유비쿼터스 환경에서 그룹 사이에서의 상호 배제 알고리즘

윤재희°, 김재훈, 조위덕
아주대학교 정보통신전문대학원
jhyoon@dmc.ajou.ac.kr ° { jaikim, chowd}@ajou.ac.kr

Group Mutual Exclusion Algorithm in Ubiquitous Environments

Jae-Hee Yoon °, Jai-Hoon Kim, We-Duke Cho
Graduate School of Information and Communication, Ajou University

요 약

유비쿼터스 환경은 이질적인 디바이스와 서비스로 구성되어 있으며 많은 컴포넌트들이 각기 목적을 가지고 여러 네트워크에 산재되어있다. 이러한 디바이스들을 연결하고 서비스를 효율적으로 관리하기 위해 비슷한 목적을 가지는 컴포넌트들을 하나의 그룹으로 묶을 수 있다. 이러한 그룹의 멤버들은 서비스를 제공하기 위해 같은 그룹의 멤버들과 리소스를 공유하며 때로는 그룹의 모든 멤버들이 동시에 하나의 리소스를 사용하기도 한다. 서로 다른 그룹이 이러한 리소스를 공유하기도 한다. 따라서 그룹 간에 상호 배제 알고리즘이 필요하다. 이 논문에서는 쿼럼-기반 알고리즘(Quorum-based algorithm)을 응용하여 유비쿼터스 환경에서 그룹간의 상호 배제문제를 해결하는 알고리즘을 제안하고 이에 대한 성능측정 결과를 설명한다.

1. 서론

유비쿼터스 환경은 사용자와 주변의 모든 컴퓨팅 자원들이 융합되어 인간이 컴퓨팅 장비를 의식하지 않고 편안하게 정보를 사용할 수 있도록 서비스를 제공하는 것을 말한다. 유비쿼터스 환경은 이질적인 디바이스와 서비스로 구성되어있으며 많은 컴포넌트들이 각기 목적을 가지고 여러 네트워크에 산재되어있다. 유비쿼터스 환경에서 이러한 디바이스들을 연결하고 서비스를 효율적으로 관리하기 위해 비슷한 목적을 가지는 컴포넌트들을 분류하여 하나의 그룹으로 묶을 수 있다. 컴포넌트들이 그룹으로 묶여서 서로 연계하여 서비스를 제공할 때 보다 효율적으로 시스템을 유지할 수 있을 뿐 아니라, 사용자나 시스템의 요구에 보다 질 높게 대처할 수 있다.

유비쿼터스 환경에서 생성된 그룹의 멤버들은 서비스를 제공하기 위해 같은 그룹의 멤버들과 리소스를 공유하여 사용한다. 그리고 그룹의 멤버들은 그룹 내부에서 공유하고 있는 자원뿐만 아니라 그룹 외부의 자원을 사용할 수 있다. 때로는 그룹의 모든 멤버들이 동시에 하나의 자원에 대해 접근해야 하는 경우도 발생하는데 특히, 그룹 외부의 자원의 경우에는 다른 그룹과 공유하게 된다. 따라서 여러 그룹이 동시에 하나의 리소스에 접근할 수 있게 되고 이에 따라 리소스에 대한 접근 충돌이나 잘못된 데이터의 조작이 일어날 수 있다. 따라서 그룹 간에 상호배제문제를 해결하기 위한 알고리즘이 필요하다.

그룹 상호 배제(Group Mutual Exclusion)는 [1]에 의해 제안되었다. [1]에서는 그룹 상호 배제를 "어떤 주어진 시점에서 서로 다른 그룹에 속한 두 프로세스는 동시에 CS(Critical Section: 임계영역)에 들어갈 수 없다" 라고 정의를 하였다. 즉, 프로세스의 그룹이 여러 개가 있을 때 같은 그룹에 속한 프로세스는 CS에 동시에 들어갈 수 있다. 반면에 서로 다른 그룹에 속한 프로세스는 CS에 동시에 들어갈 수 없다. [3]에서는 그룹 상호 배제의 예로 주크박스를 들었다. CD 주크박스가 있고 각각의 프로세스가 CD에서 어떤 데이터를 읽어오기를 원한다. 만약에 CD A가 주크박스로 로드 되었다면 CD A에 있는 데이터를 읽기 원하는 모든 프로세스들은 동시에 CD A의 데이터를 읽을 수 있다. 이 프로세스들은 같은 그룹에 속하게 된다. 반면에 CD B에 있는 데이터를 읽기를 원하는 프로세스들은 CD A가 주크박스에 로드 되어 있는 동안에는 그 작업을 할 수 없다. 이 프로세스들은 다른 그룹에 속한다.

이처럼 그룹 상호 배제는 한 리소스에 그룹 전체 멤버의 접근을 허용한다. 유비쿼터스 환경에서 생성된 그룹들은 서로 리소스를 공유하기 때문에 상호 배제 알고리즘이 필요하다. 또한 이러한 상호 배제 알고리즘은 동적인 유비쿼터스 환경의 특성을 고려해야 한다. 이 논문에서는 유비쿼터스 환경에서 발생할 수 있는 그룹 상호 배제 문제를 해결하는 방안을 제안한다.

본 논문은 다음과 같은 내용으로 구성된다. 2장에서는 관련연구를 설명하고 3장에서는 제안하는 알고리즘에 대해 설명을 한다. 그리고 4장에서는 제안한 알고리즘의 성능평가를 설명하고 마지막으로 5장에서는 결론을 설명한다.

본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 지원에 의한 것임

2. 관련연구

이 장에서는 그룹 사이에서 발생하는 상호 배제를 해결하는 알고리즘에 대한 관련연구를 알아보도록 한다.

그룹 상호 배제를 처음 제안한 [3]에서는 이에 대한 해결방안으로 m-그룹 쿼럼 시스템(m-group quorum system)을 소개하였다. [3]에 나타난 m-그룹 쿼럼 시스템 의 정의는 다음과 같다.

정의1. 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 flowing properties.

Intersection:

$\forall 1 \le i, j \le m, i \ne j, \forall Q_1 \in C_i, \forall Q_2 \in C_j : Q_1 \cap Q_2 \ne 0$.

Minimality:

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

위의 정의에서 C_i를 카르텔(cartel)이라고 하고 각 $Q \in C_i$를 쿼럼(quorum)이라고 한다. C는 다음과 같이 그룹 상호 배제문제를 풀 수 있다: 그룹 j에 속한 각 프로세스 i가 CS에 들어가기를 원할 때, '$Q \in C_i$' 이 조건을 만족하는 쿼럼에 속한 모든 멤버들에게 승인을 얻어야만 한다. 그리고 CS에서 나올 때, 프로세스 i는 그 쿼럼에 속한 멤버들에게 알려야 한다. 한 쿼럼의 멤버가 하나의 프로세스에게만 승인을 줬다고 가정했을 때, 교집합의 특성에 의해서 서로 다른 그룹에 속한 두 노드가 동시에 CS에 들어갈 수 없다. Minimality 속성은 효율성을 높이는데 사용된다.

3. 제안 알고리즘

3.1 가정사항

유비쿼터스 환경에서 그룹 사이에 일어날 수 있는 상호 배제문제를 해결하기 위해 다음과 같은 가정을 한다.

1) 그룹에 포함된 멤버들 사이에서 전송되는 메시지는 신뢰성이 있다.
2) 각 그룹에 포함된 멤버들은 member_list를 가지고 있다. member_list는 자신이 속한 그룹에 포함된 멤버들의 리스트이다. 이 리스트는 어느 멤버가 그룹을 탈퇴하거나 새롭게 참여할 경우 그러한 변화를 즉각적으로 반영한다.
3) 각 그룹과는 별도의 저장소에 <그룹 아이디, {멤버리스트}>가 저장되어있다. 이 저장소의 정보는 동적으로 변화하는 그룹의 상태를 즉각 반영한다. 그룹에 속한 멤버들은 이 저장소에서 다른 그룹의 아이디와 그 그룹에 속한 멤버들의 리스트를 얻어올 수 있다.
4) 쿼럼-기반 알고리즘(Quorum-based Algorithm)[3]에서는 프로세스의 그룹과 쿼럼의 집합이 따로 존재하지만, 본 논문에서 제안한 알고리즘에서는 그룹 자체가 쿼럼이 된다. 그리고 그룹에 속한 멤버들이 쿼럼의 멤버로서 동작을 한다.
5) 쿼럼-기반 알고리즘에서 상호 배제를 보장하기 위한 특성인 교집합 특성을 만족하기 위해서 모든 그룹에 포함되는 컴포넌트를 만든다. 유비쿼터스 환경에서 그룹 자체를 쿼럼으로 여기면 약간의 문제가 발생한다. 예를 들어 그림1을 살펴보도록 하자. 그림1을 보면 그룹 D와 그룹 C, 그리고 그룹 B와 그룹 C사이에는 교집합이 없음을 알 수 있다. 이러한 상황에서 그룹 D가 그룹B를 쿼럼으로 선택하여 리소스 R에 대한 요청메시지를 보낸 상황을 고려해보도록 한다. 이 때 그룹 C가 이미 리소스 R을 사용하고 있다고 가정하여 보자. 그룹 D와 그룹 B는 각각의 그룹에 포함되면서 그룹 C에도 포함된 멤버가 없기 때문에(즉, 그룹 C와 교집합을 이루는 멤버가 없기 때문에) 그룹 C가 리소스 R을 사용하고 있다는 사실을 모르게 된다. 이러한 상황에서 그룹 B에 속한 어떠한 멤버도 리소스 R을 사용하고 있지 않는다면 그룹 B의 모든 멤버는 그룹 D에게 리소스 R을 사용할 수 있도록 OK메시지를 보내게 된다.

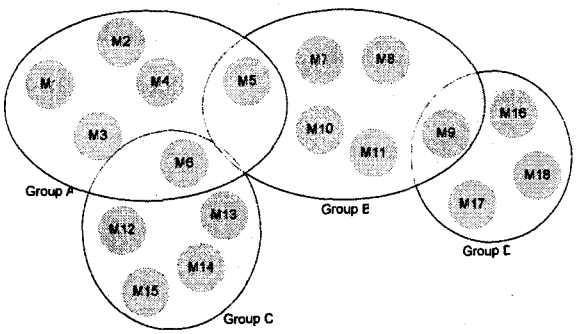

그림 1. 유비쿼터스 환경에서 그룹은 특정 패턴이나 규칙이 없이 동적으로 형성된다. 이러한 그룹형성은 그룹 상호 배제를 위한 교집합 특성을 만족하지 못하는 경우를 발생시킨다.

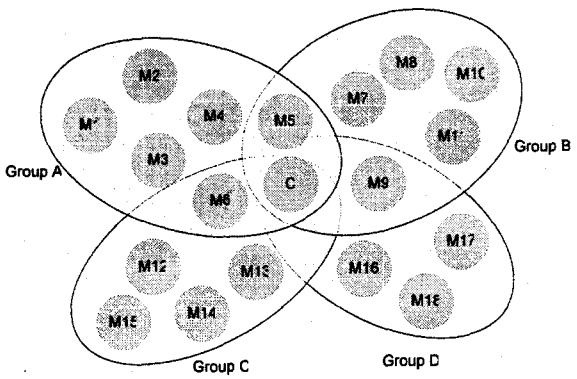

그림2. 교집합의 속성을 만족시키기 위해 각 그룹에 공통으로 포함되는 멤버를 포함시킨다. 이 멤버는 (리소스, 상태)리스트를 관리한다.

이처럼 유비쿼터스 환경에서 동적으로 형성되는 그룹은 교집합 특성을 만족하며 형성되지 않을 수 있다. 유비쿼터스 환경에서의 그룹은 사용자나 개발자가 지정한

대로 형성되지 않는다. 또한 유비쿼터스 환경에서 형성되는 그룹은 특정 규칙이나 패턴에 의해 형성되지 않는다. 이러한 특성 때문에 교집합 특성을 만족하도록 그룹을 생성하는 것은 어렵다. 따라서 이러한 문제를 해결하기 위해 각 그룹에 교집합의 역할을 하는 컴포넌트를 포함시킨다(그림2). 이처럼 공용컴포넌트가 모든 그룹에 포함되기 때문에 임의로 그룹이 생성되더라도 교집합의 속성이 만족될 수 있다.

3.2 그룹 사이에서의 상호 배제 알고리즘

이 논문에서는 그룹 사이의 상호 배제문제를 해결하기 위해 [1]에서 제안된 쿼럼-기반 알고리즘을 응용한다.

그룹 상호 배제를 위한 쿼럼-기반 알고리즘은 프로세스의 그룹과 쿼럼이 따로 존재한다. 하지만 본 논문에서 제안한 알고리즘에서는 그룹자체가 쿼럼이 되고, 그룹에 속한 멤버가 쿼럼의 멤버로 동작한다.

본 논문에서 제안하는 알고리즘은 크게 두 부분으로 나뉠 수 있다. 요청메시지를 보내는 프로세스와 요청메시지를 받는 프로세스가 그것이다. 먼저 CS에 들어가기 위해 요청메시지를 보내는 프로세스부터 살펴보도록 한다.

1) 요청 프로시저:

알고리즘 1. 그룹 G 에 속하는 멤버 m 이 CS 에 들어가기를 원할 때

```
Begin
  gState = (R, TRYING);
  select arbitrary group as quorum
  from coterie;
  send RequestCS(G, m, R, t) to all
  member of the group;
end;
```

변수:

- (R, gState): 리소스 R에 대한 그룹의 상태를 나타내는 변수이다.
- m: 메시지를 보내는 컴포넌트의 아이디를 나타낸다.
- G: 컴포넌트 m이 속한 그룹의 아이디를 나타낸다.
- R: CS 즉, 리소스의 아이디를 나타낸다.
- t: 메시지를 보내는 시간을 나타낸다.

알고리즘1에 나타나있는 그룹의 모든 멤버가 동시에 한 리소스를 사용해야 하는 경우를 먼저 살펴보도록 하자. 그룹에서 어느 한 컴포넌트가 임의의 그룹을 쿼럼으로 선택하여 쿼럼으로 선택된 그룹의 모든 멤버에게 요청메시지를 전송한다. 이때 둘 이상의 멤버가 같은 리소스에 대해 각자 쿼럼을 선택해서 요청메시지를 보내지 않는다는 것을 가정한다. 다음으로 멤버가 요청메시지를 보낸 후에 그에 대한 응답을 받았을 때를 살펴본다. 이에 관한 알고리즘은 알고리즘2에 나타나있다.

알고리즘 2. 요청 메시지에 대한 응답을 받았을 때

```
Begin
 if(Response == OK) then begin
   rCount++;
   if( rCount >= qCount ) then begin
     gState = (R, LOCKED);
     send noticeCS(G, m, R) to all
     member of group;
     /* enter the critical section
     */
     /* receive Release(G, m, R)
     from all member of group */
     if receive Release(G, m, R)
     from all member of group then
     begin
       gState = (R, UNLOCKED);
       send UnLockCS(G, m, R) to
       all member of the quorum
       group
       end /* end of receive */
   end /*end of rCount >= qCount)
 end /* end of Response == OK */
 else
     wait until receive response
 end /* end of else */
end
```

변수:

- rCount: 쿼럼의 멤버로부터 받은 응답메시지의 개수이다.
- qCount: 쿼럼에 속한 멤버의 수이다.

유비쿼터스 환경에서 그룹은 환경의 변화에 따라 동적으로 변하는데, 그룹에 속한 멤버가 그룹에서 탈퇴하거나 새로운 멤버가 그룹에 참여할 수 있다. 만일 쿼럼으로 선정된 그룹의 멤버가 탈퇴하는 경우에는 자신이 받은 요청메시지가 있는지 확인하고 그에 대한 응답메시지를 보낸 후에 탈퇴해야 한다. 그리고 요청메시지를 보낸 멤버는 요청메시지를 보낸 직후에 그룹 저장소에서 쿼럼에 속한 멤버의 수를 알아와 qCount에 저장하고 받은 응답메시지의 개수인 rCount와 비교를 하게 된다. qCount와 rCount가 같거나 rCount가 크다면 쿼럼에 속한 모든 멤버로부터 응답을 받은 것이므로 CS에 들어갈 수 있게 된다. 이제 이 멤버는 그룹의 모든 멤버들에게 CS에 들어갈 수 있음을 알리고 이 메시지를 받은 다른 멤버들은 즉시 CS에 들어간다. 그리고 CS에서 나올 때는 CS에 들어갈 수 있음을 알렸던 멤버에게 Release()메시지를 보낸다. Release 메시지를 다른 모든 멤버로부터 전송 받았다면 이제 쿼럼에게 UnLockCS메시지를 보냄으로써 CS의 사용을 마친다.

다음으로 요청메시지를 받는 멤버, 즉 쿼럼의 멤버가

동작하는 과정을 살펴보도록 한다.

2) 응답 프로시저:

알고리즘 3. 요청 메시지를 받았을 때

```
Begin
 Receive RequestCS from one member
 of other group;
 if( (R, gState) == UNLOCKED ) then
 begin
    send OK(R);
    (R, gstate) == LOCKED;
 end /* end of (R, gState) ==
 UNLOCKED */
 else
    store the RequestCS to message
    queue;
 end
end
```

쿼럼으로 선택된 그룹의 모든 멤버들이 요청메시지를 받으면 먼저 요청메시지에 나타난 리소스의 상태를 살핀다. 만약 그 리소스가 어느 누구에 의해서도 사용되지 않고 있다면(즉, UNLOCKED 상태라면) OK메시지를 보내고 그 리소스의 상태를 LOCKED로 바꿔놓는다.

알고리즘 4. UnLockCS 메시지를 받았을 때

```
Begin
 Receive unlock message from one
 member of other group;
 (R, gState) = UNLOCKED;
 if there are some message about
 resource R in message queue then
 response;
end
```

UnLockCS메시지를 받으면 해당 리소스에 대한 그룹의 상태를 UNLOCKED로 바꾸어 놓는다. 그리고 메시지 큐를 살펴보아 해당 리소스에 대한 요청메시지가 있다면 앞에서 설명하였던 과정으로 처리한다.

4. 성능평가 및 분석

이번 장에서는 제안한 알고리즘에 대해 실시한 성능평가에 대해 설명한다. 알고리즘의 성능평가를 위해 최대 20개의 그룹과 그룹들이 공유하는 리소스를 1개로 하여 시뮬레이션을 하였다. 그리고 각 그룹에 속한 멤버의 수는 10개로 고정되게 사용하였다. 각 그룹이 리소스에 대해 요청메시지를 보내는 시점은 랜덤으로 설정하였다. 성능평가를 하기 위해 사용한 기준은 다음과 같다.

- 대기시간(waiting time): 한 멤버가 그룹을 대표하여 쿼럼의 모든 멤버에게 요청메시지를 보낸 시간부터 응답을 받기까지의 시간이다.
- 메시지 수: 그룹에 속한 멤버가 CS에 들어가기를 요청한 순간부터 CS에 들어갈 때까지 보내는 메시지 전송 횟수이다.

시뮬레이션 프로그램은 자바로 작성되었으며 이들 메시지의 전송 방법은 일반적인 TCP/IP방식을 사용하였다. 한 멤버가 그룹의 모든 멤버에게 메시지를 전송하는 경우에는 멀티캐스트 방식으로 전송하였으며, 그룹의 멤버들의 통신은 로컬에서 이루어졌다. 리소스에 접근하기 원하는 그룹이 리소스를 획득한 시점(LOCK)부터 리소스 사용을 마친 시점(UNLOCK)까지의 시간은 1ms로 고정하였다. 이는 리소스 사용시간으로 인한 대기시간의 증가를 막기 위함이다. 그림3과 표1에 시뮬레이션 결과가 나타나있다.

시뮬레이션은 그룹의 개수를 3개, 5개, 7개, 10개 그리고 20개로 하여 테스트를 했으며 그룹의 개수 별로 각 4회씩 테스트를 하였다. 아래 그림3에서 볼 수 있듯이 그룹의 개수가 많아질수록 대기시간이 증가하는 것을 볼 수 있다.

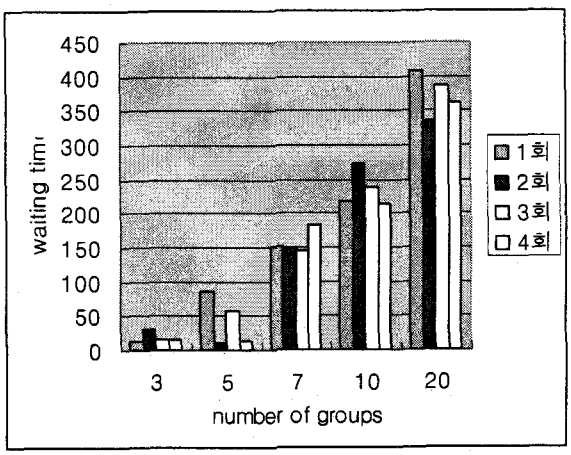

그림3. 그룹의 수에 따른 waiting time(단위: ms). 각 그룹의 수마다 4회씩 실험하였다.

아래의 표1에는 위에서 측정한 대기시간의 평균을 구한 것이 나타나있다. 그리고 한 그룹이 CS에 들어가기 위해 주고받는 메시지의 수를 측정하였다. 표1에 나타난 q는 쿼럼으로 선정된 그룹의 멤버 수, 즉 쿼럼의 멤버 수를 나타낸다. 한 멤버가 그룹내의 모든 멤버에게 메시지를 보낼 때 멀티캐스트를 사용한다고 가정하여 표1과 같은 결과가 나왔다. 즉, 한 멤버가 쿼럼에 속한 모든 멤버에게 요청메시지를 멀티캐스트로 보내는 횟수 1번과 쿼럼의 모든 멤버에게 받는 응답 메시지의 수(q)와 자신이 속한 그룹에게 CS의 사용을 알리는 메시지를 멀티캐스트로 보내는 횟수1번을 더한 것이다. 메시지의 전송회수는 그룹 수가 증가하는 것에 상관없이 동일하다. 만약에 동일한 조건에서 쿼럼-기반 알고리즘방식을 사용한다면 메시지 전송횟수는 그룹의 멤버 수 + 쿼럼의 멤버 수(q)가 된다. 이는 쿼럼-기반알고리즘에서는 CS에 들어가기 위해서 그룹의 멤버들이 개별적으로 쿼

럼에 요청메시지를 보내기 때문이다. 그룹을 대표하여 한 멤버만 쿼럼에 요청메시지를 보내고 그에 대해 받은 응답메시지를 자신이 속한 그룹의 멤버들에게 알리는 방식을 취함으로써 CS에 들어가기 위해 필요한 메시지의 전송횟수를 감소시켰다.

표 1. 제안된 알고리즘의 성능측정. (q = 쿼럼의 멤버 수)

그룹수	3	5	7	10	20
평균 대기시간(ms)	18	42	158	235	373
메시지의 수(한 멤버당)	2+q				

5. 결론

지금까지 유비쿼터스 환경에서 생성된 그룹간에 상호 배제문제를 해결하기 위한 알고리즘을 살펴보았다. 제안한 알고리즘은 그룹 상호 배제를 위한 쿼럼-기반 알고리즘[3]을 바탕으로 하여 유비쿼터스 환경의 특성에 맞도록 응용하였다.

먼저, 쿼럼 시스템을 구축하지 않더라도 그룹 자체가 쿼럼의 역할을 할 수 있도록 하였다. 그리고 동적으로 변하는 그룹의 특성이 적용되도록 그룹에서 한 멤버가 쿼럼에 요청메시지를 보내고 그에 대한 응답을 받으면 자신이 속한 그룹의 모든 멤버에게 알리는 형식을 취하였다. 쿼럼-기반알고리즘에서는 CS에 들어가기 위해서 그룹의 멤버들이 개별적으로 쿼럼에 요청메시지를 보낸다. 하지만 본 논문에서 제안한 방식은 그룹을 대표하여 한 멤버만 쿼럼에 요청메시지를 보내고 그에 대해 받은 응답메시지를 자신이 속한 그룹의 멤버들에게 알리는 방식을 취하였다. 그렇게 하여 쿼럼으로 메시지가 많이 몰리는 것을 방지하였고 CS에 들어가기 위해 필요한 메시지의 전송횟수를 감소시켰다.

6. 참고문헌

1. Y.-J.Joung: Asynchronous group mutual exclusion. Distributed Computing, Vol.13, No.4 (2000) 189-206
2. Andrew S.Tanenbaum, Maarten van Steen: Distributed System. Principles and Paradigms
3. Y.-J.Joung: Quorum-based algorithm for group mutual exclusion, IEEE Trans. On Parallel and Distributed Systems, Vol.14, No.5 (2003) 463-476
4. K.-P.Wu, Y.-J.Joung: Asynchronous group mutual exclusion in ring networks, IEEE Proceedings, Computers and Digital Techniques, Vol.147 (2000), 1-8
5. Jaehyrk Park, Sukkyu Gang, Kwangjo Kim: Group Mutual Exclusion based Secure Distributed Protocol, Proc. Of CSS2003, (2003), 445-450
6. Vassos Hadzilacos: A note on group mutual exclusion, In 20th ACM SIGACTSIGOPS Symposium on Principles of Distributed Computing, (2001), 100-106
7. Yoshifumi Manabe, Jaehyrk Park: A quorum-based extended group mutual exclusion algorithm without unnecessary blocking, Proc. of Parallel and Distributed Systems, Tenth International Conference (ICPADS'04), Vol.00, (2004) 341-348