커뮤니티 컴퓨팅 환경에서 자원 관리 서비스를 이용한 그룹 상호 배제 알고리즘

Group Mutual Exclusion Algorithm Using RMS in Community Computing Environments

박창우* · 김기영** · 정혜동*** · 김석윤****
Chang-Woo Park · Ki-Young Kim · Hye-Dong Jung · Seok-Yoon Kim

Abstract - Forming Community is important to manage and provide the service in Ubiquitous Environments including embedded tiny computers. Community Computing is that members constitute the community and cooperate. A mutual exclusion problem occurs when many processors try to use one resource and race condition happens. In the expanded concept, a group mutual exclusion problem is that processors in the same group can share the resource but processors in different groups cannot share. As mutual exclusion problems might be in community computing environments, we propose algorithm which improves the execution speed using RMS (resource management service). In this paper describes proposed algorithm and proves its performance by experiments, comparing proposed algorithm with previous method using quorum-based algorithm.

Key Words : Community Computing, Group Mutual Exclusion, Resource Management Service

1. 서론

유비쿼터스는 작은 컴퓨터가 주변 환경과 사물 등에 내재되어 사용자가 컴퓨터나 네트워크를 의식하지 않는 상태에서 장소에 구애받지 않고 자유롭게 네트워크에 접속할 수 있는 환경을 의미한다.

유비쿼터스 공간 내에서 각종 멤버들이 커뮤니티를 형성하고 상호 협력하는 것을 커뮤니티 컴퓨팅이라 한다. 커뮤니티 컴퓨팅 모델에는 세 가지 커뮤니티 모델(정적인 커뮤니티, 동적인 커뮤니티, 자율적인 커뮤니티)에 따라 구분될 수 있다. 이보다 발달된 환경의 커뮤니티(정적->동적->자율적인 커뮤니티)에서 멤버간의 복잡한 커뮤니티가 이루어질수록 다수의 커뮤니티의 멤버가 동일한 자원의 사용을 요청 했을 시 일어나는 문제커지고 그 해결 방법은 복잡해진다. 멤버들은 리소스를 사용하기 위해 커뮤니티 내에서나 커뮤니티 밖에서 다른 멤버들과 경쟁을 하는 경우가 생긴다. 특히나 외부의 리소스를 사용하기 위해 다른 커뮤니티의 멤버들과 자원을 공유하게 되는데 이럴 때 접근 충돌 등과 같은 상호 배제 문제가 생길 수가 있다.

상호 배제(Mutual Exclusion)는 분산시스템에서 기본적인 문제들 중 하나이다. 다수의 프로세서가 CS (Critical Section: 임계 영역)에 진입하는 것과 같이 공유 자원에 접속하고자 하지만, 자원은 하나의 프로세스만 사용할 수 있으므로 경쟁 상태가 발생하는 것이 상호 배제 문제이다. 또한, 확장된 개념으로 그룹 상호 배제 (Group Mutual Exclusion)가 있다[1.] 이것은 같은 그룹에 속한 프로세스들은 자원을 공유할 수 있지만, 다른 프로세스가 사용하고 있을 시 다른 그룹에 속한 프로세스들은 사용을 하지 못하는 문제이다.

지금까지 상호 배제 문제를 해결하기 위한 여러 가지 알고리즘들이 제안되어 왔으나 기존 연구들 효과적인 topology 형성에 관련된 것이 대부분이었다[2.] 본 논문에서는 topology 형성 이후 그룹 상호 배제 문제 해결에 걸리는 시간을 최소화하는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 서론에 이어 2장은 기존에 연구되었던 관련 연구를 살펴보고, 3장에서는 RMS를 이용한 알고리즘을 설명한다. 4장에서는 실험을 통하여 기존 알고리즘과 제안하는 알고리즘의 성능을 비교하고 마지막으로 5장에서는 결론을 맺는다.

2. 관련 연구

일반적으로 상호 배제를 위한 m-그룹 쿼럼시스템(qourum system)의 정의는 다음을 따른다[2].

정의. Let $P=1,...,n$ be a set of nodes. An m-group quorum system $C=C_1,...,C_m$ over P consists of m sets, where each $C_i \subseteq 2^P$ is a set of subsets of P satisfying the following properties:

intersection:

$\forall 1 \le i,j \le m, 1 \ne j, \forall Q_1 \in C_i, \forall Q_2 \in C_j : Q_1 \cap Q_2 \ne \varnothing$

minimality:

$\forall 1 \le i \le m, \forall Q_1, Q_2 \in C_i, Q_1 \ne Q_2 : Q_1 \not\subseteq Q_2$

저자 소개
* 崇實大學 컴퓨터學科 碩士課程
** 崇實大學 컴퓨터學科 博士課程
*** 電子部品硏究院 知能型 情報硏究센터, KETI
**** 崇實大學 컴퓨터學科 正敎授 · 工博

¶본 연구는 지식경제부 지역산업기술개발사업의 "PAMP용 개인화 미디어 게이트웨이 시스템 기술개발" 지원에 의한 것임.

C_i를 카르텔(cartel)이라 하고 $Q \in C_i$를 쿼럼(quorum)이라 한다. 프로세스가 한 개 이상의 그룹에 속하여 CS (Critical Section)에 진입을 원할 때, 프로세스는 속하고자 하는 그룹을 구분하여야 한다. 프로세스가 CS에 들어가고자 할 때, '$Q \in C_i$'을 만족하는 그룹 멤버의 승인을 얻어야 한다.

3. 제안 알고리즘

3.1 기존 알고리즘

유비쿼터스 환경에서 상호 배제를 보장하기 위해 교집합 특성을 만족하는 그룹 내에 컴포넌트를 만든다.

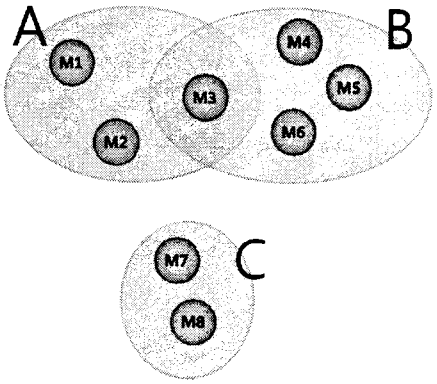

그림 1 유비쿼터스 환경에서 동적으로 형성된 그룹

그림 1은 유비쿼터스 환경에서 무작위로 형성된 그룹을 보여주고 있다. 그룹 A,B,C는 각각의 멤버를 가지고 있으며 그룹 A와 B는 교집합을 가지고 있다. 그룹 C는 어느 그룹과도 교집합을 가지고 있지 않다.

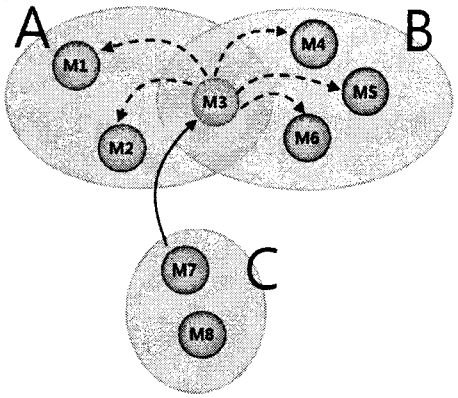

그림 2 기존 쿼럼 기반 알고리즘에서의 group merging

만약 그룹 C에 있는 멤버가 리소스를 사용하려 한다면 상호 배제 문제를 피하기 위해 그 리소스가 다른 그룹에서 사용 중인지를 판단하여야 한다. 그러기 위해선 그룹 A와 B의 교집합 멤버를 통하여 모든 멤버들에게 리소스 사용 요청 메시지를 보내야 한다.

이와 같이 기존의 쿼럼 기반 알고리즘에서 그룹의 멤버가 CS에 접근하기 위해서는 CS에 들어가 있는 모든 그룹의 멤버의 승인을 받아야 하기 때문에 그림 2와 같은 경우는 deg(C) = 5 값을 가져 비효율적인 부분이 있다. 따라서 본 논문에서는 각 멤버의 정보를 통하여 그룹의 priority를 결정해 deg(C)의 값을 줄이고자 한다.

3.2 제안하는 알고리즘

제안하는 상호 배제 알고리즘은 다음과 같은 환경에서 이루어짐을 전제로 한다.

1. 각 멤버는 최소한 자신이 포함된 하나 이상의 그룹에 포함되어 있다.
2. 각 멤버들 사이에서 전송되는 메시지는 신뢰성이 보장된다.
3. 멤버의 기본적인 정보는 다음과 같다.
   Member.Info={ID,Name,System priority,join time, reminded time, battery...}

위와 같은 환경에서 제안된 상호배제 알고리즘은 다음과 같은 정책을 따른다.

1. 그룹 통합 시 자원에 대한 요청 큐의 merging 과정은 멤버의 우선순위에 영향을 받는다.
2. 분배가 가능한 자원에 대한 경우는 라운드-로빈 알고리즘에 따라 각 자원에 대한 멤버의 권한을 time unit을 할당받아 수행된다.
3. 분배가 가능하지 않는 자원(예를 들면 Projecter ,Printer, Display..)에 대하여는 해당 멤버를 관리하는 서비스를 생성하여 공동 요청 시 그룹 priority 에 따라 그 순서를 결정한다.

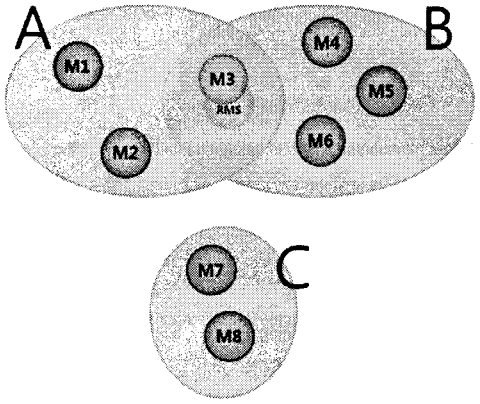

그림 3 개선된 쿼럼 기반 알고리즘에서의 group merging

다른 그룹에서 M3을 포함하여 그룹을 재구성 하려 하면 공유 불가능한 멤버인 M3을 관리하는 RMS (resource management service)를 생성하여 각 그룹의 통합 그룹 priority를 관리하게 한다. 이때 기존의 쿼럼 기반의 알고리즘에 비하여 항상 deg(C) = 1을 유지할 수 있어 이에 따라 처리 속도를 향상 시킬 수 있다.

RMS는 기존 멤버의 정보를 가지고 각 그룹에서 멤버가 추가, 삭제 될 시 그룹의 priority는 재설정된다.

수식 1. 그룹 우선순위 값

group.priority =

{∀mx ∈ group | Σ(mx.Join_time + mx. reminded* weight1 + battery*weight2 + mx.priority *weight3)}

(mx = memberx)

여기서 각 weight는 해당된 그룹의 전체 목적에 따라 그 가중을 다르게 한다.

RMS는 M3의 요청 큐가 존재할 때 그룹이 재구성되면 상기 group.priority 의 값에 따라 요청 큐를 재구성한다.

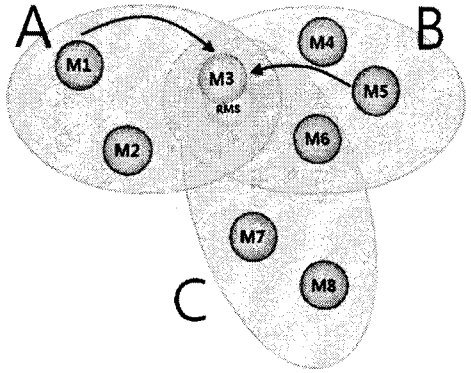

그림 4 동시에 CS에 접근하였을 경우 상호배제 상황

동시에 공유 불가능한 멤버(M3)에 대한 요청이 수신되었을 때 RMS는 이를 관리하여 group.priority를 비교하여 요청 큐에 입력하여 멤버간의 요청에 대한 충돌을 없앤다.

4. 실험 및 결과

제안하는 알고리즘의 성능을 평가하기 위해 다음과 같은 실험을 실시하였다. 그룹의 수는 최소 3개에서 최대 20개로 두고 각각의 그룹은 10개의 멤버를 가진다. 리소스 사용 멤버와 요청 멤버는 무작위로 선정하였다.

시뮬레이션 프로그램은 C로 작성되었으며, 그룹의 개수는 3개, 5개, 7개, 10개 15개, 20개로 두었고 각각의 그룹에 대해 50회씩 5번의 실험을 하였다.

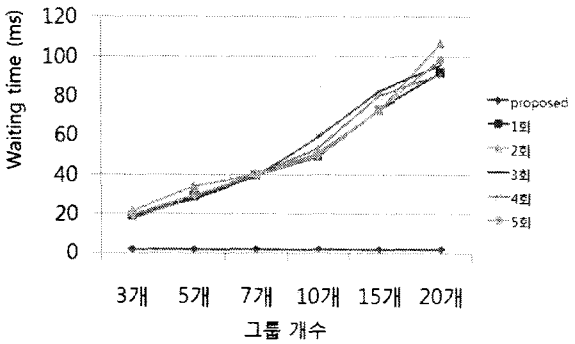

그림 5 기존 알고리즘과 제안하는 알고리즘에서 그룹 개수에 따른 waiting time의 변화

그림 5는 실험에 대한 결과로, 기존 알고리즘과 제안하는 알고리즘에서의 waiting time을 그룹 개수에 따라 나타내고 있다. 기존 알고리즘의 경우 그룹 개수가 늘어남에 따라 waiting time이 늘어난다. 즉, 그룹 개수에 따라 멤버의 수도 늘어나고, 이에 따라 리소스 사용 중인 그룹에서 승인을 받아야 하는 멤버의 수가 늘어나므로 waiting time이 늘어나게 되는 것이다. 이에 반해 제안하는 알고리즘은 모든 리소스의 상태를 전체 교집합에 속한 멤버에 있는 RMS가 관리를 하게 되므로 한 번의 요청과 승인에만 waiting time이 소요된다. 기존 알고리즘에서 각각의 그룹 수에 대해 평균 19.6ms, 30ms, 39.9ms, 52.6ms, 76,3ms, 96.8ms의 시간이 소요되었으며 제안하는 알고리즘에서는 그룹의 개수와는 상관없이 2.1ms로 일정한 시간이 소요되었다.

5. 결론

본 논문에서는 기존의 쿼럼-기반 알고리즘[3]을 응용하여 처리 속도를 더 향상시킬 수 있게 리소스를 관리하는 RMS (Resource Management Service)를 이용한 알고리즘을 제안하였다. 커뮤니티 컴퓨팅 환경에서는 수많은 멤버가 리소스를 사용하고, 이 때 리소스를 효율적으로 배분하기 위해 상호배제 알고리즘이 필요하다. 기존 알고리즘이 모든 그룹에 속하는 멤버를 두어 리소스를 사용하는 멤버로의 접근을 용이하게 했다면, 본 논문에서의 알고리즘은 RMS를 두어 RMS가 직접 리소스를 관리하게 하였다. 이는 RMS의 추가적인 저장 공간이 필요하게 되었지만, 요구하는 저장 공간의 크기가 크지 않고 이에 반해 얻게 되는 시간적인 이득이 크다. 그룹이나 멤버의 증가에 관계없이 일정한 waiting time이 걸리므로 RMS의 효율성은 높은 것으로 판단된다.

참 고 문 헌

[1] Y.-J.Joung, "Asynchronous group mutual exclusion. Distributed Computing", Vol.13, No.4 (2000) 189-206

[2] Y.-J.Joung, "Quorum-based algorithm for group mutual exclusion", IEEE Trans. On Parallel and Distributed Systems, Vol.14, No.5 (2003) 463-476

[3] 윤재희, 김재훈, 조위덕, "유비쿼터스 환경에서 그룹 사이에서의 상호 배제 알고리즘", 한국정보과학회 가을 학술발표논문집, Vol.33, No.2(A) (2006)