디지털산업정보학회 논문지 제7권 제4호-2011년 12월

# P2P 오버레이 네트워크에서 효과적인 Peer 검색을 위한 B-Chord

홍 록 지*·문 일 영**

## *Effective B-Chord look-up peer in P2P overlay network*

Hong, Rok Ji ·Moon, Il Young

〈Abstract〉

In this paper, search-efficient Bi-directional-Chord(B-Chord) is proposed in P2P (Peer-to-Peer) overlay network. Chord is the most popular P2P Look-up protocol. However, it applied to the mobile environment, the search success rate become lower and the request delay time increases. That is big problem.

Thus, by improving the existing Chord, in this paper proposed B-Chord reduces the request delay time to in a mobile environment. Proposed B-Chord have the two Finger table and can search by selecting Finger table depending on the value of Key. By use these bi-directional, it can reduce the number of nodes Hop and search delay time. Thus, As a result, it will be able to increase the search success rate in a mobile environment.

Key Words : Chord, DHT, MANET Environment

## Ⅰ. 서론

P2P방식은 최근 스마트폰의 사용 증가로 인 해 Wifi, Wibro의 이용이 확산되면서 모바일환경에서에서도 많은 각광을 받고 있다. 이는 사용자들이 Social Network Service(SNS)와 다양한 모바일 콘텐츠들을 이용하게 되면서 모바일 기기 간의 파일 공유 필요성이 증가하고 있기 때문이다[1-2]. 그러나 일반적인 P2P 컴퓨팅 네트워크를 모바일 디바이스에 적용하기에는 너무 복잡할 뿐만 아니라 적합하지 않다. 이는 모바일 기기의 이동성이라는 특성으로 인하여 각각의 기기는 통신범위라는 제약조건을 가지기 때문이다. 이러한 이유로 인하여 기존 유선환경의 P2P방식은 모바일 환경에 적용할 수 없으므로 이를 적합한 방식으로 개선해야하는 필요성이 대두되고 있다.

현재 파일 및 서비스 공유를 위한 기술 개발은 이루어져 있으나 아직 정식적인 모바일 P2P는 개발되지 않은 상태이다. 그리하여 모바일 환경에 적합한 방식으로 개선하는 연구가 필요하다. 본 논문에서는 DHT의 Look up 알고리즘 중 하나인 Chord의 문제점을 개선시켜 모바일 환경에서 적합하도록 하는 방식을 모색해보려 한다.

또한, 2장에서 기존 Unstructured P2P 방식의 단점을 보완하기 위한 방식인 DHT(Distributed Hash table)와 DHT 알고리즘[3-4] 중 많은 곳에서 연구가 진행 중인 Chord 알고리즘에 대한 이론을 알아본다. 3장에서는 기존 Chord의 문제점 및 관련연구 동향을 파악하고 4장에

* 한국기술교육대학교 컴퓨터공학부 석사과정(제1저자)
** 한국기술교육대학교 컴퓨터공학부 부교수

서 본 논문에서 제안한 B-Chord를 소개한다. 마지막으로 5장에서는 본 논문의 결론을 맺는다.

## Ⅱ. Chord

Chord는 여러 프로토콜 중 쉽고 간단한 방법으로 여러 곳에서 가장 많이 쓰이고 연구되고 있는 검색프로토콜이다. Chord는 하나의 circle을 가정하고 이 circle위에 0~2^m-1 만큼의 Key/node identifier들이 가능하다는 것으로 시작한다.

우선 8개의 identifier를 가지는 Chord를 예로 들어본다. 8개의 identifier를 가진다는 것은 m = 3 이라는 뜻이며, 결국 modulo 2^m 으로 모든 Key와 node id를 표현할 수 있다는 뜻이다[5].

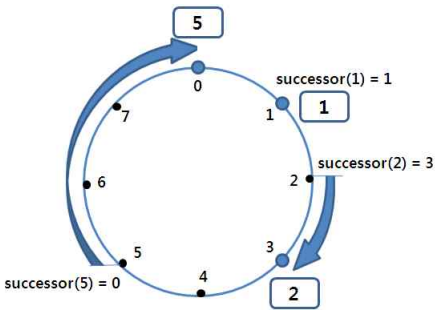



그림 1. Chord의 기본적인 구조

<그림 1>에서 볼 수 있듯이 3개의 노드가 각각 0, 1, 3 위에 배치되어 있는 것을 확인할 수 있다. 여기에서 중요한 것은 각 노드가 과연 어떠한 Key를 담당할 것이냐를 결정하는 문제이다.

Chord는 Successor라는 개념을 사용하여 하나의 노드가 담당할 Key의 range를 결정한다. 위에서 Key 1은 node 1이 담당하고, Key 2 는 node 3, 그리고 Key 6는 node 0이 담당한다는 것을 쉽게 알 수 있다. Chord는 담당해야할 Key range에 대한 효율적인 관리를 위해서 또한 인접한 노드간의 라우팅을 위해서 다음과 같은 Finger table을 각 노드에서 관리하고 있다.

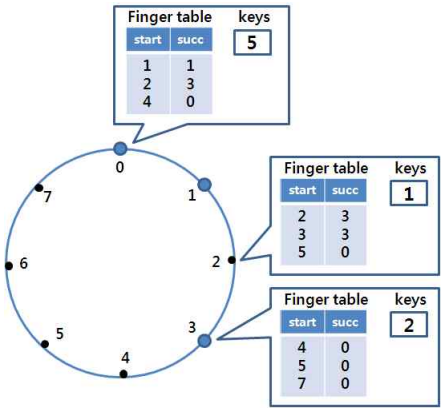



그림 2. Chord의 Finger table

Chord의 Finger table은 주소공간의 크기에 따라 테이블의 전체 크기가 변하게 된다.

우리가 살펴본 Chord의 Finger table 방식은 arbitrary Key에 대한 정보를 포함하고 있지 않다. 그리하여 Chord 알고리즘은 임의의 Key를 담당하고 있는 node를 찾는 과정을 거치게 된다. 이러한 과정으로써 O(logN) 검색 효율로 Key를 찾을 수 있게 된다[6].

## Ⅲ. 문제점 및 관련연구

### 3-1 메시지 수 감소

Chord에서는 데이터 요청 메시지를 보낼 때 Finger table을 보고 메시지를 보내게 된다. 하나의 노드가 메시지를 보내면 가장 먼저 요청 메시지를 해쉬하여 키 값을 얻어 온다. 받은 키 값과 자신의 Finger table을 비교한다. 자신의 Finger table에서 그 키 값보다 작은 값 중에 최댓값인 Successor로 메시지를 보내도록 한다. 받은 노드는 앞의 과정을 반복하여 메시지를 지정된 장소에 보

낸다. 또한 Chord는 노드의 실패에 대해서 파급효과를 최소로 하기위해 Stabilization 함수를 주기적으로 호출한다. Stabilization 함수는 ping메시지와 Find_successor메시지 두 가지로 구성된다. 먼저 ping메시지는 자신의 Successor가 살아있는지 확인하기 위해 ping메시지를 주기적으로 보낸다. ping메시지에 대한 응답이 오지 않으면 그 Successor를 자신의 Finger table에서 지운다. 그리고 Stabilization 함수는 Find_successor메시지를 보내 자신의 Successor를 주기적으로 찾고 등록한다. Find_successor메시지는 Ping 메시지 보다 많은 트래픽을 발생시킨다. 이는 Successor을 찾기 위해서 자신은 하나의 메시지만 발생시키지만 다음 노드로 가게 되면 전체 네트워크에서 여러 메시지가 발생되기 때문이다.

기존의 Chord는 모바일 환경과 같은 Join/Leave가 자주 일어나는 환경에서 요청 메시지의 성공률이 떨어진다. 이러한 환경에서 Chord는 데이터 요청 메시지의 실패가 일어나지 않기 위해 Finger table을 최신으로 유지해야한다. 이를 위하여 Stabilization 함수의 주기를 빠르게 해야 하지만 주기가 빨라지게 되면 하나의 노드가 처리해야하는 메시지의 양은 지수적으로 증가하게 된다.

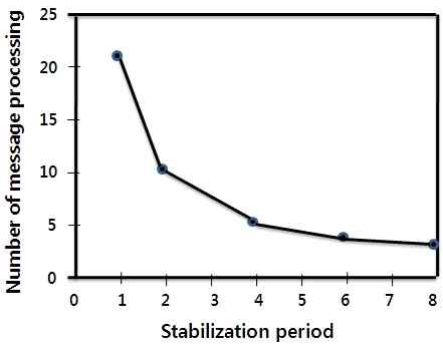



그림 3. Stabilization 주기에 따른 메시지 수

<그림 3>는 Stabilization 주기에 따른 메시지 양의 변화를 보여주는 그래프이다. x축은 주기를 나타내고 y축은 하나의 노드가 초당 처리하는 메시지의 개수를 나타낸다. 그림을 보면 주기가 느리면 메시지 양이 적고 빨라지게 되면 메시지의 양이 급격히 증가하는 것을 알 수 있다[7].

이러한 과정으로 발생하는 메시지의 양은 각각의 노드와 전체 네트워크의 트래픽을 발생시키는 문제점을 발생시키게 된다.

## 3.2 검색 성공률 향상

기존의 Chord를 모바일에 적용할 경우 이동성이라는 특성으로 인하여 검색 성공률이 매우 낮은 수치를 나타내어 P2P 검색이 거의 이루어지지 않는 문제점이 발생한다. 이러한 문제점을 해결하기 위한 연구로 Backtracking Chord가 있다. Backtracking Chord는 Chord의 Successor table을 수정해서 형성한다. 기존의 Chord는 단순한 Finger table(노드 ID에 의한 가상적으로 형성하는 ID table)에서 실재 노드를 선정하여 Successor table을 만들었다. 하지만 Backtracking Chord의 경우 단계(stage)를 두어Successor table을 만든다. 이러한 Successor table을 가진 Backtracking Chord는 처음에 검색 요청하는 Successor가 사라질 경우 단계를 증가시켜 새로운 Successor에게 검색 요청 메시지를 전송한다.

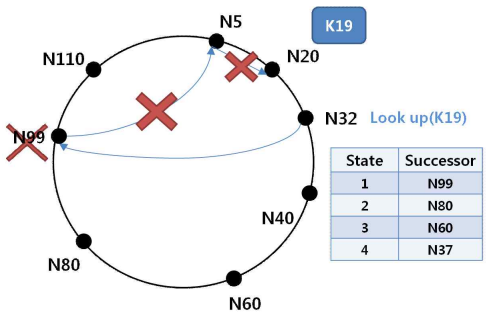



그림 4. Backtracking Chord

예를 들면 <그림 4>와 같이 N32 노드가 K19를 찾기 위해서 Successor table에서 K19를 가지고 있는 노드에 가까운 N99에 검색 요청을 했을 때 N99는 전파 영역에

서 벗어났기 때문에 검색 요청에 반응할 수 없다. 이때 Time-out이 발생하고 Time-out 이 후에는 N32의 다음 stage에 해당하는 Successor인 N80으로 검색 요청을 하게 된다. 만약 N80 역시 Time out이 일어날 경우 계속 stage을 증가시켜 다른 Successor로 검색 요청을 보낸다 [8].

### 3.3 지연시간 및 Hop 수 감소

모바일 P2P 네트워크에서는 노드의 Join/Leave가 빈번하게 이루어지므로 검색과정에서 검색 지연시간이 길어지는 경우가 발생하게 된다. 또한 Finger table의 Successor을 찾아가는 방식을 취하므로 Hop 수가 많아질수록 지연시간이 증가하게 될 것이다.

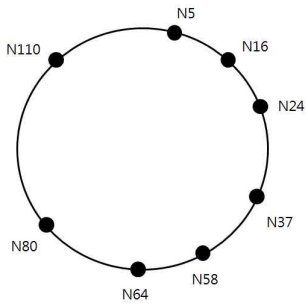
이러한 문제점을 해결하기 위하여 본 논문에서는 B-Chord(Bi-directional-Chord)를 제안한다.

## Ⅳ. 제안하는 B-Chord (Bi-directional-Chord)

기존의 Chord에서는 데이터 요청 메시지를 보낼 때 받은 Key 값을 갖고 있는 노드를 찾기 위해 자신의 Finger table을 탐색한다. 자신의 Finger table에서 그 키 값보다 작은 값 중에 최댓값인 Successor로 메시지를 보내도록 한다. 그러나 이러한 경우 Key 값이 Finger table의 Successor의 가장 작은 값보다 작은 경우 기존의 Chord는 단방향 검색을 하기 때문에 많은 Hop 수를 거쳐 검색이 이루어지게 된다. 이러한 경우 검색 지연 시간이 길어지게 되며 검색 성공률도 낮아지게 된다. 이동성이라는 특성을 갖고 있는 모바일 환경에서는 Hop 수를 줄이는 것이 기존의 P2P 환경에서 보다 더 중요해진다. 그리하여 본 논문에서는 B-Chord(Bi-directional-Chord)를 제안하여 Hop 수를 줄이고 그에 따라 검색 성공률을 높이려 한다.

### 4.1 B-Finger table 생성

B-Chord에서는 기존의 Finger table과 함께 B-Finger table을 생성하여 두 개의 Finger table을 가지게 된다. B-Finger table의 Start값은 노드ID - 2^n 으로 나타내며 그에 따른 Successor가 정해지게 된다. 만약 노드ID - 2^n 값이 0보다 작아질 경우에는 null 값을 갖게 된다.




| | Start | Successor |
|---|---|---|
| N24+2^0 | N25 | N37 |
| N24+2^1 | N26 | N37 |
| N24+2^2 | N28 | N37 |
| N24+2^3 | N32 | N37 |
| N24+2^4 | N40 | N58 |
| N24+2^5 | N56 | N58 |

| | Start | Successor |
|---|---|---|
| N24-2^0 | N23 | N16 |
| N24-2^1 | N22 | N16 |
| N24-2^2 | N20 | N16 |
| N24-2^3 | N16 | N16 |
| N24-2^4 | N8 | N5 |
| N24-2^5 | Null | Null |

그림 5. Finger table와 B-Finger table

### 4.2 Key 값에 따른 Bi-Finger table 검색

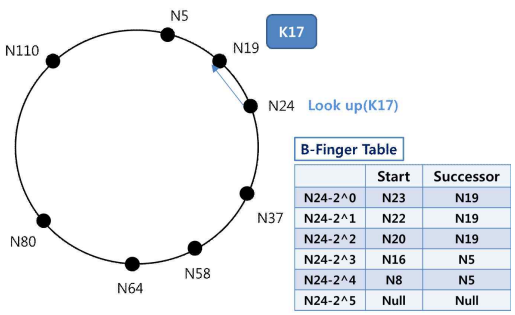
검색 알고리즘은 다음과 같다.

```
if(Key. ID < node. ID)
        search B-Finger table
        if(Key. ID < Successor. ID)
        minimum of Successor. ID
else
        search Finger table
        if(Key. ID > Successor. ID)
```

maximum of Successor. ID

다음 알고리즘을 살펴보면 Key. ID가 node. ID보다 큰 경우에는 기존의 Finger table을 탐색한다. 그리고 Key 값보다 작은 값 중에서 최댓값인 Successor을 찾아 Find_successor() 메시지를 보내게 된다. 이 과정을 반복하여 Key를 갖고 있는 노드를 찾아 검색이 완료된다. 이 경우는 기존의 Chord와 탐색 과정이 같다. 그러나 B-Chord에서는 Key. ID가 node. ID보다 작은 경우에는 B-Finger table을 탐색한다. 그리고 Key 값보다 큰 값 중에서 최솟값인 Successor을 찾아 Find_successor() 메시지를 보내게 된다. 이러한 과정을 반복하여 최종 노드를 찾게 되는 것이다.

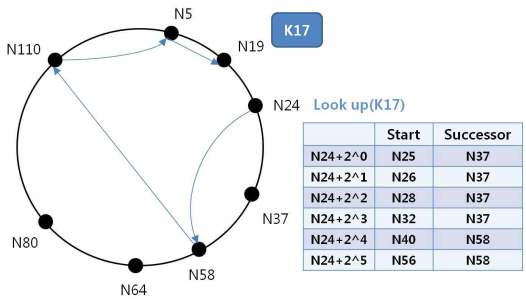
예를 들어 기존의 Chord는 <그림 6>과 같이 N24에 Key 17을 요청할 경우 Finger table에 따라 N58에 요청 메시지를 보내고 같은 과정을 거쳐 N19에 있는 Key 17을 찾게 된다. 이 과정에서 Hop수는 4가 된다.




| | Start | Successor |
|---|---|---|
| N24-2^0 | N23 | N19 |
| N24-2^1 | N22 | N19 |
| N24-2^2 | N20 | N19 |
| N24-2^3 | N16 | N5 |
| N24-2^4 | N8 | N5 |
| N24-2^5 | Null | Null |

그림 6. 기존 Chord에서 Key값이 가장 작은 Successor보다 작을 경우의 검색 과정

그러나 B-Chord에서는 요청한 Key. ID가 요청받은 node. ID 보다 작으면 B-Fimger table을 탐색하여 Key를 찾게 된다. <그림 7>을 보면 B-Finger table의 Successor인 N19에 Find_successor() 메시지를 보내 기존의 Chord보다 적은 1 Hop으로 검색되는 것을 볼 수 있다.




| | Start | Successor |
|---|---|---|
| N24+2^0 | N25 | N37 |
| N24+2^1 | N26 | N37 |
| N24+2^2 | N28 | N37 |
| N24+2^3 | N32 | N37 |
| N24+2^4 | N40 | N58 |
| N24+2^5 | N56 | N58 |

| | Start | Successor |
|---|---|---|
| N58+2^0 | N59 | N64 |
| N58+2^1 | N60 | N64 |
| N58+2^2 | N62 | N64 |
| N58+2^3 | N66 | N80 |
| N58+2^4 | N74 | N80 |
| N58+2^5 | N90 | N110 |

그림 7. B-Chord에서 Key값이 가장 작은 Successor보다 작을 경우의 검색 과정

B-Chord와 같이 양방향 탐색이 이루어질 경우 불필요한 Hop 수를 줄이고 메시지 수를 줄여 검색 시간을 단축시킬 뿐만 아니라 검색 효율을 높일 수 있다.

## V. 결 론

본 논문은 P2P(Peer-to-Peer) 오버레이 네트워크에서 검색 효율성을 높인 Bi-direction Chord를 제안하였다. 기존의 Chord에서의 단방향 검색이 아닌 양방향 Finger table 탐색이 이루어지면서 검색 과정에서 Hop 수가 줄어드는 것을 알 수 있다. MANET 환경에서는 이동성이라는 특성으로 인하여 Join/Leave가 빈번히 일어나게 된다. 이러한 경우 Hop 수를 줄임으로써 요청지연시간을 감소시켜 검색 효율을 높일 수 있다.

## 참고문헌

[1] 김대업, 이일금, 오재신, “이용수준에 따른 모바일 엔터테인먼트 서비스 만족도에 관한 연구 :게임, 음

악, 동영상 서비스를 중심으로," 디지털산업정보학회논문지, 제1권, 제2호, 2005. 9, pp. 79-89.

[2] 권준희, 김성림, "유비쿼터스 환경에서 상황 데이터 기반 모바일 콘텐츠 서비스를 위한 추천 기법," 디지털산업정보학회논문지, 제6권, 제2호, 2010. 6, pp. 1-9.

[3] M. Dischinger, "Mobility Enhancements to an Appoach for Structured Overlay Routing in Wireless Moblie Ad Hoc Networks," Diploma Thesis, System Architecture Group, University or Karlsruhe, Nov. 2005.

[4] LEWIN, D. Consistent hashing and random trees: Algorithms for caching in distributed networks. Master's thesis, Department of EECS, MIT, 1998. Available at the MIT Library, http://thesis.mit.edu/.

[5] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, "Chord : A Scalable P2P Lookup Service for Internet Applications," In IEEE/ACM Transaction on Networking, Vol. 12, No. 2, Apr 2004, pp. 205-218.

[6] Chord project, http://pdos.lcs.mit.edu/chord/.

[7] Yoon Young-Hyo, Kwak Hu-Keun, Kim Cheong-Ghil, Chung Kyu-Sik "A Reactive Chord for Efficient Network Resource Utilization in Mobile P2P Environments," Korean Institute of Information Scientists and Engineers, 36(2), April 2009, pp.80-89.

[8] 이세연, 장주욱, "Backtracking을 이용한 모바일 에드혹 네트워크에서 Chord 검색 방법," 한국정보과학회 춘계학술발표회, 31권, 1호, 2004.

## ■ 저자소개 ■

홍 록 지
Hong, Rok Ji

2010년 9월~현재
한국기술교육대학교 컴퓨터공학부 대학원 재학
2007년 3월~2010년 8월
한국기술교육대학교 컴퓨터공학부 졸업

관심분야 : 모바일, P2P 네트워크, Future 네트워크
E-mail : shj1062@kut.ac.kr

문 일 영
Moon, Il Young

22005년 3월~현재
한국기술교육대학교 컴퓨터공학부 부교수
2004년 ~ 2005년
한국정보문화진흥원 선임연구원
2005년 2월 한국항공대학교 정보통신공학과 졸업(공학박사)
2002년 2월 한국항공대학교 항공통신정보공학과 졸업(공학석사)
2000년 2월 한국항공대학교 항공통신정보공학과 졸업(공학사)

관심분야 : 무선 인터넷 응용, 모바일 인터넷, 모바일IP
E-mail : iymoon@kut.ac.kr

논문접수일 : 2011년 9월 3일
수 정 일 : 2011년 9월 22일
게재확정일 : 2011년 10월 1일