

이동 컴퓨팅 환경에서 트리 높이의 균형을 유지하는 상호 배제 알고리즘

(Distributed Mutual Exclusion Algorithm for Maintaining Tree-Height Balance in Mobile Computing Environments)

김형식[†] 엄영익^{**}

(Hyoung Shick Kim) (Young Ik Eom)

요약 이동 호스트란 시간과 공간의 제약 없이 필요한 기능을 수행하고, 또한 원하는 정보에 접근할 수 있는 휴대용 컴퓨터를 일컫는다. 이동 컴퓨팅 환경이란 이러한 이동 호스트들을 지원할 수 있도록 구성된 분산 환경이다. 따라서 이동 컴퓨팅 환경에서의 분산 알고리즘은 정적 호스트만을 고려한 기존의 분산 환경에서와는 다르게 이동 호스트의 여러 가지 특성들을 고려하여 제안되어야 한다. 즉, 이동 컴퓨팅 환경의 도래로 인하여 이동성과 휴대성, 그리고 무선 통신과 같은 특성들을 고려한 새로운 분산 상호 배제 알고리즘이 필요하다. 이제까지 이동 컴퓨팅 환경에서의 상호 배제 알고리즘은 토큰 링 구조에 기반을 두고 설계되었다. 토큰 링 구조는 이동 호스트들의 위치를 유지하기 위하여 높은 비용을 필요로 하는 단점을 가지고 있다. 본 논문에서는 균형 높이 트리(height-balanced tree)라는 새로운 모형을 제안함으로써 정적 분산 환경과 이동 분산 환경이 혼합된 환경에서 상호 배제 비용을 감소시킬 수 있는 새로운 알고리즘을 제안하며, 각 경우에 있어서의 비용을 산출하고 평가한다.

Abstract The mobile host is a portable computer that carries out necessary functions and has the ability to access desirable informations without any constraints in time and space. Mobile computing environment is a distributed environment that is organized to support such mobile hosts. In that environment, distributed algorithms of which environment not only with static hosts but with mobile host's several properties should be proposed. With the emergence of mobile computing environments, a new distributed mutual exclusion method should be required to consider properties mobile computing system such as mobility, portability, and wireless communication. Until now, distributed mutual exclusion methods for mobile computing environments are designed based on a token ring structure, which have the drawbacks of requiring high costs in order to locate mobile hosts. In this paper, we propose a distributed mutual exclusion method that can reduce such costs by structuring the entire system as a height-balanced tree for static distributed networks and for networks with mobile hosts. We evaluated the operation costs in each case.

1. 서론

상호 배제 알고리즘은 공통 메모리와 공통 클럭을 기반으로 동작하는 단일 시스템에서의 알고리즘과, 그러한 기반을 갖지 못한 분산 시스템 환경에서의 알고리즘으

로 크게 두 가지로 구분된다. 즉, 분산 시스템 환경에서는 단일 시스템의 상호 배제 알고리즘과는 다른 새로운 상호 배제 알고리즘이 필요한 것이다. 한편 이동 컴퓨팅 환경은 이러한 분산 상호 배제 문제를 더욱 더 어렵게 한다. 이동형 컴퓨팅 환경에 존재하는 이동 호스트는 시간과 공간에 대한 제약 없이 원하는 정보를 어디서나 얻을 수 있으며, 또한 네트워크와 접속한 채로 움직일 수 있다는 장점을 가지고 있지만, 한편으로는 이러한 이동성이 기존의 분산 상호 배제 알고리즘에 치명적인 문제점을 초래하게 된다. 따라서 현재까지의 분산 상호 배

[†] 비회원 : 한국과학기술원 전산학과
morethan@adam.kaist.ac.kr

^{**} 종신회원 : 성균관대학교 전기전자및컴퓨터공학부 교수
yicom@simsan.skku.ac.kr

논문접수 : 1999년 2월 2일
심사완료 : 1999년 8월 18일

제 알고리즘 설계를 위하여 제안되었던 토큰 링 모형과 신장 트리(spanning tree) 모형 등은 호스트들의 빈번한 이동과 전력의 제한 요소까지 고려하여 새로운 분산 상호 배제 알고리즘으로 확장되어야 할 것이다.

본 논문에서는 크게 두 가지 성능에 초점을 맞추어 분산 상호 배제 알고리즘을 설계한다. 본 논문에서는 메시지 지연 시간을 줄이기 위하여 네트워크의 계층화 개념을 도입하고 호스트들간의 기근 상태를 회피할 수 있는 알고리즘을 제안한다. 시스템은 모형의 구조에 따라 성능의 차이를 보이게 되는데, 본 논문에서는 시스템의 모형을 이상적인 경우와 그렇지 못한 경우로 각각 구분하여 제안 알고리즘의 성능을 산출하였다. 전체 노드의 수를 N 이라 하고 각 노드에 대한 결합가(degree)를 k 라고 하였을 때, 시스템이 이상적인 모형을 갖는 경우에는 $O(N \log_k N)$ 의 비용을 보인다는 것을 유도할 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 분산 상호 배제 알고리즘과 관련된 기존의 연구를 살펴보고, 3장에서는 시스템 모형을 설명한다. 4장에서는 그 시스템을 세부적으로 설계하고 새로운 상호 배제 알고리즘을 제시한다. 5장에서는 그 알고리즘들에 대한 운영비용을 평가하고, 마지막으로 6장에서는 결론과 앞으로의 연구 과제에 대하여 언급한다.

2. 관련 연구

2.1 기존의 상호 배제 알고리즘

상호 배제 문제 배제 알고리즘은 임계 지역 실행 조건에 따라 명제 기반(assertion-based) 알고리즘과 토큰 기반(token-based) 알고리즘으로 구분된다. 명제 기반 알고리즘은 임계 지역을 수행하고자 할 때 임계 지역에 관한 명제를 검사하여 참일 때만 임계 지역을 수행할 수 있는 권한을 갖게 되는 알고리즘이고, 토큰 기반 알고리즘은 임계 지역 수행 권한을 의미하는 토큰이라는 메시지가 존재하여 네트워크 전체에서 토큰을 소유한 노드만이 임계 지역 진입을 수행할 수 있는 권한을 갖게 되는 알고리즘이다. 전자의 알고리즘은 일반적으로 단일 시스템에서 사용되며, 반면에 후자의 알고리즘은 분산 시스템에서 주로 사용된다.

네트워크 전체의 노드 수를 N 이라 가정하였을 때, 임의의 노드로부터 임계 지역 진입이 요청되었을 경우, Ricart and Agrawala가 제안한 알고리즘[1]은 임계 지역 진입을 위하여 $2*(N - 1)$ 개의 메시지를 교환하며, 1985년에는 Suzuki와 Kasami가 N 개의 메시지를 보장하는 알고리즘[2]을 제안하였다. 또 같은 해, Maekawa에 의하여 $O(\sqrt{N})$ 의 메시지 수를 보장하는 알고리즘이

제안된 바 있다[3]

그러한 상호 배제 알고리즘들을 내용별로 분류하면 시간 기반 사건 순서화 방법, 토큰 전달 방법, 그리고 우선 순위 기반 사건 순서화 방법으로 더욱 세분화 될 수 있다[4]. 시간 기반 사건 순서화 방법에서는 요청 순서에 따라 임계 지역에 들어가는 방식으로 타임스탬프(timestamp)를 사용하는 특징을 갖고 있다. 토큰 전달 방법은 논리적인 링 구조에 의해 정의된 순서에 따라 프로세스들 간을 순환하는 토큰이라는 특별한 형식의 메시지를 제공한다 그리고 우선 순위 기반 사건 순서화 방법에 기초한 알고리즘들은 사건이 발생한 시간이 아니라 프로세스들의 우선 순위를 기준으로 사건들을 배열한다.

2.2 이동 컴퓨팅 환경에서의 상호 배제 알고리즘

이동 컴퓨팅 환경에서의 연구가 이루어져 왔던 부분은 위에서 분류한 연구 기법 중에서 토큰 전달 방법에 기반한 상호 배제 방법이 유일하다. Badrinath와 Imielinski[5]가 고안한 이 기법은 크게 세 가지 전략으로 나눌 수 있다.

Search 전략은 고정 호스트가 토큰을 인계 받았을 때, 자신에게 임계 지역 진입을 요청했던 이동 호스트의 위치를 직접 찾아내는 방법이다. Inform 전략은 임계 지역 진입을 요청한 이동 호스트가 이동할 때는 항상 요청했던 고정 호스트에게 이동 정보를 알리는 방법으로 이동 호스트들의 이동 빈도가 낮은 경우에는 Search 전략보다 성능이 우수하지만, 이동 빈도가 높은 경우에는 성능이 급격히 떨어지는 단점이 있다. 마지막으로 Proxy 전략은 Search 전략과 Inform 전략을 결합한 방법으로 네트워크 전체를 몇 개의 고정 호스트들로 구성된 영역으로 구분하고, 각 영역마다 임의의 단일 고정 호스트가 그 영역을 대표하여 토큰 링 구조에 참여하게 됨으로써 그 호스트가 자신이 속한 영역의 대리인(proxy)이 되는 것을 의미한다. 이 전략은 영역내의 고정 호스트들에 속한 이동 호스트들이 자신이 속한 영역 내에서의 이동은 빈번하나 영역간의 이동 빈도는 현저히 적은 경우에 효율적인 방법이다.

하지만 위의 세 가지 전략들은 임계 지역의 수행을 위한 메시지 전송 이외에 이동 호스트의 위치 이동이 있을 때 그 호스트의 위치 정보를 찾거나 알리기 위한 메시지 전송을 추가로 해야 하는 오버헤드를 갖고 있다

2.3 트리 구조에 기반한 상호 배제

Raymond가 제시한 트리 구조 기반의 상호 배제 기법은 정적인 호스트만으로 이루어진 분산 환경을 위한 알고리즘으로서 각 호스트에서 발생 가능한 상황과 그

에 따른 동작 알고리즘을 제시하였다[6]. Raymond는 임계 지역을 수행하기 위해 발생하는 상황을 네 가지 경우로 나누었다. 첫 번째는 임의의 노드가 임계 지역에 들어가기를 원하는 경우, 두 번째는 이웃 노드로부터 임계 지역 수행 요청 메시지를 받았을 경우, 세 번째는 임의의 노드가 특권 메시지를 받았을 경우, 그리고 네 번째는 노드가 임계 지역에서 빠져 나오는 경우이다.

시스템의 동작을 살펴보면 다음과 같다. 우선, 네트워크 구조를 논리적인 신장 트리(spanning tree) 구조로 만든 후, 임의의 노드를 특권 노드로 임명하여 특권 메시지를 소유하게 하고 그 사실을 네트워크의 모든 노드에 전파한다. 각 노드들은 그 특권 메시지를 소유하여야만이 임계 지역을 수행할 수 있으므로 특권 메시지가 바로 토큰이 된다. 그리고 각 노드들은 자신과 자신의 이웃 노드의 정보만을 유지하고 있게 된다. 임계 지역을 수행하고자 하는 노드는 토큰을 소유한 노드로 갈 수 있는 경로상의 이웃 노드에게 요청 메시지를 송신하게 되고, 이러한 송신이 계속 전파되어 그 요청 메시지는 특권 노드에게까지 전달되는 것이다. 각 노드에서는 자신이 이미 요청 메시지를 송신한 적이 있다면 또 다른 요청을 수신한 경우 그 요청 메시지를 송신하지 않고 자신의 대기 큐에 넣기만 하는데, 이는 중복되는 메시지 전송 횟수를 줄이기 위한 방법이다. 특권 노드가 임계 지역을 수행하고 있지 않거나 임계 지역 수행을 끝마쳤을 때는 자신의 큐에 대기하고 있는 노드에게 특권을 인계하게 된다. 토큰을 전달받은 노드는 자신의 대기 큐에 존재하는 상위 목록이 자신이면 임계 지역을 수행하게 되고 자신이 아니면 다시 그 특권을 인계하게 된다. 그리하여 임계 지역 수행을 요청한 순서대로 토큰을 전달받게 되는 것이다.

또한, 신장 트리 모형을 기반으로 이동 호스트가 토큰을 소유하지 않고 자기 지역을 이동하는 경우와 토큰을 소유하고 자기 지역을 이동하는 경우를 함께 고려함으로써, 이동 컴퓨팅 환경에 알맞은 분산 상호 배제 알고리즘이 제안되어 있다[7,8,9]. 이 기법에서는 토큰을 소유한 노드가 이러한 신장 트리 모형의 어느 한 끝에 편향되어 위치한 경우, 메시지 전송 횟수는 네트워크의 가장 긴 경로인 지름의 두 배가 되며, 이에 따라 임계 지역 진입에 따른 비용이 토큰을 소유한 노드가 트리의 중앙에 위치한 경우와 비교하였을 때, 대략적으로 두 배의 차를 보이게 된다. 또한, 토큰을 소유한 노드와 임계 지역 진입 요청을 한 노드와의 거리에 따라 진입 과정에 대한 거리가 노드들간에 서로 차이를 보임으로써, 토큰을 소유한 노드의 반대편에 위치한 노드들에 있어서

는 기존 상태를 발생시킬 확률이 높아지게 된다.

본 논문에서는 이러한 관련 연구를 바탕으로 이동 컴퓨팅 환경에 알맞은 새로운 분산 상호 배제 알고리즘을 제안한다. 토큰을 소유한 노드가 편향되어 위치하는 경우를 방지하기 위하여, 토큰을 소유한 노드의 인접 노드를 항상 일정한 깊이로 유지시켜 준다. 즉, 트리의 균형을 유지함으로써 기존의 기법에 비하여 메시지 전송 횟수를 줄이고, 가능한한 모든 노드들에게 임계지역 진입과 관련한 공정성이 보장되도록 하였다.

3. 시스템 모형

3.1 네트워크 구조

그림 1에서는 이동 컴퓨팅 환경의 시스템 구조를 보여주고 있다. 이동 컴퓨팅 환경의 호스트들은 고정 호스트와 이동 호스트로 구분된다. 이동 호스트는 물리적인 위치가 변할 수 있는 반면, 고정 호스트는 그 위치가 고정되어 있다. 무선 패체를 이용하여 이동 호스트와 직접적으로 통신할 수 있는 호스트를 이동 지원국(MSS : Mobile Support Station)이라 한다. 그리고 각 이동 지원국은 논리적 지리적으로 자신이 관리하고 있는 영역을 갖고 있는데 이를 셀(cell)이라고 한다. 임의의 이동 지원국의 셀안에 있어서 그 이동 지원국과 통신을 할 수 있는 이동 호스트를 그 이동 지원국의 지역 호스트(local host)라 한다. 이동 호스트는 논리적으로 한 순간에 한 셀에만 속하게 되어 오직 자신의 지역 이동 지원국(local MSS)과 메시지 전송을 하게 된다[10].

본 논문에서는 메시지 전송은 통신 네트워크 상에서 보장되며, 메시지 도착의 시간과 순서는 예측할 수 없고, 각 노드는 신뢰할 수 있다고 가정한다.

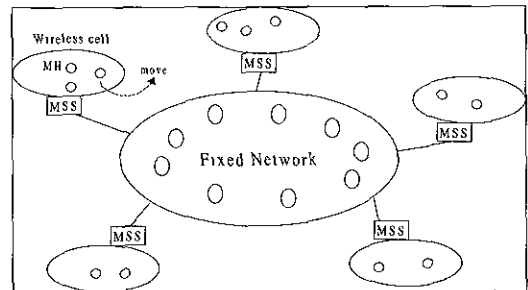


그림 1 시스템 모형의 예

논리적인 신장 트리 네트워크 구조를 구성할 때, 최초 참여 노드는 고정 호스트이다. 고정 호스트만으로 이루어진 구조가 완전한 신장 트리 구조라면, 그 때 각 이

동 지원국은 자신의 지역 내에 있는 모든 이동 호스트를 네트워크 구조에 참여시킴으로써 정적 호스트들과 이동 호스트들로 이루어진 완전한 신장 트리 네트워크 구조를 구성할 수 있다(그림 2).

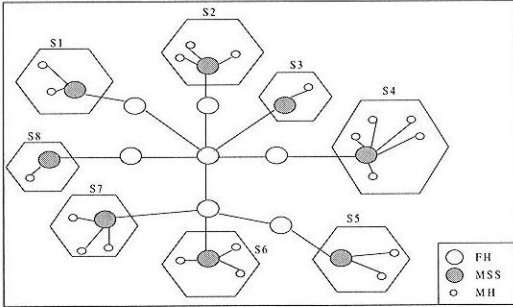


그림 2 이동 컴퓨팅 환경의 신장 트리 구조

3.2 용어

기존 기법이 노드를 임계 지역 진입 가능 노드와 임계 지역 진입 불가 노드로 구분한 것과는 달리, 본 제안 알고리즘에서는 이러한 두 종류의 노드 외에 임계 지역 진입 가능 노드에 직접적으로 인접한 노드에게까지 특별한 의미를 부여하여 노드의 종류를 모두 세 계층으로 구분한다(그림 3). 그림 4의 빗금친 노드가 새로이 지정된 노드이다. 본 논문에서는 앞으로 토큰 메시지를 소유한 노드, 즉 임계 지역 진입 가능 노드를 특권 노드(privileged node)로, 이 특권 노드와 직접적으로 인접한 노드를 후보 노드(candidate node)라 한다.

상호 배제를 보장하기 위하여 임계 지역 진입 가능 노드는 유일하며 그 노드는 네트워크를 신장 트리로 구성할 경우 루트가 된다. 또, 노드들은 서로 양방향으로 연결되어 있으며 OWNER라는 변수를 통해서 임계 지역 진입 노드를 가리킨다. 즉, 후보 노드들의 OWNER 값은 임계 지역 진입 가능 노드가 된다[8,9,10].

3.2.1 자료 구조

◆ OWNER

이 변수에는 self 또는 이웃 노드의 이름이 저장된다. 값이 self인 경우는 임계 지역 진입이 가능한 경우이다.

◆ USING

부울 변수이며 자신이 임계 지역에서 실행 중인 경우에 참이 된다.

◆ REQUEST_Q

해당 노드에게 REQUEST를 요청한 노드들의 이름을 저장한다. 알고리즘의 특성상 자신과 인접 노드의 요청만이 큐에 적체될 수 있다. 그러므로 REQUEST_Q의

최대 크기는 "이웃 노드의 수 + 1"이 된다.

◆ ASKED

부울 변수이며 자신의 OWNER에게 REQUEST 메시지를 보냈을 경우 TRUE가 된다. 이는 REQUEST가 중복 전달되지 않도록 보장하기 위한 것이다.

◆ LOCAL

부울 변수이며, 이 값이 TRUE인 노드가 후보 노드임을 의미한다.

◆ WAIT

부울 변수이며 이동 호스트의 이동시 상호 배제를 보장하기 위하여 사용한다.

◆ NEIGHBORS

인접 노드들의 집합으로 이동 호스트의 이동시 값이 변화한다. NEIGHBORS에 기록된 노드들 중 하나가 OWNER가 된다.

3.2.2 메시지 종류

인접한 노드들은 상호 배제 보장을 위하여 서로 일련의 메시지들을 교환하며 이에 따라 각 노드의 상태 변수 값이 설정된다.

◆ INITIALIZE

트리를 초기화하는 메시지이다. 네트워크의 논리적인 구조가 구성되고, 특권 노드가 선정되면, 그 노드는 자신의 이웃 노드에게 INITIALIZE 메시지를 전송한다. INITIALIZE 메시지를 받은 노드는 자신의 OWNER 값을 INITIALIZE 메시지를 보낸 노드 이름으로 할당한 후, 다시 인접 노드에게 INITIALIZE 메시지를 송신한다.

◆ REQUEST

임계 지역 진입을 요청하는 메시지이다. 이를 수신한 노드는 자신의 REQUEST_Q에 요청 노드의 ID를 적체한다.

◆ PRIVILEGE

임계 지역 진입을 허락하는 토큰을 의미한다. 논리적으로 네트워크는 항상 하나의 노드에 이 PRIVILEGE 메시지를 유지하여야 한다.

◆ CHANGE

OWNER가 바뀌었음을 알리는 메시지이다. 이 메시지를 전송 받은 노드는 이를 전송한 노드의 OWNER를 가리키도록 OWNER 값을 수정한다.

◆ REJECT

토큰에 대한 권한을 허용할 수 없다는 메시지이다. 이 메시지는 일반적으로 CHANGE 메시지와 함께 사용되는데, 이 메시지를 전송 받은 노드는 CHANGE 메시지로 정정된 OWNER에게 다시 토큰을 요청하게 된다.

◆ WAIT

이동 호스트를 위한 대기 메시지이다. 이 메시지를 전송 받으면, WAKE 메시지를 전송 받을 때까지 대기 상태가 된다. 즉, REQUEST가 있어도 QUEUE에 그 요청들을 적재하기만 한다.

◆ WAKE

대기 상태를 해제하는 역할을 하는 메시지이다.

◆ CANCEL

이 메시지를 수신한 노드는 이를 전송한 노드를 인접 노드 리스트에서 삭제한다.

◆ CORRECT

토큰을 소유한 이동 호스트가 위치를 변경하는 경우 사용되는 메시지이다.

과정이 끝나면, 일반 노드의 상태 값은 다음과 같이 할당된다.

- OWNER = 토큰을 소유한 노드를 간접적으로 가리키는 노드
- USING = FALSE
- REQUEST_Q = NULL
- ASKED = FALSE
- WAIT = FALSE
- NEIGHBORS = 인접한 노드들의 집합
- LOCAL = FALSE

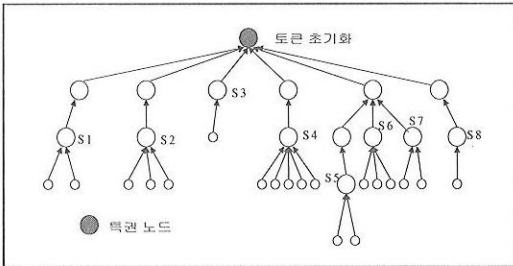


그림 3 신장 트리의 수직 계층 구조

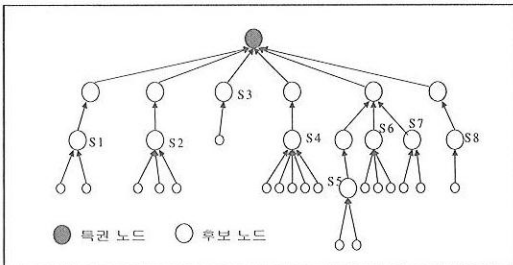


그림 4 후보 노드 추가 후의 수직 계층 구조

초기화 과정에서는 방사형 구조의 가장 중심이 되는 노드를 토큰을 소유한 노드로 초기화한다. 따라서 그 노드에 인접한 노드들은 자동적으로 후보 노드가 된다(그림 4).

4. 제안 상호 배제 알고리즘

4.1 초기화

INITIALIZE 메시지가 모든 노드에 전송되어 초기화

하면, 일반 노드가 아닌 특권 노드와 후보 노드의 경우에는 OWNER 값이 다르게 설정되는데 특권 노드의 경우에는 자기 자신이 토큰 메시지를 소유한 형태이기 때문에 OWNER 값에는 'self'가 초기화되며, 후보 노드의 경우에는 OWNER 값에 '특권 노드의 ID'가 초기화되고, 또한 LOCAL 변수가 'TRUE'로 초기화된다.

4.2 기존 알고리즘의 동작과정 및 문제점

제안 알고리즘을 설명하기에 앞서 Raymond가 제안했던 기존의 알고리즘[7]의 동작 원리를 먼저 살펴본다. 모든 호스트는 PRIVILEGE 메시지라는 토큰을 가져야 임계 지역에 들어갈 수 있는 권한을 갖게 된다. 그 PRIVILEGE 메시지를 할당받기 위해서는 REQUEST 메시지를 사용하여 요청하여야 하며 그 요청은 OWNER 변수에 할당된 노드로 전송된다(그림 5)[8,10].

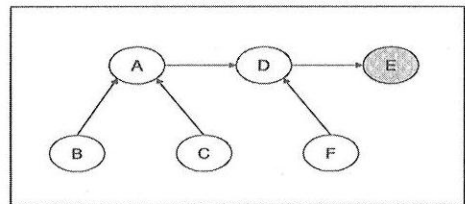


그림 5 기존 알고리즘의 동작 I

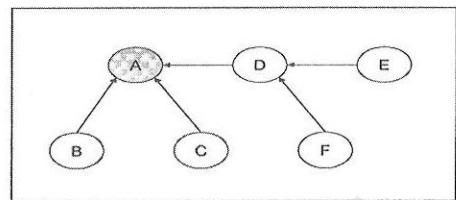


그림 6 기존 알고리즘의 동작 II

표 1 알고리즘의 동작 이전 상태

표 2 알고리즘의 동작 이후 상태

OWNER _A = D	OWNER _A = self
OWNER _B = A	OWNER _B = A
OWNER _C = A	OWNER _C = A
OWNER _D = E	OWNER _D = A
OWNER _E = self	OWNER _E = D
OWNER _F = D	OWNER _F = D

토큰을 갖고 있지 않은 노드 A가 임계 지역에 진입하고자 하는 경우에는 REQUEST 메시지를 자신의 OWNER D에게 전송한다. 토큰을 갖고 있지 않은 노드 D는 다시 자신의 OWNER E에게 REQUEST 메시지를 전송하게 된다. 토큰을 소유하고 있는 노드 E는 REQUEST_Q에 그 메시지를 저장한다. 모든 노드들은 자신과 자신의 이웃에 관한 정보만을 갖고 있기 때문에 노드 A는 토큰의 소유자(E)를 알고 있지 못하지만 토큰의 소유자에게 임계 지역 요청을 하기 위하여 자신의 이웃 노드 중 어느 노드(D)에게 요청메시지를 보내야 할 지를 알고 있다. PRIVILEGE 메시지를 소유하고 있는 노드 E가 임계 지역을 빠져 나오면 E는 PRIVILEGE 메시지를 D에게 전송하며 자신의 OWNER 값을 D로 바꾼다. 노드 D는 PRIVILEGE 메시지를 자신에게 REQUEST 메시지를 보냈던 A에게 전송한 후, 자신의 OWNER 값을 A로 바꾼다. 노드 A는 자신의 OWNER 값을 self 즉, A로 만들고 임계 지역으로 들어가게 된다(그림 6). 그러므로 이 알고리즘에서는 임의의 노드의 요청을 인접 노드가 대신하여 동작하게 되는 것이다.

즉, 노드 A는 노드 E로부터 토큰 권한을 획득하기 위하여 D라는 중간 노드를 거쳐야만 하는 것이다. 따라서 만약 이 중간 노드의 수가 많아진다면, 메시지 교환 비용이 커질 것은 자명한 사실이다. 만약 그림 7과 같은 구조에서 단말(leaf) 노드가 임계 지역 권한을 획득하는 경우, 이 트리를 재구성하면 그림 8에서와 같이 나타나

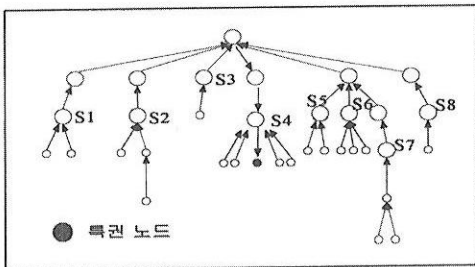


그림 7 PRIVILEGE 메시지의 전송

게 되며, 이 경우 트리의 깊이가 커지는 것을 확인할 수 있다.

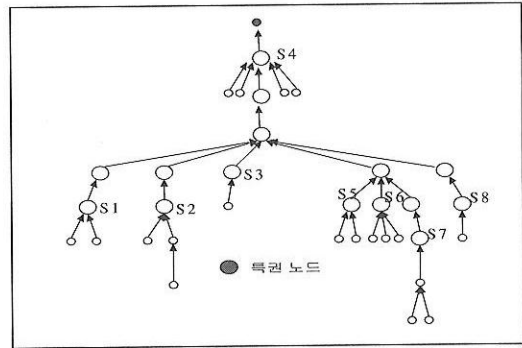


그림 8 새로운 노드를 루트로 하는 트리의 수직 계층 구조

4.3 제안 알고리즘의 동작 과정

그림 7과 같은 상태에서 새로운 노드가 임계 지역 진입 권한을 획득할 때, 트리의 균형이 유지되도록 그림 9에서와 같이 트리를 재구성할 수도 있을 것이며, 본 논문에서는 이러한 기법을 구체적으로 소개한다.

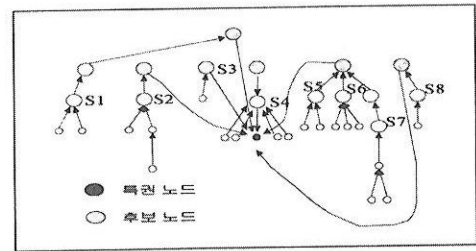


그림 9 제안하는 알고리즘의 적용

트리의 균형을 유지하기 위한 구체적인 방법을 제안하기에 앞서 제안 알고리즘의 전략을 먼저 간략하게나마 살펴보고자 한다. 우선 트리를 K개의 부분 트리로 분할하며, 분할 방법은 초기에 정적으로 결정한다. 상수 K는 알고리즘의 비용과 관련된 시스템 파라미터로서, 그 값이 클수록 트리의 깊이를 줄여주지만, 한편으로 인접 노드에 대하여 유지하여야 하는 정보가 커지는 특성을 가지므로 시스템의 성격에 따라 적절하게 선택되어야 한다. 부분 트리의 루트는 후보 노드가 된다. 후보 노드는 부분 트리 내에서는 마치 트리 전체의 특권 노드처럼 동작한다(그림 10). 한편, 특권 노드를 소유한 부분 트리는 후보 노드를 소유하지 않는다. 다시 말해 후보 노드를 소유하였다는 것은 그 부분 트리에 토큰을

소유한 노드가 존재하지 않는다는 것을 의미한다. 후보 노드는 부분 트리 내의 임의의 노드로부터 요청을 받으면, 특권 노드에게 토큰을 요청하는 중간 노드의 역할만을 담당한다.

루트 노드로부터 가장 먼 거리의 노드가 임계 지역 진입을 요청한 경우를 관찰해보면 제안한 알고리즘이 기존의 알고리즘에 비하여 성능의 향상을 보일 수 있음을 알 수 있다. 즉, 메시지 전송 비용의 관점에서 보면 기존 알고리즘이 임계지역 진입 요청 노드로부터 루트 노드까지의 왕복 거리를 상한값으로 하는 반면, 제안 알고리즘은 임계지역 진입 요청 노드로부터 부분 트리의 루트까지의 거리에 1을 더한 만큼에 비례한 상한값을 갖게 될 것이다. 물론 이러한 구조를 유지하기 위하여 지불해야 하는 보정 작업에 대한 비용이 필요할 것이다. 이러한 보정 작업에 대한 비용에 관해서는 5장에서 언급하도록 하고, 여기서는 (그림 8)과 (그림 11)을 비교하여 제안하는 알고리즘이 기존의 알고리즘과 비교하여 임계 지역 진입에 대한 권한 획득 비용이 상대적으로 적게 드는 것만을 확인한다.

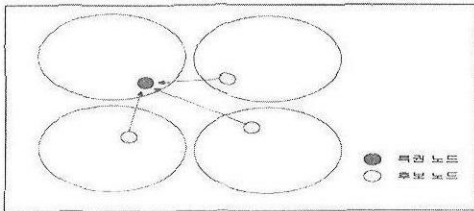


그림 10 제안하는 기법의 논리적인 대응도

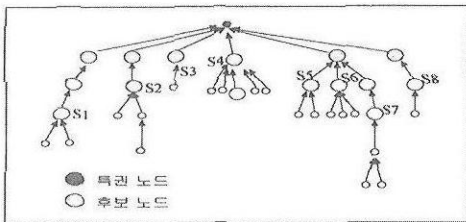


그림 11 제안 알고리즘 적용 후의 수직 계층구조

4.4 고정 네트워크 환경에서의 상호배제 알고리즘

4.4.1 HMA 알고리즘(Height-balance Maintaining Algorithm)의 개요

◆ 단계 1

임계 지역을 진입해야 하는 노드는 토큰을 요청하기

위하여 OWNER 변수에 할당된 노드에 REQUEST를 요청하고 ASKED 변수를 TRUE로 바꾼다.

◆ 단계 2

요청을 받은 노드가 후보 노드나 특권 노드가 될 때까지 요청을 계속한다.

◆ 단계 3

만약 요청 받은 노드가 특권 노드라면 토큰을 전송하고 CHANGE 메시지를 이용하여 자신을 가리키고 있는 후보 노드에게 그 요청을 한 노드의 값을 넘겨준다. 후보 노드는 OWNER 값을 정정하고 알고리즘은 ASKED 변수가 TRUE인 노드가 할당받을 때까지 동작을 계속 반복한다. 만약 요청 받은 노드가 후보 노드라면 토큰 요청 작업을 수행하지 않고 자신에게 토큰을 요청한 노드에게 CHANGE 메시지를 요청하여 OWNER 값을 특권 노드로 수정하고, REJECT 메시지를 전송하여 다시 특권 노드에게 REQUEST를 요청하도록 한다. 즉, 후보 노드의 권한을 그 토큰을 요청한 노드에게 넘겨주는 것이다. ASKED 변수가 TRUE인 노드가 할당받을 때까지 이 작업을 반복한다. 작업을 마치면 후보 노드가 된 노드가 직접 특권 노드에게 토큰을 요청하게 되고 나머지 후보 노드의 OWNER 값을 보정한다(그림 12).

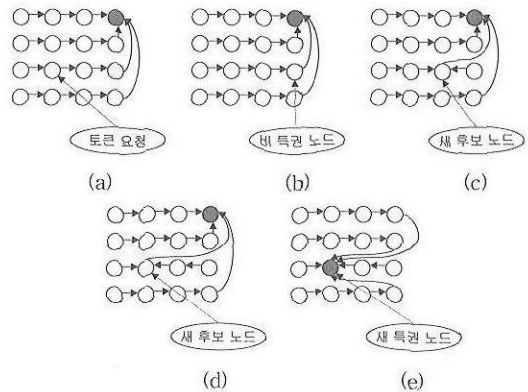


그림 12 HMA 알고리즘 개요

4.4.2 HMA 알고리즘

① 임계 지역 진입을 원하는 경우

노드가 임계 지역에 진입하기를 원하는 경우, 자신을 REQUEST_Q에 넣은 후, 임계 지역 진입 과정을 수행한다.

② 임의의 노드 X로부터 REQUEST 메시지를 받은 경우

표 3 임계 지역 진입을 원했을 때의 알고리즘

```
enqueue(REQUEST_Q, self),
enter_CS;
```

인접한 노드로부터 REQUEST를 요청 받은 경우, 그 인접 노드를 REQUEST_Q에 넣은 후, 토큰 획득 과정을 수행한다.

표 4 임계 지역 진입을 요청 받았을 때의 알고리즘

```
enqueue(REQUEST_Q, X);
enter_CS;
```

③ 임의의 노드가 토큰을 받는 경우

우선 OWNER 값을 자신으로 만든다. 만약 REQUEST_Q가 NULL이 아니면, 임계 지역 진입 과정을 수행하게 된다.

표 5 특권을 획득했을 때의 알고리즘

```
OWNER = self,
if (REQUEST_Q != NULL)
enter_CS;
```

④ 노드가 임계 지역을 빠져 나오는 경우

임계 지역을 사용하지 않는다는 표시로 USING 값을 FALSE로 만든다. REQUEST_Q가 NULL이 아니면, 임계 지역 진입 과정을 다시 수행한다.

표 6 임계 지역을 빠져 나올 때의 알고리즘

```
USING = FALSE,
if (REQUEST_Q != NULL)
enter_CS;
```

⑤ 임계 지역 진입 과정

REQUEST_Q에 REQUEST를 요청한 노드가 적재된 경우에 임계 지역 진입 과정이 발생된다.

자신이 토큰 소유자이고 임계 지역을 수행하고 있지 않다면, 그 REQUEST_Q에서 큐의 정책에 따라 최상위 목록을 꺼내어 이를 OWNER 값으로 만들고 그 값이 만일 자신이면 임계 지역을 수행한다. 만약 그 최상위 목록이 자신이 아니었다면 토큰을 OWNER에게 전송하고 자신의 인접 노드들에게 CHANGE 메시지를 전송하여 OWNER를 자신의 OWNER와 일치시킨다 자신이

표 7 임계 지역 진입 알고리즘

```
enter_CS()
{
    if (OWNER == self ^ ~USING) {
        OWNER = dequeue(REQUEST_Q);
        ASKED = FALSE;
        if (OWNER == self) {
            USING = TRUE;
            <execute Critical Section>
        }
        else
            send Token to OWNER;
    }
    else if (OWNER != self ^ REQUEST_Q != NULL
             ^ ~ASKED ^ ~WAIT) {
        if (LOCAL == TRUE) {
            CHANGE = OWNER;
            OWNER = dequeue(REQUEST_Q);
            send CHANGE, REJECT to OWNER;
        }
        else {
            send REQUEST to OWNER;
            ASKED = TRUE;
        }
    }
}
```

토큰 소유자가 아닐 경우에는 노드가 후보 노드인지 일반 노드인지에 따라 경우가 두 가지로 구분된다. 일반 노드이고 임계 지역 수행을 요청하지 않았다면 단순히 REQUEST 메시지를 OWNER에게 전송한다. 또 후보 노드일 경우에는 REQUEST_Q의 최상위 목록의 노드에 CHANGE 메시지와 REJECT 메시지를 전송하여 새로운 후보 노드로 만들고 자신의 OWNER 값을 새로운 후보 노드로 수정한다. 전송하는 CHANGE 메시지에는 후보 노드의 원래의 OWNER였던 특권 노드가 할당된다.

4.5 이동 호스트를 고려한 HMA 알고리즘

이동 호스트를 고려하면 알고리즘이 보다 복잡해진다. 우선 몇 가지의 제한 요소가 추가로 고려되어 진다. 일단 가장 먼저 고려해야 할 사항은 이동 호스트끼리의 통신을 배제해야 한다는 것이다. 따라서 만약 이동 호스트의 OWNER가 이동 호스트인 경우에는 논리적인 관계에서의 OWNER일 뿐, 실제로는 그 OWNER인 이동 호스트의 인접한 MSS를 상대로 메시지를 전달하도록 설계되어야 한다. 또, 다음으로 고려해야 할 사항은 이동 호스트가 셀을 이동하는 경우이다. 이 경우는 크게 두 가지로 구분하는데 이동 호스트가 토큰을 소유한 채로 셀을 이동하는 경우와 토큰을 소유하지 않고 셀을 이동하는 경우이다.

4.5.1 이동 호스트가 토큰을 소유하지 않고 셀을 이동하는 경우

이동 호스트는 이동 전에 자신의 지역 MSS에게

CANCEL 메시지를 보냄으로써 셀을 이탈한다는 사실을 알리고 이동 후에는 자신의 OWNER 값을 새로운 지역의 MSS로 바꾸고 ASKED가 TRUE였으면 REQUEST를 새로운 OWNER에게 다시 전송한다.

만약 이동 호스트가 후보 노드였다면, 이동 전에 몇 가지 과정을 거쳐 후보 노드의 위치를 계승하여야 한다. 따라서 자신의 인접 노드 내에서 새로운 후보 노드를 선정하여 CHANGE 메시지를 통하여 OWNER를 특권 노드로 지정하여 준 다음, 자신을 가리키고 있는 나머지 노드들을 그 후보 노드를 가리키도록 보정한다. 그 이후 이동 호스트는 앞의 경우처럼 이동할 수 있다.

표 8 토큰을 소유하지 않고 셀을 이동하는 경우의 알고리즘

```

1) 이동 전
if (LOCAL == TRUE) {
  select New_Invalid_Node in NEIGHBORS;
  CHANGE = OWNER;
  send CHANGE to New_Invalid_Node;
  CHANGE = New_Invalid_Node;
  send CHANGE to NEIGHBORS
  except New_Invalid_Node;
}
send CANCEL to MSS;

2) 이동 후
OWNER = MSS;
NEIGHBORS = MSS;
if (ASKED == TRUE)
  send REQUEST to OWNER;

```

4.5.2 이동 호스트가 토큰을 소유하고 셀을 이동하는 경우

즉, 이동 호스트가 특권 노드인 경우이다. 이동 호스트는 이동 전에 분할 트리 내에서 후보 노드를 선정하고 WAIT 메시지로 자신이 새로운 셀에 들어갈 때까지 대기시킨다

표 9 토큰을 소유하고 셀을 이동하는 경우의 알고리즘

```

1) 이동 전
select New_Invalid_Node in NEIGHBORS;
CHANGE = New_Invalid_Node;
send CHANGE to NEIGHBORS except New_Invalid_Node;
send WAIT to New_Invalid_Node;
send CANCEL to MSS;
REQUEST_Q = NULL;

2) 이동 후
NEIGHBORS = MSS;
send CORRECT to OWNER;

```

새로운 셀로 들어가서는 토큰을 소유한 노드가 왔다

는 정보를 알리기 위하여 CORRECT 메시지를 보낸다.

4.5.3 CANCEL 메시지를 받았을 경우

CANCEL 메시지를 보내온 X를 인접 노드 집합에서 삭제하고, 만일 그 X가 REQUEST_Q에 있다면 그 목록도 삭제한다. 그 후 REQUEST_Q가 NULL이면 ASKED 값을 FALSE로 만들고, OWNER 값이 자신이 아니고 인접 노드라면 CANCEL 메시지를 그 인접 노드에게 보낸다.

표 10 CANCEL 메시지를 받았을 때의 알고리즘

```

remove X from NEIGHBORS;
if (REQUEST_Q  $\ni$  X)
  remove X from REQUEST_Q;
if (REQUEST_Q == NULL) {
  ASKED = FALSE;
  if (OWNER != self  $\wedge$  OWNER  $\in$  NEIGHBORS)
    send CANCEL to OWNER;
}

```

4.5.4 CORRECT 메시지를 받았을 경우

CORRECT 메시지를 받으면 토큰을 소유한 이동 호스트가 셀내에 들어온 것을 의미한다. 따라서 새로운 트리 구조의 형성이 불가피하고 인접 노드들에게 차례로 이 CORRECT 메시지를 전송하여 분할 트리 내의 후보 노드를 제거한다. CORRECT 메시지를 후보 노드가 받으면 LOCAL 변수를 FALSE로 바꾸고 CORRECT 메시지를 전송한 노드를 OWNER로 바꾸고, 나머지 일반 노드의 경우에는 계속 자신의 인접 노드에게 CORRECT 메시지를 전송한다 한편 MSS는 이동 호스트가 넘어오기 전의 MSS에게도 WAKE 메시지를 전송하여 대기 상태에서 벗어나게 하는 동시에 논리적으로 연결하여 준다.

표 11 CORRECT 메시지를 받았을 때의 알고리즘

```

if (LOCAL = TRUE) {
  LOCAL = FALSE;
  OWNER = CORRECT;
  WAKE = OWNER;
  send WAKE to Pre-MSS;
}
else
  send CORRECT to NEIGHBORS;

```

4.5.5 WAKE 메시지를 받는 경우

대기 상태에서 깨어난다. READY 변수의 값을 FALSE로 바꾸고 OWNER를 변경한다. Queue에 쌓인 요청이 있었다면 이를 수행한다.

표 12 WAKE 메시지를 받았을 때의 알고리즘

```

if (LOCAL = TRUE) {
    OWNER = WAKE;
    READY = FALSE;
}
if (REQUEST_Q != NULL)
    enter_CS;
    
```

5. 제안 알고리즘의 성능 평가

제안 알고리즘의 성능 평가를 위하여 임계지역 진입 시의 메시지 전송 횟수를 사용한다. 그 이유는 본 논문에서 제안한 알고리즘이 메시지 전송 횟수를 줄이는 부분에 초점을 두고 설계되었으며, 그에 따른 추가적인 비용은 이전의 알고리즘들에 비하여 큰 차이가 없기 때문이다.

임의의 노드가 임계 지역 진입을 허락 받을 때, 교환하는 메시지의 전송 횟수는 전체 노드의 수를 N이라 할 때 토큰 획득에 따른 비용 ($f_a(N)$)과 보정 비용 ($f_c(N)$)의 합이 된다.

($f_a(N)$)의 기대값을 계산하기 위해 모든 노드가 한번 씩 임계지역을 진입한다고 가정하고, 이를 위한 메시지 전송 횟수의 합을 계산하면 다음과 같다.

$$\sum_{k=1}^N f_1(k)$$

여기서 $f_1(k)$ 는 토큰을 요청한 노드로부터 토큰을 소유한 노드, 즉 특권 노드까지의 왕복 거리를 의미한다. $f_1(k)$ 의 값은 각 노드로부터 특권 노드까지의 거리에 의존하므로 네트워크의 구조와 토큰의 위치에 따라 그 값이 다르게 나타난다. 이와 같이 계산된 값을 총 노드의 수 N으로 나누어 각 노드의 평균 임계지역 진입 비용 ($f_a(N)$)을 계산할 수 있게 된다. 물론, 각 노드의 임계지역 진입 빈도가 다른 경우에는 노드별로 가중치를 주어 계산할 수도 있을 것이다.

최악의 경우는 직선형의 네트워크 구조가 만들어지고, 토큰을 가진 노드가 직선형 구조의 한쪽 끝에 위치하는 경우이며, 이 경우 직선형의 구조에 위치한 N개 노드의 요청 비용의 합은 다음과 같이 계산된다.

$$\sum_{k=1}^N f_1(k) = \sum_{k=1}^N 2 \cdot (k-1) = N \cdot (N-1) = O(N^2)$$

최상의 경우는 방사형 구조의 네트워크가 만들어지고, 그 중앙부에 토큰을 소유한 노드가 위치하는 경우이다. 이러한 구조에서의 요청 비용은 부분 트리의 개수

및 그 구조와 밀접한 관계를 갖게 된다. 네트워크 전체를 x개의 부분 트리로 분할하고, 이 부분 트리들을 다시 계속해서 x개의 부분 트리로 분할한다고 가정할 경우, N개 노드의 요청 비용의 합은 다음과 같이 계산된다.

$$\sum_{k=1}^N f_1(k) = \sum_{k=1}^N 2 \cdot \lceil \log_x k \rceil \leq \sum_{k=1}^N 2 \cdot \lceil \log_x N \rceil = O(N \cdot \log_x N)$$

본 제안 기법의 경우 초기에 네트워크 구조를 최적의 방사형 구조가 되도록 설계하면, 이후 그 구조가 적절히 유지되며, 이에 따라 각 호스트의 임계지역 진입을 위한 비용은 $O(N \cdot \log_x N)$ 으로 유지되게 된다. 특히, 이 경우 각 노드의 인접 노드 수인 x의 값을 증가시킬수록 트리의 높이가 감소하여 비용이 줄어드는 반면, 인접 노드에 대한 부가적인 정보의 양이 증가하기 때문에 x의 값에 대해서는 적절한 조절이 필요할 것이다.

Raymond의 알고리즘[7]에서도 초기의 과정에서는 제안 알고리즘과 동일한 요청 비용을 갖지만, 이후에 토큰을 소유한 노드가 중앙에 위치하지 않고, 그 구조가 변형된다는 점이 단점으로 작용하게 된다. 또한, 이 기법에서는 중앙부에 위치한 노드들이 가장자리의 노드들에 비하여 보다 손쉽게 임계지역 진입 권한을 획득하는데 있어 그 형평성을 보장하지 못하고 있다.

제안 알고리즘에서는 이렇게 이상적인 토큰의 위치를 계속 유지하는 반면에 위치를 유지하기 위한 보정 비용 ($f_c(N)$)이 소비된다. 보정 비용은 후보 노드들로부터 임계 지역 진입을 요청한 노드까지의 특권 노드 지시를 위한 보정 비용이므로, 후보 노드들로부터 임계 지역 진입 요청 노드까지의 거리의 합으로 다음과 같이 계산된다.

$$f_c(N) \leq \sum_{k=1}^N \lceil \log_x k \rceil = O(N \cdot \log_x N)$$

따라서 결과적으로 제안 알고리즘에 대한 비용은 방사형 구조의 N개 노드들의 요청 비용과 보정 비용을 합산하여 다음과 같이 계산될 수 있다.

$$(f_a(N)) + (f_c(N)) \leq O(N \cdot \log_x N)$$

6. 결론

본 논문에서는 트리의 균형을 고려한 상호 배제 알고리즘과 새로운 시스템 모형을 제안하였다. 제안 알고리즘에서는 논리적인 중앙부에 토큰을 소유한 노드를 유지하기 위하여 전체의 트리를 부분 트리로 분할하여 부분 트리의 특정 권한을 획득한 뒤, 전체 네트워크의 권

한을 획득하는 분할 정복(divide and conquer) 방식을 따른다.

이 알고리즘을 사용하므로써 메시지 전송 횟수를 감소시킬 수 있을 뿐 아니라, 모든 호스트에서 임계 지역으로 접근하는 과정이 거의 비슷한 성능을 보이는 공정성까지 보장하고 있다. 또한, 고정 호스트뿐만 아니라 이동 호스트에 대한 알고리즘까지 제안함으로써 이동 컴퓨팅 환경에서의 상호 배제를 보장하고 있다.

이상적인 시스템 모형을 통한 성능 평가에 따르면 메시지 교환 횟수를 비교하였을 때, 본 제안 알고리즘의 우수성을 입증할 수 있었다. 하지만 상호 배제 알고리즘이 기존의 알고리즘에 비하여 복잡해지고 트리 모양을 유지하기 위한 보정 비용이 소비된다는 것이 그 단점이다. 따라서 앞으로 이러한 단점들을 보완하기 위한 추가적인 연구가 지속적으로 필요할 것이다.

참 고 문 헌

- [1] Ricart, G. and Agrawala, A. K., "An Optimal Algorithm for Mutual Exclusion in Computer Networks," *Comm. ACM*, Vol. 24, No. 1, Jan. 1981.
- [2] Suzuki, I. and Kasami, T., "A Distributed Mutual Exclusion Algorithm," *ACM Trans. Comput. Syst.*, Vol. 3, No. 4, Nov. 1985.
- [3] Maekawa, M. A., " \sqrt{N} Algorithm for Mutual Exclusion in Decentralized Systems," *ACM Trans. Comput. Syst.*, Vol. 3, No. 2, May. 1985.
- [4] V. Kumar, J. Place, and G. C. Yang, "An Efficient Algorithm for Mutual Exclusion Using Queue Migration in Computer Networks," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 3, No. 3, Sep. 1991.
- [5] B. R. Badrinath, A. Acharya, and T. Imielinski, "Structuring Distributed Algorithms for Mobile Networks," In *Computer Communications*, Apr. 1996.
- [6] K. Raymond, "A Tree-based Algorithm for Distributed Mutual Exclusion," *ACM Trans. on Comput. Syst.*, Vol. 7, No. 1, Feb. 1989.
- [7] 안상준, 임영익, "트리 구조에 기반한 모빌 환경에서의 분산 상호 배제 알고리즘", 제 6회 통신정보 합동학술대회 논문집, Apr. 1996.
- [8] 안상준, 임영익, "모빌 환경에서 노드 고장을 고려한 분산 알고리즘 설계 기법", 한국 정보처리학회 '96 추계 학술발표 논문집, Oct. 1996.
- [9] 안상준, 임영익, "이동 컴퓨팅 환경을 위한 분산 알고리즘 설계 기법" 정보과학회 논문지, 한국정보과학회, Vol. 24, No. 7, Jul. 1997.
- [10] B. R. Badrinath, A. Acharya, and T. Imielinski, "Structuring Distributed Algorithms for Mobile Hosts," In *Proc. of the 14th International Conference*

on Distributed Computing Systems, Jun. 1994.



김 형 식

1999년 2월 성균관대학교 정보공학과 졸업(학사). 1999년 3월 ~ 현재 한국과학기술원 전산학과 석사과정. 관심분야는 분산 알고리즘, 데이터 마이닝



엄 영 익

1983년 서울대학교 계산통계학과 졸업(학사). 1985년 서울대학교 대학원 전산과학전공(석사). 1991년 서울대학교 대학원 전산과학전공(박사). 1983년 ~ 1986년 서울대학교 도서관 전산화준비실. 1986년 ~ 1993년 단국대학교 전자계산학과 부교수. 1993년 ~ 현재 성균관대학교 전기전자 및 컴퓨터공학부 교수. 관심분야는 분산 시스템, 이동 컴퓨팅, 분산 객체 시스템, 운영체제