Journal of The Korea Society of Computer and Information
Vol. 17, No. 12, December 2012
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci/2012.17.12.111

DHT 기반 P2P 시스템을 위한 그룹 라우팅 알고리즘에 관한 연구

박 용 민*

A Study on the Group Routing Algorithm in DHT-based Peer-to-Peer System

Yong-Min Park*

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 Chord, Pastry, Tapestry 등이 제안되었으나 이 시스템들은 성능 효율이 log 으로 제한되어 있다. 이러한 제한된 성능 효율을 개선하기 위해 본 논문에서는 그룹 단위의 라우팅 알고리즘을 제안한다. 제안하는 알고리즘은 node-to-group 라우팅 알고리즘으로 원형 주소 공간을 그룹으로 나누고 각 그룹을 대표하는 포인터(Pointer)라는 개념을 사용하여, 포인터를 기반으로 라우팅이 이루어지는 알고리즘이다. 알고리즘의 성능을 평가하기 위해 P2P 시스템의 대표적인 알고리즘인 chord와 라우팅을 위한 평균 홉 수, 라우팅 테이블 크기, 전송 지연에 관해 비교 분석 하였으며, 결과적으로 비교 항목에 대해 성능이 향상되었음을 실험을 통해 확인 하였다

▸ Keywords : 피어 투 피어, 분산 해쉬 테이블, 오버레이 네트워크, 홉 카운트, 라우팅 테이블

Abstract

As the P2P System is a distributed system that shares resources of nodes that participate in the system, all the nodes serve as a role of server and client. Currently, although systematic, structured P2P systems such as Chord, Pastry, and Tapestry were suggested based on the distributed hash table, these systems are limited to $\log_2 N$ for performance efficiency. For this enhanced performance efficiency limited, the article herein suggests group routing algorithm. The

• 제1저자 : 박용민 • 교신저자 : 박용민
• 투고일 : 2012. 10. 29, 심사일 : 2012. 11. 21, 게재확정일 : 2012. 12. 8.
• 삼육보건대학교 의료정보시스템과(Dept. of Medical Information System, Sahmyook Health University College)
• 본 논문은 2012년도 삼육보건대학교 학술연구지원에 의해 연구되었음

www.kci.go.kr

suggested algorithm is a node-to-group routing algorithm which divides circular address space into groups and uses a concept of pointer representing each group, which is an algorithm where routing is performed based on pointer. To evaluate algorithm performance, a comparative analysis was conducted on average hops, routing table size, and delayed transmission for chord and routing, a signature algorithm in P2P systems. Therefore, enhanced performance is verified for comparative items from the simulation results.

▸ Keywords : P2P, DHT(Distributed Hash Table), Overlay network, Hop count, Routing table

I. 서 론

인터넷이 보편화 되어 일상생활에까지 그 쓰임새가 확대되면서 등장한 웹2.0은 사용자의 데이터 가공 및 처리에 대한 욕구를 유발시키는 동기를 마련해 주었다.[1] 그리하여 인터넷상에서 사용자에 의한 데이터 가공 및 처리 등이 인터넷의 중요한 쓰임새가 되면서, 대부분의 인터넷 기반 응용은 이미지 혹은 비디오 중심의 UCC 기반이 현실이다.[2] 다시 말해, 인터넷 응용 및 데이터는 멀티미디어화 되면서 대용량 데이터와 넓은 대역폭을 점점 더 필요로 하고 있다는 사실이다. 이를 서버 기반의 시스템으로 구축할 경우, 네트워크 트래픽은 급속하게 증가하게 되어 서버 양방향 링크의 병목 현상은 피할 수 없는 문제점으로 대두되고 있으며, 현재의 한정적인 네트워크 자원으로써는 컴퓨팅 지연 및 전송 지연을 일으킬 수밖에 없는 실정이다.[3][4] 또한 멀티미디어 데이터를 연속적으로 전송 받아 재생하여야 하는 스트리밍과 같은 기술은 데이터를 처리하는 단말의 필수적인 기능으로 요구되고 있으나, 서버 기반의 멀티 유니캐스트 전송 방식으로는 네트워크 자원의 부족 현상으로 심각한 제약을 받게 되며, 확장성 및 효율성 문제로 비용 또한 비싸지게 된다는 문제점을 안고 있는 실정이다. 이러한 서버 링크의 병목 현상을 해결하기 위한 다양한 연구가 진행되어 왔으며, 그 중 하나가 응용 계층에서의 멀티캐스팅을 가능하게 하는 Peer-to-Peer (P2P) 오버레이 네트워킹이라고 할 수 있다.[5]

P2P는 기존의 인터넷에서 사용되던 클라이언트-서버 커뮤니케이션의 단방향 특성이 정보를 동시에, 그리고 유기적으로 교환하는 휴먼 커뮤니케이션에서의 상황에 적합하지 않기에 고안된 양방향 커뮤니케이션 모델이다. 이러한 P2P는 크게 unstructured P2P와 structured P2P의 두 가지로 분류할 수 있다. Unstructured P2P는 다시 3가지 형태로 나눠지고, 또한 전체 네트워크에 대한 정보들이 모든 노드들에 의해 관리되거나 한 노드에게 집중되는 반면에, DHT 기반의 structured P2P 시스템은 각각의 노드가 전체 네트워크가 아닌 부분적인 네트워크 정보를 유지, 관리하게 함으로써, unstructured P2P 시스템의 단점을 보완한 방법이다. 이러한 DHT 알고리즘을 사용하면 네트워크를 구성하고 있는 모든 노드들이 하나의 동일한 프로토콜에 따라 다양한 서비스를 제공할 수 있다. 예를 들면, 네트워크 상에 존재하는 다양한 콘텐츠들을 해싱(hashing) 함수를 이용하여 네트워크 상에 적절히 분산시키고, lookup latency를 최소화할 수 있다. 또한, 관리자가 직접 노드에 새로 참가하거나 떠나는 것을 처리할 필요가 없어 관리 비용을 줄일 수 있다는 장점이 있다. 이러한 DHT 알고리듬의 예로, Chord[6], Pastry[7], Tapestry[8] 등이 있다. 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 데이터들이 저장되어야 할 위치는 시스템에 의해 결정되어 오버레이 네트워크 상의 노드들에 분산 배치된다. 그러므로 이 시스템들은 새로운 데이터를 저장하고 데이터가 저장된 위치를 탐색하기 위해 각 노드가 제한된 위치 정보를 가지면서 특정 노드를 찾을 수 있는 탐색 알고리즘을 제시하고 있다.[9-11]

DHT 기반의 분산 시스템들의 성능 효율은 대부분 N개의 노드를 가진 네트워크에서 홉 수(Hop count)와 라우팅 테이블 크기(Routing table size)에 따라 $\log N$)의 시간 복잡도를 가진다. 본 논문에서 제안하는 알고리즘은 이러한 DHT 시스템에서의 시간 복잡도를 향상 시킨 그룹 라우팅 알고리즘을 제안한다. 제안하는 그룹 라우팅 알고리즘은 consistent hashing 기반의 node-to-group 라우팅 알고리즘으로 라우팅 효율을 높이기 위해 그룹을 세분화하여 라우팅 성능을 향상시킨다. 제안한 알고리즘은 기존의 알고리즘에 비

www.kci.go.kr

해 그룹 단위로 라우팅이 이루어지므로, 평균 홉 수와 라우팅 테이블 크기를 고려한 평균 지연 시간이 향상되는 것을 증명하고 그 성능을 시뮬레이션을 통해 평가한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 DHT 기반의 대표적인 알고리즘인 chord 연구를 검토하고, 3장에서는 본 논문에서 제안하는 그룹 라우팅 알고리즘에 대해 설명한다. 4장에서는 제안한 알고리즘의 성능을 실험을 통해 평가하고 5장에서 결론을 맺는다.

II. 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 데이터의 삽입과 탐색을 수행한다. Chord 시스템에서 시스템 내에 존재하는 각 노드는 시스템 내의 모든 노드의 정보를 유지하지 않고 다른 노드들에 대한 일정한 수의 라우팅 정보만 유지함으로써 시스템의 확장성을 제공한다. Chord는 크기의 원형 식별자 공간을 사용하며 각 노드는 IP주소를 160비트의 SHA-1 해쉬 함수로 해쉬하여 nodeID를 할당받아 원형 식별자 공간의 해당 nodeID에 위치한다. 데이터의 위치 정보는(Key, value)쌍으로 표현되며 데이터가 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 procecessor는 0이다. 노드가 시스템에서 비정상적으로 떠났을 때 시스템을 복원하기 위해 log 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 successor노드라 부른다. 그림 1에서 키 6의 successor노드는 nodeID가 1인 노드이다. 각 노드는 전체 네트워크 상의 노드에 대한 정보를 분산된 동적 환경에 효과적으로 만족시키기 위해 라우팅 테이블을 유지한다. chord에서 라우팅 테이블 생성 규칙은 표 1과 같다. 라우팅 테이블을 바탕으로 이진 트리 기법과 비슷한 방식의 lookup 프로시저가 이뤄진다. 표에 나타난 원리를 바탕으로 만들어진 정보를 바탕으로 하여 어떤 데이터의 키 값을 포함하고 있는 범위를 찾고, 그 범위에 해당하는 데이터들을 관리하는 successor로 쿼리를 전달함으로써 라우팅이 이루어지게 되고, 그에 따른 오버헤드는 $\log_2 N$이 된다.

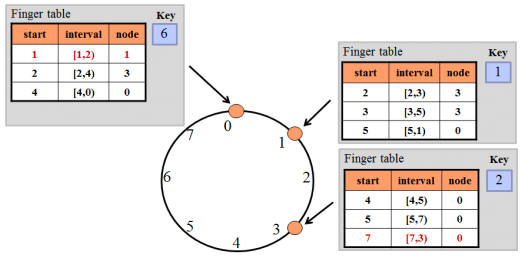


그림 1. 각 노드의 라우팅 테이블
Fig. 1. Routing table of each nodes

그림 1은 크기가 8인 원형 식별자 공간에 키 1, 2, 6을 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.

표 1. 라우팅 테이블 생성 규칙
Table 1. Rules for generating the routing tables

Notation	Definition
finger[*k*].*start*	$(n+2^{k-1}) \bmod 2^m,\ 1 \le k \le m$
.*interval*	[*finger*[*k*].*start*, *finger*[*k*+1].*start*]
.*node*	first node ≥ *n*.*finger*[*k*].*start*
successor	the next node on the identifier circle; *finger*[1].*node*
predecessor	the previous node on the identifier circle

이와 같이 범위를 지수적으로 감소하면서 탐색하기 때문에 탐색비용은 $O(\log_2 N)$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 관리 하는 successor를 탐색할 때, 먼저 노드 N의 successor에 원하는 키가 존재하는지 확인하고 존재하지 않으면 자신의 라우팅 테이블을 통해 범위에 키가 포함되는 successor 에게 키에 대한 successor를 요청하는 메시지를 보낸다. 이러한 과정을 반복함으로써 노드는 원하는 자원을 관리하는 노드에게 메시

www.kci.go.kr

지를 전달할 수 있다.

2.2 Chord 시스템의 자원 탐색

이 절에서는 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 범위에 키가 포함되는 successor에게 키에 대한 successor요청하는 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 다음과 같다.

1. 노드 1은 우선 자신의 successor인 노드 3에 키 6이 존재하는지 확인한다. 존재한다면 자신의 successor인 노드 3이 키 6의 successor가 된다. 그러나 그림3에서는 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 탈퇴

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 2와 같이 된다.

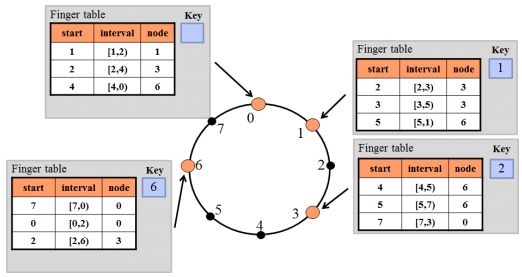


그림 2. 새로운 노드의 참여 후 라우팅 테이블
Fig. 2. Routing table after join

노드 6은 Chord 라우팅 알고리즘에 따라 자신의 라우팅 테이블을 구성한다. 그리고 시스템에 이미 존재하는 노드들은 새로 조인한 노드 6을 자신의 라우팅 테이블에 반영한다. 노드 0은 노드 6의 successor 노드가 되며 자신이 관리하고 있는 데이터 중 노드 6이 관리해야 할 데이터를 전송한다. 그림 2에서 노드 1은 키 6을 관리하고 있는데 키 6의 successor는 노드 6이 되기 때문에 노드 1은 노드 6에게 키와 데이터를 넘겨준다. 그림 2에서 노드 1이 시스템에서 정상적으로 떠나면 그림 3과 같이 된다. 노드 1의 자신이 관리하던 자원을 자신의 successor인 노드 3에게 전송하고 시스템을 떠나게 된다. 그림 3을 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.

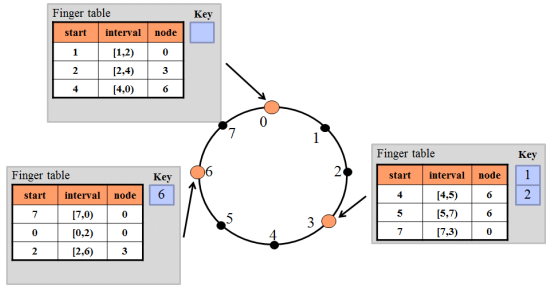


그림 3. 노드의 탈퇴 후 라우팅 테이블
Fig. 3. Routing table after leave

III. 제안하는 그룹 라우팅 알고리즘

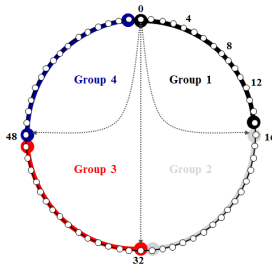
본 장에서는 제안하는 그룹 라우팅 알고리즘에 대해 설명한다. 제안하는 그룹 라우팅 알고리즘은 node-to-group 라우팅 알고리즘으로 m-bit(0~ −1)의 원형 주소 공간을 그룹으로 나누고 각각의 노드와 데이터의 키 값을 할당하는

www.kci.go.kr

방법이다. 각각의 노드와 데이터의 키 값으로부터 해싱 함수를 이용하여 주소 공간으로의 매핑이 이루어진다. 또한 각 그룹을 대표하는 포인터(Pointer)라는 개념을 사용하여 포인터를 기반으로 라우팅이 이루어져 좀 더 효율적인 라우팅이 가능하도록 제안하였다.

3.1 그룹 라우팅 알고리즘

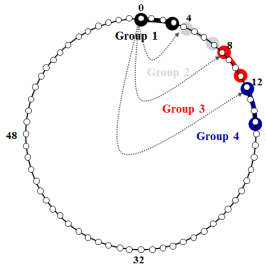
기존 chord 알고리즘의 성능 효율은 개의 노드를 가진 네트워크에서 평균적으로 log N)의 시간 복잡도를 갖는다. 하지만 제안하는 알고리즘의 성능 효율은 N개의 노드를 가진 네트워크에서 평균적으로 log (N)의 시간 복잡도를 갖는다. 여기서 G는 그룹 수를 나타낸다. 즉, G를 조절하여 시간 복잡도를 향상시킬 수 있다. 제안하는 알고리즘에서는 각각의 노드들이 Group table이란 라우팅 정보를 유지하고 있다. Group table은 주소 공간과 그룹에 따라 그 크기가 달라지는데, 노드 수 N과 그룹 수가 G인 경우 $\log_G(N)$의 행으로 이루어지게 된다. 예를 들어, 노드 수가 64이고 그룹 수가 4개인 경우 $\log_4 64 = 3$이 된다. 즉, 하나의 노드에 Group table이 3개의 행으로 이루어지게 된다. 이와 같이 노드와 그룹 수를 통해 각 노드가 가지는 테이블의 행의 수를 결정할 수 있다. 여기서 행의 수는 평균적인 홉 카운트 수를 의미한다. 본 논문에서는 각각의 행을 레벨(Level)로 정의하였으며, 각각의 레벨에는 현재 노드를 기준으로 하여, 그룹을 이루는 노드의 범위 정보를 가지고 있다. 예를 들어, 노드 수가 64이고 그룹 수가 4개인 경우 노드 0에서 Group table을 설정하는 단계는 그림 4와 같다. 먼저 그림 4의 (a)에서는 레벨이 1인 경우 4개의 그룹으로 나누고 각 그룹은 전체 노드에서 그룹의 수로 나눈 16개의 노드의 범위를 갖는다. 즉, 레벨 1의 그룹 1은 0~15의 범위를, 그룹 2는 16~31, 그룹 3은 32~47, 그룹 4는 48~63의 노드 범위를 갖는다. 그림 4의 (b)는 레벨 2를 나타낸 것으로, 각 그룹을 다시 4개의 그룹으로 나눈다. 레벨 1의 그룹 노드의 범위인 0~15의 노드의 범위를 레벨 2에서 다시 4개의 범위로 나눈다. 즉, 레벨 2의 그룹 1은 0~3, 그룹 2는 4~7, 그룹 3은 8~11, 그룹 4는 12~15의 노드의 범위를 갖는다. 그림 4의 (c)는 최종으로 설정이 되는 레벨로 레벨 2의 그룹 1의 범위인 0~3의 노드의 범위를 다시 4개의 범위로 나눈다. 즉, 레벨 3의 그룹 1은 0, 그룹 2는 1, 그룹 3은 2, 그룹 4는 3의 노드를 설정하게 된다. 이렇게 하여 최종으로 노드 0에서의 Group table은 그림 4의 (c)와 같다.



Group Table (node=0) / Level 1

node 0	Group 1	Group 2	Group 3	Group 4
Level 1	0…15	16…31	32…47	48…63

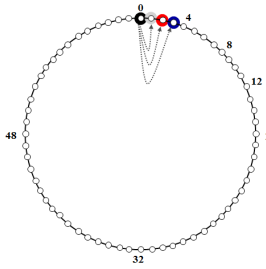
(a) 노드 0에서 Level 1 설정



Group Table (node=0) / Level 2

node 0	Group 1	Group 2	Group 3	Group 4
Level 1	0…15	16…31	32…47	48…63
Level 2	0…3	4…7	8…11	12…15

(b) 노드 0에서 Level 2 설정



Group Table (node=0) / Level 3

node 0	Group 1	Group 2	Group 3	Group 4
Level 1	0…15	16…31	32…47	48…63
Level 2	0…3	4…7	8…11	12…15
Level 3	0	1	2	3

(c) 노드 0에서 Level 3 설정

그림 4. 노드 0에서 그룹 테이블
Fig. 4. Group table from Node 0

3.2 그룹 라우팅의 자원 탐색

제안하는 알고리즘에서의 자원 탐색은 Group table을 바탕으로 하여 어떤 데이터의 키 값을 포함하고 있는 범위를 찾고, 그 범위에 해당하는 데이터들을 대표하는 포인터(Pointer)로 쿼리를 전달함으로써 라우팅이 이루어지게 되고, 그에 따른 오버헤드는 $\log_G(N)$이 된다. 여기서 포인터는 다음과 같이 정의 한다.

정의 1. 포인터(Pointer)는 자신을 제외한 각 레벨의 그룹에서 가장 첫 번째 해당하는 노드를 포인터로 정의 한다.

www.kci.go.kr

그림 5는 노드 0에서의 Group table을 바탕으로 포인터를 나타낸 것으로 정의 1에 의해 각 레벨의 그룹에서 가장 첫 번째 해당하는 노드를 포인터로 정의 한다.

Group Table (node=0)

node 0	Group 1	Group 2	Group 3	Group 4
Level 1	0…15	16…31	32…47	48…63
Level 2	0…3	4…7	8…11	12…15
Level 3	0	1	2	3

level 3(Group 2) = 1
level 3(Group 3) = 2
level 3(Group 4) = 3
level 2(Group 2) = 4
level 2(Group 3) = 8
level 2(Group 4) = 12
level 1(Group 2) = 16
level 1(Group 3) = 32
level 1(Group 4) = 48

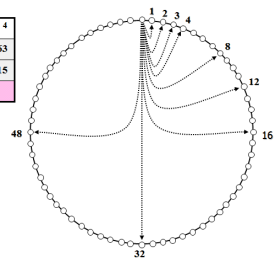


그림 5. 노드 0에서의 포인터
Fig. 5. Pointer from node 0

즉, 노드 수가 64이고 그룹 수가 4개인 경우 노드 0에서의 포인터는 1, 2, 3, 4, 8, 12, 16, 32, 48이 된다. 이것을 수식으로 표현하면 식 (1)과 같다. 여기서 (i, j)는 그룹과 레벨을 고려한 현재 노드에서의 포인터를 나타내며, n은 현재 노드, i는 그룹 항, j는 레벨 항, 는 전체 그룹 수, L은 전체 레벨 수를 나타낸다. 그림 6은 식 (1)에 의해 노드 0에서 노드 수가 64이고, 그룹 수가 4개인 경우 계산된 포인터를 나타낸다.

$$f(i,j) = n + (i-1)G^{\ j} \qquad \text{식 (1)}$$

이 포인터는 해당 범위에서의 대표를 나타낸 것으로, 어떤 범위 내의 키를 검색할 경우 레벨 1부터 순차적으로 검색하여 범위를 포함하고 있는 포인터에게 쿼리를 전송하여 탐색이 이루어진다.

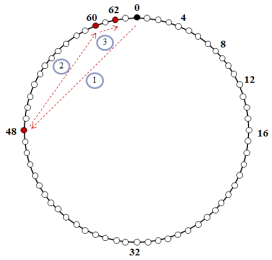
N=64, G=4일때 node=0에서의 pointer

node 0	Group 1 (i=1)	Group 2 (i=2)	Group 3 (i=3)	Group 4 (i=4)
Level 1(j=1)	f(1,1)	f(2,1)	f(3,1)	f(4,1)
Level 2(j=2)	f(1,2)	f(2,2)	f(3,2)	f(4,2)
Level 3(j=3)	f(1,3)	f(2,3)	f(3,3)	f(4,3)

node 0	Group 1 (i=1)	Group 2 (i=2)	Group 3 (i=3)	Group 4 (i=4)
Level 1(j=1)	0	16	32	48
Level 2(j=2)	0	4	8	12
Level 3(j=3)	0	1	2	3

그림 6. 노드 0에서의 포인터
Fig. 6. Pointer from node 0

예를 들어, 노드 0에서 62의 키를 탐색할 경우 그림 7과 같이 노드 0에서 레벨 1의 그룹 4의 범위는 62를 포함한 48~63의 범위를 가지고 있다. 따라서 레벨 1의 그룹 4의 포인터인 노드 48에게 첫 번째 쿼리를 전송하게 된다. 쿼리를 받은 노드 48은 자신의 Group table에서 62에 해당하는 범위를 레벨 2에서 탐색하게 된다. 62의 키 값은 레벨 2의 그룹 4에 포함됨으로 포인터인 노드 60에 두 번째 쿼리를 전송하게 된다. 쿼리를 받은 노드 60은 자신의 Group table에서 62에 해당하는 범위를 레벨 3에서 탐색하여 해당하는 최종 노드에게 쿼리를 전송하게 된다. 즉, 제안하는 알고리즘은 범위를 나누고 큰 범위에서 작은 범위로 포인터를 기반으로 키를 탐색하는 알고리즘이다.



Group Table (node=0)

node 0	Group 1	Group 2	Group 3	Group 4
Level 1	0…15	16…31	32…47	48…63
Level 2	0…3	4…7	8…11	12…15
Level 3	0	1	2	3

Group Table (node=48)

node 48	Group 1	Group 2	Group 3	Group 4
Level 1	48…63	0…15	16…31	32…47
Level 2	48…51	52…55	56…59	60…63
Level 3	48	49	50	51

Group Table (node=60)

node 60	Group 1	Group 2	Group 3	Group 4
Level 1	60…11	12…27	28…43	44…59
Level 2	60…63	0…3	4…7	8…11
Level 3	60	61	62	63

그림 7. 그룹 라우팅의 자원 탐색
Fig. 7. Lookup of Group routing

만약, 해당하는 키를 관리하는 노드가 없을 경우, 그 다음 노드를 찾아가기 위해 각각의 노드는 Greedy 알고리즘을 이용한다. Greedy 알고리즘은 최적의 해를 구하는 데에 사용되는 근사적인 방법으로, 여러 경로 중 하나를 결정해야 할 때마다 그 순간에 최적이라고 생각되는 것을 선택해 나가는 방식으로 최종적인 목적지에 도달한다. 그림 8은 그룹 라우팅을 위한 그리드 알고리즘을 나타낸다.

www.kci.go.kr

```
1: procedure n.TERMINATE(i)
2:     return i ∈ (n, succ]
3: end procedure

1: procedure n.NEXT_HOP(i)
2:     if TERMINATE(i) then
3:         return succ
4:     else
5:         r := succ
6:         for j := 1 to K do                    ▷ Node has K pointers
7:             if rt(j) ∈ (n, i) then
8:                 r := rt(j)
9:             end if
10:        end for
11:        return r
12:    end if
13: end procedure
```

그림 8. 그룹 라우팅을 위한 그리드 알고리즘
Fig. 8. Greedy algorithm for group routing

제안하는 그룹 라우팅 알고리즘에서 그림 9의 라우팅 생성 알고리즘에 의해 새롭게 참여하는 노드는 먼저 그룹 테이블을 생성한다.

```
1: procedure n.INITROUTINGTABLE()
2:     for i := 1 to (k − 1) log_k(N) do
3:         n.UPDATEENTRY(i)
4:     end for
5: end procedure

6: procedure n.UPDATEENTRY(i)
7:     S := n.LOOKUP( P,GETSLIST()) // p is pointers
8:     rt(i) := s
9: end procedure

10: procedure n.GETSLIST()
11:     return {n} ∪ succlist
12: end procedure
```

그림 9. 각 노드에서의 라우팅 생성 알고리즘
Fig. 9. Each nodes in the routing generation algorithms

IV. 성능평가

본 장에서는 제안한 알고리즘의 우수성을 확인하기 위해 기존 알고리즘과 비교 분석을 위한 성능평가를 수행한다.

4.1 실험 환경

본 논문에서 제안한 알고리즘의 성능을 평가하기 위해서 Java 언어를 사용한 시뮬레이터를 구현하고 노드 수의 증가에 따른 논리적인 라우팅의 홉 수, 라우팅 테이블 크기 및 전송 지연을 측정하였다. 이를 위해 링(ring) 토폴로지와 GT-ITM을 사용하여 만든 트랜짓-스텁 랜덤 그래프를 (Transit-Stub Random Graph)[12]를 기반 네트워크로 모델링하여, 노드수의 변화에 따른 성능을 측정하였다. 기존의 시뮬레이터에서는 오버레이 상의 모든 노드들 간에 어플리케이션 홉 간 지연시간이 일정하게 설정되어 기반 네트워크를 굳이 고려할 필요가 없었다. 그러나 이러한 구조에서는 한 노드에서 다른 모든 노드까지의 네트워크 상의 거리가 같아 홉 수가 작을수록 좋은 결과를 얻게 될 수 밖에 없다. 그러나 이러한 네트워크 구조는 광대역 네트워크의 노드간 지연시간 특성을 제대로 반영했다고 볼 수 없기 때문에, 본 실험에서는 기존 시뮬레이터를 수정하여 여러 기반 네트워크 토폴로지들 상에서 시뮬레이션이 가능하도록 하였다. 네트워크의 크기는 노드수를 초기 150개로 시작한다. 점점 노드가 추가되어 시뮬레이션 내부 시간으로 120초가 될 때 약 2000개가 네트워크에 편입되도록 구성되었다. 모든 노드들의 경우 8~10ms의 회선 지연시간과 8Mbps의 대역폭을 갖는다. aL과 aH는 [13]에서 최적의 값이라고 밝힌 0.2, 0.3으로 실험하였으며 노드 참여 주기가 올 때마다 일정한 확률로 생성 대신 노드의 탈퇴가 일어나게 하였다. 기여도 queue의 사이즈는 10으로 설정하였고, 식별자는 32bits 길이로 $0, 2^2-1]$ 범위에서 각 노드에게 무작위로 할당된다. 표 2는 시뮬레이션 파라미터를 나타낸다.

파라미터 항목	값
시작 노드 수	150
제한 노드 수	2000
시뮬레이션 시간	120 sec
지연 시간	8~10 ms
대역폭	8 Mbps
$, aH$	0.2, 0.3
$Queue$	10
traffic_generator	노드의 참여/탈퇴 및 데이터의 추가/삭제를 위한 트래픽 데이터 생성
topology_generator	Ring을 바탕으로 한 transit-stub 랜덤 그래프 생성
topology_latency	토폴로지 상의 노드간 최단 거리 계산

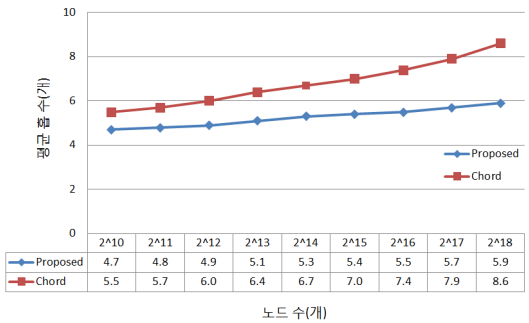
표 2. 시뮬레이션 파라미터 설정
Table 2. Simulation parameter settings

4.2 논리적 홉 수

chord는 finger table의 레코드 노드 중 검색하는 키 까지의

www.kci.go.kr

chord 거리가 가장 가까운 노드에게 질의를 라우팅 한다. 탐색 마다 키를 관리하고 있는 목적 노드까지의 chord 거리가 줄어들게 된다. 제안하는 알고리즘의 경우 노드 하나하나를 검색 하는 방법이 아닌 그룹 단위로 검색하는 방법으로 그룹의 범위에 속하는 키는 그룹의 대표인 포인터에게 전송하여 라우팅 하는 방법이다. 기본적으로 chord 알고리즘의 평균 홉 수는 log 성능을 나타내며, 제안하는 알고리즘은 log N 의 성능을 나타낸다. 이러한 성능을 바탕으로 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. 그림 10은 32-bits의 크기의 원형 식별자 공간을 가지는 네트워크에 노드 수를 증가 하였을 경우 제안한 알고리즘과 chord 알고리즘의 결과를 나타낸 그림이다. 그림에서 알 수 있듯이, 제안한 알고리즘이 노드 수가 증가하여도 chord에 비해 평균 홉 수가 약 22% 정도 적어 확장성 측면에서 우수함을 보여주고 있다.



	2^10	2^11	2^12	2^13	2^14	2^15	2^16	2^17	2^18
Proposed	4.7	4.8	4.9	5.1	5.3	5.4	5.5	5.7	5.9
Chord	5.5	5.7	6.0	6.4	6.7	7.0	7.4	7.9	8.6

그림 10. 노드 수에 따른 평균 홉 수
Fig. 10. Average hop count according to the number of nodes

4.3 라우팅 테이블 크기(size)

chord 알고리즘에 의한 라우팅 테이블 크기는 $\log_2 N$으로 그림 11과 같이 이진 탐색을 바탕으로 하고 있다. 이것을 좀 더 정확히 표현하면 $2-1)\log_2 N$의 라우팅 테이블의 크기를 가지고 있다.[10][11] 제안하는 알고리즘은 그림 12와 같이 이진 탐색이 아닌, 다차원 탐색으로 라우팅 테이블의 크기는 $(G-1)\log_G N$ 가 된다.

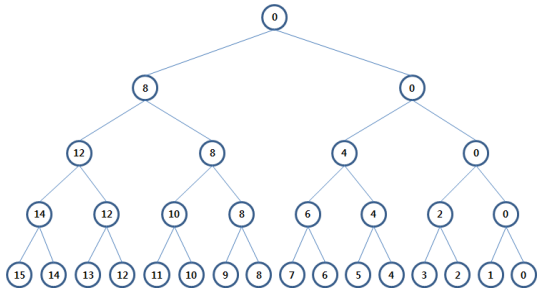


그림 11. chord의 이진 탐색 트리
Fig. 11. Binary search tree of chord

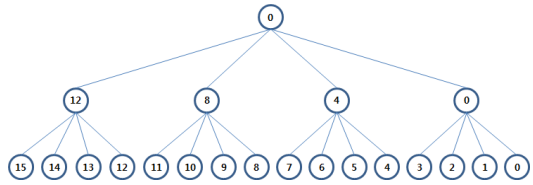
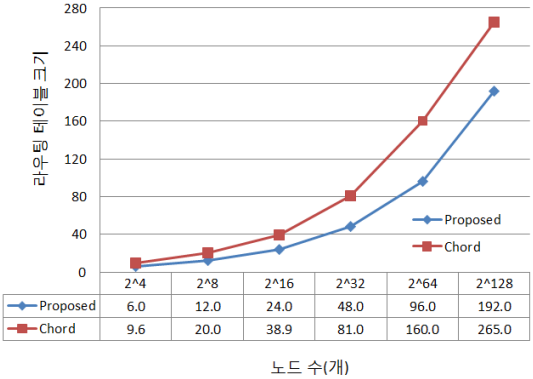


그림 12. 제안하는 알고리즘의 다차원 탐색 트리
Fig. 12. Multidimensional search tree of the proposed algorithm

그림 13은 chord와 제안한 알고리즘에 노드 수를 증가시켜, 각 알고리즘의 라우팅 테이블의 크기를 실험한 결과를 나타낸다. 제안한 알고리즘이 chord에 비해 약 33% 정도의 라우팅 테이블의 크기를 적게 생성하여 제안한 알고리즘이 chord 보다 성능이 향상되었음을 알 수 있다.



	2^4	2^8	2^16	2^32	2^64	2^128
Proposed	6.0	12.0	24.0	48.0	96.0	192.0
Chord	9.6	20.0	38.9	81.0	160.0	265.0

그림 13. 노드 수에 따른 라우팅 테이블 크기
Fig. 13. Routing table size according to the number of nodes

4.4 전송 지연(latency)

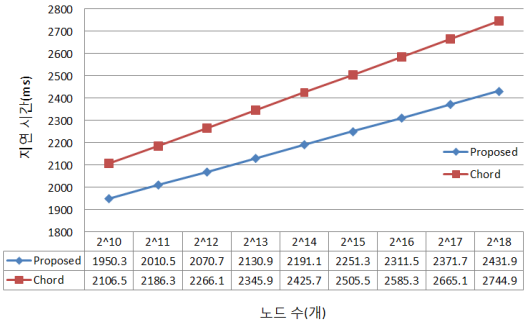
전송 지연은 각 노드가 가지는 홉 수와 테이블 크기를 고

www.kci.go.kr

려하여, 소스에서 목적지까지 쿼리를 보내는데 걸리는 평균 시간을 의미한다. 전송 지연 $_{otal}$)은 식 (2)와 같이 구할 수 있으며, L_{trans}는 쿼리 전송 지연 시간, L_{lookup}은 라우팅 테이블 검색 지연 시간, L_{churn}은 노드의 참여 및 탈퇴에 따른 지연 시간, $L_{timeout}$은 쿼리의 타임 아웃(timeout) 지연 시간을 의미한다.

$$L_{total} = L_{trans} + L_{lookup} + L_{churn} + L_{timeout}$$

식 (2)



	2^10	2^11	2^12	2^13	2^14	2^15	2^16	2^17	2^18
Proposed	1950.3	2010.5	2070.7	2130.9	2191.1	2251.3	2311.5	2371.7	2431.9
Chord	2106.5	2186.3	2266.1	2345.9	2425.7	2505.5	2585.3	2665.1	2744.9

그림 14. 노드 수에 따른 지연 시간
Fig. 14. Latency according to the number of nodes

그림 14는 노드 수를 증가 하였을 경우 제안한 알고리즘과 chord 알고리즘의 지연시간을 측정한 결과는 나타낸 그림이다. 그림에서 알 수 있듯이, 제안한 알고리즘의 평균 지연이 기존 chord 알고리즘의 평균 지연보다 약 10% 정도 지연 시간이 적어, 제안한 알고리즘이 chord 보다 성능이 향상되었음을 알 수 있다.

V. 결론

인터넷 접속 환경의 발달과 컴퓨터 및 네트워크 성능 향상으로 인해 P2P 시스템이라는 새로운 시스템이 등장하였다. 하지만 아직도 대부분의 시스템들은 클라이언트/서버 모델을 주로 사용하지만, 클라이언트/서버 시스템은 서버에 대한 과부하 문제, SPOF(Single Point Of Failure), 확장성 문제 등을 가지고 있다. P2P 시스템은 클라이언트/서버의 이러한 문제를 해결하고 클라이언트의 자원을 효율적으로 활용하는 것을 목적으로 제안되었다.

현재까지 잘 알려진 DHT 기반 P2P 시스템들은 대부분 이진 탐색을 기반으로 하고 있어, 평균 홉 수 및 라우팅 테이블의 성능 효율이 $\log_2 N$으로 제한되어 있다. 본 논문에서 제안하는 알고리즘은 그룹 단위로 라우팅이 이루어지는 알고리즘으로 다차원 검색을 기반으로 하고 있다. 제안한 알고리즘은 기존 chord 알고리즘에 비해 라우팅의 평균 홉 수, 라우팅 테이블의 크기에 대해 우수한 성능을 보였으며, 라우팅 홉 수 및 라우팅 테이블을 고려한 전송 지연에서도 우수한 성능을 보였다. 따라서 제안하는 알고리즘은 이진 탐색으로 국한된 기존의 알고리즘에 비해 다양한 네트워크에서 효율적으로 사용될 것으로 기대한다.

향후 과제로는 backup 및 전송 지연을 줄이기 위한 복제(replication) 기법 및 제안한 알고리즘을 토대로 클라우드 컴퓨팅을 위한 스토리지 시스템 개발을 위한 연구를 추가로 진행 할 것이다.

참고문헌

[1] What Is Web 2.0, http://www.oreillynet.com
[2] I.W. Lee, H.J. Park, "A Trend of P2P-Based Service and Charging Technics," Electronics and Telecommunications Trends, Oct, 2007.
[3] H.J. Park, K.R. Park, "P2P Technology Trend and Application to Home Network," Electronics and Telecommunications Trends, Oct, 2006.
[4] H.C. Kwon, Y.H. Moon, J.B. Gu, S.K. Kho, J.H. Nah, J.S. Jang, "Standardization and Technology Trend of Peer-to-Peer Communication," Electronics and Telecommunications Trends, Feb, 2007.
[5] B. O. Kim, I. W. Lee, H. J. Park, "Trend of Distributed Hash Tables-Based P2P," Electronics and Telecommunications Trends, Dec, 2006.
[6] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. "Chord: A scalable Peer-To-Peer lookup service for internet applications." In Proceedings of the 2001 ACM SIGCOMM Conference, pages 149 - 160, 2001.
[7] 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-01-1121, UC Berkeley, EECS, 2001.
[8] Antony Rowstron and Peter Druschel. "Pastry:

www.kci.go.kr

Scalable, decentralized object location, and routing for large-scale peer-to-peer systems". In IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pages 329 - 350, 2001.

[9] Evangelia Kalyvianaki and Ian A. Pratt. Building adaptive peer-to-peer systems. In Peer-to-Peer Computing, pages 268 - 269, 2004.

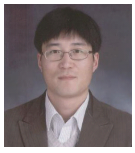
[10] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma and S. Lim, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes," IEEE Communications Survey and Tutorial, March 2004.

[11] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer File Sharing Technologies," Athens Univ. of Economics and Business White Paper (WHP-2002-03), 2002

[12] GT-ITM, http://www.isi.edu/nsnam/ns/ns-topogen.html

[13] R. Lobb, A. P. Couto da Silva, E. Leonardi, M. Mellia, and M. Meo, "Adaptive Overlay Topology for Mesh-based p2p-tv Systems" in ACM NOSSDAV, June 2009.

저 자 소 개



박 용 민

2005: 광운대학교
전자통신공학과 공학석사

2011: 광운대학교
전자통신공학과 공학박사

현 재: 삼육보건대학교
의료정보시스템과 전임강사

관심분야: 의료정보, 정보통신

Email : pym@shu.ac.kr

www.kci.go.kr