피어의 컴퓨팅 능력을 고려한 인터넷 파일 시스템을 위한 확장성 있는 자원 탐색 프로토콜 설계

(A Scalable Resource-Lookup Protocol for Internet File System Considering the Computing Power of a Peer)

정 일 동 † 유 영 호 †† 이 종 환 ††† 김 경 석 ††††
(Il-dong Jung) (Young-ho You) (Jong-hwan Lee) (Kyongsok Kim)

요 약 인터넷과 PC의 발달은 정보의 분산과 공유를 가속화하여, 사용자 스스로 컴퓨터 자원과 서비스를 개인의 컴퓨터 사이에 서로 공유하는 P2P (Peer-to-Peer) 컴퓨팅이 등장하였다. 대부분의 P2P 시스템에서 가장 중요한 기능은 효율적으로 데이타를 위치시키고 (location) 탐색하는 것이다. 지금까지 개발된 P2P 시스템은 피어의 성능을 똑같은 것으로 가정한다. 이는 알고리즘의 분석이 쉽기 때문에 학문의 관점에서는 유용하지만, 실제로는 모든 피어가 비슷한 기능을 가지는 것이 아니다. 본 논문에서는 기존의 P2P 시스템에서 무시하고 있는 피어의 컴퓨팅 능력을 고려한 P2P 프로토콜을 제안하고, 또한 제안한 프로토콜의 응용 분야와 활용 가능성을 제시한다. 그리고 시뮬레이션을 통해 메시지의 피어 사이의 라운드 시간과 메시지의 평균 홉 (Hop) 수를 측정하여 프로토콜의 성능을 검증하고, 마지막으로 수식으로 성능을 분석한다. 본 논문에서 제안한 P2P 프로토콜의 이름을 Magic Square 라고 한다. 마방진에서 각각의 숫자는 특별한 의미가 없지만, 그 수를 적절히 배치한 마방진은 어느 방향에서 보아도 같은 합을 가진다. 본 논문에서 제안하는 Magic Square도 각 피어가 가지는 정보는 의미가 없더라도 전체 시스템에서는 의미를 가지고, 어떤 피어에서 질의를 하더라도 비슷한 성능이 나올 수 있도록 설계하였다.

키워드 : 피어-투-피어, 인터넷 컴퓨팅, 인터넷 파일 시스템

Abstract Advances of Internet and PC accelerate distribution and sharing of information, which make P2P (Peer-to-Peer) computing paradigm appear. P2P computing paradigm is the computing paradigm that shares computing resources and services between users directly. A fundamental problem that confronts Peer-to-Peer applications is the efficient location of the node that stores a desired item. P2P Systems treat the majority of their components as equivalent. This purist philosophy is useful from an academic standpoint, since it simplifies algorithmic analysis. In reality, however, some peers are more equal than others. We propose the P2P protocol considering differences of capabilities of computers, which is ignored in previous researches. And we examine the possibility and applications of the protocol . Simulating the Magic Square, we estimate the performances of the protocol with the number of hop and network round time. Finally, we analyze the performance of the protocol with the numerical formula. We call our p2p protocol the Magic Square. Although the numbers that magic square contains have no meaning, the sum of the numbers in magic square is same in each row, column, and main diagonal. The design goals of our p2p protocol are similar query response time and query path length between request peer and response peer, although the network information stored in each peer is not important.

Key words : P2P, Internet Computing, Internet File System

· 이 연구는 부산대 해외 장기 파견에 의하여 일부 지원 받았습니다.
† 비 회 원 : LG전자 DAC연구소 연구원
idjung96@daum.net
†† 비 회 원 : 부산대학교 컴퓨터및정보통신연구소 교수
yhyou@asadal.pusan.edu
††† 비 회 원 : 오토파워 부설연구소 연구원
jhwlee@asadal.pnu.edu
†††† 종신회원 : 부산대학교 전자전기정보컴퓨터공학부 교수
gimgs@asadal.pnu.edu
논문접수 : 2004년 3월 9일
심사완료 : 2004년 9월 14일

1. 서 론

인터넷과 PC의 발달은 정보의 분산과 공유를 가속화 하였으며, 기존 클라이언트-서버 컴퓨팅 환경은 분산 다중 서버 환경으로 바뀌었다. 또한 정보의 원천이 웹과 데이타베이스에서 개인 PC까지 다양해졌고, 검색 포털의 한계, 즉 정보 탐색 범위 제한, 부정확성, 개인들의 검색에 대한 욕구 불만 등의 문제로 사람들은 다른 대안을 찾기 시작했다. 사용자 스스로 컴퓨터 자원과 서비스를 개인의 컴퓨터 사이에 서로 공유하는 P2P(Peer-to-Peer) 컴퓨팅이 등장하였다[1,2].

P2P 컴퓨팅은 인터넷에서 저장 공간, CPU 사이클, 콘텐츠, 사람과 같은 자원을 활용하는 응용 분야이다. 분산된 자원에 접근하는 것은 IP가 고정적이지 않고 접속이 불안정한 환경에서 시스템이 작동하는 것을 의미하며, 각 피어는 충분히 혹은 완전히 분산되어 자율성을 가져야 한다[3]. 최근에는 P2P 컴퓨팅 분야에서 중복 저장(redundant storage), 저장의 영구성(permanence), 인접 피어 선택(selection of nearby servers), 익명성(anonymity), 검색(search), 인증(authentication), 이름 붙이기(naming) 등과 같은 연구가 진행 중이다. 하지만, 이런 여러 가지 특징에도 불구하고, 대부분의 P2P 시스템에서 가장 중요한 기능은 효율적으로 데이타를 위치시키고(location) 탐색하는 것이다[4].

초기 P2P 시스템에서는 자원의 목록과 그 위치를 가지고 있는 서버를 두거나(Napster), 찾는 자원에 대한 질의를 자신이 알고 있는 피어들에게 방송(broadcast) 함으로써 (Gnutella) 자원에 대한 탐색을 실행하였다. Napster와 Gnutella의 방식은 확장 가능성(scalability) 문제를 안고 있다. 그래서 확장 가능성의 문제를 극복하기 위해서 CAN, Chord, Pastry와 같은 구조화된 시스템을 발표하였다. 이러한 구조화된 시스템은 분산 해시 테이블(Distributed Hash Table, DHT)을 사용하여 확장 가능성과 라우팅 성능을 향상하였다[5].

지금까지 개발된 P2P 시스템은 피어의 성능을 똑같은 것으로 가정한다. 이는 알고리즘의 분석이 쉽기 때문에 학문의 관점에서는 유용하다. 하지만, 실제로는 모든 피어가 비슷한 기능을 가지는 것이 아니다. P2P 시스템에 참여하는 컴퓨터의 CPU의 성능과 사용할 수 있는 메모리 혹은 디스크의 공간의 크기, 네트워크의 성능이 다르다. 물리적으로 어떤 컴퓨터는 네트워크의 허브에 위치할 수도 있고, 어떤 컴퓨터는 네트워크의 말단에 위치할 수도 있다[6].

본 논문에서는 기존의 P2P 시스템에서 무시하고 있는 피어의 컴퓨팅 능력을 고려한 P2P 프로토콜을 제안하고, 또한 제안한 프로토콜의 응용 분야와 활용 가능성을 제시한다. 그리고 시뮬레이션을 통해 메시지의 피어 사이의 라운드 시간과 메시지의 평균 홉(Hop) 수를 측정하여 프로토콜의 성능을 검증하고, 마지막으로 수식으로 성능을 분석한다.

본 논문에서 제안한 P2P 프로토콜의 이름을 Magic Square(마방진)라고 한다. 마방진은 가로, 세로, 및 대각선에 있는 각각의 합이 같도록 배열한 것을 말한다. 마방진에서 각각의 숫자는 특별한 의미가 없지만, 그 수를 적절히 배치한 마방진은 어느 방향에서 보아도 같은 합을 가진다. 이처럼 본 논문에서 제안한 Magic Square도 각 피어가 가지는 정보는 의미가 없더라도 전체 시스템에서는 의미를 가지고, 어떤 피어에서 질의를 하더라도 비슷한 성능이 나올 수 있도록 설계하였다.

2장에서는 관련 연구를 살펴보고, 3장에서는 본 논문에서 제안하는 P2P 프로토콜인 Magic Square의 설계한다. 사용자 혹은 피어가 원하는 데이타를 찾기 위해서 분산 해시 테이블을 어떻게 사용하는가를 살펴보고, 피어의 가용 네트워크 능력을 반영하여 시스템을 향상하는 방법과 P2P 프로토콜의 기본 기능이라고 할 수 있는 피어의 접속과 떠남, 결함을 처리하는 방법에 대해서 살펴본다. 4장에서는 시뮬레이션을 통해서 제안한 프로토콜의 성능을 검증한다. 마지막으로 5장에서는 결론과 향후 연구를 제시하고 논문을 마친다.

2. 관련 연구

2.1 Skip list

이진 트리(binary tree)가 특정 상황에서 성능이 떨어지는 문제를 보완하여 균형 트리(balanced tree)가 제시되었다. 균형 트리는 트리의 균형 조건을 유지하기 위해 트리를 재구성하는 연산을 하는데, 트리가 커짐에 따라 트리의 재구성 연산의 시간 비용이 증가한다[7].

스킵 리스트는 랜덤 수 생성기를 사용하여 균형 트리(balanced tree)를 확률적으로 변형한 형태이다. 스킵 리스트가 가장 나쁜 성능을 보이는 경우가 있더라도 이진 트리처럼 입력되는 순서가 성능에 영향을 미치는 경우는 없다. 스킵 리스트의 균형을 이루는 속성은 탐색 트리에 랜덤 삽입(insertion)을 하여 균형을 이루는 속성과 비슷하다[7].

확률적으로 균형을 만드는 것이 명시적으로 균형을 유지하는 연산을 하는 것보다 쉬우며, 알고리즘도 더 간단하다. 스킵 리스트의 알고리즘이 단순하기 때문에 구현하기도 쉽고, 균형 트리보다 속도도 더 빠르다. 또한 스킵 리스트는 공간 효율성이 아주 높다.

스킵리스트는 기존의 연결 리스트 (linked list) 에 임의의 노드마다 몇 개의 연결을 추가한 것으로 그림 1과 같은 모양을 이룬다. 각 노드가 가지는 연결의 수를 레벨(level)이라 하며, 이 레벨은 랜덤하게 생성하기 때문

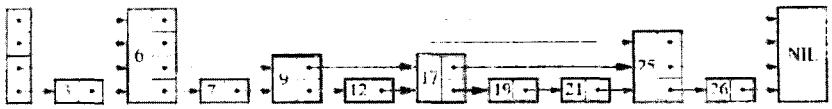


그림 1 스킵리스트의 구조

에 균형을 이루며 임의의 노드를 찾는데 O(log N)의 시간 복잡도(time complexity)를 가진다[7].

Magic Square의 시스템 구조는 스킵 리스트와 비슷하다. 따라서, Magic Square의 각 피어는 스킵 리스트의 노드에 대응되고, Magic Square에서 피어를 찾는 과정은 스킵 리스트에서 데이타를 찾는 과정과 비슷하다.

2.2 P2P(오버레이 네트워크)의 라우팅

오버레이 네트워크(Overlay Network, =P2P 네트워크) 분야에서 사용하는 라우팅의 의미는 일반적인 네트워크 분야에서 사용하는 라우팅의 의미와 달리, 피어가 가진 다른 피어들의 리스트 중에서 목적지와 가장 가깝거나 효율적으로 전달할 것 같은 피어에게 메시지를 전달하는 것을 말한다.

Chord는 손끝 테이블(finger table)을 사용해서 소스 피어와 ID 영역의 거리가 1/2이 되는 피어, 1/4, 1/8, 2의 지수승의 거리에 있는 피어의 IP를 알아낸다. 이 구조는 스킵 리스트 데이타 구조와 비슷하다. 피어는 키 K를 요청하는 질의를 손끝 테이블에서 K를 넘지 않는 가장 큰 ID의 피어(K의 계승자(successor))에 전달한다. 손끝 테이블이 2의 지수 구조로 되어 있기 때문에 메시지를 전달하는 피어와 K까지의 ID 공간의 1/2을 전달할 수 있다. 따라서 질의를 처리하는데 O(log N) 메시지가 필요하다[4]. Chord의 손끝 테이블과 라우팅 경로는 그림 2와 같다.

Pastry의 각 피어 n은 리프 셋(leaf set) L을 유지하는데, L의 절반은 n보다 크면서 가까운 피어의 집합을 유지하고, 나머지 절반은 n보다 작으면서 가까운 피어의 집합을 유지한다. 피어 n이 키 K를 질의하면 n의 리프 셋 중에서 K와 가장 가까운 피어에게 메시지를 전달한다. 이 리프 셋의 정확하면 메시지가 정확하게 전달되기 때문에 시스템의 정확성이 보장된다. 단, 인접한 ID를 가지는 리프 셋의 절반 이상 동시에 에러가 나지 않아야 한다[8].

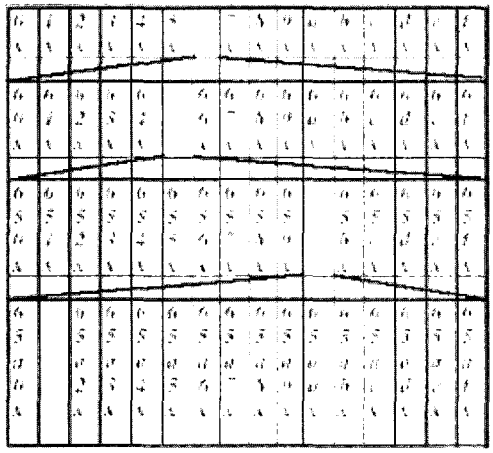
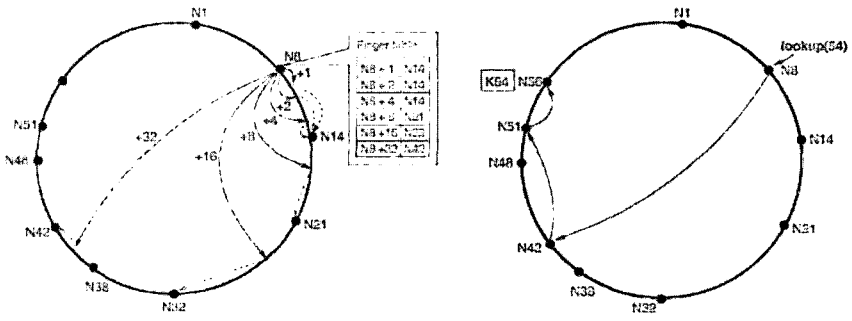


그림 3 Pastry에서 피어 ID가 65a1x인 피어의 라우팅 테이블

CAN(Content-Addressable Network)은 이전에 나온 시스템과 달리 DHT를 구현하기 위해서 다차원 좌표 공간을 사용한다. 좌표 공간은 존(zone)이라고 불리는 하이퍼 네모(Hyper Rectangle)로 분할된다. 각 피어



(a) (b)

그림 2 (a) Chord에서 피어 8의 손끝 테이블 모양 (b) 피어 8에서 K54를 질의했을 경우의 라우팅 경로

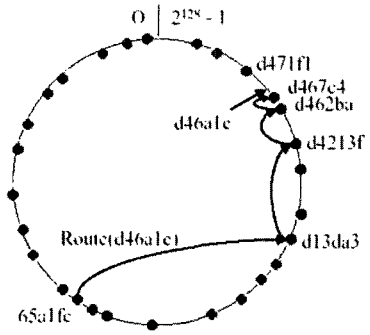


그림 4 Pastry에서 피어 ID가 65a1fc인 피어가 d46a1c의 키를 가지는 데이타를 질의했을 경우의 라우팅 경로

는 자신의 존을 담당하며, 존의 경계로 피어를 구분한다. 키는 좌표 공간의 한 점으로 대응되고, 그 점의 좌표를 포함하는 존을 담당하는 피어에 저장된다[9]. 그림 5는 2차원 영역 [0,1] *[0,1]에 6개의 피어가 존재하는 CAN과 라우팅 경로를 보여준다[10].

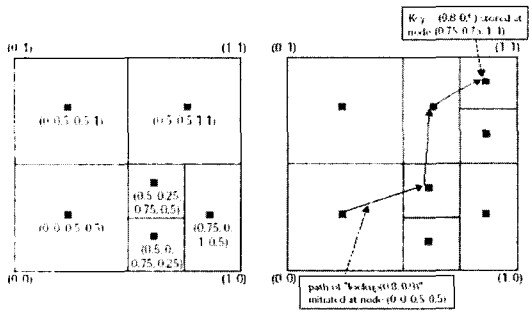


그림 5 (a) 2차원 [0,1]*[0,1]의 CAN에 6개의 피어가 존재함 (b) 키 (0.8, 0.9) 를 찾아가는 경로

위에서 소개한 P2P 시스템은 피어의 컴퓨팅 능력을 고려하지 않고 있다. 하지만, 실제 컴퓨팅 환경에서는 모든 피어의 성능이 다르기 때문에 컴퓨팅 성능의 차이를 고려하여 프로토콜을 설계한다면 저장 공간, CPU 활용, 라우팅 측면에서 더 좋은 성능을 보일 것으로 본다.

구조화된 P2P 시스템(Structured P2P System)은 시스템이 안정적일 때는 자원 탐색 성능이 우수하긴 하지만, 피어가 자주 접속하고 떠나서 시스템이 불안정할 때는 구조를 변경하기 위해서 많은 비용이 든다. 반면에 Magic Square는 스킵 리스트처럼 랜덤 요소를 사용해서 자원을 탐색하기 때문에 별도의 자료 구조를 변경할 필요 없이 새로 접속한 피어의 ID를 추가하거나, 떠난 피어의 ID를 제거하고 다른 피어의 ID를 삽입하면 된다.

2.3 컴퓨팅 능력의 예측

성능을 예측하고 평가하는 것은 분산 컴퓨팅 중, 특히 그리드 컴퓨팅에서 중요한 요소이다. 특히, 사용할 수 있는 자원(CPU, Network bandwidth, Storage capacity 등)의 성능은 효율적인 작업 스케줄링을 구현할 때 사용할 수 있다. 컴퓨팅 상황이 자주 변하기 때문에 스케줄러는 작업을 실행할 때 얻을 수 있는 성능을 예측할 수 있어야 한다. 예측을 기반으로 스케줄러는 작업을 가장 효율적으로 처리할 수 있도록 사용할 수 있는 자원을 조합하여 선택할 수 있어야 한다[11].

그리드 컴퓨팅에서는 그리드를 구성하는 컴퓨터, 네트워크, 저장 장치 등의 성능이 가장 중요하며, 그 중 컴퓨터의 CPU 클럭과 같은 정적이면서 알려진 속성은 기록하여 작업을 스케줄할 때 사용한다. 하지만, CPU 클럭은 컴퓨터의 구조가 같을 때만 의미가 있을 뿐, 그리드 컴퓨팅처럼 구조가 다른 컴퓨터가 시스템에 참여하는 경우에는 의미가 없다. 게다가, 그리드 컴퓨팅에서는 활용하는 자원이 컴퓨터의 CPU만 있는 것이 아니라, 네트워크나 저장 장치와 같이 다양한 자원을 활용하기 때문에 성능에 영향을 미치는 요소가 많다.

[11]은 그리드의 자원의 성능을 예측하는 방법을 제안하였다. 제안한 예측 방법은 그리드 환경의 설정이나 온라인 에러에 따라 동적으로 프로그램의 스케줄링에 사용할 수 있다. 사용할 수 있는 많은 컴퓨터 중에서 어디에서 프로그램을 실행할 것인지 결정하기 위해서는 스케줄러에 다음 2가지가 필요하다.

- 자원 성능 특성(Resource Performance Characteristics)을 파라미터화했을 때 프로그램 실행 성능을 정확하게 예측할 수 있는 성능 모델
- 프로그램이 실행될 때 필요할 것으로 보이는 자원의 성능 특성을 평가하는 방법

이 중 후자를 만족시키는 기술을 중점적으로 연구하였다. 시간을 두고 관찰된 성능 요소 데이타로부터 자원의 성능 소모를 예측하기 위해서 정적인 시계열 분석을 사용하여, 다양한 그리드 스케줄에서 효율적으로 사용하였으며 여러 가지 그리드 자원이 같이 사용될 때 나타나는 여러 가지 정적인 특징을 논하였다.

[12]는 그리드 환경에서 네트워크 대역폭을 찾고 할당하는 것을 문제로 지적하고, 네트워크 대역폭을 측정하여 그리드 환경에서 응용할 수 있도록 글로버스 그리드 컴퓨팅 툴킷(Globus Grid Computing Toolkit)의 부분으로 Gloperf를 개발하였다. Gloperf는 쉽게 사용할 수 있도록 설계하였고, 단순한 방법을 사용하여 종단간 TCP 측정을 할 수 있도록 하였다.

위에서 언급한 연구에서는 그리드 컴퓨팅과 같은 분산 컴퓨팅에서 컴퓨팅 능력을 측정하거나 예측하여 성

능을 개선하였다. 하지만, 위의 연구는 연결된 컴퓨터를 관리하고 각 컴퓨터의 성능을 기록하는 컴퓨터가 있어서, 순수(pure) P2P 컴퓨팅에는 바로 적용할 수가 없다.

3. Magic Square 프로토콜 설계

3.1 Naming & Key Location

시스템에서 피어를 구별하기 위해 고유의 ID가 필요하다. 본 논문에서 제안하는 Magic Square에서는 SHA-1을 이용하여 고유의 ID를 만든다. 이 ID는 고정 IP를 가지는 피어는 IP를 해시하고, 동적 IP를 가지는 피어는 사용자의 고유 ID를 해시하여 만든다. 또한 자원의 ID는 자원의 key를 해시하여 만든다. ID의 길이는 160비트를 사용하기 때문에 두 개의 피어 혹은 두 개의 자원이 같은 ID를 가질 가능성이 희박하다. 또한, 해시 함수로 SHA-1을 사용하기 때문에 피어의 ID(또는 자원 ID)는 해시 공간에 균일(uniform)하게 분포한다.

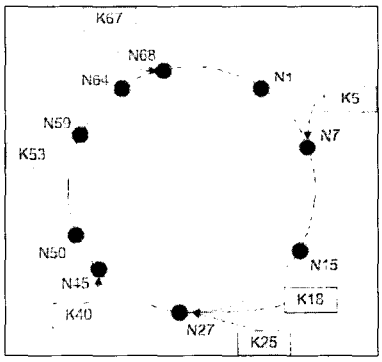


그림 6 6개의 키를 저장한 9개의 피어가 구성하는 Magic Square(해시 공간)

정의 1.

- 담당자(manager): 키의 값을 저장하는 피어이며, 키의 값 K와 같거나 시계방향으로 바로 뒤에 있는 피어이다. 키 K를 저장하는 계승자는 manager(k)로 표기한다.
- 계승자(successor): 어떤 피어의 시계 방향으로 바로 옆에 있는 피어이며, 어떤 피어 p의 계승자는 successor(p)로 표기한다.
- 선행자(predecessor): 어떤 피어의 시계 반대 방향으로 바로 옆에 있는 피어이며, 어떤 피어 p의 선행자는 predecessor(p)로 표기한다.

(a, b]는 ID 공간에서 피어 a를 포함하지 않고, 시계 방향으로 뒤에 있는 피어 b를 포함하는 구간을 표기한다. 피어 a와 b가 바로 인접한 피어라면 (a, b]의 담당자는 피어 b이고, 선행자는 피어 a이다. 마찬 가지로 [a, b)에서 피어 a와 b가 바로 인접한 피어라면 [a, b)의 담당자는 피어 a이고, 계승자는 피어 b이다.

그림 6은 Magic Square에 9개의 피어가 연결되어 있고, 6개의 키를 저장하고 있는 그림이다. 키 K25와 K18의 담당자는 N27이며, N27에 K25와 K18이 저장된다. N27의 선행자는 시계 반대 방향으로 바로 옆에 있는 피어인 N15이고, N27의 계승자는 시계 방향으로 바로 옆에 있는 피어인 N45이다.

3.2 시스템 구조

P2P 시스템에서는 피어의 접속과 떠남, 데이타의 탐색이 중요한 연산이지만, 이 모든 연산을 관장하는 분산 해시 테이블을 생성하고 사용하는 방법은 P2P 시스템의 구조에 영향을 받는다. 이 절에서는 피어의 탐색 비용을 O(log N)으로 유지하면서 피어의 성능을 반영하여 성능을 향상할 수 있는 Magic Square의 구조에 대해서 살펴본다. Magic Suqare의 구조는 그림 7과 같다.

Magic Square는 그림 7과 같이 스킵리스트와 비슷한 모양을 가진다. 스킵리스트의 각 노드는 Magic Square의 피어에 대응되며, 링크는 피어 사이의 연결에 대응된다. 피어 사이의 연결 정보와 다른 피어의 정보는 해당 피어의 *라우팅 테이블*에 저장한다. 그러므로 시스템에서 가장 많은 연결을 가지는 피어는 높이가 가장 높은 피어이며, 2*높이 개의 피어 정보를 관리한다. 보기를 들

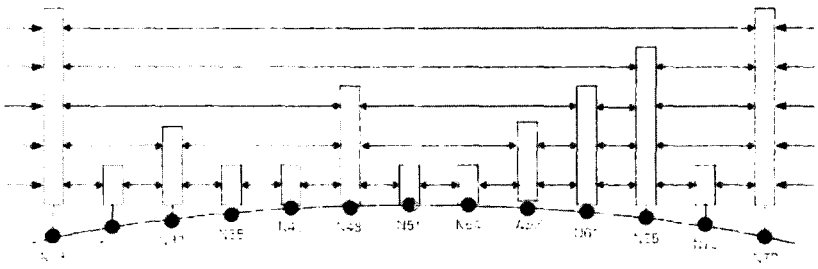


그림 7 Magic Square의 구조(전체 ID공간 중 일부분). N23과 N72의 네트워크 성능이 가장 우수한 집합에 포함되며, 높이는 5라고 가정한다.

어, N46 피어는 시스템에서 6개의 피어와 연결을 가지므로, N46이 관리하는 라우팅 테이블에는 {N23, N32, N42, N51, N57, N61}을 저장한다. N61 피어는 {N57, N61}을 저장한다. N61과 같이 경우에 따라 높이는 높지만 연결을 적게 가질 수 있지만, 전체 분포에는 큰 영향을 미치지 못하기 때문에 성능 저하는 없을 것으로 본다.

Magic Square는 피어의 가용 네트워크 대역폭을 반영하여 피어의 높이를 정한다. 그림 7의 각 피어는 높이를 스스로 결정하는 것이 아니고, 자신의 가용 네트워크 대역폭을 측정하여 결정한다. 높이를 결정하는 방법은 3.4.3 절에서 살펴본다. Magic Square는 피어의 높이가 높은 피어 집합이 메시지 라우팅에 자주 참여하는데, 이 피어 집합의 네트워크 성능이 우수하기 때문에 메시지 라우팅 성능이 향상될 것이다. 결과적으로 Magic Square의 가용 네트워크 성능이 우수한 피어 집합이 시스템의 성능에 결정적인 역할을 한다.

3.3 데이타 찾기

피어가 Magic Square에 접속하기 위해서 피어의 위치를 찾는 연산을 수행하는데, 이 연산은 데이타의 담당자를 찾는 연산과 같다. 피어가 Magic Square에 접속하는 절차를 살펴보기에 앞서 데이타의 담당자를 찾는 연산에 대해서 먼저 살펴본다.

P2P 컴퓨팅을 이용한 인터넷 파일 시스템에서는 피어가 시스템에 저장된 파일에 대한 정보를 모두 가지고 있지 않다. 피어가 시스템에 저장된 데이타를 찾기 위해서는 데이타의 담당자를 찾는 방법을 알고 있으면 된다. Magic Square에 참여하는 피어는 ID와 높이에 따라 연결되는 피어가 정해지며, 연결을 유지하기 위해서 라우팅 테이블에 정보를 유지한다.

데이타를 요청하는 피어는 자신이 관리하는 라우팅 테이블에서 목적 데이타를 저장한 피어에 가장 가까운 피어를 선택해서 메시지를 전달하면, 메시지를 전달받은 피어도 역시 자신이 관리하는 라우팅 테이블에서 목적 데이타를 저장한 피어에 가장 가까운 피어를 선택하여 메시지를 전달한다. 이 과정을 반복하여 데이타를 저장하고 있는 피어에 연결한다. 이처럼 데이타를 요청하는 질의는 데이타의 담당자를 만날 때까지 ID 공간을 따라 전달된다. ID 공간에서 데이타의 담당자를 찾는 알고리즘은 그림 8과 같다. 그림 8에서 피어의 ID가 앞에 붙어 있는 것은 리모트 호출과 리모트 변수 사용을 의미하고, 피어의 ID가 생략된 것은 지역 변수와 지역 함수 호출을 의미한다.

그림 8의 find_manager()는 데이타의 담당자가 인접했으면 담당자를 반환하고, 그렇지 않은 경우에는 현재 피어의 라우팅 테이블에서 담당자와 1) 시계 반대 방향으로 가장 가까운 피어와 2) 시계 방향으로 가장 가까운 피어를 선택하여 메시지를 동시에 전달한다. 메시지에는 최종 전송한 피어의 ID가 기록되어 있으므로 같은 피어에 다시 전달되지 않는다. 양방향으로 전달하면 메시지가 많이 생성되지만, 높이가 높은 (네트워크 성능이 좋은) 피어를 만나서 데이타의 담당자를 빨리 찾을 가능성이 커지기 때문에 시스템의 응답 시간이 빨라질 것으로 보인다.

```
n.find_manager(k)  // manager(k)를 찾을 때
{
  if k ∈ n (predecessor, n] then
    return n;
  else if k ∈ (n, successor] then
    return successor;
  else
    n' = closest_peers(k);
    // 병렬로 실행
    for(i=0;i<n'.length;i++)
      make_thread(return n'[i].find_manager(k));
}
n.closest_peers(k)
{
  do
  {
// 0번째: 시계 반대 방향 피어 중 가장 높이가 높은 피어
// 1번째: 시계 방향 피어 중 가장 높이가 높은 피어
    highest_peers = find_highest_peers(n.route_table);
  } while( highest_peers 가 k를 지남 )
    // k를 지나지 않으면 루프를 빠져나옴
  return highest_peers;
}
```

그림 8 키의 담당자를 찾는 의사(Pseudo) 코드

closest_peers()는 자신이 관리하는 라우팅 테이블에서 데이타의 담당자와 가장 가까울 것으로 예상되는 두 개의 피어를 반환한다. 스킵리스트에서 높이가 낮은 노드를 선택하는 것보다 높이가 높은 노드를 선택했을 때의 검색의 진행이 더 빠르다. 따라서 Magic Square에서도 스킵리스트처럼 데이타의 검색 속도를 높이기 위해서 피어가 진행 방향으로 담당자를 지나치지 않으면서 높이가 높은 피어를 선택한다.

그림 9는 Magic Square에서 K53의 담당자를 찾기 위해서 메시지를 라우팅하는 경로를 보여준다. 보기에서는 K53의 담당자는 N54이며, 높이가 높은 피어를 선택해서 탐색을 진행하다가 메시지가 담당자를 지나치지 않도록 피어를 선택하는 과정이 스킵리스트의 데이타 검색과 비슷함을 알 수 있다.

각 피어는 스킵 리스트를 사용해서 라우팅 테이블을 관리하기 때문에 피어를 찾는 연산은 분산된 스킵 리스트 조각을 이용해서 특정 값을 찾는 연산과 같다. 분산

된 스킵 리스트 조각을 사용하더라도 특정 값을 찾을 때 레벨이 높은 노드(리스트 안의 노드)를 사용해서 진행하기 때문에 비용은 O(log N)에 근접할 것이다. 4장에서 실험으로 증명하겠다.

3.4 피어 접속

P2P 컴퓨팅에서 피어는 메시지를 다른 피어에 효율적으로 라우팅하기 위해서 다른 피어에 대한 정보를 유지한다. 다른 피어에 대한 정보는 라우팅 테이블에 저장하는데, 라우팅 테이블이 유효하지 않으면 시스템의 성능이 저하된다. P2P 컴퓨팅에서는 피어가 임의로 접속하고 떠나는 경우에도 탐색 시간을 일정하게 유지할 수 있어야 한다. 그러므로 실제로 P2P 프로토콜을 사용하기 위해서는 라우팅 테이블을 항상 유효하게 유지하는 것이 중요하다.

정의 2.

- 피어의 접속(join): 피어가 P2P 시스템의 인증을 받고 라우팅 테이블을 생성한 후, P2P 시스템의 일원이 되는 것을 말한다. 인터넷 파일 시스템에서는 피어가 담당하는 키의 데이타를 이동시키는 연산도 포함한다.
- 피어의 떠남(leave voluntarily): 피어가 P2P 시스템의 허가를 받고, P2P 시스템에서 담당하던 기능을 반납하는 것을 말한다. 인터넷 파일 시스템에서는 피어가 담당하던 키의 데이타를 계승자에게 이동시키는 연산을 포함한다.
- 피어의 결함(failure): 피어가 P2P 시스템의 허가를 받지 않고, P2P 시스템을 떠나는 것을 말한다. 다른 피어가 해당 피어가 존재하는 것으로 알고 있기 때문에 라우팅 에러와 키를 탐색하는 연산의 에러, P2P 네트워크가 분리되는 현상이 발생할 수 있다.

본 논문에서 제안한 Magic Square의 각 피어는 양방향 스킵 리스트를 이용해서 라우팅 테이블을 관리한다. 따라서 피어가 빈번하게 접속하고 떠나는 P2P 컴퓨팅 환경에서 라우팅 테이블을 효율적으로 관리할 것으로 본다. 또한 라우팅 테이블에 나타나는 다른 피어의 정보 중 높이(레벨)에 P2P 시스템의 속도와 가장 관계가 깊은 네트워크 대역폭을 반영하도록 하였다.

3.4.1 피어의 접속

Magic Square의 피어는 효율적으로 메시지를 라우팅하기 위해서 2*피어의 높이* 개의 피어의 정보를 유지하는데, 이 정보가 유효하지 않으면 성능이 감소한다. 이 정보를 관리하기 위해서 피어 p는 다음과 같은 순서로 Magic Square에 접속한다.

1. Magic Square에 접속하려는 피어 p는 네트워크에 방송을 하여 Magic Square에 접속되어 있는 피어의 집합 q를 찾는다. q가 존재하지 않으면 새로운 Magic Square를 형성한다.
2. 집합 q 중 네트워크 성능이 가장 우수한 피어를 선택하여, manager(p)를 찾는다.
3. manager(p)를 찾는 동안 연결되는 모든 피어는 p와 메시지를 교환하여 메시지의 전달 시간을 측정한다. 메시지의 전달 시간에 적절한 가중치를 곱한 뒤 계산을 하여 피어의 네트워크 성능으로 사용한다. 이 과정은 3.4.2 절에서 자세하게 기술한다.
4. manager(p)를 찾아서 선행자, 담당자, 계승자를 수정하고, 자신이 관리할 데이타를 manager(p) 로부터 가지고 온다.
5. 3의 과정에서 메시지를 교환한 피어 집합과 메시지를 교환하여 새로운 피어가 접속했음을 알리고, 자신의 라우팅 테이블을 생성한다.

3.4.2 전역 라우팅 테이블 구성

스킵리스트는 항상 높이가 가장 높은 헤더에서 탐색을 시작하기 때문에 탐색 성능이 O(log N)에 근접한다. 하지만, Magic Square는 성능이 가장 우수한 피어(높이가 가장 높은 피어)가 항상 질의를 하는 것이 아니라, 시스템에 참가하는 모든 피어가 질의를 할 수 있기 때문에 질의 처리 속도를 높이기 위한 방법이 필요하다. 그 방법은 다음과 같다.

1. 질의를 처리하거나 시스템에 처음 접속하면서 알아낸

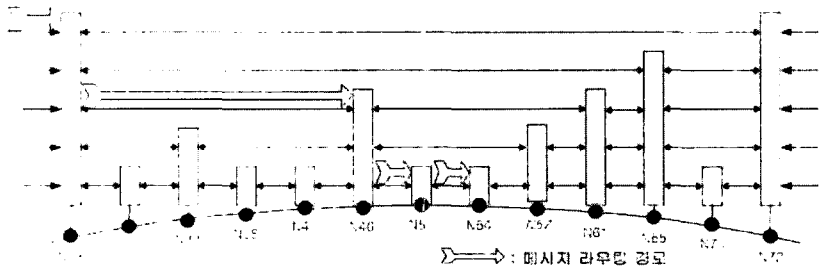


그림 9 Magic Square 중 일부. Magic Square에서 manager(K53)을 찾기 위해서 메시지가 라우팅되는 경로

높이가 높은 피어의 ID를 저장한다. 이후에 자신의 질의를 처리할 때 높이가 높은 피어를 바로 이용하기 때문에 필요하지 않은 홉을 제거한다. 따라서 시스템의 질의 처리 비용을 O(log N)에 근접한다.

2. 피어에서 멀리 떨어진 피어의 ID를 사용하여 높이가 낮은 피어 집합을 메시지가 통과하지 않도록 한다. 이는 메시지를 처리하는 피어와 목적 피어 사이의 거리가 멀면 목적 피어를 지나지 않는 범위에서 최대한 멀리 뛸 수 있도록 한다.

전역 라우팅 테이블은 3.4.1절에서 만든 라우팅 테이블과 같이 저장하되 라우팅 테이블 중 일부분을 할당해서 관리한다. 전역 라우팅 테이블은 캐시와 마찬가지로 시스템이 작동하면서 계속 바뀐다. 전역 라우팅 테이블에는 현재까지 자신이 발견한 피어 중 가장 좋을 것 같은 것(높이가 높은 것)을 골라서 저장한다. 이로써 시스템이 질의를 처리하는 시간을 줄일 수 있다.

3.4.3 가용 네트워크 대역폭 예측

기존의 P2P 시스템에서는 피어의 성능이 모두 같다는 가정을 하고 있는데, 이는 실제 컴퓨팅 환경을 잘 반영하지 못한다. 컴퓨팅 성능이 다른 점을 P2P 시스템에 반영하면 실제 환경을 반영하기 때문에 성능, 가용성, 신뢰성 등을 개선할 수 있다.

본 논문에서 제안하는 Magic Square는 각 피어의 컴퓨팅 성능을 피어가 관리하는 라우팅 테이블에 반영하여, 시스템의 성능과 가용성, 신뢰성을 개선하였다. 라우팅 테이블을 스킵 리스트로 관리하는데, 스킵 리스트의 높이(높이가 높으면 레벨이 높다.)에 컴퓨팅 능력을 반영하여 라우팅 테이블을 구성하였다.

피어의 컴퓨팅 능력이라고 할 수 있는 것은 다음과 같다.

- 피어의 계산 속도와 연관: CPU 속도, 버스의 속도, 메모리의 속도 등
- 시스템의 메시지 전달과 연관: 네트워크 대역폭
- 시스템의 크기와 연관: 하드디스크와 같은 저장 장치
- 기타

피어의 컴퓨팅 능력 모두를 고려하면 시스템의 성능을 극대화할 수 있지만, 스킵 리스트의 높이에 여러 가지 특성을 모두 반영할 수 없다. 따라서 이 중 P2P 시스템의 속도와 가장 관계가 깊은 네트워크 대역폭만을 시스템에 반영하도록 하겠다. 응용에 따라서 다른 성능 특성을 고려하여 라우팅 테이블을 구성할 수 있을 것으로 본다.

피어 p가 Magic Square에 접속하기 위해서 manager(p)를 찾을 때 find_manager 메시지가 지나는 피어에서 보내주는 메시지를 사용해서 p의 가용 네트워크 대역폭을 측정한다. 측정한 네트워크 대역폭에 임의의 값을 더하거나 빼주어서 피어의 높이를 다양하게 만든다. 왜냐하면 피어의 높이가 비슷해지면 탐색 성능이 떨어지기 때문이다. p의 가용 네트워크 대역폭을 측정하는 구체적인 과정은 그림 10과 같다.

```
while( not find peer's position )
{
    probe_msgs ← msgs from some peers;
}
for each element of probe_msgs
{
    perform the test;
    write the results in the memory;
}
estimate bandwidth with results;
```

그림 10 피어의 가용 네트워크 대역폭을 측정

그림 10의 probe_msgs는 피어의 위치를 찾을 때 메시지가 전달되는 피어에게서 받은 메시지를 저장한 것이다. 피어의 위치를 찾는 비용은 O(log N)이기 때문에 probe_msgs에 저장되는 메시지의 수는 O(log N) 개이며, 결과적으로 피어의 가용 네트워크 대역폭을 예측하는 비용도 O(log N)에 근접할 것이다.

3.5 안정화(stabilization)

P2P 컴퓨팅 환경은 피어의 참여와 떠남, 결함이 자주 발생하기 때문에 P2P 프로토콜에는 여러 가지 상황에 적절하게 대응할 수 있는 기능이 필요하다. 피어의 집합이 변하는데도 원하는 자원을 정확하게 찾기 위해서 Magic Square는 주기적으로 안정화 연산을 실행한다. 안정화 연산을 실행하고 나면 선행자와 계승자의 연결을 확인하고 라우팅 테이블이 정확하지 않으면 수정하는 연산을 한다.

Magic Square에 참여하는 모든 피어는 주기적으로 그림 11의 *stabilize()*를 백그라운드로 실행하여 피어의 라우팅 테이블이 유효할 수 있도록 관리한다. 여러 개의 피어가 동시에 접속하고 떠나면 라우팅 테이블의 내용이 실제 접속되어 있는 피어의 상황을 반영하지 못하는 경우가 생기는데, stabilize()를 실행하여 라우팅 테이블에 빠져 있는 피어를 찾는다.

새로운 피어가 시스템에 접속하면서 피어 정보를 관리할 피어 집합을 선택하여 피어 집합과 메시지를 교환한다. 이 과정에서 새로운 피어와 피어 집합의 라우팅 테이블이 수정되지만, 네트워크에서 메시지가 전달되는 속도의 차이 때문에 피어간 연결이 완전하지 못하여 시스템에 왜곡이 생길 수 있다. 시스템의 왜곡을 해결하기 위해서 시스템 안정화 연산이 필요하다. 안정화 연산은 Magic Square의 피어간 연결과 순서를 조정하기 때문

```
n.stabilize()
{
    x = successor.predecessor;
    y = predecessor.successor;
    if( x ∈ ( n, successor ) )
        successor = x;
    successor.notify(n);
    if( y ∈ ( predecessor, n) )
        predecessor = y;
    predecessor.notify(n);
}

n.notify(n')
{
    if( predecessor is nil or n' ∈ (predecessor, n) )
        predecessor = n';
    if( successor is nil or n' ∈ (n, successor) )
        successor = n';
    if( routing table is changed )
        notify to neighbors;
}
```

그림 11 안정화를 위한 의사 코드

에 시스템의 도달성(Reachability)를 유지하는 역할을 한다.

3.6 피어의 결함(failure)

P2P 시스템에서 정확성은 각 피어가 원하는 데이타의 담당자를 정확하게 찾는 것을 의미한다. 그러나 어떤 피어에 결함이 발생하면 원하는 데이타를 찾을 수가 없다. 보기를 들어 그림 9의 피어 N32와 N46에 동시에 결함이 발생하면 N38과 N42는 Magic Square에 참여하고 있지만, 다른 피어가 접근할 수도 없고 이 피어들도 다른 피어에 접근할 수 없다.

따라서 시스템이 결함에도 잘 작동할 수 있도록(robust) 각 피어는 추가의 정보를 유지할 필요가 있다. 본 시스템에서 각 피어는 인접한 r 개의 선행자와 r개의 계승자를 리스트로 관리하여 인접한 피어에 결함이 발생하여도 Magic Square 네트워크가 붕괴되지 않는다. 보기를 들어 그림 9의 피어 N46은 선행자 리스트에 N32, N38, N42를, 계승자 리스트에 N51, N54, N57을 저장하여 인접한 피어에 결함이 발생하면 다음 선행자나 계승자를 리스트에서 선택하여서 대치한다.

피어의 ID는 해시를 사용해서 할당하기 때문에 ID 공간에서 인접한 피어가 물리적인 네트워크에서 인접할 가능성은 거의 없다. 따라서 인접한 2*r개의 피어에 동시에 결함이 발생할 가능성이 적기 때문에 시스템에는 별 영향을 미치지 못할 것이다. 이는 4장에서 실험으로 증명하도록 한다.

4. 성능 평가

이 절에서는 시뮬레이션을 통해서 본 논문에서 제안하는 Magic Square 프로토콜의 성능을 평가한다. 본 논문에서 서술한 의사 코드를, Java를 사용해서 메시지 수준의 시뮬레이터를 작성하였다.

4.1 탐색

Magic Square의 성능은 질의에 대한 응답을 처리할 때 방문하는 피어 수에 의존하는데, Magic Squre의 탐색 방법은 N개의 노드를 가지는 스킵리스트의 탐색 방법과 비슷하기 때문에 O(log N)에 근접할 것이다. Magic Suqare의 성능을 알아보기 위해서 표 1과 같은 실험 조건을 이용해서 시뮬레이션하였다. 시뮬레이션에서 선택된 각 피어는 시스템에서 랜덤 키를 찾는 질의를 요청하고, 질의를 처리할 때 접속하는 피어의 수를 측정하여서 탐색의 성능을 알아보았다.

표 1 실험 환경

중요 실험 인자	값
시스템에 참가한 피어의 수 (N)	100 ~ 100,000
질의의 수	N/2
메시지의 네트워크 지연 시간	10ms ~ 1000ms

아래의 그림 12는 시스템에 참가하는 피어의 수가 늘어남에 따라 시스템의 성능, 즉 메시지의 홉 수의 변화를 보여준다. 피어의 수와 홉 수의 그래프가 log N 그래프에 근접하고 있음을 볼 수 있다. 따라서 시스템에서 데이타 탐색 비용은 O(log N)에 근접한다.

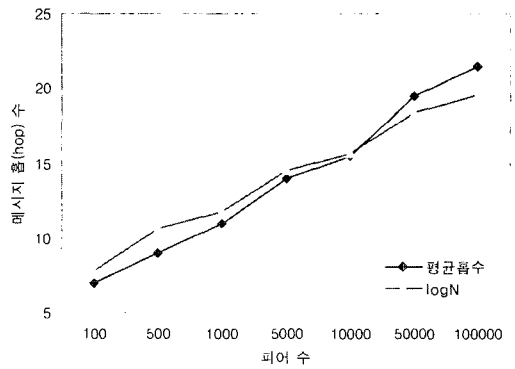


그림 12 시스템에 참여하는 피어 수의 변화에 따른 메시지 홉 수

이전 연구에서 개발한 P2P 시스템에서는 질의의 홉 수가 시스템의 중요한 평가 요소가 되었다. 하지만, 본 논문에서 제안하는 Magic Square는 이전에 나온 P2P 시스템과 달리 피어의 네트워크 성능을 고려하여 질의를 처리한다. 피어를 찾을 때의 메시지 홉 수는 네트워크의 성능의 척도로 볼 수 없기 때문에 시스템의 응답 시간으로 네트워크의 성능이 반영되어 있는지 살펴보았

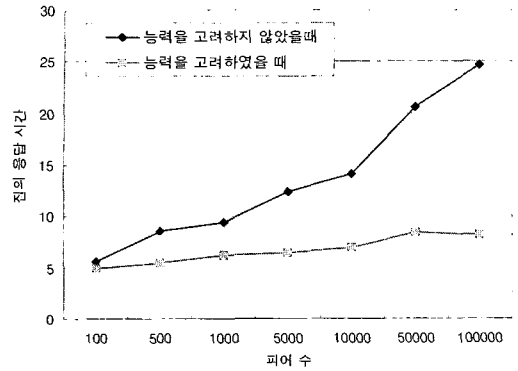


그림 13 시스템에 참여하는 피어 수의 변화에 따른 시스템의 응답 시간

다. 그림 13은 Magic Square 시스템의 라우팅 테이블에 네트워크 성능을 반영한 경우와 그렇지 않은 경우의 시스템의 응답 시간을 보여 준다. 피어의 네트워크 능력을 고려한 시스템의 응답 시간의 증가율이 피어의 네트워크 능력을 고려하지 않은 시스템의 그것의 증가율에 비해 낮다. 따라서 피어의 수가 급격하게 늘어나더라도 시스템이 쉽게 적응할 수 있다.

4.2 안정성

이 실험에서는 여러 개의 피어에 결함이 Magic Square의 성능에 어떤 영향을 미치는지, 그리고 정확하게 탐색하여 키를 찾는지 확인한다. 본 논문에서 제안한 Magic Square의 안정화 단계의 성능을 확인하기 위해서 실험에서 사용한 가정은 다음과 같다.

ㄱ) 탐색 성공: 원하는 키의 계승자를 찾았을 경우이다.

ㄴ) 탐색 실패: 원하는 키의 계승자를 찾지 못한 경우이다.

ㄷ) 피어 결함: 피어 n의 평균 패킷 전달 시간의 10배(타임 아웃)가 지나도 패킷이 다른 피어 n으로 전달되지 못하면 n에 결함이 발생한 것으로 판단한다(타임 아웃: time out).

1000개의 피어(N=1000)가 참여하고, 피어의 선행자 리스트와 계승자 리스트의 크기는 $10(\log_2 N=10)$인 Magic Square를 구성하였다. 네트워크가 안정되고 난 후, 일정한 비율로 시스템에 참여한 피어를 결함 상태로 만들면서 질의를 요청하였다. 피어에 결함이 발생하면 P2P 네트워크가 끊어져서 탐색 실패의 원인이 되는데, 피어의 결함에 시스템이 어떻게 적응하는지 알아보기 위해서 각 질의에 대해서 계승자를 찾았는지의 여부를 기록하였다.

그림 14는 주기적으로 안정화 단계를 실행하는 Magic Square와 그렇지 않은 Magic Square에 대해서 피어의 결함 비율과 질의 실패율로 Magic Square 시스템의 적응 정도를 보인다. 안정화 단계가 없는 시스템의 그래프는 피어의 결함 비율이 높아지면 그에 따라 질의의 실패율이 증가함을 보여준다. 안정화 단계가 있는 시스템에서는 타임 아웃 이전에 P2P 네트워크를 복구하기 때문에 담당자 피어에 결함이 발생한 경우를 제외하고는 시스템에서 키를 찾을 수 있다. 따라서 위 그래프에서 질의 실패율이 5%를 초과하지 않음을 알 수 있다. MagicSquare의 각 피어는 선행자 리스트와 계승자 리스트를 제대로 관리하여 안정화 단계를 통해 피어 결함을 시스템 차원에서 잘 해결하고 있음을 알 수 있다.

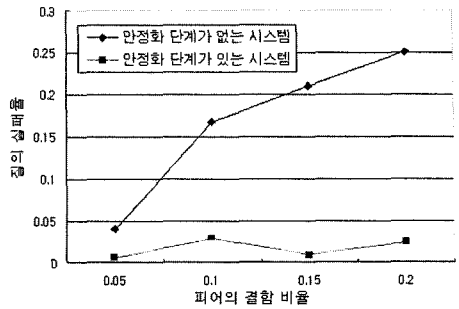


그림 14 피어 결함에 대한 시스템의 적응

5. 결 론

대부분의 P2P 시스템에서 가장 중요한 기능은 효율적으로 데이타를 위치시키고(location) 탐색하는 것이다. 이미 개발된 Chord나 Pastry와 같은 P2P 시스템에서 분산 해시 테이블을 사용하는 분산적인 방법으로 이를 해결하였다. 본 논문에서 제안한 Magic Square도 이전 연구에서 제안한 P2P 시스템과 마찬가지로 분산 해시 테이블을 사용하여 분산적인 방법으로 데이타 위치 문제를 해결하였다.

Magic Square는 스킵리스트와 비슷한 구조로 구성되기 때문에 스킵리스트와 비슷하게 랜덤 속성을 가진다. Magic Square는 라우팅 테이블을 랜덤하게 구성하여 피어의 구성이 최악의 경우에도 라우팅 테이블을 구성하는 비용과 키를 탐색하는 비용이 평균 탐색 비용인 O(log N)에 근접한다. Magic Square는 시스템에 피어가 새로 들어오거나 떠나더라도 부분적으로 정확하지 않은 라우팅 정보를 가지고 라우팅을 할 수 있다. 시뮬레이션을 통해서 피어의 수가 증가하더라도 Magic Square가 잘 적응함을 알 수 있었다.

Magic Square에 부적절한 피어가 참여하면 Magic Square 네트워크가 결함에 노출되거나 분리될 수 있다. 시스템에 참여하는 피어는 암호화된 정보를 교환하면 부적절한 피어가 정보를 가로채는 문제를 해결할 수 있

다. 하지만, 암호화된 정보를 교환하는 것은 시스템의 가용성에 좋지 못하다. 또한, 시스템에 참여하는 피어를 랜덤하게 선택해서 여러 가지 질의와 응답을 조합해서 부적절한 피어를 찾아서 제거하는 방법이 있다. P2P 환경에서 랜덤하게 피어를 선택하는 것은 비용이 아주 크기 때문에 적합하지 않다. 앞으로 Magic Square에 부적절한 피어를 찾아서 제거하는 방법에 대한 연구가 필요한 것으로 보인다. 이 연구는 Magic Square 뿐만 아니라 대부분의 P2P 시스템에 필요하다.

참 고 문 헌

[1] A. Oram, "Peer-to-Peer," O'Reilly, Mar, 2001.

[2] ThinkStream, "A Technical Review of the Next Generation Internet Architecture."

[3] C. Shirky, "What is P2P... and what Isn't," http://www.openp2p.com/pub/a/472.

[4] I. Stoica, R. Morris, D.Liben-Nowell, D. R. Karger, M. Frans Kaashoek, F. Dabek, H. Balakrshinan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," IEEE/ACM Transactions on Networking, Vol. 11, Feb 2003.

[5] V. A. Mesaros, B. Carton, P. V. Roy, "S-Chord: Using Symmerty to Improve Lookup Efficiency in Chord," Proceedings of PDPTA03, Jun 2003.

[6] J. Kubiatowicz, "Extracting Guarantees from Chaos," Communications of the ACM Vol. 46, Feb 2003.

[7] William Pugh, "Skip Lists: A Probabilistic Alternative to Balanced Trees," Communications of the ACM, Vol. 33, Jun 1990.

[8] A. Rowstron, P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," Proceedings of the 18th IFIP/ACM Int'l Conf. on distributed Systems Platforms, Nov. 2001.

[9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "A Scalable Content-Addressable Network," Proceedings of ACM SIGCOMM, Aug. 2001.

[10] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, "Looking Up Data in P2P Systems," Communications of the ACM Vol. 46, Feb 2003.

[11] R. Wolski, "Experiences with predicting resource performance on-line in computational grid settings," ACM SIGMETRICS Performance Evaluation Review, Vol. 30 Issue 4, Mar. 2003.

[12] C. A. Lee, J. Stepanek, R. Wolski, C. Kesslman, I. Foster, "A Network Performance Tool for Grid Environments," Proceedings of the 1999 ACM/IEEE conference on Supercomputing, Jan 1999.

정 일 동

2000년 부산대학교 전자계산학과(이학사). 2002년 부산대학교 대학원 전자계산학과(이학석사). 2002년~부산대학교 대학원 전자계산학과 박사과정. 2003년 11월~LG전자 DAC연구소 주임 연구원. 관심분야는 정보가전, 인터넷 컴퓨팅, 임베디드 시스템 등

유 영 호

1994년 부산대학교 전자계산학과(이학사). 1997년 부산대학교 전자계산학과(이학석사). 2003년 부산대학교 전자계산학과(이학박사). 2004년~현재 부산대학교 컴퓨터및정보통신연구소 기금교수. 관심분야는 XML, Mobile Computing, P2P

이 종 환

1996년 부산대학교 전자계산학과(이학사). 1998년 부산대학교 전자계산학과(이학석사). 2003년 부산대학교 전자계산학과(이학박사). 2004년 9월~현재 (주)오토파워 부설연구소 선임 연구원. 관심분야는 데이타베이스, 인터넷응용, 스마트 센서

김 경 석

1977년 서울대학교 무역학과(경제학사) 1979년 서울대학교 전자계산학과(이학석사). 1988년 일리노이 주립대(어바나-샴페인) 전자계산학 박사. 1988년~1992년 미국 노스다코타 주립대학교 전자계산학과 조교수. 1992년~현재 부산대학교 전자전기정보컴퓨터공학부 교수. 관심분야는 데이타베이스, 멀티미디어, 한글/한말 정보처리, 인터넷 컴퓨팅 등