Backtracking을 이용한 모바일 에드혹 네트워크에서 Chord 검색 방법

이세연° 장주욱
서강대학교 서강대학교
seiyonl@eecal.sogang.ac.kr jjang@sogang.ac.kr

Backtracking Chord over Mobile Ad-hoc Network

Sei-yon Lee° Ju-wook Jang
Sogang University Sogang University

요 약

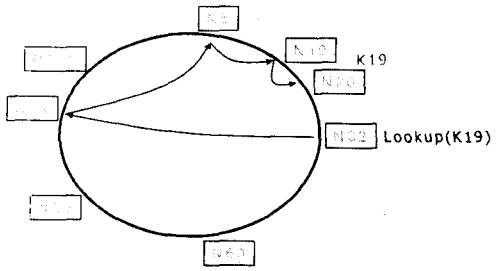
Chord[6]는 N개의 노드로 이루어진 P2P(Peer-to-Peer)네트워크에서 검색에 사용되는 메시지를 $O(\log N)$으로 줄인 P2P 검색 알고리즘이다. 하지만 모바일 Ad-hoc 네트워크에 이를 적용할 경우 검색 성공률이 매우 떨어져 (1000개의 노드가 도보속도(2m/s)로 움직이는 경우: 검색 성공률 30%이하)P2P 검색이 거의 이루어지지 않는 문제점이 발생한다. 본 논문에서는 이같은 문제점을 극복하기 위한 알고리즘인 Backtracking Chord을 제안한다. Backtracking Chord 방식은 $O(\log N)$메시지를 사용하여 순차적으로 t번까지 검색을 요청함으로써(t: Timeout의 횟수 $(0 < t < \log N)$) t에 따라 최고 88%(t>4)까지 검색 성공률을 높일 수 있다.

1. 서 론

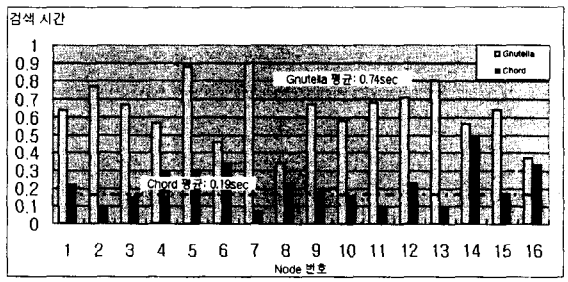
기존의 모바일 Ad-hoc네트워크에 사용되는 P2P 알려진 검색 알고리즘은 Gnutella[2]와 같은 Query flooding방식이었다. 하지만 이러한 방식은 대역폭이 제한된 모바일 Ad-hoc 네트워크에서는 효율이 크게 떨어지는 문제점이 있다.이러한 문제점을 해결하고자 본 논문에서는 유선 P2P 네트워크에서 가장 효율적인 검색 알고리즘으로 알려진 Chord[6] 방식을 무선 모바일 Ad-hoc 네트워크에 적합하게 개선하는 방법을 제안하였다.

2. Chord의 문제점

분산형 P2P 알고리즘인 Chord방식은 검색 메시지(Query)를 모든 네트워크에 flooding하는 Gnutella방식과는 달리 $O(\log N)$크기의 Successor Table을 이용하여 Successor(자료를 찾아가기 위한 포인터(인덱스)노드)에게만 검색 요청 메시지를 보내고 그에 의한 응답을 받아서 검색을 하는 방식이다. 이 방식은 모든 노드에게 검색 요청을 보내는 Query flooding 방식보다 효율적이고 빠른 검색을 할 수 있는 것이 특징이다.[2]

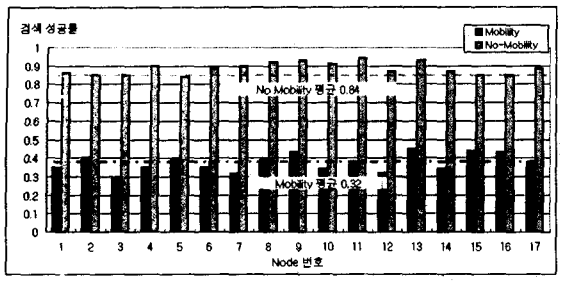


[그림 1] Chord 검색 방식



[그림 2] 모바일 Ad hoc 네트워크에서의 검색 시간 비교

Chord를 모바일 Ad-hoc 환경에서 적용하였을 때의 NS-2 시뮬레이션 결과는 [그림 2]와 [그림 3]과 같다. 검색 시간이 Query flooding 방식인 Gnutella 방식보다 평균 0.4초(70% 검색 시간 단축)단축되었음을 알 수 있다. 하지만 검색 성공률에서의 결과는 그림 2의 결과 모바일 Ad-hoc 환경에서는 검색 성공률(Hit ratio)이 45%이하로 떨어지는 것을 보였다. Chord 검색 방식은 모바일 Ad-hoc 환경에서 검색 시간 단축시킬 수 있으나 검색 성공률이 낮기 때문에 검색이 불가능하였다 이는 [그림 4]

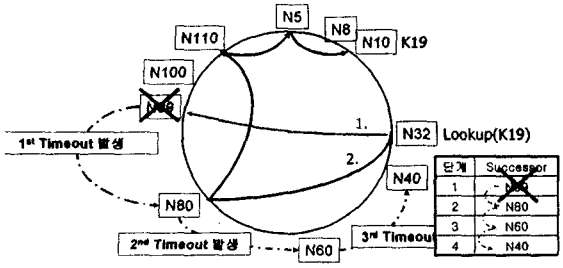


[그림 3] 모바일 Ad-hoc 네트워크에서 Chord의 쿼리 검색 성공률 비교
각 노드들이 사람의 도보 속력(2m/s)으로 움직인다.

와 같이 노드가 이동할 경우 순간적으로 노드가 전파 영역에서 사라지는 경우가 발생하기 때문에 Successor Table중의 Successor가 갑자기 사라지는 경우가 발생한다. 이 때 검색 경로에 있는 Successor의 Successor table이 함께 사라지기 때문에 그림 4와 같이 검색이 이루어지지 않게 된다.

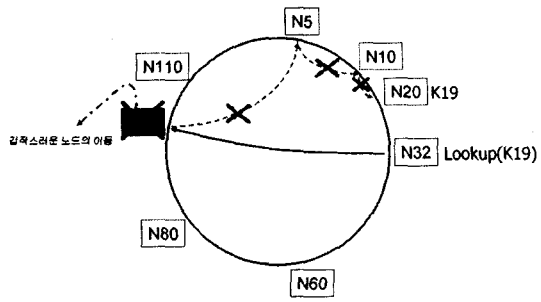
3. 제안 방식 : Backtracking Chord 방식

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 두어 Successor table을 만든다. 이러한 Successor table을 가진 Backtracking Chord는 처음에 검색 요청하는 Successor가 사라질 경우 단계를 증가시켜 새로운 Successor에게 검색 요청 메시지를 전송한다. 예를 들면 [그림 5]와 같이 N32 노드가 K19를 찾기 위해서 Successor table에서 K19를 가지고 있는 노드에 가까운 N99에 검색 요청을 했을 때 N99는 전파 영역에서 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.



[그림 5] Backtracking Chord

이 때 stage를 증가하는 횟수는 t(Timeout의 횟수 $(0 < t < \log N)$)로 결정한다. 만약 Successor table의



[그림 4] Chord 쿼리 검색 실패

stage를 증가하면서 검색을 요청했을 경우에도 검색이 이루어지지 않을 경우 1번 stage로 초기화 시켜서 다시 1번 stage에 해당하는 Successor에게 검색 요청을 보낸다. 이 경우에도 검색이 이루어지지 않을 때는 최종적으로 검색 불가능이라는 결과를 도출한다.

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. 하지만 검색 요청에 드는 대역폭은 한 번 보내는 메시지의 양은 $O(\log N)$로 기존의 $O(N)$ 메시지를 사용하는 Query flooding 방식보다 절약된다.

4. 시뮬레이션 및 결과

4.1 실험 환경

NS-2 Simulator에서 1000개의 노드가 존재하는 모바일 Ad hoc 네트워크를 구축하였다. 모바일 Ad hoc 네트워크에서의 라우팅 프로토콜은 AODV(Ad hoc On-demanding Distance Vector)을 하였다. 각 노드들은 도보 속도에 해당하는 2m/s의 속도로 움직이게 하였다.

검색 성공률(H)은 검색 응답 메시지(M_r)을 검색 요청 메시지(M_q)의 비율로 구했다.

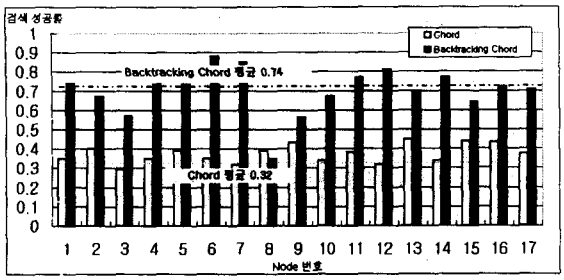
$$H = \frac{M_r}{M_q} \quad (1)$$

검색 시간은 검색 응답 시간(T_r)에서 검색 요청 시각(T_q)을 제해서 구했다.

$$T_s = T_r - T_q \quad (2)$$

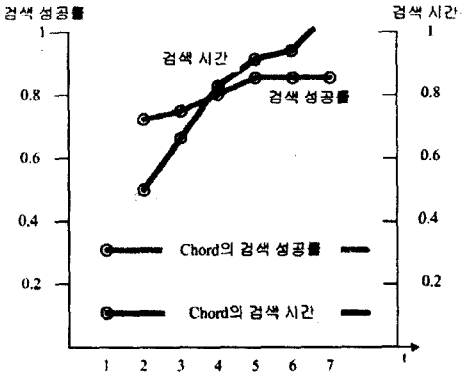
4.2 실험 결과

[그림 6]은 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 성공



[그림 6] Backtracking Chord과 Chord의 검색 성공률 비교

률이 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]



[그림 7] t값에 의한 Backtracking Chord의 성능 분석

[그림 7]은 Backtracking Chord의 t값에 의한 검색 시간, 검색 성공률의 성능 분석 그래프이다. 여기서의 검색 성공률과 검색 시간은 모든 노드들의 평균 결과이다. Backtracking Chord의 경우 t값(전송 대역폭)을 증가시킬수록 검색 시간이 계속 증가하였다. 검색 성공률은 88%에서 수렴하였다. 검색 시간이 계속해서 증가하는 이유는 모바일 Ad hoc 네트워크에서 t번만큼의 timeout이 발생하는 현상이 증가하기 때문이다. 검색 성공률의 경우는 일부 Successor에게만 검색 요청을 보내기 때문에 모든 노드에게 검색 요청을 보내는 Query flooding 방식과 같은 최대 검색 성공률을 얻을 수 없고 88%의 한계점을 가지게 된다. t값이 5이상일 때 최대의 검색 성공률을 보임으로써 t값을 5이상 늘리는 것은 무의미하다. 이유는 t값에 따라 검색 시간이 비례하기 때문이다. t값이 5 일 때와 7일 때를 비교해보면 그 이유를 잘 알 수 있다. 물론 모바일 노드가 1000개일 때의 실험만을 했기 때문에 노드의 수가 달라지면 최대 검색 성공률에 수렴하는 t값은 달라질 수 있다. 하지만 수렴하는 t값이 존재하는 것을 보여줌으로써 반드시 t를 증가시키는 것이 검색 성공률 향상에 도움이 되지 않는다는 것을 알 수 있다.

4. 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. Backtracking Chord의 경우에는 기존 Chord 알고리즘 보01다 검색 성공률(평균 74%, 최고 88%) 향상시키면서 모바일 Ad-hoc 네트워크에서 P2P 검색을 가능하게 하였다. 이러한 실험 결과를 바탕으로 실제 무선 대역폭의 사용양(utilization, Bandwidth) 측정과 노드 수를 변화시키는 실험에 따른 노드 수와 최적의 t값의 관계, 그리고 t값에 의한 검색 성공률과 검색 시간의 상관관계에 관한 분석을 할 예정이다.

참고문헌

[1]Napster, http://www.napster.com/.

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

[3]M. Ripeanu, "Peer-to-Peer Architecture case study: Gnutella Network", Proceedings of the First International Conference on Peer-to-Peer Computing, 2001.

[4]Alexander Klemm, Christoph Lindemann and Oliver P. Waldhorst, "A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc networks", Proc. IEEE Semiannual Vehicular Technology Conference, October 2003.

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

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

[7]T. H. Hu, B. Thai and A. Seneviratne, "Supporting Mobile Devices in Gnutella File Sharing Network with Mobile Agents", ISCC, Kemer - Antalya, Turkey, July 2003.