Magic Square : 노드의 능력을 고려한 자원 탐색 프로토콜

박선미°, 정일동, 손영성†, 김경석‡
부산대학교 전자계산학과, 한국 전자통신 연구소†, 부산대학교 정보컴퓨터공학부‡
{smpark°, idjung, ysson, gimgs}@asadal.cs.pusan.ac.kr

Magic Square : Resource lookup protocol considering computing power of node

Sun-mi Park°, Il-dong Jung, Young-sung Son†, Kyongsok Kim‡
Dept. of Computer Science, Pusan National University, Electronics and Telecommunications Research Institute†,
Division of Computer Science and Engineering, Pusan National University‡

요 약

P2P 시스템의 주요 이슈는 자원 (resource) 을 효율적으로 저장하고 찾는 것이다. 자원 탐색 프로토콜은 초기의 Napster , Gnutella와 같은 형태에서 발전하여, 현재는 분산 해시 테이블 (Distributed Hash Table) 을 사용한 형태로 발전하고 있다. 본 논문에서는 분산 해시 테이블을 사용한 P2P 프로토콜인 Magic Square를 제안한다. Magic Square에 참여하는 각 노드는 양방향 스킵리스트로 구성된 지역 라우팅 테이블과 임의의 노드로 구성된 전역 라우팅 테이블을 가진다. 지역 라우팅 테이블은 각 노드의 능력을 고려하여 구성된다. 스킵리스트를 사용하였기 때문에 탐색과 노드의 추가와 삭제 과정이 간단하며, P2P 네트워크가 자주 바뀌어도 큰 영향을 받지 않는다.

1. 서 론

지금까지의 인터넷 서비스는 클라이언트/서버 모델을 기반으로 하고 있다. 클라이언트/서버 모델은 클라이언트의 능력이 서버에 비해 떨어진다는 점을 가정한다. 그러나 최근 클라이언트의 컴퓨팅 능력이 향상되고 초고속 정보통신망이 보급되면서 클라이언트와 서버의 구분이 모호하게 되었다.

이러한 상황에서 최근 P2P 시스템이 각광받고 있으며, P2P 시스템에 관한 많은 연구가 진행 중이다. P2P 시스템은 각 노드 사이에 직접적인 연결을 통해 자원과 서비스를 공유한다. 각 노드는 서버와 클라이언트, 두 가지의 역할을 한다. 단일 고장 지점이자 병목 현상의 원인이 될 수 있는 서버를 제거함으로써 P2P 시스템은 전체 시스템의 신뢰성 (reliability) 과 성능 (performance) 을 향상시킬 수 있다 [1].

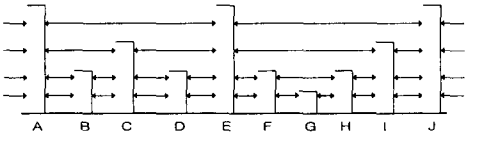
현재 P2P 시스템의 주요 이슈는 자원 (resource) 을 효율적으로 저장하고 자원의 위치를 찾는 것이다. 초기 P2P 시스템에서는 자원의 목록 및 위치를 가지고 있는 서버를 두거나 (Napster) , 찾고자 하는 자원에 대한 질의를 방송 (broadcast) 함으로써 (Gnutella) 자원에 대한 탐색을 실행하였다. 그러나 이러한 방식은 확장 가능성 (scalability) 에 제한을 가지고 있다.

최근 P2P 시스템에서의 탐색에 관한 연구에서는 분산 해시 테이블 (Distributed Hash Table) 을 사용한 탐색 기법을 제시하고 있다. 분산 해시 테이블을 사용한 프로토콜로는 CAN, Chord, Pastry 등이 있다. 이러한 프로토콜들은 노드의 컴퓨팅 능력이 같다고 가정하고 있지만, 실제 현실에서는 노드들의 능력은 서로 다르다. 본 논문에서는 노드의 능력을 고려한 분산 해시 테이블을 사용한 탐색 프로토콜인 Magic Square를 제안한다 [2,3,4].

2. 관련 연구

2.1 스킵리스트

Magic Square에 연결된 노드는 스킵리스트를 사용하여 라우팅 테이블을 관리한다.



[그림 1] 스킵리스트

스킵리스트는 균형 트리 (balanced tree) 보다 간단하면서 확률적으로 비슷한 성능을 내는 자료구조로 1990년에 소개되었다 [5]. 스킵리스트는 기존의 연결 리스트 (linked list) 에 임의의 노드마다 몇 개의 연결을 추가한 것으로 [그림 1]과 같다. 각 노드가 가지는 연결의 수를 레벨 (level) 이라 한다. 레벨이 임의로 정해지기 때문에 균형을 이루며 임의의 노드를 찾는데 O(log N)의 시간 복잡도 (*time complexity*) 를 가진다. 이 수치는 최적화된 균형 트리의 시간 복잡도와 같다. 스킵리스트는 균형 트리보다 알고리즘이 간단하여 구현이 쉽지만, 균형 트리 만큼의 성능을 보장하는 장점이 있다.

2.2 자원 탐색

P2P 시스템의 주요 이슈는 효율적으로 자원을 저장하고 찾는 것이다. 자원을 찾기 위한 프로토콜로 다음과 같은 것이 있다. 첫째, Napster와 같은 자원의 이름과 위치를 관리하는 서버를 사용하는 형태이다. 이 형태는 확장가능성이 떨어지고, 중앙 서버의 고장 (failure) 에 취약하다는 단점이 있다. 둘째, Gnutella와 같이 자원을 찾는 방송 (broadcast) 을 통해 자원을 가진 노드를 찾아 자원을 제공받는 형태이다. 이 방법은 방송 메시지가 계속 전달 (forward) 되어 루프를 가지지 않도록 해야 하며, 네트워크 대역폭이 낭비되는 문제가 있다. 셋째, 분산 해시 테이블을 사용하는 것이다. CAN , Chord, Pastry 등의 연구에서 사용된 방법이다. 이 방법은 최근에 많

이 연구되고 있으며, 각 노드가 전체 노드에 대한 정보가 아닌 전체 노드 중 일부에 대한 정보를 라우팅 테이블에 저장하는 방법이다. 일부 노드에 대한 정보를 가지고 각 프로토콜에서 정의한 방법에 따라 자원을 찾아가는 형태이다.[2,3,4]

3. Magic Square

3.1 Naming

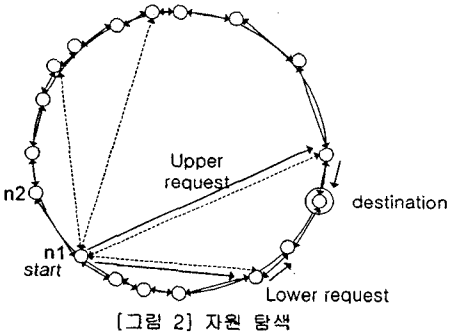
해시 공간에서 각각의 노드를 구별하기 위해 노드에는 고유의 ID를 부여한다. 노드의 ID는 고정 IP를 가진 경우 IP를 , 고정 IP를 가지지 못한 경우 사용자가 정의한 ID를 해시하여 만든다. 자원 ID는 자원의 key를 해시하여 만든다. Magic Square에서는 해시 함수로 SHA-1을 사용한다. SHA-1을 사용함으로써 노드 ID (또는 자원 ID) 는 해시 공간에 균일(uniform) 하게 분포한다.

[그림 2]와 같이 n1, n2 두 개의 노드[1] (n1<n2) 가 연속으로 존재한다고 가정하자. 이때 n1은 자원 ID가 n1과 같거나 크고 n2보다 작은 자원을 관리하게 된다. 즉, 하나의 노드는 시계 방향으로 자신의 오른쪽에 있는 노드 범위까지의 자원을 관리한다. 임의의 노드가 Magic Square에 참여하거나 떠나게 될 때, 시계 방향으로 왼쪽에 있는 노드의 데이터베이스와 (자원 관리를 위해) 이웃한 노드의 라우팅 테이블만이 수정되므로 노드가 네트워크에 자주 접속하고 연결이 끊어지는 것에 큰 영향을 받지 않는다.

3.2 자원 탐색

Magic Square에 참여하는 노드는 자원을 찾기 위해 지역 테이블 및 전역 테이블을 사용한다. 지역 테이블은 양방향 스킵리스트로 유지하여 이웃한 노드에 대한 정보가 들어간다. 전역 테이블에는 임의로 선택된 노드에 대한 정보가 들어간다. 지역 테이블을 스킵리스트로 구성하기 때문에 O(log N)의 탐색 시간이 보장된다. 전역 테이블은 스킵리스트와는 별개의 임의의 노드를 가짐으로써 지름길 (shortcut) 의 효과를 기대할 수 있다 [6].

지역 테이블에는 nodeId와 IP주소 그리고 연결된 레벨 수준에 대한 정보를 포함한다. 전역 테이블은 nodeId와 IP 주소만을 가진다. 임의의 자원을 찾으려는 노드는 레벨이 높으면서 자원ID에 가까운 노드로 요청을 보내게 된다. 이때, 좀더 빠른 탐색을 위하여 양방향으로 요청을 보낸다. [그림 2]는 Magic Square에서의 탐색을 나타내고 있다.



[그림 2] 자원 탐색

1) 노드의 ID를 표기하는 방법이다.

탐색 메시지에는 다음과 같은 정보를 포함한다.

msgId	type of msg	source IP	resourceId

msgId : 중복된 메시지에 대한 서비스인지 확인할 때 사용 [*nodeId : msg_sequence_num*] 으로 구성한다. msgId는 메시지를 보내는 각 노드별로 관리를 하게 된다. 새로운 탐색 메시지를 보낼 때마다 msg_sequence_num을 증가시킨다.

type of msg : SRCH (탐색 메시지)

자원 탐색 알고리즘은 [알고리즘 1]과 같다.

[알고리즘 1] 자원 검색 알고리즘

```
if(msgId is in msgIdQue) terminate;
else{
   search resourceId in database;
   if(resourceId is in database){
       send find message;
   }
   else{
       divide routing table into two fraction;
       select total two nodes which is near to
resourceId and highest level in each fraction;
       forward search message to two nodes;
   }
}
```

3.3 노드 추가

노드를 추가할 때에는 자원 탐색 방법을 활용한다. 추가할 노드를 n이라고 할 때, n을 Magic Square에 추가하기 위해서는 먼저 n의 레벨과 ID를 정한다. 레벨이 높은 노드일수록 더 많은 탐색 요청을 받을 확률이 커진다. 또한 스킵리스트에 연결된 노드에 자신의 자원을 중복 저장 (replication) 한다. 이와 같은 점을 반영하여 Magic Square에서는 노드의 능력을 고려하여 레벨을 결정한다. 즉, 노드의 컴퓨팅 능력이 뛰어날수록 높은 레벨을 할당받는다.

Magic Square에 노드를 추가하는 과정은 [알고리즘 2]와 같다.

[알고리즘 2] 노드 추가

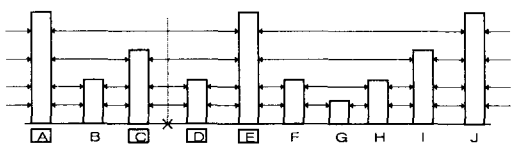
```
//a node to be inserted => n
//si is a node in magic square which is found using
broadcast
request si that search a node which is neighbor of n at
level 1;

// n1, n2 : neighbor of n
Ask node k which is linked n1 or n2 whether pass n;

while(curlinklevel <= myMaxLevel){
    receive link information;
    update routing table;
}

take data of n from n2;
```

n의 ID가 결정되면 n이 들어갈 위치가 결정된다. n의 라우팅 테이블을 구성하기 위해 먼저 레벨 1에서 n의 양옆에 존재하는 노드 n1, n2 (n1<n<n2) 를 찾는다. n1, n2는 data 탐색 알고리즘을 활용하여 찾는다. n1, n2를 찾게 되면, n1, n2를 시작으로 n을 지나는지를 묻는 메시지를 보낸다. 이 메시지를 받은 노드는 현재 연결된 레벨보다 높은 레벨에서 n을 지나갈 경우, 자신의 라우팅 테이블을 고치고, 그 사실을 n에게도 알려준다. 그리고 현재 연결된 레벨을 수정한 후, 같은 메시지를 현재 연결된 레벨 이상의 레벨을 가지는 노드에게 보낸다.

이 메시지는 n의 최고 레벨에서 연결될 노드를 찾을 때까지 다음 노드로 전송된다. 모든 레벨에서의 연결이 이루어지면 n의 라우팅 테이블 구성은 끝난다. 마지막으로 n은 자신의 데이터를 n2로부터 받는다.

3.4 자원의 중복 저장

자원을 한 곳의 노드에만 저장시킬 경우, 자원이 저장된 노드가 고장 (failure) 날 경우 자원을 복구할 수 없다. 노드가 고장나더라도 자원을 사용하기 위하여 Magic Square에서는 목적지를 포함하는 모든 스킵리스트 상의 노드에 자원을 중복 저장한다. [그림 3]은 자원 ID가 C와 D 사이의 값일 경우, 노드의 목적지를 포함하는 노드 A,C,D,E에 저장됨을 보인다.



[그림 3] 자원의 중복 저장

4. 성능평가

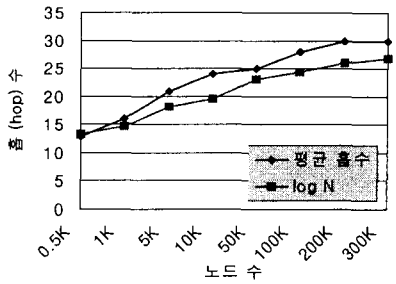
자원을 탐색할 때 필요한 메시지 수는 분산 해시 테이블을 사용하는 P2P 시스템의 성능을 평가하는 척도이다. 따라서 Magic Square의 성능을 평가하기 위해서 자원 탐색에 필요한 메시지 수를 측정한다. 시뮬레이터는 JDK 1.3.1을 사용해서 작성했으며, 메시지 수는 홉 (Hop) 수로 표시하였다.

Magic Square의 성능을 평가하기 위해서 [표1]의 변수를 사용하였다.

[표1] 실험 환경

변수	값
노드 수	500 ~ 300,000
요청 수	노드 수의 50%

[그림 4]와 같이 탐색에 필요한 평균 메시지의 수는 log N에 근접하고 있다. Magic Square에 참여하는 노드의 수가 더 많아지면 log N 의 값과 같아질 것으로 보인다. 이는 탐색의 시간 복잡도가 가장 작은 균형 트리의 탐색 비용과 같다.



[그림 4] 노드 수에 따른 평균 탐색 시간

5. 결론

본 논문에서는 분산 해시 테이블을 사용하는 P2P 프로토콜인 Magic Square를 제안하였다. Magic Square는 노드가 네트워크에 자주 접속하고 연결이 끊어지는 P2P 환경에서 자원을 공유/탐색할 수 있음을 알 수 있었다. 자원을 탐색하기 위해 분산 해시 테이블을 사용하였으며, 분산 해시 테이블 상의 노드들은 스킵리스트로 연결되어 있다. 스킵리스트를 사용하였기 때문에 O(log N)의 탐색 비용이 보장된다. 스킵리스트에서의 탐색 및 조인 방법을 활용하여 노드 탐색 및 조인 방법이 간단하다.

본 논문에서는 자원을 공유/탐색하는 프로토콜을 설계해 보았다. 향후 연구과제로는 첫째, 여러 개의 노드가 동시에 참여 할 때의 안정성 검사, 둘째, Magic Square 프로토콜의 구현, 셋째, 분산 해시 테이블을 적절히 이용한 복구 기법 등이 있다.

5. 참고 문헌

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

[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, 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

[5] William Pugh, "Skip Lists : a probabilistic alternative to balanced Trees" , communications of the ACM, 33권 6호 pp.668-676 , June 1990

[6] Hui Zhang, Goel A., Govindan R., "Using the small-world model to improve Freenet performance", INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies., vol.3 pp.1228 -1237, 2002

[7] FIPS 180-1, Secure Hash Standard, U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, VA, Apr.1995