노드 복구를 고려한 Magic Square 확장

이현주[0] 정일동 손영성[†] 김경석[‡]
부산대학교 전자계산학과[0] 한국전자통신연구소[†] 부산대학교 정보컴퓨터공학부[‡]
{hjlee[0], idjung, ysson, gimgs}@asadal.cs.pusan.ac.kr

Design Improvement of Magic Square Considering Node Recovery

Hyun-ju Lee[0] Il-dong Jung Young-song Son[†] Kyongsok Kim[‡]
Dept. of Computer Science, Pusan National University[0], Electronics and Telecommunications Research Institute[†],
Division of Computer Science and Engineering, Pusan National University[‡]

요 약

Magic Square는 자원 탐색을 제공하는 P2P 프로토콜이다. 그러나 복구비용이 높고 인접한 하나 이상의 노드가 고장났을 경우에는 복구하지 못한다. 본 논문에서는 Magic Square의 복구를 위한 정보를 저장하는 복구 노드를 사용하였다. 복구 노드는 관리하는 영역 전체의 라우팅 테이블을 유지하기 때문에 노드가 고장이 나더라도 즉시 연결을 복구할 수 있다. 본 논문에서 제안하는 방법은 시스템의 성능을 떨어뜨리지 않고 모든 링크 복구와 대부분의 자원 복구를 할 수 있는 장점이 있다.

1. 서 론

인터넷이 급속하게 성장하고 초고속 인터넷이 보급되면서 클라이언트-서버 모델은 자원과 네트워크 대역폭을 활용하는데 한계를 드러내고 있다. P2P 모델은 기존 시스템의 문제점을 해결하고 P2P에 참여하는 컴퓨터끼리 직접 자원을 주고받음으로써 자원과 서비스를 공유한다.

Magic Square 시스템은 개발 중인 P2P 프로토콜이며 해시를 이용해서 자원을 저장하기 때문에 확장가능한 (scalable) 자원 탐색 기법을 사용한다. 하지만 Magic Square에서는 자원 탐색을 각 노드가 가지고 있는 이웃 노드 집합만을 이용해서 하기 때문에 원하는 자원을 노드가 고장났을 경우 찾지 못할 수도 있다. 자원 탐색 기법을 제안하는 P2P 프로토콜에서 노드 고장을 복구하는 기법은 필수적인 요소이다 [1].

본 논문에서는 노드 고장을 고려하여 Magic Square 영역을 관리하는 복구 노드를 사용하였다. 복구 노드는 각 영역의 전체 라우팅 테이블을 유지하기 때문에 노드가 고장났을 경우 즉시 복구를 할 수 있다. 뿐만 아니라 라우팅 테이블을 이용해서 대부분의 자원복구를 할 수 있다.

2. 관련 연구

2.1 Magic Square

Magic Square는 여러 가지 네트워크 환경에 독립적이고 확장 가능한 자원 탐색 기법을 제공하는 P2P 프로토콜이다. Magic Square를 구성하는 노드들은 해시 공간에 할당되며 양방향 스킵 리스트를 사용해서 라우팅 테이블을 관리한다. 노드의 레벨은 각 노드의 컴퓨팅 능력 (네트워크 대역폭, 중앙처리 장치)에 따라 결정된다. 각 노드는 라우팅을 위해 전역 테이블과, 지역 테이블을 저장한다. 전역 테이블은 Magic Square의 임의의 노드 정보로 구성되고 지역 테이블은 각 노드와 연결된 노드의 정보로 구성된다.

자원 탐색은 각 노드에서 양방향으로 검색 요청을 보내는데 스킵리스트로 라우팅 테이블이 관리되기 때문에 시간복잡도가 O(log N)이다.

또한 노드 복구와 서비스 속도 향상을 위해 각 노드와 연결된 모든 이웃 노드들에게 자원을 복사한다.

2.2 P2P에서 복구 기법

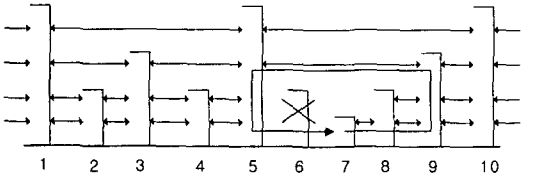
P2P 프로토콜에 대한 이전 연구에서는 각 시스템에 맞게 여러 가지 복구기법을 제시하였다.

CAN에서는 노드의 정보를 이웃 노드 집합에 중복 저장하여 노드가 고장났을 때 그 자원과 연결 정보를 복구할 수 있다. 하지만 인접한 이웃 노드 집합 모두가 고장이 난 경우는 복구할 수 없다 [2].

Chord에서는 자원이 저장된 노드를 그 자원의 successor 노드라 부른다. 노드들은 복구를 위해 가장 가까운 노드에 대한 successor-list를 유지하는데 만약 successor가 고장났다면 successor-list 내의 작동중인 노드로 대체함으로써 successor를 유지한다 [3].

3. Magic Square 확장

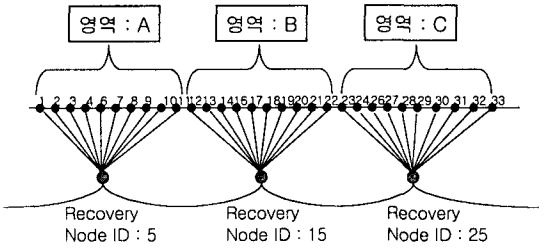
3.1 기본 Magic Square 프로토콜의 문제점



[그림 1] 노드 6이 고장났을 경우

Magic Square는 자신과 연결된 노드의 연결 정보만 유지하기 때문에 인접한 노드가 여러 개 고장났을 경우에는 모두 복구하는 것이 어렵고, 만일 링크를 복구하지 못한 경우 P2P 네트워크가 여러 부분으로 분리되는 현상이 발생한다. 이 때 사용자들이 원하는 자료가 Magic Square에 존재한다 할지라도 찾을 수 없는 문제가 생긴다. 그 뿐만 아니라 고장난 노드의 레벨이 높은 경우에는 인접한 이웃 노드만을 사용해서 복구하지 못하고 연결될 노드를 찾아야하므로 비효율적이다.

3.2 복구를 고려한 Magic Square 확장



[그림 2] Magic square 확장 구조

노드 복구를 위한 노드 (복구 노드) 는 스킵리스트를 사용해서 M개의 노드의 라우팅 정보를 유지한다. [그림 2]와 같이 Magic square를 구성하는 노드들은 그 영역을 관리하는 복구 노드와 연결되어 있다. 노드들은 일정한 시간 간격으로 alive message를 복구 노드에게 보내고 복구 노드는 3회 이상 노드에 대한 메시지를 받지 못했을 경우, 노드 고장으로 보고 복구 과정을 실행한다. 복구 노드의 고장에 대비해서 복구 노드들은 서로 연결되어 있으며 이웃한 복구 노드의 라우팅 테이블도 함께 유지한다.

그림 2에서는 복구 노드가 최대 10개의 노드를 관리한다고 가정하였다. 영역 B에 속하는 노드 10개는 복구 노드인 ID가 15인 노드가 라우팅 정보를 관리하게 된다. 노드 ID가 15인 복구 노드는 노드 5와 노드 25가 이웃 노드 집합이 되는데 복구 노드 5와 복구 노드 25의 라우팅 테이블도 복구 노드 5가 관리한다.

Magic Square는 노드의 삽입, 삭제가 빈번하게 있어나기 때문에 복구 노드가 관리하는 영역은 동적으로 결정해야 한다. 노드의 삽입이 자주 발생해서 복구 노드가 관리하는 영역의 노드 수가 M개 이상이 되면 영역을 분할하여 새로운 복구 노드를 Magic Square에 참여하는 노드 중에서 선정한다. 노드의 삭제가 많아서 이웃 복구 노드의 영역에 있는 노드의 수와 자기 영역의 노드의 수를 합쳐서 M개 이하가 되면 노드 합병이 발생하여 복구 노드 1개를 제거한다.

복구 노드가 저장하는 정보는 아래와 같다.

1) 복구 노드의 ID

확장성을 위해 기본 설계에서 생성한 노드의 ID를 그대로 쓴다.

2) start_ID, end_ID

복구 노드 영역의 시작 노드 ID, 끝 노드 ID

3) Node_Count

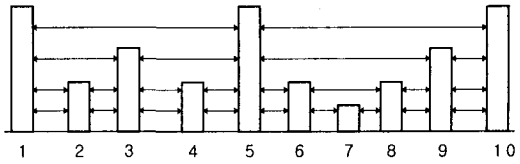
복구 노드가 관리하는 각 영역에 포함된 노드의 수

4) left_ID, right_ID

왼쪽에 있는 복구 노드의 ID와 오른쪽에 있는 복구 노드의 ID

5) 복구 노드 영역에 속하는 노드들의 링크 정보

복구 노드 1개가 관리하는 Magic Square 영역의 모습이 [그림 3]과 같은 스킵리스트로 유지하며, 최대 레벨이 4라고 가정한다. 복구 노드가 유지하는 라우팅 테이블은 [표 1]과 같이 표현할 수 있다. 복구 노드가 관리하는 노드의 ID와 IP, 그리고 그 노드의 레벨을 이용해서 노드의 정보를 저장한다. 라우팅 테이블에는 노드가 가지는 level*2 개의 링크 정보가 저장된다. 노드 1과 같이 레벨이 4인 경우 1 레벨부터 4 레벨까지의 왼쪽에 링크된 노드의 ID, 오른쪽에 링크된 노드의 ID를 유지한다. 단, 링크된 노드가 없을 경우에는 null의 값을 저장한다.



[그림 3] 영역을 구성하는 스킵리스트 연결 구조

[표 1] 라우팅 테이블의 구조

노드 정보			링크 정보								
			1 level		2 level		3 level		4 level		
node ID	node IP	node level	왼쪽 노드	오른쪽 노드	왼쪽 노드	오른쪽 노드	왼쪽 노드	오른쪽 노드	왼쪽 노드	오른쪽 노드	
1	~	4	null	2	null	2	null	3	null	5	
2	~	2	1	3	1	3					
3	~	3	2	4	2	4	1	5			
4	~	2	3	5	3	5					
5	~	4	4	6	4	6	3	9	1	10	
6	~	2	5	7	5	8					
7	~	1	6	8							
8	~	2	7	9	6	9					
9	~	3	8	10	8	10	5	10			
10	~	4	9	null	9	null	9	null	5	null	

6) left_routing_table, right_routing_table

왼쪽 복구 노드 영역의 라우팅 테이블, 오른쪽 복구 노드 영역의 라우팅 테이블

7) global_link_table

각 노드가 가지고 있는 전역 링크를 테이블로 유지한다.

8) left_global_link_table, right_global_link_table

왼쪽 복구 노드 영역의 전역 링크 테이블, 오른쪽 복구 노드 영역의 글로벌 링크 테이블

3.3 동작 과정

3.3.1 복구 노드 영역 분할

현재 복구노드가 관리하는 영역을 두 개로 분할하여 앞 부분은 기존의 복구노드가 그대로 관리하고 뒷 부분은 새로운 복구 노드를 삽입하여 관리한다. 추가되는 복구 노드는 컴퓨팅 능력이 높은, 즉 새로 분할된 영역의 노드들 중 레벨이 가장 높은 것 중 1개를 선택하기로 한다. 복구 노드의 분할과 삽입은 [알고리즘 1]을 따라 이루어진다.

[알고리즘 1] 복구 노드 삽입

```
기존의 복구 노드 : r
추가되는 복구 노드 : r'

a. 중간 노드 m = (start_ID + end_ID) /2
b. r' = m~end 영역에서 레벨이 높은 노드, Magic Square에서 삭제
c. r의 영역을 start~m 사이로 설정
d. r'의 영역을 m~end 사이로 설정
e. m~end 영역의 라우팅 테이블 r -> r'로 복사
f. r과 r'의 이웃 노드 정보를 갱신한다.
```

3.3.2 복구 노드 영역 합병

복구 노드가 관리하는 노드의 수가 적은 노드들이 많을 경우

복구비용이 높아진다. 그러므로 주기적으로 이웃한 복구 노드와의 통신을 하며 이웃한 복구 노드가 관리하는 노드의 수의 합이 M개 이하일 때 두 노드를 합병한다. 과정은 [알고리즘 2]와 같다.

[알고리즘 2] 복구 노드 합병

```
복구 노드 : r 과 r'
제거되는 복구 노드 : del_node

a. if( r의 컴퓨팅 능력이 r'보다 높다){
     new_node = r ;
     del_node = r';
  }
  else {
    new_node = r';
    del_node = r;
  }
b. new_node의 end_ID = del_node의 end_ID
e. 라우팅 테이블에서 del_node 삭제, Magic Square에 삽입
d. new_node와 del_node의 이웃한 노드들에게 갱신 메시지 보냄
```

3.3.3 복구 노드 삭제

복구 노드의 삭제는 두 가지 경우에 발생할 수 있다. 첫째 노드가 탈퇴 메시지를 이웃 노드들에게 보내고 정상적으로 종료하는 경우이다. 둘째 복구 노드의 합병으로 복구 노드가 더 이상 필요 없어진 경우이다. 두 번째 경우는 노드가 종료하는 것이 아니므로 복구 노드에서 탈퇴시키고 Magic Square에 삽입을 한다.

3.3.4 노드 복구

시스템은 복구 동작을 시작한 후부터 종료 될 때까지 다른 갱신 동작은 하지 않는다고 가정한다. 그리고 실제로 연속된 많은 수의 노드가 동시에 고장나는 경우가 발생할 가능성이 거의 없다. 본 논문에서는 연속된 복구 노드 3개가 고장나고, 가운데 복구 노드가 관리하는 영역에서 인접한 2개 이상의 노드가 동시에 고장나는 경우는 없다고 가정한다.

노드의 복구는 링크와 자원 복구로 나눌 수 있으며 링크를 복구한 뒤에 자원을 복구한다.

① 복구 시작

복구 동작 이외의 시스템의 모든 동작이 중단된다.

② 링크 복구 과정

먼저 복구 노드에서 라우팅 테이블의 정보를 이용해서 링크를 먼저 복구한 다음 Magic Square의 노드들에 적용한다. 과정은 [알고리즘 3]과 같다.

[알고리즘 3] 고장난 노드 복구

```
a. fail_node : 고장난 노드 ID

  for( 1 레벨 to fail_node의 레벨){
    left_ID = 레벨의 left_link;
    right_ID = 레벨의 right_link;
    left_ID의 right_link = right_ID;
    right_ID의 left_link = left_ID;
  }
b. 복구 노드의 라우팅 테이블에서 fail_node 삭제
c. 이웃한 복구 노드에게 갱신 메시지 보냄
```

[표 2] 링크 복구 과정

6	~	2	5	7	5	8 9	null				
7	~	1	6	8 9	null						
9	~	3	8 7	10	8 6	10	5	10	null		

[표 2]는 [표 1]의 일부분이다. 노드 8이 고장났을 경우 라우팅 테이블에서 노드 8을 찾는다. 노드 8은 레벨이 2이므로 4개의 링크를 복구해야 한다. 먼저 1 레벨의 왼쪽 노드 7번의 1 level의 오른쪽 링크에 노드 8의 값 9를 복사한다. 두 번째로 1 level의 오른쪽 노드 9의 왼쪽 링크에 노드 8의 값 7을 복사한다. 2 level 링크도 같은 방식으로 노드 6과 노드 9의 링크 값을 복구한다.

③ 자원 복구를 한다.

Magic Square는 자원을 링크로 연결된 노드들에게 복사본을 저장한다. 그러므로 한 두개의 노드가 고장났을 경우에는 자원이 적절한 노드에 복사본이 존재하게 된다. 인접한 여러 개의 노드가 고장이 났을 경우는 라우팅 테이블을 참조하여 적절한 노드에 자원을 복사해야 한다. 자원 복구 뒤에는 자원이 저장되는 노드가 바뀌었으므로 새로운 이웃 노드 집합에 자원의 버전을 하나 올려서 복사본을 저장한다.

④ 복구 종료

4. 결론

Magic Square는 복구비용이 높고 인접한 하나 이상의 노드 고장은 복구하지 못하는 문제를 가지고 있었다. 이에 본 논문에서는 복구 노드를 도입하여 복구 노드가 특정 노드 집합의 전체 라우팅 테이블을 유지함으로써 복구비용을 줄이고 모든 링크와 대부분의 자원을 복구할 수 있게 하였다. 그 뿐만 아니라 기능이 많은 복구 노드의 특성을 고려하여 컴퓨팅 능력이 좋은 노드를 복구 노드로 선택하므로 성능이 급격하게 떨어지지 않을 것으로 보인다.

본 논문은 Magic Square의 복구 기능을 개선하는 방법을 연구하는 중간 결과물이다. 복구 노드와 같은 관리 노드를 사용하지 않고 Magic Square의 복구 기능을 개선할 수 있는 연구가 필요하다. 복구 기법에 대한 성능을 평가할 수 있는 평가 모델에 대한 연구도 필요할 것으로 보인다.

5. 참고 문헌

[1] 박선미, 정일동, 손영성, 김경석, Magic Square : 노드의 능력을 고려한 자원 탐색 프로토콜, 정보과학회 춘계학술대회 (심사 중), 2003

[2] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, "A Scalable Content-Addressable Network", ACM SIGCOMM , 2001

[3] Ion Stoica, Robert Morris, David Karger, M.F *Kaashoek, Hari Balakrishnan* , "*Chord: A Scalable* Peer-to-Peer Lookup Service for Internet Applications" , ACM SIGCOMM , 2001

[4] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale Peer-to-Peer systems", IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp. 329-350, November 2001

[6] Matei Ripeanu, "Peer-to-Peer Architecture Case Study: Gnutella Network", in proceedings of 2001 IEEE International Conference on Peer-to-Peer Computing , 2001