모바일 Ad-hoc 환경을 위한 피어 투 피어 검색 방법

이세연° 이경근 조태경 장주욱
서강대학교 세종대학교 SKTELECOM 서강대학교
seiyonl@eeca1.sogang.ac.kr kglee@sejong.ac.kr tkcho@sktelecom.com jjang@mail.sogang.ac.kr

Peer-to-Peer Query Search over Mobile Ad-hoc Network

Sei-yon Lee° Kyung-geun Lee Tae-kyoung Cho Ju-wook Jang
Sogang University Sejong University SKTELECOM Sogang University

요 약

Chord는 N개의 노드로 이루어진 P2P(Peer-to-Peer)네트워크에서 검색에 사용되는 메시지를 $O(logN)$으로 줄인 P2P 검색 알고리즘이다. 하지만 모바일 Ad-hoc 네트워크에 이를 적용할 경우 검색 성공률이 30%이하를 보여서 운동성(Mobility)이 있는 네트워크에서는 P2P 검색이 거의 이루어지지 않는 문제점이 발생한다. 본 논문에서이 같은 문제점을 극복하기 위한 알고리즘인 Backtracking Chord와 Redundant Chord를 제안한다. Backtracking Chord 방식은 $O(\log N)$메시지를 사용하여 순차적으로 t번까지 검색을 요청함으로써(t: Timeout의 횟수 $(0 < t < SuccessorID)$) t에 따라 최고 88%까지 검색 성공률을 높일 수 있다. Redundant Chord는 검색 요청에 $O(r\log N)$ (r: 복수 검색 요청 메시지의 양$(0 < r < \log N)$)메시지를 사용하여 Successor table의 entry에 속해 있는 노드들에게 동시에 복수의 검색 요청을 보냄으로써 r에 따라 최고 81%까지 검색 성공률을 높일 수 있다. NS-2 시뮬레이션을 통한 검증 결과 제안한 방식 모두 50%이상의 검색 성공률을 보임으로써 모바일 Ad-hoc 환경에서 효율적인 P2P 검색 알고리즘임을 입증하였다.

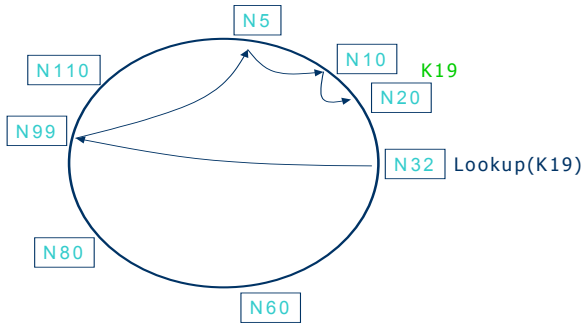
1. 서 론

기존의 모바일 Ad-hoc네트워크에 사용되는 P2P 검색 알고리즘은 Gnutella와 같은 Query flooding방식이었다. 하지만 이러한 방식은 모바일 Ad-hoc 네트워크에서 P2P 검색을 하기 위해서 많은 대역폭을 필요로 하는 문제점을 보였다.

이러한 문제점을 해결하고자 본 논문에서는 유선 P2P 네트워크에서 효율적인 검색 알고리즘인 Chord 방식을 무선 모바일 Ad-hoc 환경에 적용시켜보았다.

분산형 P2P 알고리즘인 Chord[1]방식은 검색 메시지(Query)를 모든 네트워크에 flooding하는 Gnutella[8]방식과는 달리 $O(\log N)$크기의 Successor Table(Successor: 자료 인덱스)에 해당하는 노드만을 검색함으로써 보다 빠른 시간에 검색을 할 수 있는 방식이다.[2] (그림 1 참조)

Chord에서는 Napster보다 작은 $O(\log N)$의 상태 저장 메모리 용량을 사용하고 검색 메시지는 $O(\log N)$으로 Gnutella보다 줄어들게 되어 검색 시간이 Gnutella보다 단축되게 된다.(표 1 참조)



[그림 1] Chord 검색 방식

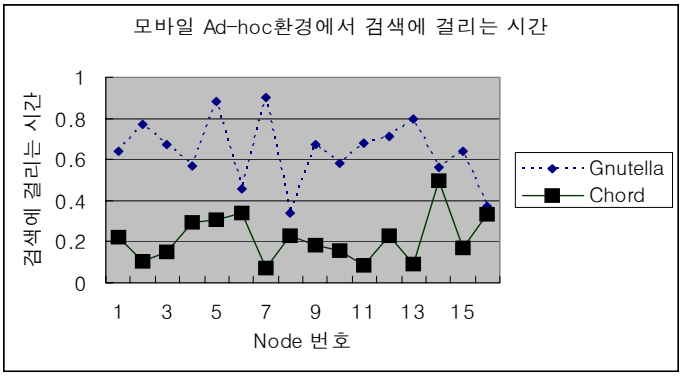
본 논문에서는 모바일 Ad-hoc 환경에 Chord를 적용해보고 그에 대한 문제점 제시 및 해결 방안을 제시해보았다.

검색 구조	Status Memory	Query Message
Napster	$O(N)$	$O(1)$
Gnutella	$O(1)$	$O(N)$
Chord	$O(\log N)$	$O(\log N)$

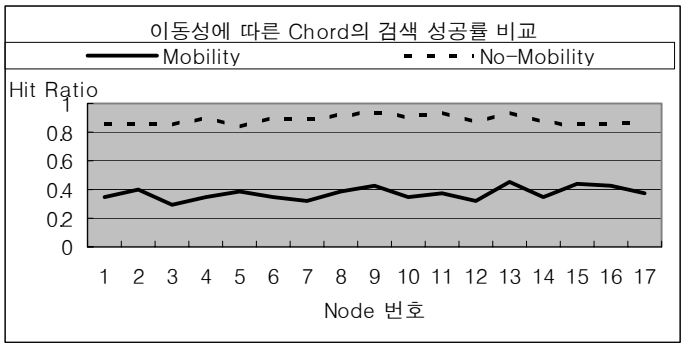
[표 1] P2P 구조에 따른 메모리와 메시지

2. 모바일 Ad-hoc 환경에서의 Chord의 문제점

Chord를 모바일 Ad-hoc 환경에서 적용하였을 때의 NS-2 시뮬레이션 결과는 그래프 1-1과 같다. 검색 시간이 Query flooding 방식인 Gnutella 방식보다 평균 0.4초(70% 검색 시간 단축)단축되었음을 알 수 있다. 하지만 검색 성공률에서의 결과는 그래프 1-2의 결과 모바일 Ad-hoc 환경에서는 검색 성공률이(Hit ratio)가 45%이하로 떨어지는 것을 보였다. Chord 검색 방식은 모바일 Ad-hoc 환경에서 검색 시간 단축시킬 수 있으나 검색 성공률이 낮기 때문에 검색이 불가능하였다 이는 그림2와 같이 노드가 이동할 경우 순간적으로 노드가 전파 영역에서 사라지는 경우가 발생하기 때문에 Successor Table 중의 Successor가 사라지는 경우가 발생한다. 이 때 중간의 Successor table이 함께 사라지기 때문에 그림 2와 같이 검색이 이루어지지 않게 된다.

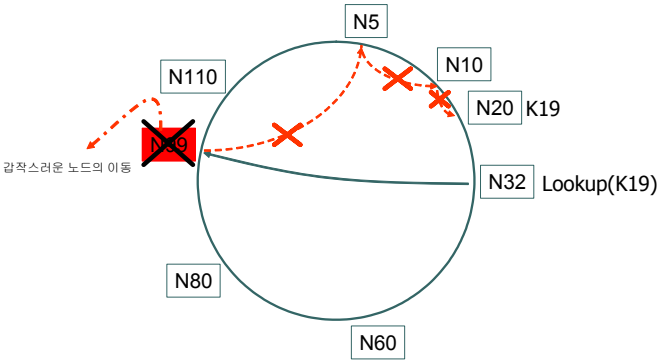


[그래프 1-1] 검색에 걸리는 시간



[그래프 1-2] 이동성에 따른 Chord의 쿼리 검색 성공률 비교:
이동성에 따른 랜덤 Seed를 1로 준다.
100% = Hit ratio '1'

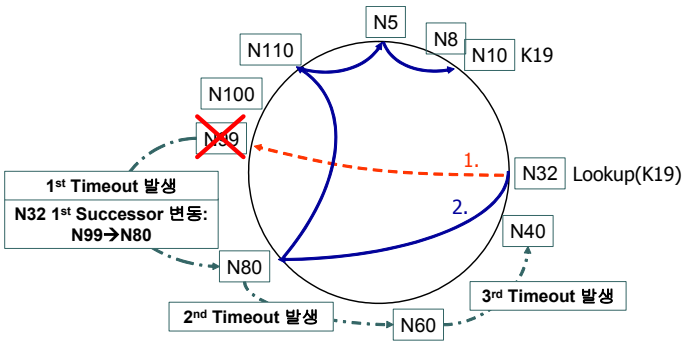
따라서 모바일 Ad-hoc 환경에서 Chord 방식을 바로 적용할 수 없다. 본 논문에서 위의 문제점을 다음 2개의 제안한 방식으로 모바일 Ad-hoc 환경에서 Chord의 장점을 이용할 수 있는 방법을 제시하였다.



[그림 2] Chord 쿼리 검색 실패

3.1 제안 방식 I : Backtracking Chord

Backtracking Chord는 일정한 시간(Time-out)을 설정해 두어서 성공자가 갑자기 사라지더라도 검색을 멈추지 않고 새로운 성공자로 검색 요청을 하도록 한다.
(그림 3 참조)



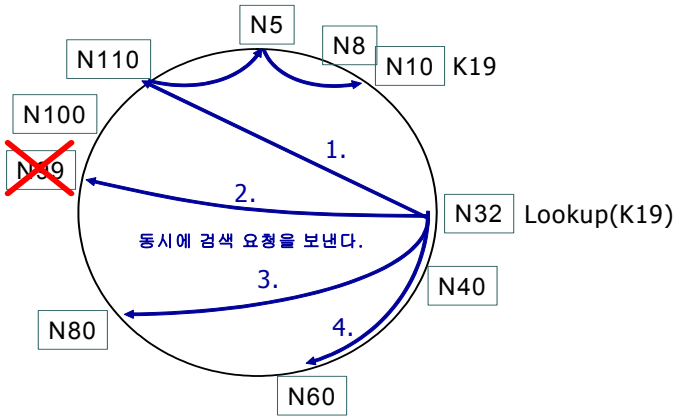
[그림 3] 제안된 방식에 따른 Timeout이 발생했을 때 검색 요청에 따른 동작: Backtracking Chord

이 때 새로운 Successor로 지정되는 노드는 현재 Chord를 구성하고 있는 노드들 중에서 사라진 Successor의 바로 앞 ID Number의 노드(Predecessor)가 설정된다. 이 Predecessor는 요청 했던 노드에게 새로운 성공자로서 Successor Table에 등록되게 된다. 이러한 방식은 $O(\log N)$ 검색 요청 메시지의 사용으로 t배 만큼의 검색 성공률을 향상시킬 수 있다. 하지만 기존의 Chord 방식보다 검색시간이 $t \times Timeout$만큼 더 지연되게 된다. 이 방식은 검색 시간을 $t \times Timeout$만큼 증가시켜서 대역폭을 절약하면서 검색 성공률을 높이는 방식이다.
(t: Timeout의 횟수$(0 < t < SuccessorID)$)

3.2 제안 방식 II : Redundant Chord

검색 요청 메시지를 다중의 Successor들에게 전송한다. 이 경우에 검색 요청에 걸리는 메시지는 $O(r\log N)$으로 증가하게 되지만 검색에 걸리는 시간의 가치 척도인 쿼리 서치에 드는 메모리 크기는 $O(\frac{\log N}{r})$으로 감소하게 됨으로써 보다 빠른 속도로 검색을 할 수 있다.

(r: 복수 검색 요청 메시지(Query)의 전송 횟수$(0 < r < logN)$)



[그림 4] 검색 시간을 줄이기 위한 제안된 다중 검색 방식: Redundant Chord

중복된 테이블에 의한 릴레이 검색으로 인해서 중복된 검색 요청 중 검색이 보다 빠르게 이루어지는 검색 노드를 이용함으로써 그림 4와 같이 노드가 없어져도 검색 요청이 성공적으로 이루어지는 쪽을 동시에 선택하기 때문에 최소 $O(\log N)$의 검색 시간을 유지하면서 모바일 Ad-hoc 환경에서 안정적인 검색을 유지할 수 있다. 하지만 동시에 복수의 검색 요청을 보낸 노드 모두가 사라질 경우에는 검색이 이루어지지 않는 단점이 있다.

4.1 실험 환경

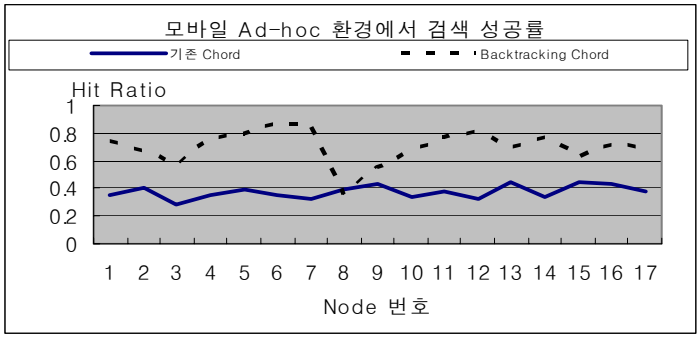
NS-2 Simulator환경에서 1000개의 임의의 모바일 클라이언트들을 설정하고 그중에서 무작위로 일부 클라이언트들을 On-Off 시킨다. 이러한 동작은 일종의 전파 영역에서 벗어나을 때를 Off시키는 것으로 다시 전파 영역에 있을 때를 On시키는 것으로 동작시킴으로써 모바일 네트워크에서의 동작과 비슷하게 움직이도록 환경 설정을 했다. 또한 1000개의 노드 중 검색 요청을 자주 하는 모바일 클라이언트 15개를 선정해 두었다.

4.2 실험 방법

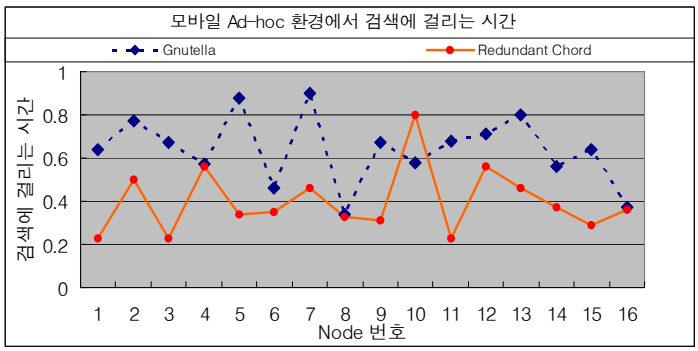
설정한 모바일 네트워크 환경에서의 Chord동작 결과와 안정한 네트워크 환경에서의 Chord 동작 결과를 비교 분석하였다. 또한 모바일 Ad-hoc 네트워크에서의 제안된 두가지 방식에 대한 각 각의 알고리즘을 적용한 실험 결과를 모바일 Ad-hoc환경에서의 기존 Chord, Gnutella와 비교 분석하였다.

4.3 실험 결과

그래프 2-1는 기존 Chord 방식과 Backtracking Chord방식의 검색 성공률에 대해서 비교 분석한 결과이다. 기존 Chord의 경우 검색 성공률이 40% 미만의 결과를 보이지만 Backtracking Chord의 경우에는 유동적이지만 평균 72%의 높은 검색 성공률을 보이는 것을 볼 수 있었다. P2P 검색 특성상 Replication 성격에 의해서 검색 성공률이 50%를 기준으로 50%미만이면 검색이 거의 이루어지지 않고 50%이상이면 검색이 잘 이루어진다는 성격에 의해서 Backtracking Chord는 검색이 잘 이루어진다는 것을 볼 수 있다.[3]



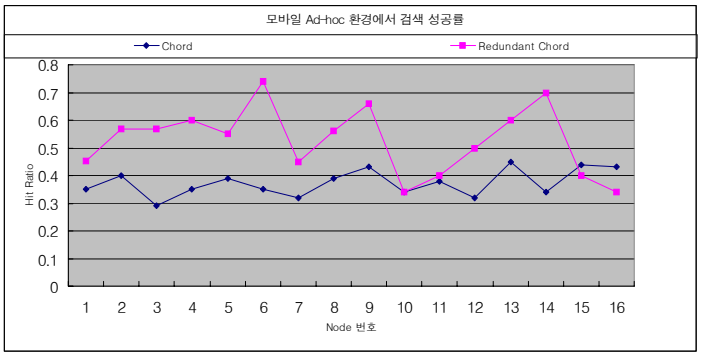
[그래프 2-1] Backtracking Chord와 Chord의 검색 성공률 비교



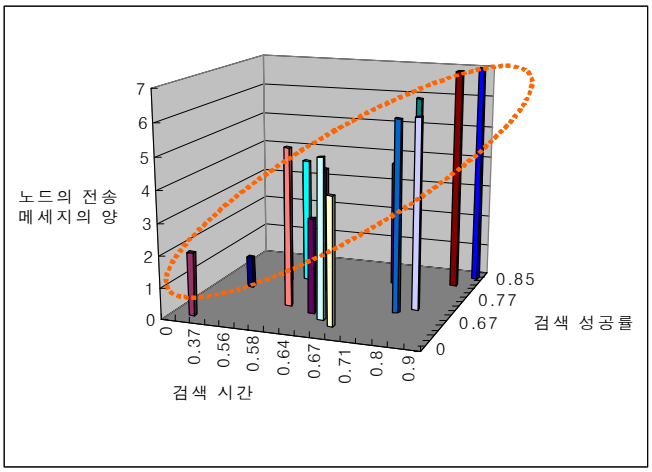
[그래프 2-2] Gnutella 와 Redundant Chord 검색에 걸리는 시간 비교

그래프 2-2는 Gnutella와 Redundant Chord와의 검색 시간 비교 분석 결과이다. 모든 검색 메시지를 flooding하는 Gnutella 방식보다 검색 시간이 최고 50% 단축되었다. Redundant Chord는 검색 메시지를 복수로 보내고 모든 노드들에게 검색 요청을 보내지 않기 때문에 검색시간이 단축하였다. 그래프 2-3의 결과는 모바일 Ad-hoc 환경에서 기존 Chord와 Redundant Chord를 비교한 결과이다. Redundant Chord가 81%의 검색 성공률을 보임으로써 모바일 Ad-hoc 환경에서의 기존의 Chord보다 높은 검

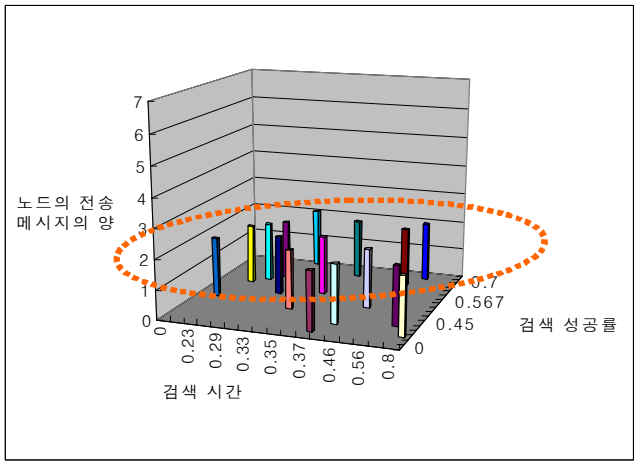
색 성공률을 보여주었다.



[그래프 2-3] Redundant Chord와 Chord 검색 성공률 비교



[그래프 2-4] Backtracking Chord의 결과 분석



[그래프 2-5] Redundant Chord의 결과 분석

그래프 2-4와 그래프 2-5은 제안한 방식들의 노드 전송 메시지 양과 검색 시간, 검색 성공률의 상관 관계를 분석한 그래프이다. Backtracking Chord의 경우 높은 검색 성공률을 보이기 위해서 검색 시간과 전송되는 메시지 양이 선형적으로 증가하는 분포를 보여주고 있다.(Linear increase) 이로 인해 대역폭 절약과 검색 시간의 단축을 위해서는 검색 성공률이 중요 파라메터로 작용하는 것을 알 수 있다. Redundant Chord의 경우에는 메시지 양과 검색 시간과 검색 성공률이 전혀 관계없는 분포를 보이고 있다. 이는 검색 시간과 검색 성공률을 결정짓는 요소가 복수 검색 요청 메시지의 수 때문이다. 따라서 모바일 노드들의 검색 성공률의 보장을 위해서는 검색 성공률이 성능 결정 요소가 되는 Backtracking Chord 방식이 적합하고 모바일 노드들의 Load balancing 측면에서는 전송 메시지 수가 성능 결정 요소가 되는 Redundant Chord 방식이 적합하다.

5. 결론 및 앞으로 할 일

본 논문에서 제안한 방식들의 경우 어떠한 네트워크 환경에서도 검색 성공률이 평균 50%의 검색 성공을 보임으로 Chord가 불안정한 네트워크에서 보이는 검색 성공률(평균 37.4%)보다 탁월한 검색 성공을 보였다. Redundant Chord의 경우에는 기존 Chord 알고리즘보다 검색 성공률(최고 82%)을 향상시키고 검색 시간을 Gnutella방식보다 최고 50% 단축함으로써 모바일 Ad-hoc 환경에서 효율적이고 빠른 P2P 검색 알고리즘임을 보여주었다. 위의 제안한 알고리즘들이 모두 모바일 Ad-hoc 환경에서 P2P 파일 검색 방식으로 적합함을 확인하였다. 이를 바탕으로 앞으로는 무선 대역폭에서의 효율적인 Utilization을 위해서 보안하도록 하겠다.

[참고 문헌]

[1] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnany Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications MIT Laboratory for Computer Science.(ACM Conf.2002)

[2] CHEN, Y., EDLER, J., GOLDBERG, A., GOTTLIEB, A., SOBTI, S., AND YIANILOS, P. A prototype implementation of archival intermemory. In *Proceedings of the 4th ACM Conference on Digital libraries* (Berkeley, CA, Aug. 1999), pp. 28-37.

[3] CLARKE, I. A distributed decentralised information storage and retrieval system. Master's thesis, University of Edinburgh, 1999.

[4] CLARKE, I., SANDBERG, O., WILEY, B., AND HONG, T.W. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability* (Berkeley, California, June 2000). http://freenet.sourceforge.net.

[5] DABEK, F., BRUNSKILL, E., KAASHOEK, M. F., KARGER, D., MORRIS, R., STOICA, I., AND BALAKRISHNAN, H. Building peer-to-peer systems with Chord, a distributed location service. In *Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII)* (Elmau/Oberbayern, Germany, May 2001), pp. 71-76.

[6] DABEK, F., KAASHOEK, M. F., KARGER, D., MORRIS, R., AND STOICA, I. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)* (To appear; Banff, Canada, Oct. 2001).

[7] DRUSCHEL, P., AND ROWSTRON, A. Past: Persistent and anonymous storage in a peer-to-peer networking environment. In *Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS 2001)* (Elmau/Oberbayern, Germany, May 2001), pp. 65-70.

[8] Gnutella. http://gnutella.wego.com/.