TP2P: 효율적인 자원탐색을 위한 토폴로지 기반의 P2P 시스템

(TP2P: Topology-based Peer-to-Peer System for Efficient Resources Lookup)

차 봉 관 † 한 동 윤 † 손 영 성 †† 김 경 석 †††
(Bongkwan Cha) (Dongyun Han) (Youngsong Son) (Kyongsok Kim)

요 약 P2P 시스템은 시스템에 참여하는 노드들의 자원을 공유하는 분산 시스템으로 여기에 참여하는 노드들은 서버와 클라이언트의 역할을 모두 수행한다. 현재 분산 해쉬 테이블(Distributed Hash Table)을 기반으로 한 체계적이고 구조화된 P2P 시스템들인 CAN, Chord, Pastry, Tapestry 등이 제안되었으나 이 시스템들은 물리적 거리를 고려하지 않아서 안정적인 성능을 보장하기 어렵다는 약점을 가지고 있다. 이 문제를 해결하기 위해서 우리는 TP2P시스템을 제안한다. 이 시스템은 스스로 조직을 관리하는 계층적 오버레이 네트워크 시스템으로 자원 탐색을 위해 Chord의 라우팅 메커니즘을 사용한다. 이 시스템은 물리적인 거리가 매우 가까운 노드들로 구성된 서브넷과 모든 노드들로 구성된 글로벌 네트워크로 이루어진 시스템이다. 각 서브넷에서 한 노드가 어떤 자원을 탐색하면 그 자원을 서브넷 안에 저장하기 때문에 Chord 시스템에 비해 물리적인 지연이 줄어들 가능성이 높다. 또 자원을 탐색할 때 각 노드들이 가지고 있는 Global-nodeID 정보를 이용함으로써 물리적 지연이 줄어드는 것은 물론이고 탐색 홉 수도 25%정도 감소한다.

키워드 : P2P 시스템, 분산 해쉬 테이블, Chord, 물리적 거리, 계층적 오버레이 네트워크

Abstract P2P systems are distributed data sharing systems, and each node in them plays the role of client as well as server. There are several studies using Distributed Hash Table, such as Chord, CAN, Tapestry, Pastry, but these systems don't consider the physical latency, therefore they have a weakness of difficulty to guarantee stable performance. To improve this problem, we present the TP2P system. This system is a self-organizing hierarchical overlay system and it uses Chord routing mechanism for lookup data. This system is organized by several subnets, each subnet is organized by physically close nodes, and global network organized by all nodes. In each subnet, one node finds a data, it saves in a node in the subnet, therefore it has higher probability to reduce physical lookup latency than Chord system. And each node has global information of some nodes in its subnet, and it is used to lookup data, therefore the number of hops decrease about 25% as well as the physical lookup latecy are reduced.

Key words : P2P system, DHT, Chord, physical latency, hierarchical overlay network

1. 서 론

초고속 인터넷이 급속히 보급되고 PC의 기능이 급속히 발전함에 따라 P2P(Peer-to_Peer) 시스템이라 불리는 새로운 패러다임이 출현하였다. 현재 대부분의 네트워크들은 클라이언트/서버(Client/Server) 모델을 사용하고 있지만, 이 모델은 확장성(scalability)의 문제를 가지고 있다. P2P 시스템은 클라이언트/서버의 확장성 문제를 해결하고 클라이언트의 자원을 효율적으로 활용할 수 있다는 장점을 가지고 있다. P2P 시스템의 가장 중요한 기능은 데이타를 효율적으로 배치하고 탐색(lookup)하는 것이다.

P2P 시스템은 참여하는 노드들의 자원을 공유하는

· 이 논문은 부산대학교 자유과제 학술연구비(2년)에 의하여 연구되었음
† 학생회원 : 부산대학교 컴퓨터공학과
bgcha@asadal.cs.pusan.ac.kr
dyhan@asadal.cs.pusan.ac.kr
†† 정 회 원 : 한국정보통신연구 홈네트워크 주임연구원
ysson@asadal.cs.pusan.ac.kr
††† 종신회원 : 부산대학교 컴퓨터공학 교수
gimgs0@asadal.cs.pusan.ac.kr
논문접수 : 2006년 9월 1일
심사완료 : 2007년 1월 29일

분산 시스템으로 여기에 참여하는 노드들은 서버와 클라이언트의 역할을 모두 수행한다. 대부분의 P2P 시스템에서 가장 핵심이 되는 기능은 데이타를 효율적으로 저장하는 일이다. 현재 잘 알려진 P2P 시스템으로는 Napster[1]와 Gnutella[2]가 있으며, 이 시스템들은 각 노드에 파일을 저장하고 파일을 원하는 노드는 그 시스템에 속한 그 파일을 저장하고 있는 노드로부터 직접 파일을 전송받는다. Napster는 인덱스(Index) 서버를 두고 서비스를 제공하고, Gnutella는 서버를 두지 않고 플러딩(flooding) 방식을 통하여 서비스를 제공한다. 두 시스템은 각 서버의 부담과 플러딩에 의한 네트워크 트래픽으로 인한 확장성의 제약을 안고 있다.

확장성의 문제를 해결하기 위해 분산 해쉬 테이블(Distributed Hash Table)을 기반으로 한 체계적이고 구조화된 시스템이 제안되었다. 이에 대한 대표적인 연구가 CAN[3], Chord[4], Pastry[5], Tapestry[6] 등이다. 이러한 시스템들은 분산 해쉬 테이블을 사용하여 오버레이 네트워크(Overlay Network)를 형성하고 파일을 완전히 분산시킴으로써 효율적인 탐색 알고리즘을 제공하여 확장성 문제를 해결하고자 한다. 이 시스템들에서 데이타들이 저장되어야 할 위치는 시스템에 의해 결정되어 오버레이 네트워크 상의 노드들에 분산 배치된다. 그러므로 이 시스템들은 새로운 데이타를 저장하고 데이타가 저장된 위치를 탐색하기 위해 각 노드가 제한된 위치 정보를 가지면서 특정 노드를 찾을 수 있는 탐색 알고리즘을 제시하고 있다.

P2P 오버레이 네트워크에서는 해쉬 함수를 이용하여 노드의 IP주소를 해쉬하여 나온 결과값에 따라 시스템에 조인되므로 노드들 사이의 물리적 거리를 고려하지 않는다. 즉 P2P 시스템에서는 오버레이 상에서는 서로 이웃한 노드라 할지라도 실제 물리적 네트워크에서는 여러 라우터를 경유할 수 있기 때문에 물리적인 지연(Latency)이 길어질 가능성이 있다. 그러므로 실제 탐색 과정에서 홉을 몇 번 뛰지 않더라도 하나의 홉의 지연이 길어질 수 있기 때문에 탐색이 비효율적인 경우가 많이 생길 수 있다.

본 논문은 이 문제점을 해결하기 위해 토폴로지(Topology)를 고려한 계층구조인 TP2P를 제안한다. TP2P는 물리적 가까운 노드들을 하나의 서브넷(Subnet)으로 구성하고 각 노드들은 Global-nodeID 테이블을 유지하여 물리적인 지연을 줄이며 효율적으로 자원을 탐색할 수 있다. 또한 자신의 서브넷에 데이타의 복제본(Replica)을 두어 탐색할 때 물리적인 지연을 보다 효율적으로 감소시킨다. 이와 같은 방법으로 물리적인 거리를 고려한 P2P시스템들에는 Grapes[7], Brocade[8] 등이 있다. 이들 시스템들은 일반적인 P2P 시스템과 비슷한 홉 수를 가지나 TP2P는 Global-nodeID 테이블을 사용함으로써 물리적 지연은 물론 홉 수를 평균 25% 정도 줄일 수 있다.

본 논문의 구성은 다음과 같다. 2 장에서 관련 연구로 본 논문의 탐색 알고리즘이 되는 Chord의 구성과 탐색, 노드 조인, 노드 탈퇴를 간략하게 설명하고 Chord시스템의 문제점을 설명한다. 3장에서 TP2P의 구조에 대한 설명과 함께 탐색 알고리즘을 제안한다. 4장에서는 시뮬레이션을 통해 성능을 비교하고 마지막으로 5장에서는 본 논문이 주장하는 TP2P에 대해 정리하고 결론을 맺는다.

2. 관련 연구

2.1 Chord 시스템

Chord 시스템은 P2P 응용을 위한 분산 해쉬 테이블을 기반으로 하는 분산 처리 자원탐색 프로토콜로 분산 탐색을 지원하며 해쉬 함수를 이용하여 데이타의 삽입과 탐색을 수행한다. Chord 시스템에서 시스템 내에 존재하는 각 노드는 시스템 내의 모든 노드의 정보를 유지하지 않고 다른 노드들에 대한 일정한 수의 라우팅(routing) 정보만 유지함으로써 시스템의 확장성을 제공한다.

Chord는 2^k 크기의 원형 식별자 공간을 사용하며 각 노드는 IP주소를 160 비트의 SHA-1 해쉬 함수로 해쉬하여 nodeID를 할당받아 원형 식별자 공간의 해당 nodeID에 위치한다. 데이타의 위치 정보는 (key, value) 쌍으로 표현되며 데이타가 저장될 노드의 위치는 키(key)를 SHA-1 해쉬 함수로 해쉬한 값에 의해 정해진다. 각 노드는 successor, predecessor의 정보를 유지하여 링 형태의 오버레이 구조를 형성한다. 그림 1에서 nodeID가 1인 노드의 successor는 3이고 prececessor는 0이다. 노드가 시스템에서 비정상적으로 떠났을 때 시스템을 복원하기 위해 log N 개의 successor 정보를 list로 유지한다.

Chord에서 키를 해쉬한 값을 원형 식별자 공간에 대응시킬 때 원형 식별자 공간에서 해쉬된 키 값과 같은 nodeID를 가진 노드에 저장하고, 같은 nodeID가 없을 때는 바로 앞의 노드에 저장한다. 이 노드를 successor 노드라 부른다. 그림 3에서 키 6의 successor노드는 nodeID가 1인 노드이다. 각 노드는 전체 네트워크 상의 노드에 대한 정보를 분산된 동적 환경에 효과적으로 만족시키기 위해 라우팅 테이블(finger table)을 유지한다. 라우팅 테이블은 데이타를 삽입하거나 키를 관리하는 노드를 찾기 위해 탐색 메시지를 해당 노드에게 전달하는데 이용된다.

그림 1은 크기가 8인 원형 식별자 공간에 키 1, 2, 6

을 관리하는 successor 노드들의 라우팅 테이블을 나타내고 있다. 라우팅 테이블의 정보는 자신의 노드에서 지수적으로 증가되는 식별자의 시작값(start)과 그 범위(interval) 그리고 시작값에 대한 successor 노드로 이루어진다. 메시지의 라우팅은 각 노드의 라우팅 테이블에 따라 범위를 지수적으로 감소시키면서 키에 대한 successor 노드를 찾는다. 이와 같이 범위를 지수적으로 감소하면서 탐색하기 때문에 탐색비용은 O(logN)이다. N은 원형 식별자 공간의 크기를 나타낸다.

자원을 시스템에 저장하거나 시스템에서 저장된 자원을 탐색할 때 라우팅 테이블을 사용한다. Chord에서 자원을 탐색하길 원하는 노드 N이 키를 해쉬한 값을 관리하는 successor를 탐색할 때, 먼저 노드 N의 successor에 원하는 키가 존재하는지 확인하고 존재하지 않으면 자신의 라우팅 테이블을 통해 범위(interver)에 키가 포함되는 successor 에게 키에 대한 successor를 요청하는 메시지를 보낸다. 이러한 과정을 반복함으로써 노드는 원하는 자원을 관리하는 노드에게 메시지를 전달할 수 있다.

2.2 Chord 시스템의 자원 탐색

이 장에서는 각 노드가 유지하는 라우팅 테이블을 이용하여 원하는 자원을 탐색하는 방법에 대해 설명한다. Chord에서 자원을 탐색하길 원하는 노드 N이 키를 해쉬한 값을 관리하는 successor를 탐색할 때, 먼저 노드 N의 successor에 원하는 키가 존재하는지 확인하고 존재하지 않으면 자신의 라우팅 테이블을 통해 범위(interver)에 키가 포함되는 successor에게 키에 대한 successor를 요청하는 메시지를 라우팅한다. 이러한 과정을 반복함으로써 노드는 원하는 자원을 관리하는 노드에 메시지를 전달할 수 있다.

그림 1과 같은 Chord 시스템에 노드 0, 1, 3이 있다고 했을 때 노드 1이 키 6의 successor를 찾는 과정은

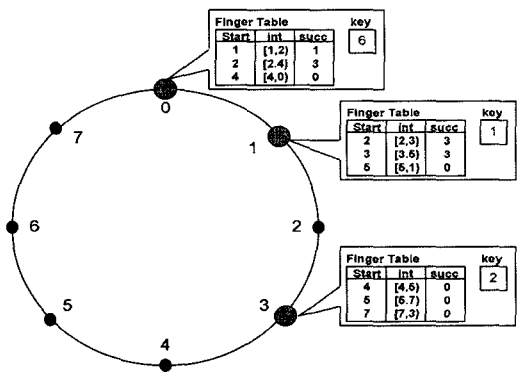


그림 1 키 1, 2, 6을 관리하는 successor 노드 0, 1, 3의 라우팅 테이블

다음과 같다.

1. 노드 1은 우선 자신의 successor인 노드 3에 키 6이 존재하는지 확인한다. 존재한다면 자신의 successor인 노드 3이 키 6의 successor가 된다. 그러나 그림 3에서는 존재하지 않기 때문에 노드 1은 자신의 라우팅 테이블에서 범위 [5,1)에 키 6이 포함되는 것을 확인하고 successor 0에게 라우팅 메시지를 보낸다.
2. 메시지를 받은 노드 0은 키 6의 successor이므로 질의를 요청한 1에게 응답 메시지를 보낸다.
3. 응답 메시지를 받은 노드 1은 노드 0에게 직접 연결하여 원하는 자원을 얻을 수 있다.

모든 노드는 위와 같은 방법을 통해 원하는 자원을 찾을 수 있다.

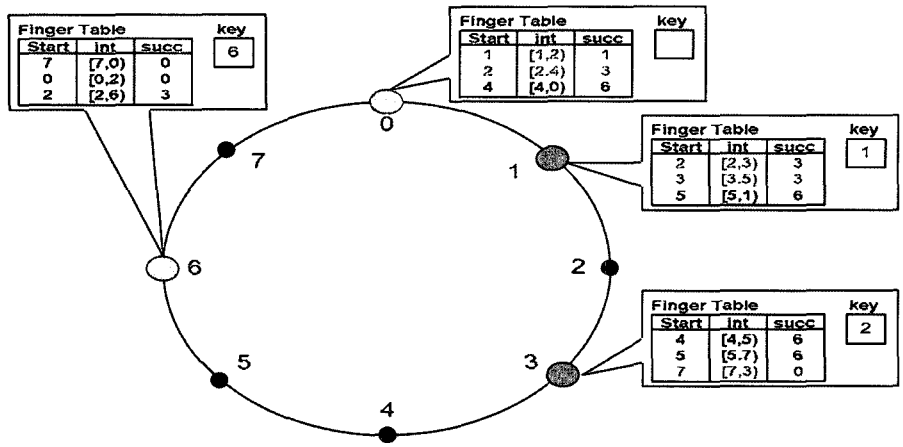
2.3 노드의 참여와 해제

조인을 원하는 노드가 시스템에 참여하거나 참여 중인 노드가 시스템을 정상적으로 떠날 때 라우팅 테이블이 어떻게 유지되는지 살펴보자. 조인을 원하는 노드는 반드시 Bootstrap노드를 알고 있으며 그 노드를 통해 시스템에 조인한다. 그림 1에서 노드 6이 시스템에 새로 조인하면 그림 2의 (a)와 같이 된다. 노드 6은 Chord 라우팅 알고리즘에 따라 자신의 라우팅 테이블을 구성한다. 그리고 시스템에 이미 존재하는 노드들은 새로 조인한 노드 6을 자신의 라우팅 테이블에 반영한다. 그림 2의 (a)에서 각 노드들의 라우팅 테이블의 엔트리가 진하게 표시된 부분이 노드 6이 시스템에 새로 참여한 후 업데이트된 부분이다. 노드 0은 노드 6의 successor 노드가 되며 자신이 관리하고 있는 데이타 중 노드 6이 관리해야 할 데이타를 전송한다. 그림 2의 (a)에서 노드 1은 키 6을 관리하고 있는데 키 6의 successor는 노드 6이 되기 때문에 노드 1은 노드 6에게 키와 데이타를 넘겨준다.

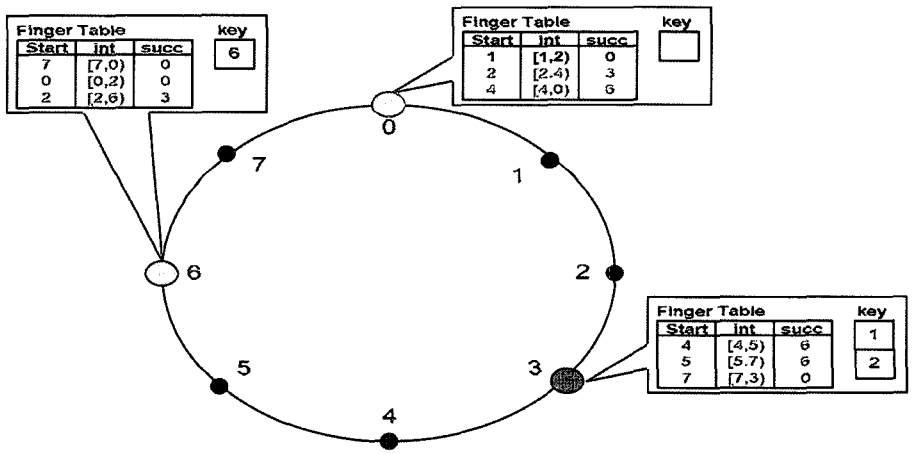
그림 2의 (a)에서 노드 1이 시스템에서 정상적으로 떠나면 그림 2의 (b)와 같이 된다. 노드 1의 자신이 관리하던 자원을 자신의 successor인 노드 3에게 전송하고 시스템을 떠나게 된다. 그림 2의 (b)를 보면 노드 3이 관리하는 자원의 키에 1이 추가된 것을 볼 수 있다. 노드가 떠날 때도 마찬가지로 시스템에 남아있는 노드들은 노드 1이 시스템을 떠난 사실을 자신들의 라우팅 테이블에 반영한다. 그림 2의 (b)에서 각 노드들의 라우팅 테이블의 진하게 표시된 부분이 노드 1이 시스템을 떠난 후에 업데이트된 부분이다.

2.4 Chord시스템의 문제점

그림 3은 노드 a에서 노드 h를 찾아갈 때 홉 경로를 보여주는 그림이다. 그림 3의 (a)는 실제 네트워크에서 Chord 시스템의 탐색 알고리즘에 따라 다음 노드로 이동할 때 이동 경로를 보여주는 그림이다.



(a) 그림 1에서 노드 6이 새로 참여한 후 변화된 라우팅 테이블



(b) (a)에서 노드 1이 정상적으로 떠난 후에 변화된 라우팅 테이블

그림 2 그림 1에서 노드 6이 조인되고 그 후에 노드 1이 정상적으로 시스템을 떠날 때 라우팅 테이블의 변화 과정

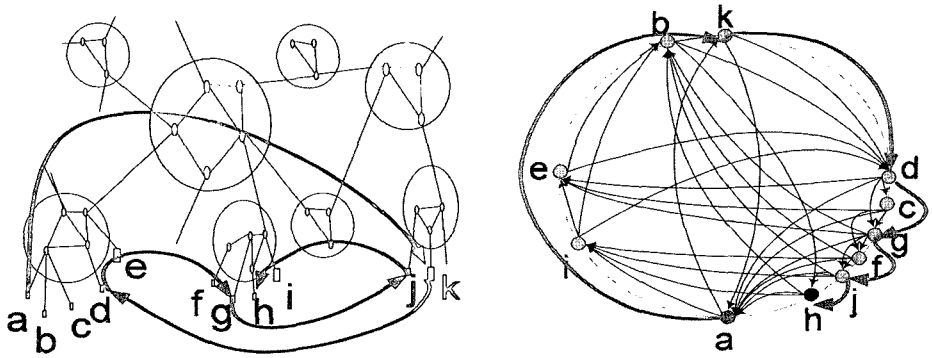


그림 3 Chord 시스템에서 자원 탐색 과정과 실제 네트워크에서 라우팅하는 과정

그림 3과 같이 Chord 시스템은 기존의 네트워크 위에 가상의 오버레이 네트워크를 형성하기 때문에 탐색 과정에서 여러 노드를 경유할 경우 실제 네트워크에서 여러 토폴로지를 경유할 수 있기 때문에 물리적인 지연

이 증가할 수 있다. 이와 같이 Chord는 실제 네트워크 정보를 고려하지 않기 때문에 자원 탐색의 효율이라는 측면에서 제한을 가지고 있으며 전체 시스템 성능에도 영향을 미칠 가능성이 높다.

본 논문에서 제안하는 TP2P는 토폴로지 정보를 이용하여 물리적으로 가까운 노드를 하나의 서브넷으로 만들고 각 노드는 자신의 서브넷에 있는 노드들의 글로벌 정보를 이용하여 자원을 탐색하므로 Chord 보다 탐색 홉 수가 감소하고 다음 홉이 자신의 서브넷으로 이동할 확률을 높여서 물리적 지연을 감소시킨다.

3. TP2P 설계

3.1 TP2P의 구조

TP2P는 모든 노드로 이루어진 글로벌 네트워크(Global network)와 물리적으로 가까운 노드들로 이루어진 여러 개의 서브넷(서브넷)으로 이루어진 시스템이다. 각 서브넷은 하나의 토폴로지로 이루어진다. 각 노드들은 글로벌 네트워크에 존재하고 그와 동시에 하나의 서브넷에 속해 있다. 각 노드는 각 네트워크에서 자신의 식별자를 가지기 위해 자신의 IP주소를 2개의 해쉬 함수(h1, h2)를 사용하여 global-nodeID와 local-nodeID를 구한다. 해쉬 함수는 SHA-1과 같은 해쉬 함수를 사용하며 global-nodeID는 글로벌 네트워크에서 원하는 자원을 탐색하거나 새로운 노드가 글로벌 네트워크에 조인할 때 사용하는 식별자이고, local-nodeID는 각 노드가 속해 있는 서브넷에서 원하는 자원을 탐색하거나 새로운 노드가 서브넷에 조인할 때 사용하는 식별자이다. TP2P는 ping을 통해서 노드 사이의 물리적인 거리를 측정하고 해당 서브넷에 조인한다. 각 서브넷은 Chord의 탐색 알고리즘을 사용하지만 글로벌 네트워크에서는 Chord의 자원 탐색 알고리즘과 global-nodeID Table을 이용하여 자원을 탐색한다.

그림 4는 크기가 16인 원형 식별자 공간을 가지는 글로벌 네트워크에 각 크기가 4인 원형 식별자 공간을 가지는 4개의 서브넷으로 이루어진 TP2P 시스템을 나타낸 그림이다. 각 노드는 글로벌 네트워크에 속해 있으며 또한 4개의 서브넷 중 하나의 서브넷에 속해 있다. 각 서브넷에는 반드시 한 개의 슈퍼 피어가 있으며 시스템에 조인을 원하는 노드가 슈퍼 피어에 ping을 함으로써 물리적 거리를 측정한다. Ping을 통해 나온 값이 임계값(threshold) 보다 작으면 그 슈퍼 피어의 서브넷에 조인한다.

그림 5는 일반 노드가 시스템을 유지하기 위해 알고 있어야 할 정보들이다. 각 노드가 유지하는 정보는 글로벌 네트워크와 서브넷에서 자원을 탐색과 시스템을 회복하기 위해 사용된다. 각 테이블의 기능은 다음과 같다.

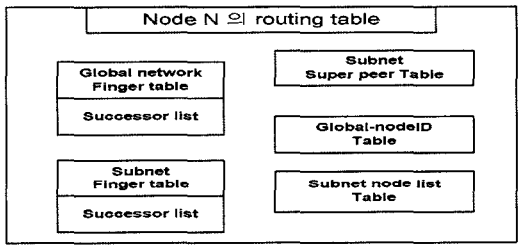


그림 5 노드 N 유지하는 정보

• Finger table: 2개의 라우팅 테이블은 글로벌 네트워크과 서브넷에서 독립적으로 사용된다. 테이블 구성

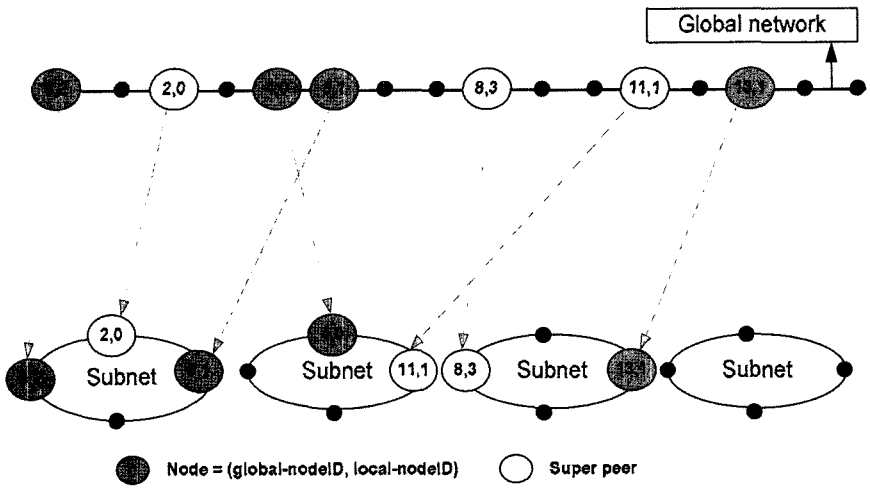


그림 4 TB-Chord의 구조

방법은 Chord 시스템의 탐색 알고리즘에 따른다.

- Successor list: 2개의 successor list는 글로벌 네트워크와 서브넷에서 독립적으로 사용되고 시스템에 존재하는 노드가 비정상적으로 시스템을 떠날 때 각 네트워크를 회복하기 위해 사용된다. Successor list의 크기는 log N이다.
- 서브넷 Super peer Table: 각 슈퍼 피어는 자신의 속해 있는 서브넷의 노드들을 관리한다. 새로 조인한 노드는 자신이 속해 있는 슈퍼 피어에게 자신의 상태 정보(컴퓨팅 파워, 대역폭 등)를 보내기 위해 유지하는 정보이다. 또한 시스템에서 정상적으로 탈퇴할 때도 슈퍼 피어에게 leave 메시지를 보내기 위해 사용되는 정보이다.
- Global-nodeID Table: 자원 탐색을 효율적으로 하기 위해 각 노드가 자식의 서브넷에 존재하는 노드들의 Global-nodeID를 유지하는 정보이다. P2P시스템은 노드들의 조인과 시스템을 떠나기 때문에 서브넷의 모든 노드 정보를 유지하는 것은 네트워크에 큰 부담을 준다. 그래서 본 논문이 제안한 TP2P는 일정한 크기만큼만 정보를 유지한다. 즉 각 노드는 자식의 서브넷 Finger Table의 Successor 노드들이 유지하는 Successor List에 속한 노드들의 Global-nodeID의 정보를 유지한다.
- Super peer list Table: 슈퍼 피어가 시스템을 떠날 때 슈퍼 피어 교체 비용이 많이 들기 때문에 보다 성능이 우수한 성능을 가지는 노드를 슈퍼 피어로 유지하기 위해 사용하는 테이블이다. Super peer list Table의 크기는 log N이다. 각 노드가 시스템에 조인할 때 자신의 상태 정보를 슈퍼 피어에게 전송한다. 전송받은 슈퍼 피어는 자신의 테이블의 노드들과 비교하여 테이블을 갱신한다. 또한 테이블에 있는 노드들은 다음 슈퍼 피어의 후보가 된다. 테이블에 있는 모든 노드들은 Super peer list Table의 정보를 동일하게 유지한다.

그림 6은 슈퍼 피어가 시스템에서 슈퍼 피어의 기능을 하고 자신의 서브넷을 관리하기 위한 정보들이다. 슈퍼 피어가 유지하는 정보들 중 Neighbor Super peer Table을 제외한 나머지 4개의 테이블은 일반 노드와 같다. Neighbor Super peer Table은 글로벌 네트워크에서 자신과 가장 이웃한 슈퍼 피어의 정보를 양쪽으로 하나씩 유지한다. 새로운 노드가 시스템에 조인할 때 슈퍼 피어와 물리적 거리를 측정하는데 만약 임계값보다 클 경우 다음 슈퍼 피어로 이동하기 위해 사용되는 정보이다. 슈퍼 피어는 자신의 서브넷에 새로운 노드가 조인하면 상태 정보를 받는데 만약 슈퍼 피어보다 성능이 우수한 노드이면 새로 조인한 노드가 슈퍼 피어가 된다.

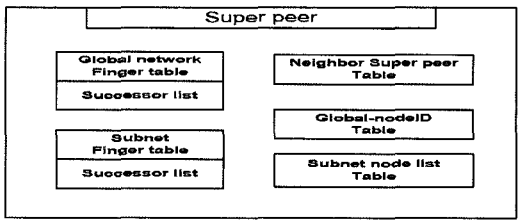


그림 6 슈퍼 피어가 유지하는 정보

3.1 노드 조인

TP2P 시스템에 조인을 원하는 노드는 시스템에 존재하는 bootstrap 노드를 반드시 알고 있다. 그림 7은 새로운 노드가 시스템에 조인하는 과정을 보여주는 그림이다.

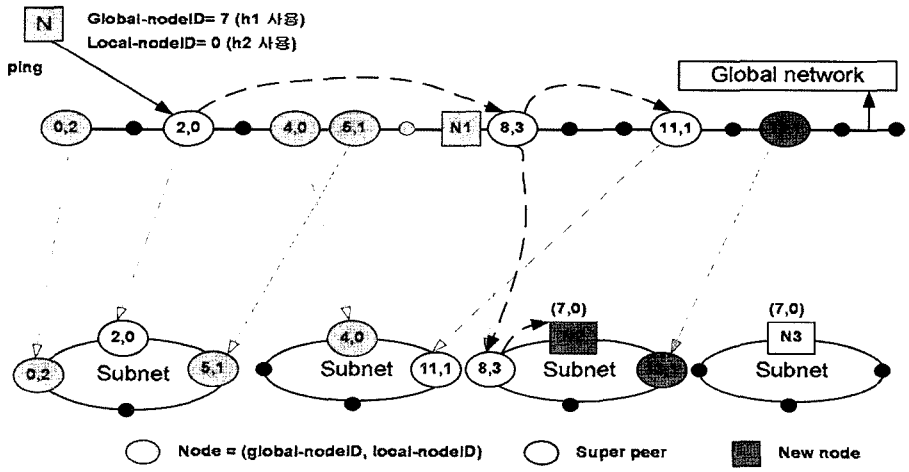


그림 7 새로운 노드의 조인

TP2P 시스템에 조인을 원하는 노드는 자신 IP 주소를 2개의 해쉬 함수(h1, h2)로 해쉬하여 global-nodeID와 local-nodeID를 구한다. 먼저 global-nodeID를 사용하여 글로벌 네트워크에 global-nodeID와 일치하는 위치에 조인한다. 그림 7에서 N1의 상태이다. 다음으로 새로운 노드는 ping을 이용하여 슈퍼 피어(2,0)와 물리적 거리를 측정한다. 물리적 거리를 측정한 값이 임계값보다 작으면 그 슈퍼 피어가 속해 있는 서브넷에 local-nodeID와 일치하는 위치에 조인한다. 만약 물리적 거리를 측정한 값이 임계값보다 크면 슈퍼 피어의 Neighbor Super peer table을 참조하여 이웃한 슈퍼 피어의 정보를 구한 뒤, 똑같은 과정을 거친다. 그림 7에서 N2의 경우이다. 이 과정을 반복하여 조인을 원하는 새로운 노드는 자신이 있어야 할 서브넷을 찾을 수 있다.

시스템에 속한 모든 슈퍼 피어와 물리적 거리를 측정한 값이 임계값보다 클 때는 2가지의 상황에 따라 다르게 조인된다. 첫 번째로 시스템이 가질 수 있는 서브넷 수가 최대가 아닐 때는 시스템에 조인 노드는 새로운 서브넷을 만들어 그 서브넷의 슈퍼 피어가 된다. 그림 7에서 N3의 경우이다. 두 번째로 시스템이 최대로 가질 수 있는 서브넷이 존재하면 조인을 원하는 새로운 노드는 임계값을 올려 다시 조인한다.

TP2P 시스템에 새로운 노드가 조인하면 새로운 노드는 자신의 상태 정보를 자신이 속해 있는 서브넷의 슈퍼 피어에게 보낸다. 상태 정보를 받은 슈퍼 피어는 자신의 서브넷 node list table 정보와 비교하여 테이블을 갱신한다. 또한 서브넷 node list table에 속한 모든 노드에 update 메시지를 전송한다. 메시지를 받은 노드들은 자신의 서브넷 node list table 을 갱신한다. 만약 새로 조인한 노드의 상태 정보가 슈퍼 피어보다 성능이 좋으면 새로 조인한 노드는 자신의 서브넷의 슈퍼 피어로 교체된다. 새로 슈퍼 피어가 된 노드는 서로 이웃한 슈퍼 피어에게 자신이 새로운 슈퍼 피어라는 메시지를 보낸다. 메시지를 받은 슈퍼 피어는 자신의 Neighbor super peer table의 정보를 갱신한다.

3.2 데이타 삽입

데이타의 삽입 노드는 데이타의 키를 2개의 해쉬 함수(h1, h2)를 사용하여 global-nodeID와 local-nodeID를 구한다. 노드는 먼저 local-nodeID를 이용하여 데이타의 삽입 노드의 서브넷에서 Chord의 탐색 알고리즘을 사용하여 local-nodeID를 관리하는 노드에게 데이타를 삽입한다. 그리고 global-nodeID를 이용하여 글로벌 네트워크에서 TP2P의 탐색 알고리즘을 사용하여 global-nodeID를 관리하는 노드에게 데이타를 삽입한다.

데이타를 서브넷과 글로벌 네트워크에 삽입함으로써 같은 서브넷에 있는 노드가 같은 데이타를 찾을 때 자신의 서브넷에서 찾을 확률을 높일 수 있고 이로 인해 홉 수와 물리적인 지연을 Chord 시스템에 비해 줄일 수 있다.

3.3 데이타 탐색

데이타의 탐색 노드는 데이타의 키를 2개의 해쉬 함수(h1, h2)를 사용하여 global-nodeID와 local-nodeID를 구한다. 먼저 데이타의 탐색 노드는 Chord의 자원 탐색 알고리즘에 따라 local-nodeID를 이용하여 자신이 속해 있는 서브넷에서 local-nodeID를 관리하는 노드를 찾는다. 만약 원하는 데이타가 있으면 local-nodeID를 관리하는 노드에게 접속하여 원하는 데이타를 받는다. 자신의 서브넷에 원하는 데이타가 없을 때는 글로벌 네트워크에서 데이타를 찾는다. 글로벌 네트워크에서 자원을 탐색할 때는 탐색 노드의 Global-nodeID Table과 글로벌 라우팅 테이블 사용하여 자원 탐색을 한다.

자원 탐색 노드는 자신의 Global nodeID Table에서 키를 해쉬한 Global-nodeID와 가장 가까운 Global-nodeID를 찾고 Chord시스템의 탐색 알고리즘에 따라 다음 홉으로 이동할 노드의 Global-nodeID를 구한다. 2개의 Golbal-nodeID를 비교하여 키를 해쉬한 Global-nodeID와 가까운 노드로 메시지를 전송한다.

그림 8은 네트워크에서 Chord 알고리즘을 사용하여 탐색하는 방법과 TP2P 알고리즘을 사용하여 탐색하는 과정을 보여준다. 그림 9는 데이타 탐색을 원하는 globa-nodeID 2와 local-nodeID 0인 노드가 유지하는 라우팅 정보를 보여준다. 전체 TP2P시스템은 크기가 32 인 원형 식별자 공간을 가지는 글로벌 네트워크와 크기가 8인 원형 식별자 공간을 가지는 4개의 서브넷으로 이루어진 시스템이다. TP2P의 자원 탐색 과정은 다음과 같다.

1. 자원 탐색 노드 (2,0)가 데이타의 키를 해쉬하여 나온 값이 global-nodeID = 27, local-nodeID = 1이고 자신의 서브넷에 원하는 데이타가 없다고 가정한다.
2. 노드 (2,0)는 자신의 Global-nodeID Table에서 27과 가장 가까운 global-nodeID를 구한다. 그림 9에서는 24이다.
3. 노드 (2,0)는 Chord 시스템의 자원 탐색 알고리즘에 따라 글로벌 라우팅 테이블을 참조하여 다음 홉인 노드의 global-nodeID는 19이다.
4. 노드 (2,0)는 2개의 테이블에서 구한 값 중 키를 해쉬한 global-nodeID 27과 가장 인접한 노드인 노드 (24,1)로 탐색 메시지를 보낸다.
5. 메시지를 받은 노드는 위 과정을 반복하여 다음 홉인 노드로 메시지를 전송한다.

TP2P의 탐색 알고리즘은 Chord 시스템보다 평균 홉 수를 줄일 수 있으며 자신의 서브넷에 있는 노드에게

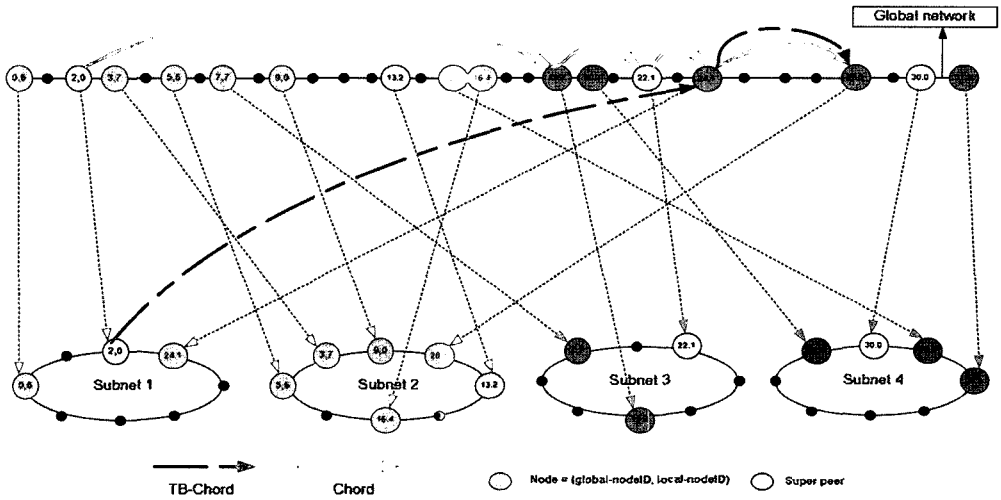


그림 8 global-nodeID = 2 local nodeID = 0인 노드에서 키를 해쉬한 global-nodeID 27를 탐색하는 과정

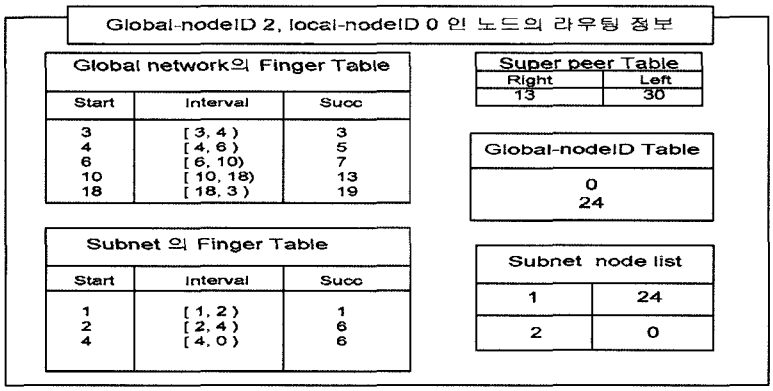


그림 9 global node 2, local nodeID 0인 노드의 라우팅 테이블

메시지를 전송할 확률이 높기 때문에 물리적 지연도 자연스럽게 줄일 수 있다. 자세한 내용은 4장에서 설명하겠다.

3.4 정상적 탈퇴

TP2P 시스템에서 일반 노드가 정상적으로 탈퇴할 때 Chord 시스템의 탐색 알고리즘과 같은 방법으로 자신의 successor 에게 leave 메시지를 보내고 자신이 관리하는 데이타를 successor에게 전송한다. 각 노드는 글로벌 네트워크에 속하면서 서브넷에 속하기 때문에 글로벌 네트워크와 서브넷의 successor에게 자신이 관리하는 글로벌 네트워크의 자원과 서브넷의 자원을 전송한 후 시스템을 떠난다. 또한 자신이 속해 있는 서브넷의 슈퍼 피어에게 leave 메시지를 전송하고 메시지를 받은 슈퍼 피어는 서브넷 node table을 갱신한다. 서브넷 노드 테이블에 등록된 모든 노드들에게 메시지를 보내 각 노드들은 자신의 서브넷 node table 을 갱신한다.

슈퍼 피어가 시스템을 정상적으로 탈퇴할 때 서브넷 노드 테이블에 있는 노드 중 가장 성능이 좋은 노드가 슈퍼 피어가 되며 자신의 서브넷에 있는 모든 노드와 Neighbor super peer Table에 있는 노드에게 자신이 새로운 슈퍼 피어라는 메시지를 전송하고 메시지를 받은 노드는 자신의 정보를 갱신한다.

4. 실 험

이 장에서는 TP2P와 Chrod의 성능 비교를 위해 두 시스템의 메시지 전송 지연을 수학적인 방법을 이용해서 비교하고 TP2P의 시스템 성능을 시뮬레이션을 통해 검증하였다.

4.1 Chord와 TP2P의 성능 비교

Chord와 TP2P에서의 글로벌 네트워크에서 원하는 데이타를 찾기 위해 라우팅할 때 총 메시지 전송 지연 비용을 수식으로 표현하면 다음과 같다.

시스템 환경은 다음과 같다.

- TP2P 시스템의 원형 식별자 공간의 크기: N
  서브넷의 수: M
- 하나의 서브넷에 존재하는 평균 노드의 수: $L_M = \frac{N}{M}$
- Chord 시스템에서의 평균 라우팅 홉: Hc
- TP2P 시스템에서 평균 라우팅 홉 : Ht
- 서브넷에서 평균 메시지 지연 : DL
- 글로벌 네트워크에서 평균 메시지 지연 : DG
- 서브넷과 글로벌 네트워크 간의 평균 메시지 지연의 비율은 상수배라 가정 : $D_G = C \cdot D_L$
- 각 노드의 global-nodeID 테이블 크기: L
- 각 노드의 글로벌 라우팅 테이블의 크기: R

Chord 시스템에서 라우팅 알고리즘의 총 지연 비용은 다음과 같이 표현된다.

- 메시지를 전송할 노드를 자신의 서브넷에서 찾을 확률: $\frac{L_m}{N}$
- 메시지를 전송할 노드를 다른 서브넷에서 찾을 확률: $\frac{N-L_m}{N}$

$$\left(\frac{L_m}{N}\cdot D_L + \frac{N-L_m}{N}\cdot D_G\right)^{Hc} = \left(\frac{L_m}{N}\cdot D_L + \frac{N-L_m}{N}\cdot C\cdot D_L\right)^{Hc}$$
$$= \left(\frac{L_m + (N-L_m)\cdot C}{N}\cdot D_L\right)^{Hc} = \left(\frac{L_m + C\cdot N - C\cdot L_m}{N}\cdot D_L\right)^{Hc}$$
$$= \left(\frac{C\cdot N + (1-C)\cdot L_m}{N}\cdot D_L\right)^{Hc} \approx (C\cdot D_L)^{Hc}$$

TP2P 시스템에서 라우팅 알고리즘의 총 지연 비용은 다음과 같이 표현된다.

- 메시지를 전송할 노드를 자신의 서브넷에서 찾을 확률 : P_L
- Global Routing Table에서 정해진 노드가 자신의 서브넷에 있을 확률은 Chord 시스템과 동일하므로: $\frac{L_M}{N}$
- Global-nodeID Table에 있는 노드가 Global Routing Table에서 정해진 노드보다 목적 노드에 가까이 있을 확률: P_T

$$P_L = \frac{L_M}{N} + \frac{N-L_M}{N}\times P_T$$

Chord 시스템에서 임의의 한 노드를 탐색할 때 Routing Table의 Interval의 크기가 각각 다르기 때문에 그 노드가 그 간격에 위치할 확률은 $\frac{1}{2^k}$ 이다. (k는 finger table의 아래에서 몇 번째 행인지를 나타낸다.) 그러므로 각 Interval에서 successor와 임의의 한 노드 사이의 평균 거리는 $\frac{1}{2^{2k}}$ 이 된다. 이 때 Global nodeID Table에 있는 각각의 노드가 이 범위 안에 있을 확률:

$$PE = \frac{1}{2}\sum_{i=1}^{\log N}(1/4)^i \approx \frac{1}{6}$$

그러므로 1에서 Global nodeID Table에 있는 모든 노드가 그 범위 안에 있지 않을 확률을 뺀 값이 $P_T = 1-(\frac{5}{6})^L$

$$P_L = \frac{L_M}{N} + \frac{N-L_M}{N}\times P_T$$
$$P_G = 1-(\frac{L_M}{N} + \frac{N-L_M}{N}\times P_T)$$

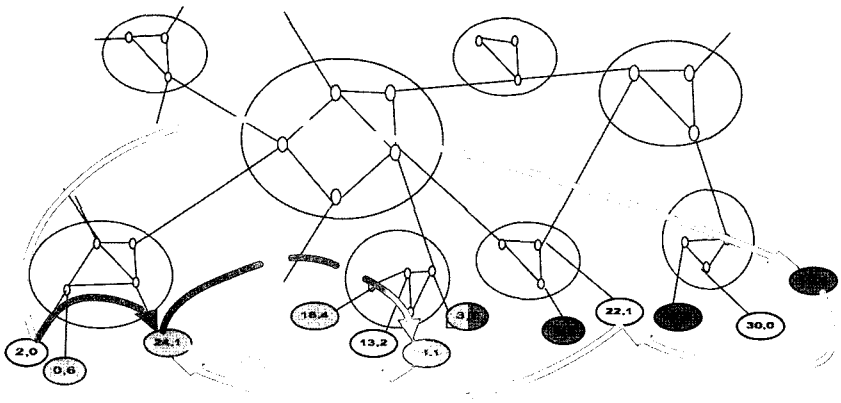
$$\left((1-(\frac{L_M}{N}+\frac{N-L_M}{N}\times P_T))\cdot D_G + (\frac{L_M}{N}+\frac{N-L_M}{N}\times P_T)\cdot D_L\right)^{Ht}$$
$$= \left(\frac{N-L_M-NP_T+L_MP_T}{N}\cdot D_G + (\frac{L_M(1-P_T)}{N}+P_T)\cdot D_L\right)^{Ht}$$
$$= \left((1-P_T-\frac{L_M(1-P_T)}{N})\cdot C\cdot D_L + (\frac{L_M(1-P_T)}{N}+P_T)\cdot D_L\right)^{Ht}$$
$$\approx \left((C(1-P_T)+P_T)\cdot D_L\right)^{Ht}$$

데이타를 탐색할 때 소요되는Chord와 TP2P의 평균 홉 수 역시 다음에 나올 실험을 통해서 TP2P가 Chord보다 적다는 것을 증명하였다. (Ht < Hc) 그리고 수식에서 보듯이 한 번의 홉에 소요되는 지연 역시 TP2P가 Chord에 비해 평균적으로 작다. 이것은 TP2P에서의 각각의 노드가 서브넷에 있는 여러 노드들의 Global-nodeId 정보를 유지하기 때문에 다음 홉이 자신의 서브넷이 될 확률(Pt) 때문이다. 이 확률에 따라 자원 탐색할 때 다음 홉이 자신의 서브넷에서 찾기 때문에 물리적인 지연을 줄일 수 있다. 즉 TP2P의 서브넷은 토폴로지와 일대일로 매핑되므로 로컬 네트워크에서 자원을 탐색하는 것과 비슷한 성능을 가질 수 있다.

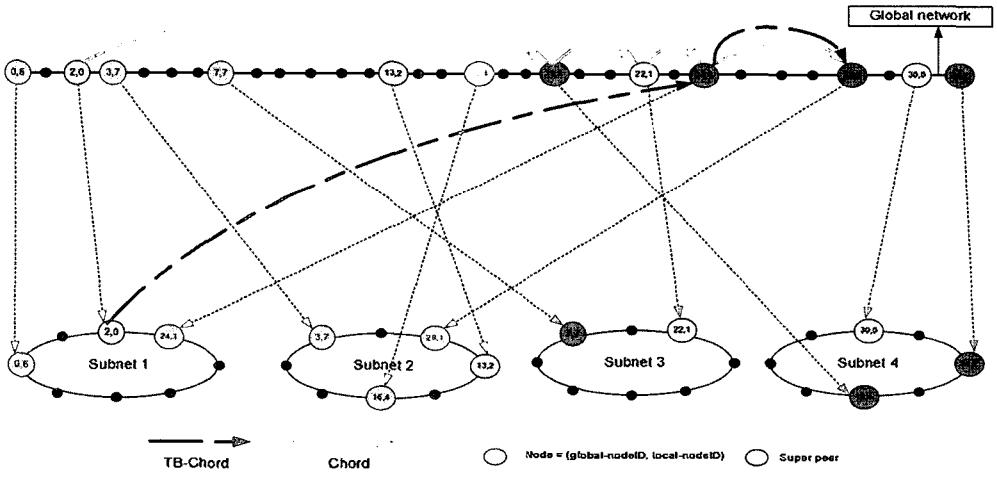
그림 10은 실제 네트워크 그림 10(a)와 TP2P시스템에서 탐색 과정을 매핑시키는 그림이다. 각 토폴로지는 하나의 서브넷이다. 그림 10(b)는 오버레이 네트워크에서 노드 (2,0)가 목적지 노드 (28,1)를 찾아 갈 때 메시지 전송 과정을 나타낸다. Chord는 4홉만에 목적지 노드를 찾고 TP2P는 2홉만에 찾는다. 그림 10(a)에서는 Chord는 4개의 토폴로지를 경유하는 반면에 TP2P는 자신의 서브넷 안에 존재하는 노드에게 메시지가 전송되므로 물리적인 지연도 줄어들 것이라 예상할 수 있다.

4.2 자원 탐색 실행후 홉 수에 따른 탐색 수의 분포

시스템 구조는 다음과 같이 가정하였다. 물리적 네트



(a) 실제 네트워크



(b) TB-Chord

그림 10 노드 (2,0)에서 키를 해쉬한 값 27를 탐색하는 과정

워크의 구조는 256개의 AS(Autonomous System)를 가상으로 구성하고 하나의 AS를 하나의 서브넷으로 간주한다. 오버레이 네트워크는 크기가 131072(217)인 원형 식별자 공간을 가지는 글로벌 네트워크와 크기가 512인 원형 식별자 공간을 가지는 256개의 서브넷으로 이루어진 TP2P 시스템에서 65536(216)개의 노드가 조인되어 있다. 각 서브넷의 노드의 수는 동일하고 Chord 시스템도 동일한 크기의 원형 식별자 공간에 동일한 노드의 수가 존재한다. 그림 11은 이 시스템에서 자원 탐색을 10000회 실행한 후 각 탐색이 자신이 원하는 데이타의 successor에게 도달할 때까지의 홉 수를 측정한 결과이다.

TP2P의 자원 탐색 알고리즘에 따라 노드는 Global-nodeID Table과 라우팅 테이블을 이용하여 원하는 데이타를 관리하는 successor에 가장 인접한 노드에게 메

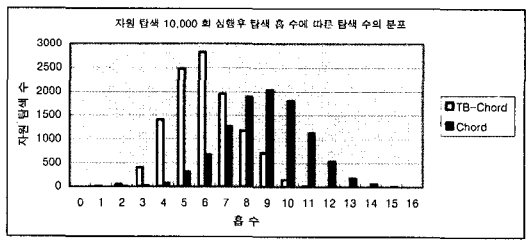


그림 11 탐색 수에 따른 평균 홉 수

시지를 전달하기 때문에 평균 홉 수가 Chord 보다 적은 것을 알 수 있다. 그림 9를 보면 TP2P는 홉 수가 4에서 6사이에 많은 자원 탐색 메시지가 집중되어 있으며 Chord는 홉 수가 7에서 11사이에 넓게 분산되어 있다. 이는 곧 TP2P가 오버레이 네트워크 상에서도 Chord 보다 더 효율적으로 탐색한다는 것을 의미한다.

4.3 노드 수와 평균 홉 수의 관계

이 장에서는 시스템의 노드 수의 증가에 따른 평균 홉 수를 측정하였다. P2P의 주요 장점 중 하나는 서버/클라이언트 시스템에서 제기되는 확장성 문제를 해결한다는 것이다. 확장성이란 노드의 수가 증가하더라도 시스템의 성능에 큰 문제가 없어야 한다는 것을 의미한다. 본 논문은 이 실험을 통해서 일반적인 P2P 시스템보다 확장성이 우수함을 보여 주고 있다. 그림 12는 크기가 131072(217)인 원형 식별자 공간을 가지는 글로벌 네트워크와 크기가 512인 원형 식별자 공간을 가지는 256개의 서브넷으로 이루어진 TP2P 시스템에서 노드 수가 212에서 216로 증가함에 따른 평균 홉 수를 나타낸 그림이다. 각 서브넷의 노드의 수는 동일하고 Chord 시스템도 동일한 크기의 원형 식별자 공간에 동일한 노드의 수가 존재한다.

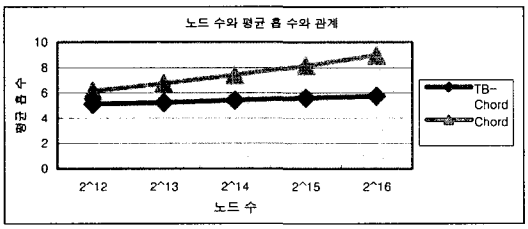


그림 12 노드 수의 증가에 따른 평균 홉 수

그림 12에서 TP2P는 노드 수가 증가해도 평균 홉 수의 5에서 6사이로 거의 변화가 없는 반면 Chord는 노드의 수가 2배가 될 때 평균 홉 수가 0.75배 정도 증가하는 것을 볼 수가 있다. 이는 TP2P가 Chord 보다 확장성 측면에서 보다 낫다는 것을 보여준다.

메시지 전송 지연에 대한 분석을 통해 물리적 지연 측면에서 TP2P가 Chord보다 우수하다는 것을 알 수 있었다. 또 실험을 통해 TP2P가 Chord에 비해 홉 수가 평균 25% 정도 줄어든다는 것을 알 수 있었다. 즉, TP2P는 Chord에 비해 물리적 지연과 탐색 홉 수라는 두 가지 측면에서 모두 좋은 성능을 가지고 있으므로 전체 시스템의 탐색 효율을 상당히 개선한 시스템이라고 이야기할 수 있다.

5. 결 론

인터넷 접속 환경의 발달과 시스템 및 네트워크 성능 향상과 컴퓨터의 성능의 발달로 인해 P2P 시스템이라는 새로운 시스템이 등장하였다. 아직도 대부분의 시스템들은 클라이언트/서버(Client/Server) 모델을 사용하지만 클라이언트/서버 시스템은 서버에 대한 과부하 문제, 즉 확장성(scalability)의 문제를 가지고 있다. P2P 시스템은 클라이언트/서버의 확장성 문제를 해결하고 클라이언트의 자원을 효율적으로 활용하는 것을 목적으로 제안되었다. 그 중에서도 DHT 기반 P2P 시스템들은 Napster나 Gnutella에서 완전히 해결하지 못한 확장성 문제를 극복하기 위해 제안되었다.

현재까지 잘 알려진 DHT 기반P2P 시스템들은 응용계층에서 메시지 경로를 결정하기 때문에 제한된 홉 수 안에서의 탐색을 보장하지만 물리적인 토폴로지를 고려하지 않고 경로를 결정하기 때문에 여러 토폴로지를 경유할 가능성이 있어 물리적인 지연이 매우 클 수 있다는 단점이 있다.

이런 문제를 해결하기 위해 TP2P는 피어의 능력을 고려하여 슈퍼 피어를 결정하고 이것이 비정상적으로 시스템을 떠날 확률을 줄임으로써 시스템을 안정화시켰다. 각 노드는 데이타를 삽입하거나 탐색할 때 서브넷에 데이타를 복제하여 저장하므로 서브넷에 있는 다른 노드가 같은 데이타를 찾을 때 서브넷에서 찾을 확률을 높임으로써 탐색의 효율성을 증가시킨다. 또한 서브넷에서 찾지 못하여 글로벌 네트워크에서 탐색할 때에도 자신의 Global-nodeID Table과 라우팅 테이블을 이용하므로 홉 수를 줄이고 서브넷에 속하는 노드에게 메시지를 보낼 수 있는 가능성을 높임으로써 실질적인 물리적 지연을 감소시킨다. TP2P는 홉 수와 물리적 토폴로지, 두 가지를 모두 고려함으로써 실제적인 지연을 줄이기 위해 제안되었다.

참 고 문 헌

[1] Napster. http://www.napster.com
[2] Gnutella. http://www.gnutella.com
[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," In Proceedings of SIGCOMM, ACM, August 2001.
[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrkshnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," In Proceedings of SIGCOMM, ACM, August 2001.
[5] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peerto-peer Systems," In Proceedings of IFIP/ACM Middleware 2001, November 2001.
[6] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," Tech. Rep. UCB/CSD-01-1141, UC Berkeley, EECS, 2001.
[7] Shin, S. Lee, G.. Lim, H. Yoon, and J. S. Ma, "Grapes: Topology-based Hierarchical Virtual Network for Peer-to-peer Lookup Services," In Proceedings of the International Conference on Parallel Processing Workshops (ICPPW' 02), 2002.

[8] B. Y. Zhao, Y. Duan, and L. Huang, "Brocade: Landmark Routing on Overlay Networks," In Proceedings of the 1st International Workshop on Peer-to-Peer Systems, March 2002.
[9] B. Yang and H. Garcia-Molina, "Designing a super-peer network," ech. Rep. 2002-13, Stanford University, 2002.

차 봉 관
2003년 경성대학교 컴퓨터공학과(학사) 2006년 부산대학교 대학원 컴퓨터공학과(석사). 2006년~현재 동양시스템즈 생명 차세대팀 TA. 관심분야는 P2P 컴퓨팅, P2P 보안

한 동 윤
2004년 부산대학교 정보컴퓨터공학부(학사). 2006년 부산대학교 대학원 컴퓨터공학과(석사). 2006년~현재 부산대학교 대학원 컴퓨터공학과 박사과정. 관심분야는 P2P 컴퓨팅 , 홈네트워크, 시맨틱 웹

손 영 성
1995년 부산대학교 전자계산학과(학사) 1997년 부산대학교 대학원 전자계산학과(석사). 2006년 부산대학교 대학원 컴퓨터공학과(박사). 1997년~1999년 시스템공학연구소(SERI) 네트워크컴퓨팅연구부 연구원. 1999년~현재 한국전자통신연구원(ETRI) 디지털홈연구단 선임연구원. 관심분야는 홈네트워크 미들웨어, 유비쿼터스 컴퓨팅, P2P 컴퓨팅

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