모바일 환경을 위한 리액티브 방식의 Chord

윤영효[O#1] 곽후근[#2] 김정길[*3] 정규식[#4]
[#]숭실대학교 정보통신전자공학부
[*]남서울대학교 컴퓨터학과
{[1]yyhpower, [2]gobarian, [4]kchung }@q.ssu.ac.kr
[3]cgkim@nsu.ac.kr

Reactive Chord for Mobile Environment

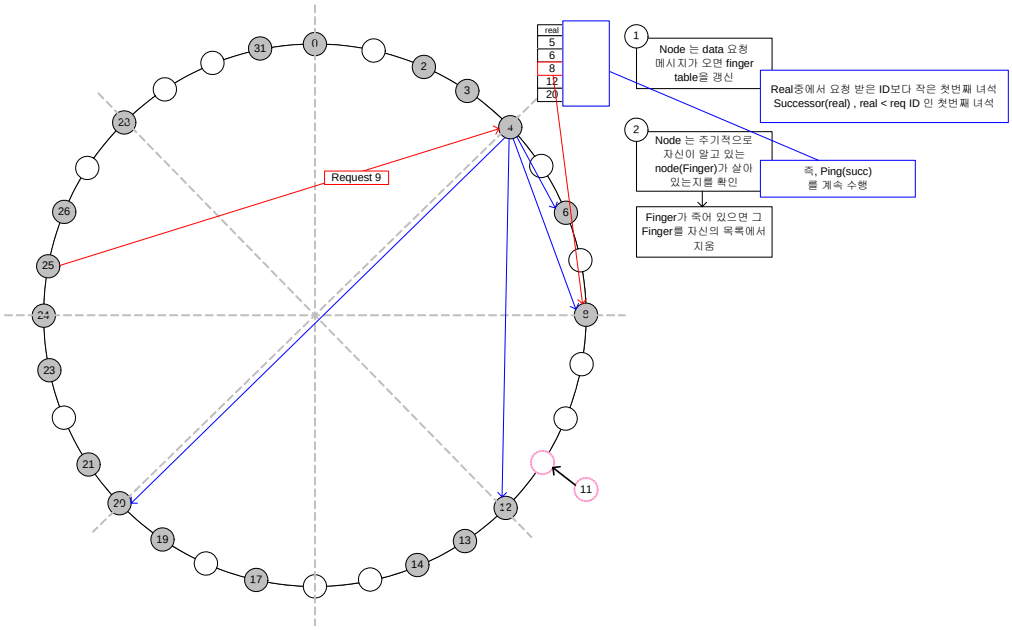
YoungHyo Yoon[O#1] Hukeun Kwak[#2] Cheongghil Kim[*3] Kyusik Chung[#4]
[#]School of Electric Engineering, Soongsil University
[*]Department of Computer Science, Namseoul University

DHT 방식의 P2P 는 기존 Unstructured P2P 방식의 단점을 보완하기 위한 방식이다. DHT 알고리즘을 사용하면 빠른 데이터 검색을 할 수 있고, 검색 효율과 상관없이 피어 개수를 증가 시킬 수 있다. 이러한 DHT 방식은 자신의 프로토콜을 유지하기 위하여 주기적인 메시지를 사용 한다. 모바일 환경이 될 경우, DHT 프로토콜에서는 프로토콜을 유지하고 요청 실패를 줄이기 위해서 빠른 주기로 메시지를 보내야 한다. 하지만 이로 인해, 전체 네트워크의 트래픽은 커지게 된다.

본 논문에서는 Reactive 테이블 업데이트 방식을 이용하여 기존 DHT에서의 테이블 업데이트를 위한 Overhead를 줄이는 기법을 제안한다. 주기적으로 자신의 테이블을 업데이트하는 방식을 사용하던 기존 방식과 달리, 제안된 방식에서는 데이터 요청이 들어 왔을 때만 테이블을 업데이트하는 방식을 사용한다. 제안된 방식은 많은 연구가 진행 되고 있는 Chord를 기반으로 실험을 하였으며, 버클리 대학에서 만들어진 Chord 시뮬레이터를 이용하였다.

전체 구조 및 동작 과정

제안된 방식에서는 Stabilization에서 발생하는 Find_successor 메시지를 주기적인 방식에서 Finger table을 데이터 요청 시에 업데이트하는 방식으로 바꾸는 것을 제안한다. (그림 1)은 제안한 방식의 전체적인 그림이다.

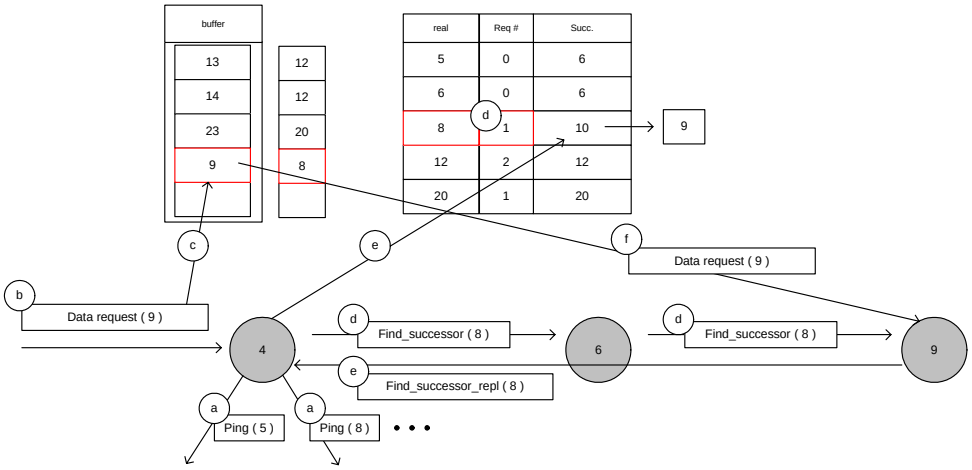


(그림 1) 제안된 방식에서 Finger table 업데이트

제안된 방식에서는 데이터 요청 메시지가 왔을 경우에만 Find_successor 메시지를 보내서 Finger table을 업데이트한다. 이전 Chord에서의 Ping 메시지는 그대로 사용을 한다. 자신이 Find_successor 메시지를 보낼 노드가 없는 상태라면, 메시지를 보내도 응답이 오지 못한다. 그렇기 때문에 노드가 살아있는지를 확인하기 위해서 Ping 메시지는 그대로 사용을 한다. 이 Ping 메시지는 Find_successor 메시지에 비하면 네트워크에서 차지하는 비중은

크지 않기 때문에 크게 문제 되지는 않는다.

제안된 방식의 동작 과정은 (그림 2)과 같다.



(그림 2) 제안된 방식의 자세한 동작 과정

단계 a. 노드는 주기적으로 자신이 유지하고 있던 Successor에게 Ping 메시지를 보내서 죽었는지 살았는지를 확인한다. 만약 죽었다면, 자신의 테이블에서 지운다.

단계 b. 다른 노드로부터 데이터 요청을 받는다.

단계 c. 노드는 받은 데이터 요청 메시지를 Buffer에 넣어둔다.

단계 d. 노드는 자신의 Start 값 중에 요청 받은 메시지의 키 값보다 작은 값 중 가장 큰 값을 찾는 Find_successor 메시지를 보낸다. 하지만 모든 요청에 대해서 Find_successor를 보내게 되면, 요청이 많을 시에 문제가 생길 수 있다. 그래서 제안하는 방식에서는 이전에 요청했던 것과 똑같은 것을 요청해야 하면 Find_successor 메시지를 보내지 않고 바로 Buffer에 넣는다. 한번만 보내고 나중에 안 보내면 다시 업데이트할 수 없는 문제가 생기기 때문에 요청을 보내지 않는 수를 제한 한다. n 개의 숫자를 제한하게 되면 그 요청에 대해서 n번의 요청이 오기 전 까지는 Find_successor 메시지를 한번만 보내고, n번 요청 후에 다시 보내게 된다.

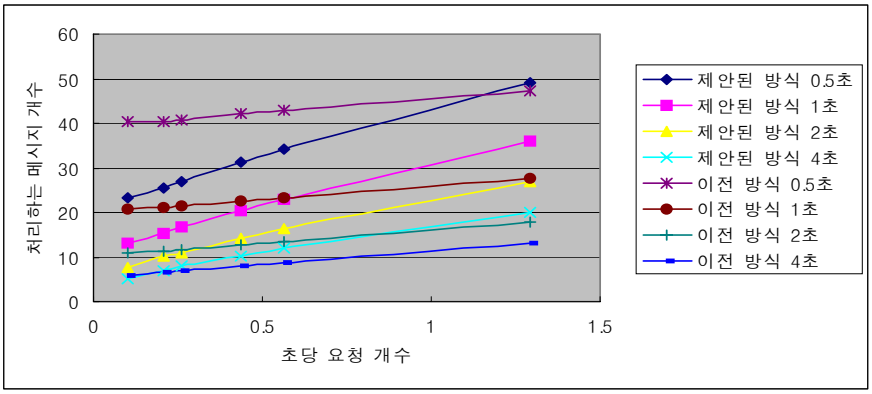
단계 e. 단계 d 에서 보낸 Find_successor는 나중에 Find_successor_reply를 받는다.

단계 f. 노드는 Find_successor_reply를 받았을 때, 자신의 Finger table을 업데이트하고, 단계 c에서 넣어 두었던 데이터 요청 메시지를 다음 노드로 전달하게 된다.

단계 g. 전달 받은 노드는 위와 같은 과정을 반복 하게 된다.

실험 결과

(그림 3)은 하나의 노드가 초당 보내는 메시지의 양에 따라서 처리 하는 총 메시지의 개수를 보여준다. 기존의 방식은 요청 메시지가 증가함에 따라 처리하는 메시지 양이 작은 기울기로 증가하는 반면, 제안한 방식은 요청 메시지가 증가함에 따라 처리하는 메시지 양이 기존 방식보다 큰 기울기로 증가 하게 된다.



(그림 3) 하나의 노드가 초당 처리 하는 총 메시지의 개수