효율적인 분산 저장 시스템을 위한 대칭 트리 복제 프로토콜

(Symmetric Tree Replication Protocol for Efficient Distributed Storage System)

최 성 춘 † 윤 희 용 †† 이 강 신 ††† 이 호 재 ††††
(Sungchune Choi) (Hee Yong Youn) (Kang Shin Lee) (Ho Jae Lee)

요 약 최근 분산 컴퓨팅 환경에서 데이타와 서비스의 복제는 통신비용의 감소, 데이타 가용성 증가, 그리고 단일 서버의 병목현상을 피하기 위해 필수적이다. 기존의 대표적인 복제 프로토콜로 네트워크를 논리적으로 구성하는 Tree quorum 프로토콜과 Grid프로토콜이 있다. Tree quorum프로토콜은 최선의 경우 가장 우수한 읽기 성능을 보이는 반면 트리의 높이가 증가할수록 노드의 수가 기하급수적으로 증가한다는 단점을 가지고 있다. Grid프로토콜은 읽기 동작에 있어 높은 가용성을 가지는 반면 고장이 없는 환경에서도 같은 읽기 및 쓰기 성능을 보이는 단점을 가지고 있다. 따라서 본 논문에서는 기존의 복제 프로토콜이 가지는 문제점을 해결하고, 대칭적 트리 구조를 이용하여 노드의 장애가 발생하는 환경에서도 우수한 성능을 갖는 복제 프로토콜을 제안한다. 제안된 복제 프로토콜은 Tree quorum 프로토콜에 비해 적은 읽기 비용을 가지며, 적은 수의 노드 구성 환경에서도 높은 읽기 가용성을 가진다. 또한 응답시간 면에서도 우수한 성능을 보인다.

키워드 : 복제 프로토콜, tree quorum, 가용성, 분산 시스템, 대칭 트리

Abstract In large distributed systems, replications of data and service are needed to decrease communication cost , increase availability , and avoid single server bottleneck. Tree Quorum protocol is a representative replication protocol, which exploits a logical structure. Tree quorum protocol is one of the replication protocols allowing low read cost only in the best case, while the number of replicas exponentially increases as the level grows. In this paper, thus, we propose a new replication protocol, called symmetric tree protocol which efficiently solves the problem. The proposed symmetric tree protocol also requires much smaller read cost than previous protocols. We conduct cost and availability analysis of the protocols, and the proposed protocol displays comparable read availability to the tree protocol using much smaller number of nodes. Also, the symmetric tree protocol has much smaller response time than the logarithmic protocol.

Key words : Replication protocol, tree quorum, availability, distributed system, symmetric tree

1. 서 론

오늘날 대용량 분산 컴퓨팅 시스템에서 성공적인 동작을 방해하는 노드와 통신 장애는 자주 발생한다. 이러한 장애는 분산 시스템에서 고장 감내 기법의 개발이 필요하도록 만들었다[1]. 장애 외의 다른 문제는 시스템의 규모가 커지면서 통신비용이 증가하고, 많은 프로세스들이 같은 작업을 수행하기 위해 경쟁함으로서 서버의 부하가 증가하게 된다. 분산 컴퓨팅 환경에서 데이타와 서비스의 복제는 이러한 문제를 해결하기 위한 기술 중의 하나이다. 복제 기법은 데이타의 가용성을 높이고, 전체 시스템의 성능을 향상시킬 수 있다. 또한 다중 노드를 사용함으로써 단일 노드의 사용으로 인한 병목현상 문제를 해결할 수 있다. 그러나 노드의 수가 증가할수록 통신비용은 증가하게 되므로, 전체 시스템을 구성하는 노드 중에 읽기/쓰기 동작을 수행하는 노드의 수는 가능하면 적은 수로 유지되어야 한다[2,3].

† 비 회 원 : 성균관대학교 정보통신공학부 컴퓨터공학
choisc@ece.skku.ac.kr
†† 종신회원 : 성균관대학교 정보통신공학부 교수
youn@ece.skku.ac.kr
††† 비 회 원 : 한국정보보호진흥원 기반보호기술팀 팀장
kslee@kisa.or.kr
†††† 비 회 원 : LG전자 전자기술원 주임연구원
hjlee72@lge.com
논문접수 : 2004년 10월 22일
심사완료 : 2004년 5월 13일

데이타가 여러 노드에 복제된 분산 시스템은 복제된 데이타에 대한 읽기와 쓰기의 두 가지 동작을 수행한다. 읽기 동작은 쓰기 동작에 의해 가장 최근에 쓰여진 데이타를 제공한다. 읽기 동작이 가장 최신의 데이타를 제공하고, 읽기/쓰기 동작과 쓰기/쓰기 동작이 동시에 수행되지 않도록 하기 위해서는 적당한 동기화 기법이 필요하다. 각 노드는 노드 자체의 복사본에 대한 접근을 동기화 하기 위해 중앙 집중 일관성 제어 기법을 사용하고, 다양한 복제 데이타의 접근을 조정하기 위해 복제 제어 프로토콜을 사용한다[5-7]. 복제된 데이타의 일관성을 유지하기 위한 기본 원리는 공통된 복사본을 필요로 하는 동작들 간에 최소한 하나 이상의 공통된 복사본이 존재하도록 하는 것이다. 일관성 제어를 위해 기존의 대표적인 방식은 읽기/쓰기 동작을 수행하는데 필요한 노드의 집합을 정의하는 Quorum이 존재한다. Quorum은 읽기 동작을 위한 RQ(read quorum)과 쓰기 동작을 위한 WQ(write quorum)으로 정의된다. 여기서 WQ과 RQ은 적어도 하나 이상의 노드를 중복하여 포함하므로 공통된 복사본이 존재하게 된다.

많은 복제 프로토콜은 다른 목적을 위하여 여러 연구를 통해 제안되었다. ROWA(Read-One/Write-All) 알고리즘[4]은 최소 읽기 비용과 최대 읽기 가용성을 가지는 반면 쓰기 동작에 있어 최대 통신비용을 가지는 단점을 가진다. Quorum Consensus[5]와 Dynamic Voting[6] 프로토콜은 우수한 읽기와 쓰기 가용성을 가진다. 하지만 높은 읽기 비용을 가지는 단점을 가진다. 더욱 낮은 읽기 동작 비용을 가지는 복제 제어 프로토콜은 상대적으로 더욱 높은 쓰기 동작 비용을 가지게 된다. 일반적인 기법들은 동작 비용으로 O(n)을 갖는다. 이것은 동작 비용이 시스템내의 노드의 수와 선형적인 관계를 갖는다는 것을 의미한다. Multi-Level Voting 프로토콜[8], Weighted Voting 프로토콜[9], 그리고 Grid 프로토콜[10]과 같이 복제 노드들의 논리적 구성을 통하여, 복제 제어 프로토콜은 통신비용을 감소시킬 수 있다. 그러나 이와 같은 프로토콜은 여전히 복제 노드의 수가 많을 경우 상대적으로 높은 읽기 비용을 가지며, 복제 노드의 장애와 상관없이 높은 동작 비용을 가진다. 최근에는 Grid 네트워크 환경에서 많은 데이타 복제 기법[11,12]들이 연구 및 개발되고 있다.

기존의 대표적인 복제 프로토콜은 Tree quorum 프로토콜과 Grid 프로토콜이다. Tree quorum 프로토콜[13]과 Grid 프로토콜 [14]은 노드들의 논리적인 구성을 통하여 전체 노드 중 일부의 노드만을 이용하여 읽기/쓰기 동작을 수행하는 복제 프로토콜이다. 물론 일부 노드만을 사용함으로써 발생하는 일관성 문제를 해결하기 위해 이전에 설명한 Quorum 프로토콜을 사용한다. Tree quorum 프로토콜은 트리의 루트 노드를 이용하여 루트 노드가 장애가 발생하지 않을 경우 1의 적은 읽기 비용을 가지는 반면 트리의 높이가 증가할수록 노드의 수가 기하급수적으로 증가한다는 단점을 가지므로 더 많은 노드 접근으로 인한 높은 읽기 비용을 가지는 문제점을 가진다. Grid 프로토콜은 노드를 행과 열로 구성하여 논리적인 그리드 형태로 구성된다. Grid 프로토콜은 Tree quorum 프로토콜에 비해 높은 가용성을 보이는 반면 노드의 장애가 발생하지 않는 환경에서도 항상 같은 읽기/쓰기 비용을 갖는다는 단점을 가지고 있다. 따라서 본 논문에서는 기존의 Tree quorum 프로토콜과 Grid 프로토콜이 가지는 장점을 모두 가지면서 기존 프로토콜의 단점을 해결할 수 있는 대칭 트리 복제 프로토콜을 제안한다. 제안된 대칭 트리 복제 프로토콜은 대칭적인 구조의 트리를 사용하는 방식으로 안정한 읽기 성능과 높은 읽기 가용성을 가지게 된다. 또한 효율적인 노드 접근 방식으로 기존의 프로토콜에 비해 적은 응답시간과 높은 처리율을 가진다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 복제 프로토콜에 대하여 설명하고, 3장에서는 본 논문에서 제안하는 대칭 트리 복제 프로토콜을 소개한다. 4장에서는 기존 프로토콜과의 읽기/쓰기 비용, 가용성, 그리고 응답시간을 비교하고, 마지막으로 5장에서는 논문의 결론을 제시한다.

2. 기존 연구

2.1 Logarithmic 프로토콜

이번 절에서는 Tree quorum 프로토콜 중 Logarithmic 프로토콜에 대하여 설명하려 한다. Tree quorum 프로토콜은 그림 1과 같이 트리 구조의 논리적인 구성을 이용하며, 장애가 발생하지 않을 경우 읽기 비용 1을 가진다. 기존의 복제 프로토콜이 장애의 발생 유무에 상관없이 항상 같은비용과 가용성을 가지는 반면 트리 구조를 사용하는 복제 프로토콜은 노드의 분산 정도와 장애가 발생한 노드의 수에 따라 다양한 성능과 가용성을 가진다.

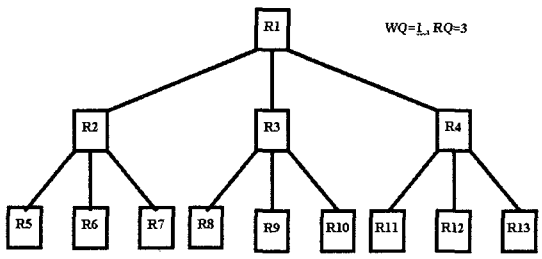


그림 1 13개의 노드를 갖는 높이 3의 트리

높이 h의 트리로 구성된 n개의 노드가 있다고 가정하면, 잎 노드를 제외한 각 노드 R_i는 S_{R_i}수의 자식 노드를 갖는다. 각 노드들을 위한 읽기와 쓰기 집합을 정의하기 위해 rq_{R_i}와 wq_{R_i}을 정의한다.

정의. RQ(Read Quorum): 복제된 데이타를 저장하는 노드들 중에 읽기 동작을 수행하기 위해 필요한 노드들의 집합을 의미한다. 선택된 노드들 중에 가장 최신의 데이타를 가지고 있는 노드가 선택된다. Tree qourum 프로토콜에서 가용한 RQ은 임의의 레벨에서 정상적으로 동작하는 노드와 장애가 발생한 노드의 하위레벨에서 정상적으로 동작하는 노드들이다. 하위 레벨로의 이동은 각 레벨에서 모든 노드가 정상적으로 동작할 때까지 재귀적으로 트리의 끝까지 계속 된다.

WQ(Write Quorum): 전체 노드들 중에 쓰기 동작을 수행하기 위해 필요한 노드들의 집합을 의미한다. Tree quorum 프로토콜에서, 가용한 WQ은 트리의 각 레벨에서 임의의 한 노드를 선택하게 된다. 결국 트리의 레벨과 동일한 수의 노드가 선택되어진다.

• 알고리즘

읽기 동작 : 루트 노드 또는 만약 루트 노드가 장애가 발생하면 루트의 rq_{R_i}를 읽게 된다. 루트의 자식 노드는 다시 루트 노드와 같이 동작하게 된다.

쓰기 동작 : 루트 노드 그리고 루트의 wq_{R_i}를 쓰게 된다. 선택된 자식 노드는 다시 루트 노드와 같이 동작하게 된다.

그림 1은 13의 노드로 구성된 3 레벨의 트리 구조이다. 그림 1에서 가용한 RQ은 만약 R1이 가용 하다면 {R1}이 되고, 만약 R1이 장애가 발생한다면 {R2, R3, R4}가 된다. 그리고 만약 R1, R2, R4가 장애가 발생한다면 {R5, R6, R7, R3, R11, R12, R13}이 된다. 가용한 WQ은 {R1, R2, R5}, {R1, R3, R10}, 그리고 {R1, R4, R12}이다.

복제된 노드들의 일관성을 유지하기 위해, 선택된 RQ와 WQ은 몇 가지 조건을 만족해야한다. 읽기와 쓰기 동작이 동시에 발생하지 않도록, 즉 읽기/쓰기 동작의 충돌을 감지하기 위해 임의의 가용한 RQ은 가용한 WQ과 반드시 하나 이상의 동일한 노드가 존재해야 한다. 이 조건은 $rq_{R_i}+ wq_{R_i} > V_{R_i}$에 의해 만족될 수 있다. 여기서 V_{R_i}는 각 노드에 대한 자식 노드의 수를 의미한다. 쓰기/쓰기 동작의 충돌을 감지하기 위해, 임의의 WQ은 반드시 공통된 하나의 노드를 가져야 한다. 이것은 모든 가용한 WQ이 루트 노드를 반드시 포함해야 하므로 만족될 수 있다. Tree quorum 프로토콜은 다양한 rq_{R_i}와 wq_{R_i}값을 가질 수 있으며, 그에 따라 다른 성능을 보인다. 이때 $rq_{R_i}=S_{R_i}$의 값과 $wq_{R_i}=1$의 값을 가지는 프로토콜을 Logarithmic 프로토콜이라 하며, Tree quorum 프로토콜에서 가장 좋은 성능을 나타낸다.

2.2 그리드(Grid) 프로토콜

그리드 프로토콜은 그림 2와 같이 행과 열의 논리적인 그리드 형태로 구성된다. 읽기와 쓰기 동작들은 읽기/쓰기와 쓰기/쓰기와 같은 동작간의 충돌을 감지하기 위해 행과 열에 대한 RQ과 WQ을 요구한다. Grid 프로토콜에서 RQ은 각 열에서 임의의 하나의 노드를 선택하여 구성되고, WQ은 각 열에서 임의의 하나의 노드와 임의의 하나의 전체 열로 구성된다. 그림 2는 4행 4열의 16개 노드를 갖는 그리드 네트웍 구조를 보여준다.

• 알고리즘:

읽기 동작 : 각 열의 임의의 하나의 노드를 읽는다. 각 열에서 선택된 노드의 집합을 C-cover라고 부른다.

쓰기 동작 : C-cover와 임의의 열 전체를 쓴다.

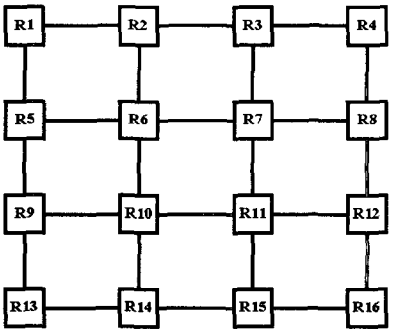


그림 2 16개의 노드를 갖는 그리드 네트워크

그림 2에서 읽기 동작을 위해 가능한 노드들의 집합은 {R1, R6, R3, R12}와 {R1, R6, R7, R8}이 된다. 쓰기 동작을 위해 필요한 노드들의 집합은 {R1, R10, R3, R4, R2, R6, R14}와 {R9, R6, R3, R8, R7, R11, R15}이다.

여기서 쓰기 동작은 각 열에서 하나의 노드와 임의의 전체 열에 대하여 동작하게 되므로 두 개의 쓰기 동작은 동시에 수행될 수 없다. 만약 하나의 쓰기 동작이 전체 열에 대하여, 즉 C-cover를 사용되고 있다면 다른 쓰기 동작은 그 열에 접근할 수 없으므로 쓰기 요청은 대기 상태가 된다. 물론 이것은 쓰기와 읽기 동작이 동시에 수행될 수 없는 이유가 된다. 그리드 프로토콜은 장애의 발생 여부에 상관없이 RQ과 WQ의 수가 항상 동일하므로 고정된 동작 비용을 가지게 된다. 하지만, 그리드 프로토콜은 Logarithmic 프로토콜에 비해 높은 가용성을 가진다. 다음 장에서는 Logarithmic 프로토콜의 단점을 해결하고, 그리드 프로토콜과 같이 보다 안정된 성능을 보이는 대칭 트리 프로토콜에 대하여 설명한다.

3. 제안하는 대칭 트리 프로토콜

3.1 동기

기존의 대부분의 복제 프로토콜은 읽기와 쓰기 동작을 위해 복사본을 가지고 있는 대부분의 노드를 접근해야 한다. 예를 들어, 16개의 노드를 갖는 Tree Quorum Consensus 프로토콜의 경우, 읽기 동작과 쓰기 동작을 위해 필요한 노드 수의 합은 전체 노드의 개수 보다 많은 노드를 필요로 한다. 하지만, 만약 노드를 논리적 형태로 구성한다면, 동작을 위해 접근하는 노드의 수는 더욱 줄어들게 된다. 예를 들어, 16개의 노드를 갖는 그리드 프로토콜의 경우, 읽기 동작을 위해 필요한 노드의 수는 4이고, 쓰기 동작을 위해 필요한 노드의 수는 7이 된다. 하지만 그리드 프로토콜은 장애의 발생 여부에 상관없이 RQ과 WQ의 수가 항상 동일하므로 고정된 동작 비용을 가지게 된다. 그러므로 장애가 발생하는 않는 환경을 위해 더욱 효과적인 다른 구조를 필요로 하게 되었다. 이를 위해 제안된 프로토콜이 Tree Quorum 프로토콜이다. 13개의 노드를 갖는 Tree Quorum 프로토콜은 최선의 경우 읽기 비용으로 1을 가지며, 쓰기 비용으로 3을 가진다. 하지만 Tree Quorum 프로토콜은 트리의 레벨이 증가함에 따라 노드의 수가 급격하게 증가한다는 단점을 가지고 있다. 예를 들어, 5레벨의 Logarithmic 프로토콜의 경우는 최악의 경우 읽기 비용으로 81개의 노드를 필요로 하게 된다. 또한 그리드 프로토콜에 비해 낮은 가용성을 가지게 된다.

그러므로 본 논문에서는 이전에 언급한 기존 프로토콜의 문제점을 해결할 수 있는 새로운 대칭 트리 프로토콜을 제안하려 한다. 제안된 프로토콜은 대칭적 트리 구조를 사용하므로 Tree Quorum 프로토콜과 달리 트리의 레벨이 증가함에 따라 노드의 수가 급격하게 증가하지 않으며, 두 개의 루트 노드를 사용하여 읽기 동작을 수행함에 따라 하나의 루트 노드 장애에도 불구하고 낮은 읽기 비용을 가질 수 있다. 대부분의 복제 프로토콜과 동일하게, 본 논문에서 제안한 프로토콜에서는 노드의 장애는 발생할 수 있지만, 비잔틴 고장은 고려하지 않는다.

3.2 제안된 대칭 트리 프로토콜

이전 장에서 설명한 것과 같이, Tree quorum 프로토콜과 Grid 프로토콜은 몇 가지 단점을 가지고 있다. 따라서 본 논문에서는 계층적 트리 구조를 이용하여 노드의 장애가 발생하지 않을 경우 우수한 읽기 성능을 가지는 Tree quorum 프로토콜의 장점을 가지며, 노드의 급격한 증가를 막을 수 있는 새로운 대칭적 트리 구조의 복제 프로토콜을 제안한다. 또한 제안된 프로토콜은 대칭 트리 구조를 사용하여 노드의 장애가 발생할 때 읽기 비용이 급격하게 증가하는 기존 복제 프로토콜의 문제를 해결한다. 제안된 대칭 트리 프로토콜은 그림 3과 같이 노드들이 구성되며, 두 개의 트리 구조는 중앙의 노드들을 기준으로 대칭적으로 구성된다. 대칭적인 두 개의 트리 중에 위의 트리를 상향 트리, 아래의 트리 구조를 하향 트리라고 부른다.

• 알고리즘

읽기 동작 : 동작은 상향 트리나 하향 트리의 루트 노드부터 시작한다. 만약 루트 노드가 장애가 발생하여 읽기 동작을 수행하지 못할 경우, 루트 노드의 모든 자식 노드에 대하여 읽기 동작을 수행하게 된다. 루트 노드의 각 자식 노드는 대칭 트리의 h/2-레벨에 도달할 때까지 루트 노드와 동일하게 동작하게 된다. 여기서 h는 트리의 레벨을 의미한다. h/2-레벨 후에는, 부모 노드들 중에 하나의 노드라도 장애가 발생하면 다음 레벨을 읽게 된다.

쓰기 동작 : 동작은 두 트리의 임의의 루트 노드부터 시작된다. 루트의 자식 노드 중에 반드시 하나의 노드는 쓰기 동작을 수행해야 하며, 선택된 자식 노드는 루트 노드와 동일하게 동작하게 된다. h/2-레벨 후에는, 각 레벨에서 하나의 노드만 존재하기 때문에 마지막 레벨까지 하나의 노드에 쓰기 동작을 수행하게 된다.

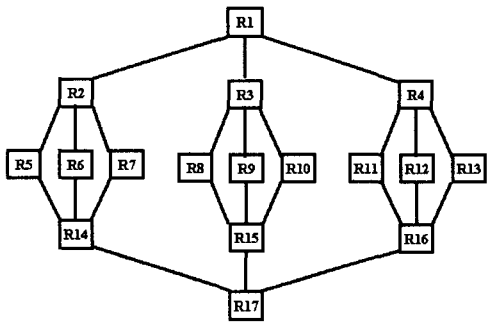
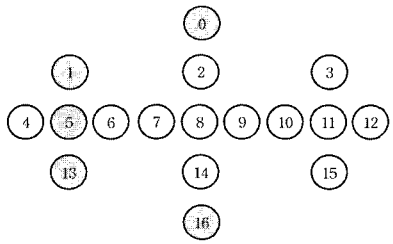


그림 3 17개의 노드를 갖는 제안된 대칭 트리 프로토콜

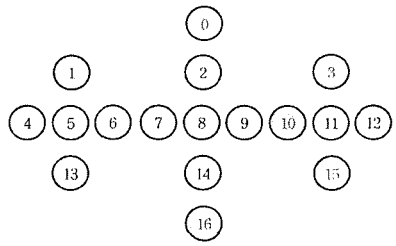
그림 3의 구조에서, 읽기 동작을 위해 가능한 노드들의 집합은 {R1}, {R17}, {R3, R4, R5, R6, R7}, {R2, R4, R15}, 그리고 {R5, R6, R7, R17}이 된다. 쓰기 동작을 위해 필요한 노드들의 집합은 {R1, R3, R10, R15, R17}와 {R1, R2, R5, R14, R17}이 된다.

3.3 제안된 대칭 트리 프로토콜의 동작 예

그림 4는 제안된 대칭 트리 프로토콜의 쓰기 동작 예를 보여준다. 이 예는 17개의 노드를 갖는 레벨 5의 구조를 사용한다. 그림 4의 (a)와 (b)는 쓰기 동작을 위해 가능한 두 가지 경우를 보여준다. 쓰기 알고리즘에 의해 각 레벨의 하나의 노드에 대하여 쓰기 동작을 수행한다.

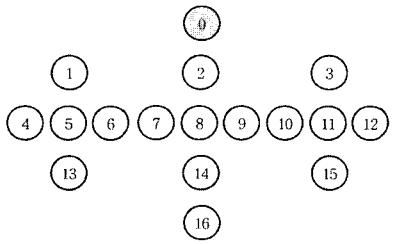


(a) WQ={0, 1, 5, 13, 16}

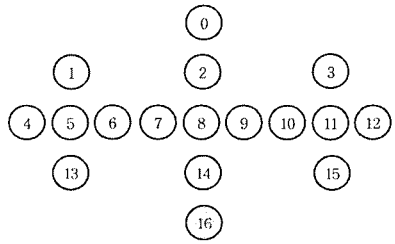


(b) WQ={0, 3, 10, 15, 16}

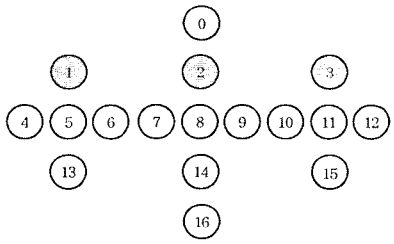
그림 4 쓰기 동작의 예



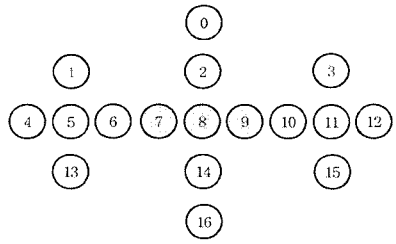
(a) RQ={0}



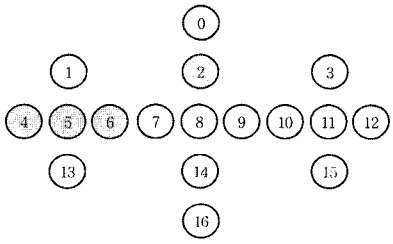
(b) RQ={16}



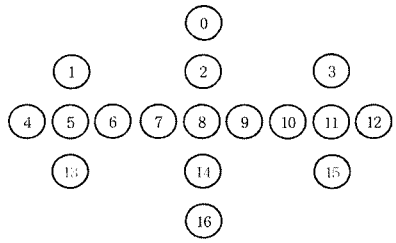
(c) RQ={1, 2, 3}



(d) RQ={1, 3, 7, 8, 9}



(e) RQ={2, 4, 5, 6, 15}



(f) RQ={13, 14, 15}

그림 5 다양한 장애 환경에 따른 읽기 동작의 예

그림 5는 제안된 프로토콜의 읽기 예를 보여준다. 만약 루트 노드가 읽기 가능하면, 오직 루트 노드만 읽기를 수행한다. 상위 트리의 루트 노드의 장애가 발생할 경우는 (b)에서와 같이 하위 트리의 루트 노드에 대하여 읽기 동작을 수행한다. 그러므로 노드의 장애 환경에서도 기존의 트리 프로토콜에 비해 낮은 평균 읽기 비용을 가지게 된다. 노드-0과 노드-16 모두 장애가 발생할 경우는 상위 트리에서부터 알고리즘에 따라 루트 노드의 자식 노드 전체에 대하여 (c)와 같이 읽기 동작을 수행하게 된다. 만약 (d)와 같이 자식 노드 중에 장애가 발생할 경우는, 장애가 발생한 노드의 자식 노드에 대하여 같은 동작을 수행하게 된다. h/2 레벨에서 장애가 발생할 경우는 하위 레벨에서 읽기 동작을 수행하게 된다. (e)의 경우와 같이, 만약 노드-10, 노드-11, 노드-12 중

에 임의의 하나의 노드에 장애가 발생할 경우에는 하위 레벨의 노드-15에 대하여 읽기 동작을 수행한다. 그러므로 노드의 장애가 계속 발생 할 경우에도 기존의 복제 프로토콜과 같이 읽기 비용이 계속 하여 증가하지 않으며, 안정된 성능을 보일 수 있다.

다음의 정의는 제안된 대칭 트리 프로토콜의 동작의 정확성을 보여준다.

정의 1. 읽기 동작은 항상 마지막에 쓰기 동작에 의해 쓰여진 최신 결과를 반환한다.

증명. 그림 4와 그림 5에서 보인 것과 같이, 임의의 RQ와 WQ를 정할 때 RQ∩WQ≠0이다. 그러므로, 제안된 프로토콜은 임의의 RQ는 항상 최신의 데이타를 갖는 노드를 포함하도록 보증하게 된다. □

정의 2. 읽기/쓰기 또는 쓰기/쓰기 동작은 동시에 수행될 수 없다.

증명. 데이타에 대한 일관성은 선택된 RQ와 WQ가 읽기/쓰기와 쓰기/쓰기 충돌을 감지할 수 있기 때문에 유지될 수 있다. 제안된 대칭 트리 프로토콜에서 읽기/쓰기 충돌은 읽기 동작은 하나의 레벨 전체에 대하여 수행하고 쓰기 동작은 최소한 각 레벨의 하나의 노드에 대하여 수행하기 때문에 읽기 동작을 수행하는 레벨에서 충돌을 검출할 수 있다. 쓰기/쓰기 충돌은 모든 쓰기 동작이 반드시 루트 노드에서부터 시작해야 하기 때문에 검출 가능하다. 그러므로 하나의 노드가 쓰기 동작을 수행하면 다른 노드는 루트 노드에 대한 접근이 불가능하게 되어 두개의 쓰기 동작이 동시에 수행될 수 없다. □

4. 시뮬레이션 및 성능 분석

이번 장에서는 제안된 대칭 트리 프로토콜의 성능을 평가하고, Tree quorum 프로토콜과의 성능을 비교하려 한다. 제안된 프로토콜의 읽기 및 쓰기 비용, 가용성을 평가하기 위해 수학적인 증명을 수행하고, 서비스 요청에 따른 응답 속도를 평가하기 위해 실제 시스템과 동일한 환경으로 프로그램을 이용해 구현하여 시뮬레이션을 수행한다. 정확한 성능평가를 수행하기 위해 다음과 같은 전제를 사용한다.

- 링크의 실패는 발생하지 않는다.
- 노드의 장애는 발생할 수 있다.
- 노드들의 실패는 독립적이며 실패율은 일정하다.
- 모든 노드는 확률 p의 가용성을 갖는다.
- Logarithmic 프로토콜의 잎 노드를 제외한 모든 노드와 대칭 트리 프로토콜의 상위 트리는 동일한 S 개의 자식 노드를 갖는다.

4.1 비용 분석

읽기/쓰기 비용은 전체 노드들 중에 읽기와 쓰기 동작을 수행하는 노드의 수를 의미한다. Tree quorum 프로토콜과 제안된 대칭 트리 프로토콜 모두 알고리즘에 따른 읽기 동작과 쓰기 동작은 모두 루트 노드를 처음으로 접근하게 된다. 이러한 알고리즘에 따라 수행하게 될 경우 이론적으로 1의 최소 읽기 비용을 가지지만, 이로 인해 루트 노드는 병목현상이 발생하게 된다. 이러한 루트 노드의 병목 현상을 해결하기 위해 알고리즘의 수정이 필요하다. 두 프로토콜의 읽기 동작은 임의의 레벨에서 시작하게 되고, 쓰기 동작은 원래의 알고리즘에 따라 루트 노드부터 동작하게 된다.

수정된 알고리즘의 읽기 동작을 위해 두 개의 변수 X와 f가 사용된다. X와 f는 0과 1사이의 값으로 f는 읽기 동작을 위해 선택되어진 상수이고, X는 균일 분포(uniform distribution)에 의해 생성된 난수이다. 첫 번째 난수 x가 생성되고, 만약 $x \le f$의 식을 만족하면 첫 번째 레벨의 노드에 대한 읽기 동작을 수행한다. 위의 조건을 만족하지 않으면 다음 레벨에 대하여 새로운 x를 생성하고 $x \le f$의 식에 의해 동일한 동작을 수행하게 된다.

4.1.1 Logarithmic 프로토콜

높이 h를 갖는 Tree quorum 프로토콜의 평균 읽기 비용은 식 (1)과 같이 계산되며, 최소 읽기 비용은 식 (2)와 같다. 최소 읽기 비용은 루트 노드가 선택되어 성공적으로 읽기 동작을 수행한 경우이다. 여기서 f는 각 레벨이 선택될 확률에 의해 결정된다.

Tree quorum 프로토콜에서 최적의 비용과 가용성을 갖는 프로토콜을 Logarithmic 프로토콜이라고 부르며, RQ=S 와 WQ=1의 값을 갖는다. 여기서 S는 부모노드가 갖는 자식 노드의 수를 의미한다. Logarithmic 프로토콜의 조건에 따른 평균 읽기 비용과 쓰기 비용은 식 (3)과 식 (4)와 같이 각각 나타난다.

$$\begin{aligned} C_{read} = {} & f + (1-f)f \cdot RQ + (1-f)^2 f \cdot RQ^2 + (1-f)^3 f \cdot RQ^3 \\ & + \cdots + (1-f)^{h-1} f \cdot RQ^{h-1} \\ & + (1 - (f + (1-f) \cdot f + (1-f)^2 \cdot f \\ & + \cdots + (1-f)^{h-1} \cdot f)) \cdot RQ^h \\ = {} & f \sum_{k=0}^{h-1} (1-f)^k \cdot RQ^k + \left(1 - f \sum_{k=0}^{h-1} (1-f)^k\right) \cdot RQ^h \qquad (1) \\ = {} & f \cdot h + 1 \qquad \text{if } (1-f) \cdot RQ = 1 \\ & f \frac{(1-f)^h RQ^h - 1}{(1-f)RQ - 1} + (1-f)^h \cdot RQ^h \qquad else \\ = {} & \min(C_{read}) = 1 \qquad (2) \end{aligned}$$

$$C_{read} = p \frac{((1-p)S)^{h-1}}{(1-p)S-1} + ((1-p)S)^h \qquad (3)$$

$$C_{write} = h + 1 \qquad (4)$$

4.1.2 대칭 트리 프로토콜

$$C_{read}=f+(1-f)f\cdot S+(1-f)^2f\cdot S^2+(1-f)^3f\cdot S^3+\cdots$$
$$+(1-f)^{\frac{h}{2}}f\cdot S^{\frac{h}{2}}+(1-f)^{\frac{h}{2}}\cdot(1-f^S)\cdot f\cdot S^{\frac{h}{2}-1}$$
$$+(1-f)^{\frac{h}{2}}\cdot(1-f^S)^2\cdot f\cdot S^{\frac{h}{2}-2}$$
$$+\cdots+(1-f)^{\frac{h}{2}}\cdot(1-f^S)^{\frac{h}{2}-1}\cdot f\cdot S$$
$$+(1-(f+(1-f)\cdot f+(1-f)^2\cdot f$$
$$+\cdots+(1-f)^{\frac{h}{2}}(1-f^S)^{\frac{h}{2}-1}\cdot f))$$
$$=f\left(\sum_{k=0}^{\frac{h}{2}}(1-f)^k\cdot S^k+\sum_{k=1}^{\frac{h}{2}-1}(1-f)^{\frac{h}{2}}\cdot(1-f^S)^k\cdot S^{\frac{h}{2}-k}\right)$$
$$+\left(1-\left(f\sum_{k=0}^{\frac{h}{2}}(1-f)^k+f\sum_{k=0}^{\frac{h}{2}-1}(1-f)^{\frac{h}{2}}(1-f^S)^k\right)\right) \quad (5)$$

제안된 대칭 트리 프로토콜의 평균 읽기 비용은 식(5)와 같이 계산되고, 최소 읽기 비용과 평균 쓰기 비용은 Tree quorum 프로토콜과 같다. 평균 읽기 비용을 계산하기 위해서는 그림 6에서 보는 것과 같이 각 레벨에서의 읽기 비용을 통하여 계산하여야 한다. 이전 장에서 말한 것과 같이 f는 각 레벨이 선택될 확률을 의미한다. 따라서 두 번째 레벨이 선택될 확률은 첫 번째 레벨이 선택되지 않을 확률과 각 레벨이 선택될 확률의 곱으로 다시 표현될 수 있다. 최종적으로 가장 하위 레벨은 전체 확률에서 나머지 상위 레벨들이 선택될 확률을 제거함으로써 구할 수 있게 된다. 평균 읽기 비용은 그림 6에서의 각 레벨에서의 읽기 비용을 모두 합한 값으로 표현된다.

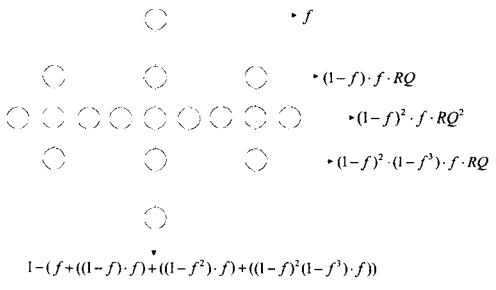


그림 6 각 레벨에서의 읽기 비용 분석

그림 7은 121개의 노드를 갖는 Logarithmic 프로토콜과 161개의 노드를 갖는 제안하는 대칭 트리 프로토콜의 평균 읽기 비용을 비교한 결과이다. 그림 7에서 보는 것과 같이 대칭 트리 프로토콜의 f값에 따른 읽기 비용은 최악의 경우와 최선의 경우 모두 최소 읽기 비용을 가지며, 평균 24개의 노드를 사용하는 Logarithmic 프로토콜에 비해 평균 7개의 노드를 사용하여 읽기 동작을 수행할 수 있으므로, 보다 안정된 읽기 성능을 가진다. 또한 그림 7에서 주목할 점은 대칭 트리 프로토콜이 Logarithmic 프로토콜에 비해 더 많은 노드로 구성되지만 읽기 비용은 더욱 낮게 나타난다는 것이다. 그러므로 같은 수의 노드로 구성할 경우 읽기 비용은 더욱 좋아지게 되며, f의 값이 적어질수록 제안된 대칭 트리 프로토콜의 읽기 비용은 Logarithmic 프로토콜의 읽기 비용과 더욱 큰 차이를 보이게 된다.

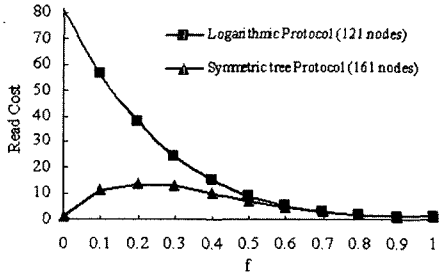


그림 7 평균 읽기 비용 비교

평균 쓰기 비용은 전체 레벨에 대하여 각 레벨에서 하나의 노드에 쓰기 동작을 수행하므로 노드를 구성하는 레벨에 비례하여 증가하게 된다.

4.2 가용성 분석

4.2.1 Logarithmic 프로토콜

레벨 $l>0$의 트리로 구성된 Tree quorum 프로토콜의 읽기 동작을 위한 가용성은 식 (6)과 같이 나타낼 수 있다. 식 (6)에서 p는 각 노드의 성공 확률이고, $\wp_{read}^{(0)}=p$이다. 식 (6)에서 보는 것과 같이 Tree quorum 프로토콜의 읽기 가용성은 이전 레벨의 읽기 가용성에 의존하게 되고, 재귀적인 식에 의하여 나타나게 된다.

$$\wp_{read}^{(l)}=p+(1-p)\sum_{k=RQ}^{S}\binom{S}{k}\left(\wp_{read}^{(l-1)}\right)^k\left(1-\wp_{read}^{(l-1)}\right)^{S-k} \quad (6)$$

RQ=S의 값을 갖는 Logarithmic 프로토콜의 읽기 가용성은 식 (6)으로부터 식 (7)로 정리할 수 있다.

$$\wp_{read}^{(l)}=p+(1-p)\left(\wp_{read}^{(l-1)}\right)^S \quad (7)$$

레벨 $l>0$의 트리로 구성된 Tree quorum 프로토콜의 쓰기 동작을 위한 가용성은 식 (8)과 같이 나타낼 수 있고, Logarithmic 프로토콜의 쓰기 가용성은 식 (8)로부터 식 (9)로 정리된다.

$$\wp_{read}^{(l)}=p\sum_{k=RQ}^{S}\binom{S}{k}\left(\wp_{read}^{(l-1)}\right)^k\left(1-\wp_{read}^{(l-1)}\right)^{S-k} \quad (8)$$

$$\wp_{read}^{(l)}=p\left(1-\left(1-\wp_{read}^{(l-1)}\right)^S\right) \quad (9)$$

4.2.2 대칭 트리 프로토콜

레벨 $\frac{h}{2}\le l<0$로 구성된 대칭 트리 프로토콜의 읽기

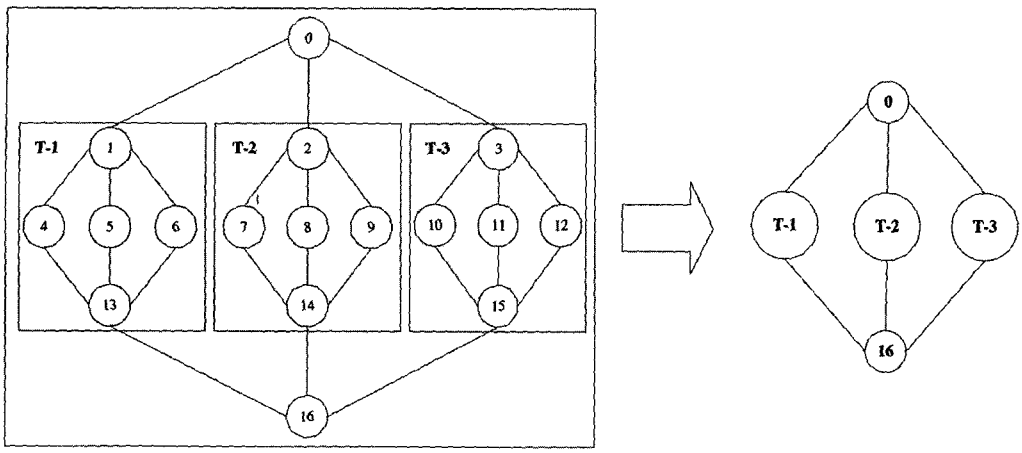


그림 8 가용성 성능 평가 모델

동작을 위한 가용성은 식 (10)과 같이 나타낼 수 있고, 쓰기 가용성은 식 (12)와 같다. 읽기 및 쓰기 동작을 위한 가용성은 아래 식에서 보는 것과 같이 순환적으로 구할 수 있다. 그림 8에서 보면 대칭 트리 구조는 기본적으로 가장 적은 크기의 3-레벨 대칭 트리들로 구성된다. 따라서 5-레벨 대칭 트리는 그림 8에서와 같이 순환적으로 3-레벨 대칭 트리로 구성될 수 있다. 가장 기본적인 3-레벨 대칭 트리의 읽기 가용성은 식 (10)과 같이 나타낼 수 있으며, 레벨의 증가에 따른 일반식은 식 (11)과 같이 구할 수 있다.

$$8^{3-lev}_{read} = p + (1-p)\cdot(p^3 + (1-p^3)\cdot p) \quad (10)$$

$$8^{(l)}_{read} = p + (1-p)\left(\left(8^{(l-1)}_{read}\right)^S + \left(\left(1-8^{(l-1)^S}_{read}\right)p\right)\right) \quad (11)$$

$$8^{(l)}_{write} = p^2 \sum_{k=1}^{S} \binom{S}{k} \left(8^{(l-1)}_{write}\right)^k \left(1-8^{(l-1)}_{write}\right)^{S-k} \quad (12)$$

그림 9는 Logarithmic 프로토콜과 대칭 트리 프로토콜의 읽기 가용성을 비교한 결과이다. 비교를 위해 17개의 노드를 갖는 5레벨 구조, 53개의 노드를 갖는 7레벨 구조, 그리고 161개의 노드를 갖는 9레벨 구조의 다른 경우에 대한 대칭 트리 프로토콜에 대하여 성능을 분석한다. 그림 9에서 보는 것과 같이, 읽기 가용성은 대칭 트리의 레벨이 증가할수록 높은 가용성을 가지며, 제안된 대칭 트리 프로토콜은 더 적은 노드의 구성에도 불구하고 Logarithmic 프로토콜에 비해 높은 가용성을 가진다.

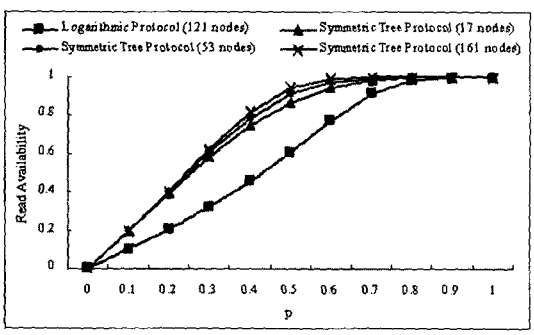


그림 9 읽기 가용성 비교

4.3 응답 시간 분석

응답 시간을 비교하기 위한 시뮬레이션은 C 프로그래밍을 이용하여 모듈 프로그래밍으로 구현하였다. 시뮬레이션 환경 및 변수는 표 1에 나타난 것과 같다. 응답시간은 임의의 노드에서 동작이 요청된 시간으로부터 동작이 끝나는 시간으로 계산된다. 정확한 시뮬레이션 결과를 위해 다음과 같은 가정 아래에서 시뮬레이션이 수행된다.

표 1 시뮬레이션 환경

구분	사양
시스템	펜티엄-III 800MHz , 256MB
컴파일러	비쥬얼-C++ 6.0
변수	락 요청 통신 시간 : 0.0001초 락 시간 : 0.0002초 동작 요청 및 회답 시간 : 0.0003초 동작 시간 : 0.00005초 전체 요청 횟수 : 1,000,000회
시뮬레이션 횟수	100,000,000회

가정 :

- 노드들은 이더넷에 의하여 연결되고, 노드 간에 통신은 브로드캐스트에 의하여 이루어진다.
- 이더넷 회선의 특성상 두 개의 노드가 동시에 회선을 공유할 수 없기 때문에, 하나의 노드가 회선을 사용중이면 다른 노드는 회선을 사용할 수 없다.
- 노드의 동작 요청 간격은 지수 분포에 따른다.

- 모든 노드는 전체 노드의 수와 동일한 크기의 버퍼를 가지며, 이것은 모든 다른 노드들로부터 요청을 받을 수 있음을 의미한다.
- 모든 노드는 FIFO 스케줄링 순서에 의하여 다른 노드들로부터의 요청에 응답한다.
- 노드는 동작을 수행하기 전에 동작을 수행하는 모든 노드들에 락을 이용하여 동시성 제어를 수행한다.
- 시뮬레이션 시작과 동시에 노드의 장애 여부를 결정하게 되며, 시뮬레이션이 끝날 때까지 장애가 발생한 노드는 동작하지 않는다.

그림 10은 제안된 대칭 트리 프로토콜의 응답시간을 구하기 위한 시뮬레이션 흐름도이다. 시뮬레이션은 세 부분으로 나누어진다. min_site는 전체 노드 중에 가장 작은 동작 시간을 가지고 있는 노드를 찾는 동작을 수행한다. 동작 시간은 각 노드가 동작을 시작하는 시간을 의미한다. read_op와 write_op는 실제 동작을 수행하기 위해 구성된 부분이다. 마지막으로 anaysis_func는 노드의 타임 스탬프를 관리하고, 시뮬레이션 동안 응답시간과 처리율을 계산한다. 각 노드의 동작 시작 시간은 지수 분포 함수에 의하여 결정되고, 시뮬레이션의 초기화 과정에서 전체 노드에 할당되며, 동작을 수행하고 끝날 때마다 현재 시간의 이후 시간으로 다시 할당된다. 각 노드의 동작 유형(읽기와 쓰기)은 균일 분포 함수에 의하여 결정된다.

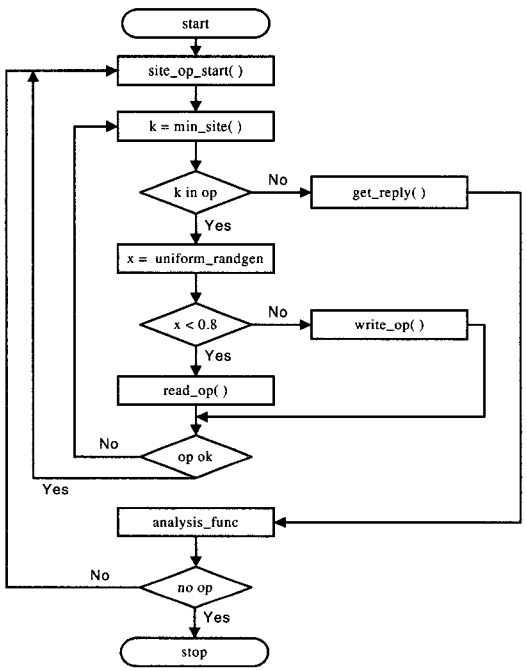


그림 10 대칭 트리 프로토콜의 시뮬레이션을 위한 흐름도

시뮬레이션을 위한 읽기와 쓰기 동작을 위한 의사코드는 그림 11과 같다. 복제된 데이타의 일관성 유지를 위해, 락킹 동작은 읽기와 쓰기 동작이 수행되기 전에 반드시 수행되어야 한다. RQ와 WQ에 포함된 노드들은 avail_site 함수에 의하여 락킹 동작을 수행하게 된다. 시뮬레이션 결과는 노드의 장애 발생과 동작 시작 시간의 임의성으로 인하여 100,000,000번의 시뮬레이션을 수행하여 평균치로 계산된다.

그림 12부터 그림 14는 두 프로토콜의 읽기 및 쓰기에 대한 응답시간을 나타낸다. 그림에서 보는 것과 같이 제안된 대칭 트리 프로토콜의 경우는 161개의 노드를

```
read(min_node)
  lev = uniform_randgen( );
  WHILE(lev < last_level)
    IF( avail_site())
/* all sites of level(lev) available */
       get reply times for quorum;
       RETURN(reply time);
    ELSE
       lev = lev + 1;
    ENDIF
  ENDWHILE
  /* get next time for next operation*/
RETURN(exponential_randgen( ))
```

```
write()
  l=0;
  WHILE(l>no_of_lev)
    IF(avail_site())
    /* at least one site available */
     l=l+1;
    ELSE
  /* get next time for next operation*/
     RETURN(not OK)
    ENDIF
  ENDWHILE
  get reply times for quorum;
RETURN(reply time);
```

```
avail_site(i)
  IF(site i is available)
    RETUN(OK)
  ELSE
    IF(site(i) is not last level)
      k=0;
      FOR(all sites j of level)
        A[k] = avail_site(j)
        k=k+1;
      ENDFOR
    ELSE
      RETURN(not OK);
    ENDIF
  ENDIF
  IF(all sites are OK)
    RETUN(OK)
  ELSE
  RETURN(not OK)
  ENDIF
```

그림 11 읽기와 쓰기 동작을 위한 알고리즘

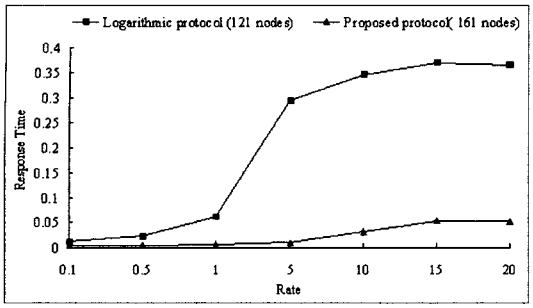


그림 12 읽기 동작에 대한 응답시간 비교

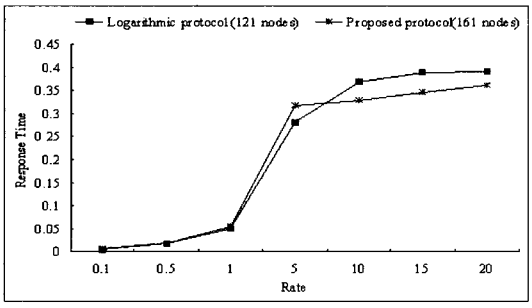


그림 13 쓰기 동작에 대한 응답시간 비교

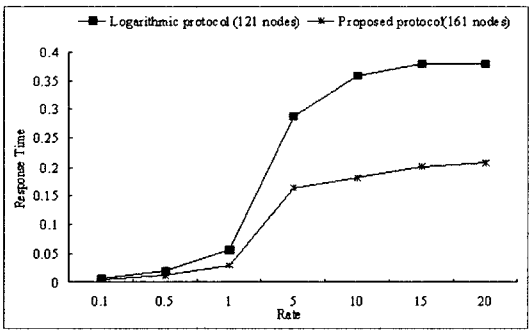


그림 14 전체 응답시간 비교

갖는 9-레벨 구조에 대하여 시뮬레이션을 수행하였다. 121개의 노드를 갖는 Logarithmic 프로토콜과 161개의 노드를 갖는 대칭 트리 프로토콜의 응답시간의 비교 결과이다. Logarithmic 프로토콜에 비해 제안된 대칭 트리 프로토콜이 월등하게 적은 응답시간을 가진다. 이러한 결과는 제안된 대칭 트리 프로토콜이 Logarithmic 프로토콜에 비해 읽기 비용이 적기 때문이다. 그림에서 보는 것과 같이 두 프로토콜의 응답시간 차이는 노드간 요청시간 간격이 짧을 경우 더욱 명확하게 나타난다. 요청 간격이 짧아질수록 단위 시간동안 요청량이 늘어나기 때문에 두 프로토콜간의 차이는 더욱 커지게 된다.

5. 결 론

본 논문은 기존의 Logarithmic 프로토콜에 비해 읽기 비용, 가용성, 그리고 응답시간 면에서 우수한 성능을 보이는 대칭 트리 프로토콜을 제안한다. 제안된 대칭 트리 프로토콜의 가장 큰 장점중의 하나는 노드의 실패 확률이 증가하더라도 이전의 프로토콜에 비해 훨씬 적은 읽기 비용과 응답시간을 가지는 것이다. 복제 프로토콜의 선택은 환경에 따라 필요한 주요 성능 요소에 의존하여 사용하여야 한다. 그러므로 제안된 대칭 트리 프로토콜은 노드의 장애 확률이 높은 환경에서 읽기 동작을 주로 수행하는 작업에 적합하다. 또한 제안된 프로토콜은 최근 데이타의 가용성을 높이기 위해 연구 중인 서바이벌(survival) 저장 시스템에 효율적으로 적용 가능하다.

향후 과제는 실제 시스템 상에서 제안된 프로토콜의 구현을 통해 보다 시스템 성능에 접근한 성능분석을 수행하고 보다 향상된 복제 프로토콜의 제안을 수행하려 한다.

참 고 문 헌

[1] C. Amza,, A.L. Cox, W. Zwaenepoel, Data replication strategies for fault tolerance and availability on commodity clusters, Proc. Intl Conf on Dependable Systems and Networks (DSN), 2000, 459-467.

[2] H. Y. Youn, B. Krishnamsetty, D. Lee, B. K. Lee, J. S. Choi, H. G. Kim, C. W. Park, and H. S. Lee, An Efficient Hybrid Replication Protocol for Highly Available Distributed System, Proc. Intl Conf on Communication and Computer Networks (CCN), Nov, 2002.

[3] K. Arai, K. Tanaka, M. Takizawa, Group protocol for quorum-based replication, Proc. Seventh Intl Conf on Parallel and Distributed Systems, 2000, 57-64.

[4] G. Alonso, Partial Database Replication and Group Communication Primitives, Proc. of the 2nd European Research Seminar on Advances in Distributed Systems (ERSADS97), March 1997, 171-176.

[5] P.A. Bernstein and N. Goodman, An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases, ACM Trans on Distributed Systems, 9(4), 1984, 596-615.

[6] R.H. Thomas. A Majority Consensus Approach to Concurrency Control for Multiple Copy Databased, ACM Trans on Database Systems, 4(2),1979,180-207.

[7] D. Davcev, A Dynamic Voting Scheme in Distributed Systems. IEEE Trans on Software

Engineering, 15(1), 1989, 93-97.
[8] D. Saha, S. Rangarajan, S.K. Tripathi, An Analysis of the Average Message Overhead in Replica Control Protocols, IEEE Trans on Parallel and Distributed Systems, 7(10), Oct. 1996, 1026-1034.
[9] B. Freisleben, H.H. Koch, and O. Theel, Designing Multi-Level Quorum Schemes for Highly Replicated Data. Proc. of the 1991 Pacific Rim Intl Symp on Fault Tolerant Systems, IEEE, 1991, 154-159.
[10] D.K. Gifford, Weighted Voting for Replicated Data, Proc. of the 7th ACM Symp on Operating Systems Principles, 1979, 150-162.
[11] Lamehamedi, H., Zujun Shentu, Szymanski, B., and Deelman, E., Simulation of dynamic data replication strategies in Data Grids, Proc of the Intl Symp on Parallel and Distributed Processing, 2003, 22-26.
[12] H. Lamehamedi, B. K. Szymanski, Z.Shentu, and E. Deelman, Data Replication Strategies in Grid Environments, Proc of ICAP'03, 2002, 378-383.
[13] D. Agrawal and A. El Abbadi, The tree Quorum protocol: An Efficient Approach for Managing Replicated Data, Proc of the 16th Very Large Databases (VLDB) Conf, 1990, 243-254.
[14] S. Cheung, M. Ammar, and M. Ahamad, The Grid Protocol: A High Performance Scheme for Maintaining Replicated Data, Proc of the 6th Intl Conf on Data Engineering, 1990, 438-445.

최 성 춘

2001년 남서울대학교 정보통신공학과 학사. 2003년 성균관대학교 전기전자컴퓨터공학부 석사. 2004년~현재 성균관대학교 정보통신공학부 컴퓨터공학 박사과정. 관심분야는 분산컴퓨팅 및 네트워크, 모바일 컴퓨팅, 유비쿼터스컴퓨팅

윤 희 용

1977년 서울대 전기공학과 학사. 1979년 서울대 전기공학과 석사. 1988년 Univ. of Massachusetts, 컴퓨터공학과 박사 1988년~1991년 Univ. of North Texas, 조교수. 1991년~1999년 Univ. of Texas at Arlington, 부교수. 1999년~2000년 정보통신대학교 교수. 2000년~현재 성균관대학교 정보통신공학부 교수. 관심분야는 모바일 컴퓨팅, 분산처리, 시스템 소프트웨어

이 강 신

1987년 한양대학교 수학과(석사). 1989년 한양대학교 수리통계학과(석사). 1990년~1992년 (주)데이콤종합연구소 연구원. 1992년~2000년 한국전산원 부장. 2000년~현재 한국정보보호진흥원 기반보호기술팀 팀장. 관심분야는 정보보안, 침입감내, 무선보안 등

이 호 재

2002년 성균관대학교 석사. 2002년~2004년 한국정보보호진흥원 연구원. 2004년~현재 LG전자 전자기술원 주임연구원. 관심분야는 네트워크 보안, 침입감내 기술