으로써, 지역적인 질의의 인기도에 적응적으로 복제본을 생성할 수 있다.

3.3.2 캐싱을 위한 알고리즘

캐싱을 위한 알고리즘 중 캐시 선택 알고리즘은 다음과 같다. 먼저 과거 t초 동안 콘텐츠들에 대한 요청이 있었던 경우, 해당 콘텐츠들에 한해서 (식 2)와 같은 확률로 캐시를 만들 콘텐츠를 구한다.

$$P(select(i)) = q_i \qquad \text{(식 2)}$$

여기서 q_i는 특정 콘텐츠의 인기도로, $\Sigma q_i = 1$이 성립한다. 이 방법은 캐시의 전체적인 생성율을 줄일 뿐만 아니라, 최근의 인기도에 따라 캐시를 생성하기 때문에, 사용자의 질의에 적응적으로 대처할 수 있다.

아울러 캐싱할 노드 선택 알고리즘은 검색 동안에 N개의 노드를 차례로 방문하면서 이웃 노드의 수가 많은 노드라 생각되는 곳에 캐싱한다. 이 문제는 인터뷰 문제로 잘 알려져 있는데, 기존의 문제와 다른 점은 N이 가변적이라는 점이다. 따라서 인터뷰 문제의 해결 방법을 적용하기 위해서는 N을 예측하는 기법이 필요한데, 여기에서는 가중치 평균을 통해 N을 추정하였다. 가중치 평균에 대한 계산은 해당 노드에서 생성한 질의 메시지에 대한 검색이 종료되면 이루어진다. 가중치 평균을 통해 구해진 N을 N_{avg} 라고 하고, 현재 질의 메시지의 검색에 걸린 홉 수를 N_{new} 라고 할 때, 새로 측정한 노드수와 기존 예측치를 같은 비중으로 새로운 예측치에 반영하기 위하여 N_{avg} 는 (식 3)과 같이 다시 계산된다.

$$N_{avg} = \frac{1}{2} \cdot N_{avg} + \frac{1}{2} \cdot N_{new} \qquad \text{(식 3)}$$

캐싱할 노드를 선택하기 위해서 메시지가 k 홉 동안 현재 네트워크상의 노드들을 돌아다니면서, 그들의 이웃 노드의 수를 관찰한다. 여기에서 k 는 (식 4)와 같다.

$$k = \max_{t \geq 1} \left\{ \left(\frac{1}{t} + \frac{1}{t+1} + \cdots + \frac{1}{N_{avg}} \right) \right\} < 1 \qquad \text{(식 4)}$$

생성된 각 메시지는 k 홉 동안 노드들을 돌아다니면서, 가장 많은 이웃 노드 수 C_{max} 를 저장한다. k 홉 이후에 도착한 노드의 이웃 노드 수가 C_{max} 보다 크다면 (식 5)의 확률로 그 노드에 캐시를 저장하게 된다.

$$P(inject\,cache) = \min\left(\frac{t}{N_{avg}}, 1\right) \qquad \text{(식 5)}$$

3.3.3 잘못된 캐시를 처리하기 위한 알고리즘

피어-투-피어 네트워크에서는 노드의 입출이 잦기 때문에, 잘못된 캐시를 처리하기 위한 알고리즘이 필요하다. 하지만 캐시의 무결성을 유지하기 위하여, 캐시가 가리키고 있는 복제본을 소유한 노드를 계속적으로 모니터링 하는 것은 매우 큰 비용을 요구한다. 따라서 확장성 있는 알고리즘을 위해서는 지역적인 정보를 이용하여 잘못된 캐시를 처리하여야 한다. 이를 위하여 여기에서는 캐시에 대한 생존 시간을 설정하는 방법을 사용한다.

캐시 정보가 캐싱되면 해당 캐시 i에는 생존 시간 $T_{life}(i)$가 설정된다. 생존 시간은 시간이 지남에 따라 감소되며, 최종적으로 0이 되면 해당 캐시는 캐시 테이블에서 삭제된다. 하지만 캐시가 가리키고 있는 콘텐츠가 바로 이웃에 존재하여 네트워크 내에서 활동 중이거나, 혹은 해당 콘텐츠를 소유한 노드로부터 메시지가 전달되어 왔다면, 캐시의 생존 시간을 연장시킨다. 그리고 자신의 이웃 중 한 노드가 네트워크를 떠나게 되면, 캐시 테이블로부터 해당 노드와 관련된 모든 캐시들을 삭제한다.

3.3.4 복제 분산 알고리즘

앞에서 언급한 바와 같이, 이웃 노드의 수가 많은 노드에 콘텐츠를 복사하는 것은 해당 노드를 과부하로 만들기가 쉽다. 따라서 본 알고리즘에서는 이웃 노드 수가 많은 노드에는 캐쉬를 저장하고 그 이웃 노드 중에 하나를 선택하여 복제본을 저장한다. q_i가 이웃 노드 i로부터 요청이 전달된 비

율이고, s_i가 이웃 i 의 현재 용량이라고 할 때, i에 복제본이 생성될 확률은 (식 6)과 같다.

$$P(replicate_i) = \frac{q_i}{q_{total}} \times \frac{s_{total}/s_i}{\sum_j (s_{total}/s_j)} \quad (식\ 6)$$

복제본 생성 확률은 요청 비율 q_i 에 비례하고, 노드의 현재 용량 s_i 에 반비례함을 알 수 있다. 즉, 이웃 노드 중 한 노드로부터 그 복제할 데이터를 검색하는 요청이 많이 전달된다면, 그 이웃 노드에 복제본이 생성될 확률이 높다. 또한 한 이웃 노드가 많은 양의 데이터를 가지고 있다면 그 노드에 복제본을 생성하는 것은 피해야 한다.

4. 성능 평가

본 장에서는 앞서 제안한 효율적 복제 알고리즘의 성능을 측정하였다. 이를 위하여 비구조적 피어-투-피어 시뮬레이터로 잘 알려진 Neurogrid 시뮬레이터[11]를 확장하여 사용하였다.

복제 기법들의 소요 비용과 적중률은 복제 기법의 효용성을 측정하는데 좋은 측정 기준이 된다. 최적의 복제 기법은 모든 콘텐츠를 모든 노드에 복제하는 것이다. 하지만, 이러한 방법은 소요 비용이 많아 현실적으로 구현 불가능하기 때문에, 적절한 비용을 가지고 효율적인 성능을 내야 할 것이다.

따라서 본 장에서는 알고리즘의 효율성을 실험하기 위하여, 각각의 알고리즘을 유지하기 위한 비용을 측정하였다. 알고리즘의 성능을 평가하기 위하여, 평균 검색 시간, 성공률 등을 측정하였으며, 알고리즘의 유지비용은 복제 발생 횟수와 복제 적중률을 통해 측정하였다.

본 논문에서는 앞서 소개한 복제 기법인 한 단계 복제 기법(One hop Replication, 이하 OR), 경로 복제 기법(Path Replication, 이하 PR), 경로 임의 복제 기법(Path Random Replication, 이하 PRR), LAR 등을 직접 모델링하였고, 아무런 복제 기법을 쓰지 않은 임의 경로 검색(Random Walk, 이하 RW)도 모델링하여 본 논문에서 제안하는 비구조적 피어-투-피어 네트워크에서의 효율적인 복제기법 (Efficient Replication in Unstructured Peer-to-peer networks, 이하 ERUP)과 비교 분석하였다.

4.1 시뮬레이션 모델

시뮬레이션을 위한 복제 기법 모델은 하드웨어 사양이 같은 동질의 서버로 측정하였다. 해당 서버의 하드웨어는 AMD Athlon(tm) XP 1600+ CPU, PC133 타입의 512MB 메모리, 그리고 ATA 133 규격의 7200 RPM을 지원하는 하드 디스크로 구성되었다. 그리고 윈도우즈 XP 운영체제를 OS로 사용하였으며, Java 1.4.2, Neurogrid 시뮬레이터 0.2.1 버전 하에서 시뮬레이션 하였다.

복제 기법을 테스트하기 위하여 네트워크의 토폴로지는

〈표 1〉 시뮬레이션 파라미터

Topology	Power-Law Topology
alpha (α)	2.3
# of node	2000
Searching Method	Random Walk
# of messages per query	8
max_ttl	32
Contents Distribution	Uniform Distribution
# of contents per node	10
# of items	200
# of contents in network	20000
max # of contents per node	50
Query Distribution	Zipf like
alpha (α)	1.2
avg # of queries per 1 sec.	500
Cache Replacement	LRU
# of max cache item	32
# of max cache bin	50

비구조적인 피어-투-피어 네트워크의 대표적 토폴로지인 파워-로 토폴로지로 구성하였다. 파워-로 토폴로지는 노드의 이웃 노드의 수에 대한 분포가 파워-로 분포를 따르는 토폴로지로 w 가 상수일 때, i 번째로 많은 연결수를 갖는 노드는 w / i^{a} 개의 이웃을 갖는다. 한편 질의율의 분포는 연구[12]를 통해 잘 알려진 Zipf-like 분포[13]를 사용하였다. Zipf-like 분포는 i 번째의 질의율이 q_i 라고 할 때, q_i 는 $1 / i^{a}$ 에 비례한다. 그리고 피어-투-피어의 콘텐츠 검색 방법으로는 복제 기법의 효율성을 측정하기 위하여 앞서 제시한 임의 경로 검색 방법을 사용하였다. 기본적인 시뮬레이션은 <표 1>의 파라미터를 사용하였다. 한편 시뮬레이션에서는 노드가 소유할 수 있는 최대 콘텐츠의 수와 최대 캐시의 수를 제한하였다. 이는 현실적인 실험을 위하여 제한한 것으로, 뒤에, 이들 파라미터가 성능 상에 미치는 영향을 제시하도록 하겠다.

본 시뮬레이션에서는 복제 알고리즘의 적응도와 성능, 효율성 등을 측정하기 위하여 다음과 같은 측정 기준을 사용하였다.

• 검색 성공률 (Query Success Rate)

검색 성공률은 알고리즘의 성능과 적응성을 평가하기 위한 측정 기준으로, 생성된 전체 질의 중 성공한 질의의 비율이다. 검색 성공률은 (식 7)과 같이 구할 수 있다. 만일 복제본이 네트워크상에 효율적으로 배치된다면, 임의 경로 검색 방법의 성공률이 높아질 것이다.

$$S = \frac{\#\ of\ success\ query}{\#\ of\ total\ query} \quad (식\ 7)$$

• 평균 검색 거리(Average Searching Hops)

평균 검색 거리는 알고리즘의 성능을 평가하기 위한 측정

기준으로, 검색에 소요된 평균 홉 수로 구한다. 평균 검색 거리를 구하기 위해 여기에서는 실패한 검색 길이를 최대 검색 길이로 설정하고 평균을 구하였다. 만약 성공한 검색에 대해서만 평균 검색 거리를 구한다면, 평균을 구하기 위한 표본 집단의 수가 틀려지게 되고, 결국 이는 공정한 비교가 될 수 없다. 평균 검색 길이는 (식 8)과 같이 구할 수 있다.

$$D_{avg} = \frac{\sum hops\ of\ first\ success}{\#\ of\ query} \qquad (식\ 8)$$

• 복제 적중률(Replica Hit Ratio)

복제 적중률은 복제 알고리즘의 효율성을 평가하기 위한 측정 기준으로, 검색에 적중한 복제 데이터의 수를 전체 복제 데이터의 수로 나누어 구한다. 여기에서 복제 데이터란 복제본과 캐시 모두를 가리킨다. 만일 복제 적중률이 낮다면, 그만큼 사용되지 않는 복제 데이터의 비율이 높다는 의미이므로 알고리즘의 효율성이 떨어진다고 평가할 수 있다. 복제 적중률은 (식 9)와 같이 구할 수 있다.

$$Hit\ Ratio = \frac{\#\ of\ replication}{\#\ of\ total\ replication} \qquad (식\ 9)$$

4.2 성능 측정 및 분석

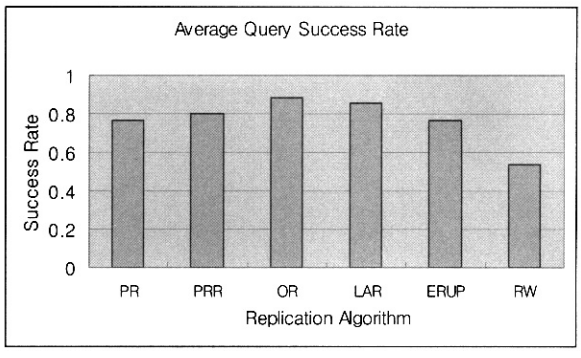
본 성능 측정에서는 앞서 제시한 측정 기준에 따라 각 알고리즘의 성능을 측정하였다. 먼저 노드의 출입이 없고, 질의율이 고정된 상황에서 각 알고리즘의 성능을 측정하였다. 그리고 일정 시간 마다 노드를 출입 시키면서, 각 노드의 출입이 각 알고리즘에 미치는 영향을 측정하였다.

복제 알고리즘의 정확한 성능 측정을 위해서는 어느 정도 복제 알고리즘이 실행되어 복제본이 생성된 후 각 성능을 측정해야 한다. 따라서 본 실험에서는 측정을 위해 60회의 검색 프로세스 실행 이후의 값들에 대하여 평균을 측정하였다. 1회의 검색 프로세스는 각 노드들이 자신에게 들어온 질의 메시지들에 해당하는 콘텐츠가 있는지 확인하고, 복제 알고리즘을 수행한 뒤, 다른 노드들에게 전달하는 과정을 포함한다.

4.2.1 검색 성공률

각 알고리즘 별로 검색 성공률을 측정한 결과는 (그림 5)와 같은데, 복제가 검색의 성공률에 높은 영향을 미치는 것을 알 수 있다. 즉, 아무런 복제 알고리즘을 사용하지 않은 Gnutella의 경우 평균 0.53 정도의 낮은 성공률을 보이지만, 복제 알고리즘을 적용한 경우는 0.7에서 0.9 사이의 높은 성공률을 보였다. 이는 복제가 임의 경로 검색 방법에서 꼭 필요하다는 것을 나타낸다.

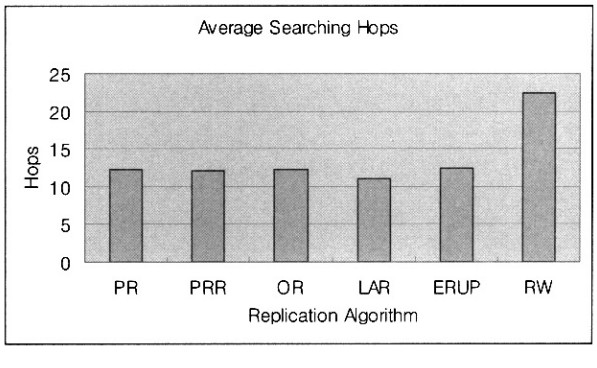
복제 알고리즘 중에서도 한 단계 복제 방법이 가장 좋은 검색 성공률을 보였는데, 이는 가장 많은 복제본(또는 인덱스)을 생성하므로 당연한 결과로 생각된다. LAR 프로토콜은 프로세스가 진행될수록 점차 나은 결과를 보였는데, 이는 프로세스가 진행되면서, 좀 더 많은 캐시를 네트워크상에 전파하기 때문이다. 한편 본 논문에서 제시하는 효율적 알고리즘은 한 단계 복제 방법에 비해 매우 적은 통신비용을 사용하면서 검색 성공률은 대략 13% 정도 성능 저하를 보였다. 하지만 그 외의 다른 알고리즘과는 비슷한 검색 성공률을 보임으로써, 복제의 비용을 줄임으로 인한 손실을 최소화하였음을 알 수 있다.



(그림 5) 복제 알고리즘들의 평균 검색 성공률

4.2.2 평균 검색 거리

각 알고리즘 별로 평균 검색 거리를 측정한 결과는 (그림 6)과 같다. 복제 알고리즘을 사용하지 않은 RW의 경우 평균 22에 가까운 매우 긴 검색 거리를 보였다. 하지만 나머지 알고리즘들은 비슷한 성능을 보였다. 이 실험 결과에서는 나타나지 않지만, 이 중 LAR 프로토콜이 프로세스를 진행하면서 점차로 검색 거리를 향상 시켰는데, 이는 캐시로 인한 라우팅의 효과로, 캐시를 네트워크의 여러 곳에 전파함으로써 보다 검색 거리를 짧게 하였다. 본 논문에서 제시한 효율적 복제 알고리즘 또한 처음에는 복제 알고리즘들 중 가장 나쁜 성능을 보이다가, 프로세스가 진행 되면서 비슷한 성능을 보이게 되었다. 따라서 여기에서도, 본 알고리즘이 다른 알고리즘들에 비해 성능이 크게 나빠지지 않았음을 알 수 있다.

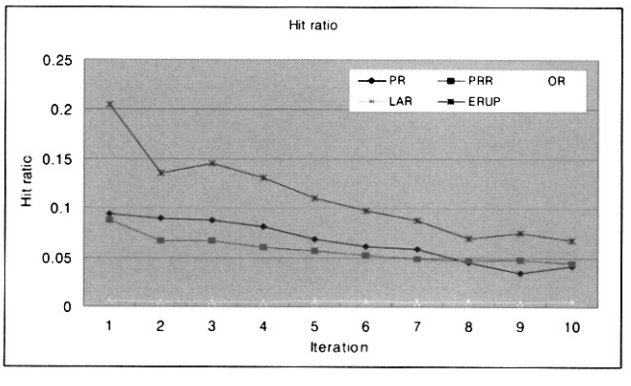


(그림 6) 복제 알고리즘들의 평균 검색 거리

4.2.3 복제 적중률

복제 적중률의 측정에서는 복제 알고리즘을 사용하지 않는 Gnutella는 실험에서 제외하였다. 그 결과는 (그림 7)과 같은데, 그림에서 보는 바와 같이 제안하는 효율적 복제 알고리즘(ERUP)이 복제 적중률이 가장 높게 나타났다. 이는 복제본과 캐시의 생성 수를 최소화하고, 질의 메시지가 잘 도달하는 곳에 위치시킴으로써 나타난 결과라고 할 수 있다. 반면에, 많은 수의 복제본을 생성하는 한 단계 복제 방법과 메시지를 생성 시킬 때마다 캐시를 피기-배킹하여 전파하는 LAR 알고리즘은 복제 적중률 면에서 나쁜 성능을 나타냈다. 이는 해당 알고리즘들이 효율적으로 복제를 생성하지 않고 임의로 복제를 많이 생성하여 성능을 높였기 때문이다.

전반적으로 시간이 지남에 따라 복제 적중률이 낮아지는 이유는 생성되는 복제본은 많아지는데 비하여, 측정 시간 동안 이루어지는 검색 비율은 일정하기 때문이다.



(그림 7) 복제 알고리즘들의 복제 적중률

4.2.4 소요 비용 측정

각 알고리즘에서 복제에 사용한 소요 비용을 측정한 결과는 <표 2>와 같다. 복제에 사용한 비용으로 복제본의 생성 개수와 캐시의 생성 개수를 측정하였으며 복제본의 생성 개수가 많을수록 복제에 사용되는 비용이 커짐을 알 수 있다

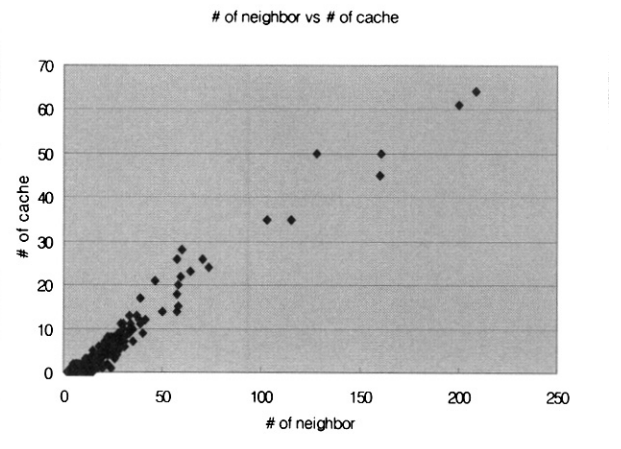
측정 결과 경로 복제 알고리즘과 경로 임의 복제 알고리즘은 비슷한 수의 복제본을 생성하였으며, 한 단계 복제 알고리즘이 가장 많은 수의 복제본을 생성하여 가장 소요 비용이 컸음을 알 수 있다. LAR은 생성 복제본 수는 적으나 캐시 생성 개수가 많았고, 본 논문에서 제시한 알고리즘은 가장 적은 수의 복제본을 생성하였으며, 캐시의 개수 또한 LAR보다 많이 줄였다.

〈표 2〉 복제 소요 비용

알고리즘	복제본 생성 수	캐시 생성 수
PR	7170	0
PRR	7482	0
OR	108554	0
LAR	1135	267912
ERUP	95	3528

4.2.5 이웃한 노드의 수 대비 캐시의 개수

이번 실험에서는 이웃한 노드의 수 대비 캐시의 개수를 측정하였다. 본 알고리즘에서는 로컬 정보만을 이용해 허브에 많은 캐시를 가지도록 설계하였다. 따라서 이웃 노드의 수가 많을수록 많은 캐시를 가져야 할 것이다. (그림 8)은 100회의 검색 프로세스 이후 각 노드에 대한 이웃한 노드의 수 대비 캐시의 개수를 측정한 결과이다. 실험 결과를 보면 본 알고리즘에서 의도한 바와 같이 이웃의 노드 수가 많을수록 많은 양의 캐시를 가지고 있음을 볼 수 있다. 한편 많은 양의 캐시를 가지기 위해서는 많은 메시지가 방문하여야 하고, 노드 선택 알고리즘에 합당한 이웃한 노드의 수를 가지고 있어야 한다. 따라서 이 실험 결과에선 파워-로 토폴로지 상에서 이웃한 노드의 수가 많은 노드에 메시지가 끌리는 현상도 간접적으로 추론할 수 있다.

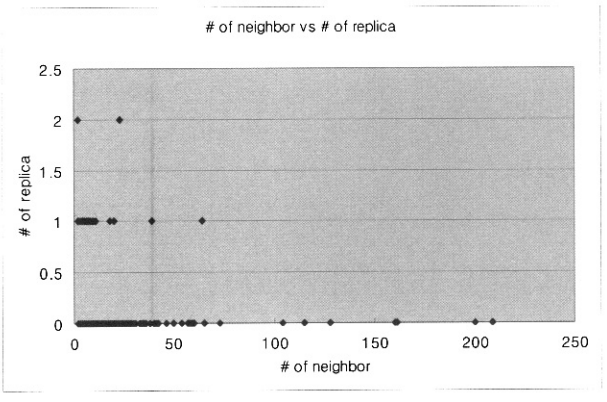


(그림 8) 이웃한 노드의 수 대비 캐시의 개수

4.2.6 이웃한 노드의 수 대비 복제본의 개수

앞서 언급한 바와 같이 파워-로 토폴로지 상에서 허브에게 많은 복제본을 생성한다면, 허브는 그 특성 상 과부하 상태에 빠지기 쉽다. 따라서 본 알고리즘에서는 복제 분산 알고리즘을 통해 이웃한 노드의 수가 적은 노드에 복제본을 생성하도록 하였다. 따라서 본 실험에서는 복제 분산 알고리즘이 복제 생성에 미치는 영향을 검증하기 위하여 이웃한 노드의 수 대비 복제본의 개수를 측정하였다. 본 알고리즘에서 의도한 바에 의하면 이웃한 노드의 수가 적을수록 복제본을 많이 가지고 있어야 하며, 복제본의 수는 되도록 특정 노드의 집중되는 현상이 없이 균형 있게 분산되어야 한다.

이웃한 노드의 수 대비 복제본의 개수를 측정한 결과는 (그림 9)와 같다. 본 알고리즘에서 의도한 바와 같이 이웃한 노드의 수가 많은 노드들은 복제본을 가지고 있지 않고, 이웃한 노드의 수가 적은 노드들에게 복제본이 생성되어 있음을 알 수 있다. 또한 생성된 복제본도 대체적으로 한 노드에 집중되지 않고 균등하게 분산되어 있음을 볼 수 있다.



(그림 9) 이웃한 노드의 수 대비 복제본의 개수

5. 결론 및 향후 과제

기존의 복제 알고리즘은 토폴로지에 대한 정보 없이 복제본의 생성 방법이나 생성 개수에 초점을 맞추어 연구가 진행되어 왔다. 따라서 네트워크 내의 임의의 노드에 복제하는 경우가 많았고, 이는 복제의 효율을 떨어뜨리는 결과를 가져오게 되었다.

따라서 본 논문에서는 비구조적인 피어-투-피어 시스템의 토폴로지로 잘 알려져 있는 파워-로 토폴로지에 알맞은 효율적인 복제 알고리즘을 제시하였다. 복제본 생성을 최소화하고 성능 향상을 위하여 무작위적인 캐시의 전파 대신에 질의율에 따른 캐시 전파를 선택하였다. 또한 이러한 캐시를 허브 상에 위치시켜 검색율의 저하를 막았으며, 이를 위해 지역적인 정보를 가지고 허브를 선택하는 알고리즘을 제시하였다. 그리고 허브의 부하 집중으로 인한 과부하 현상을 막기 위하여 허브의 주변에 복제본을 생성하는 방법을 사용하였다.

본 논문에서는 제안한 알고리즘의 성능을 측정하기 위하여 검색 프로세스를 진행시키면서 검색 성공률과 평균 검색 거리, 검색 적중률을 비롯하여 소요 비용을 측정하였다. 실험 결과 검색 성공률과 평균 검색 거리는 다른 알고리즘들과 비슷한 성능을 보임과 동시에, 검색 적중률은 100% 가까이 향상 시켰다. 또한 소요 비용 측정 결과 복제에 가장 적은 비용을 소요함을 알 수 있었다.

하지만 본 논문에서는 파워-로 토폴로지를 가정하고, 허브 상에 캐시들을 위치시켰기 때문에, 허브의 잦은 출입은 성능 상에 큰 영향을 미친다. 따라서 추후 노드들의 출입이 잦은 동적인 환경에서 허브의 출입으로 인한 성능 저하를 완화하기 위한 연구가 필요하다.

참 고 문 헌

[1] James F. Kurose and Keith. "Computer Networking: A top-down approach featuring the internet" Addison Wesley.

[2] Edith Cohen and Scott Shenker. "Replication Strategies in Unstructured Peer-to-Peer Networks." Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications, Vol.32, Issue 4, pp.177-190, August, 2002.

[3] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham and Scott Shenker. "Making Gnutella-like P2P Systems Scalable." Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp.407-418, August, 2003.

[4] Qin Lv, Pei Cao, Edith Cohen, Kai Li and Scott Shenker. "Search and Replication in Unstructured Peer-to-Peer Networks." Proc of SIGMETRICS 2002, pp.258-259, June, 2002.

[5] Saroiu, S., Gummadi, P. K., and Gribble, S. D. A. "Measurement Study of Peer-to-Peer File Sharing Systems." Proc of Multimedia Computing and Networking, 2002.

[6] Gnutella Hompage. http://www.gnutella.com/

[7] Matei Ripeanu, adriana Iamnitchi and Ian Foster. "Mapping the Gnutella Network." Internet Computing, IEEE, Vol.6, Issue 1, pp.50-57, January-February, 2002.

[8] The Gnutella Protocol Specification v0.4, http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

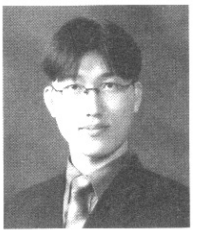
[9] Freenet Homepage, http://freenet.sourceforge.net

[10] Vijay Gopalakrishnan, Bujor Silaghi, Bobby Bhattacharjee and Pete Keleher. "Adaptive Replication in Peer-to-Peer Systems." Proceedings of the 24th International Conference on Distributed Computing Systems, pp.360-369, March, 2004.

[11] Sam Joseph. "An Extensible Open Source P2P Simulator." P2P Journal. http://p2pjournal.com/

[12] Kunwadee Sripanidkulchai. "The popularity of Gnutella queries and its implications on scalability." http://www-2.cs.cmu.edu/~kunwadee/research/p2p/peer.html

[13] Reference on Zipf's law. http://linkage.rockefeller.edu/wli/zipf/



최 우 락

e-mail : lazylune@gmail.com
2003년 서강대학교 컴퓨터학과(공학사)
2005년 서강대학교 컴퓨터학과(공학석사)
2005년~현재 퓨처시스템 VPN팀 연구원
관심분야 : 피어투피어 컴퓨팅, 분산처리 시스템

한 세 영

e-mail : syhan@sogang.ac.kr
1991년 포항공과대학교 수학과(이학사)
2003년 서강대학교 정보통신대학원 (공학석사)
2004년~현재 서강대학교 컴퓨터학과 박사과정 재학중
1996년~현재 (주)이엔지 기술본부
관심분야 : 피어투피어 컴퓨팅, 분산처리 시스템

박 성 용

e-mail : parksy@sogang.ac.kr
1987년 서강대학교 컴퓨터학과(공학사)
1994년 미국 Syracuse University 대학원 (공학석사)
1998년 미국 Syracuse University (공학박사)
1998년~1999년 미국 Bell Communication Research 연구원
1999년~현재 서강대학교 컴퓨터학과 부교수
관심분야 : Autonomic Computing, Peer to Peer Computing, High Performance Cluster Computing and System