분산 시스템의 코디네이터 선출 알고리즘 성능 비교

김기[1], 최은미[2]
[1]한동대학교 정보통신공학과
[2]국민대학교 비즈니스 IT 학부
bart@seed.handong.edu, emchoi@kookmin.ac.kr

Performance Comparison of Coordinator Election Algorithms in a Distributed System

Ki Kim[1], Eunmi Choi[2]
[1]Department of IT, Handong Global University
[2]School of Business IT, Kookmin University

요 약

분산 시스템에서의 대표적인 코디네이터 선출 알고리즘으로 Bully 와 Invitation 알고리즘이 존재한다. 본 논문에서는 기존의 코디네이터 선출 알고리즘의 단점을 보안하여, fail이 존재하는 네트워크상에서 사용 가능한 안정성 있는 효과적인 코디네이터 선출 알고리즘을 제안하고, 실제 분산 시스템상의 실험을 통하여 기존의 코디네이터 알고리즘들과의 선출과 합병에 걸리는 시간을 측정하고 비교한다.

1. 서론

분산 시스템에서 구성된 노드들간의 전역 정보나 상태 정보 등을 이용한 동기화된 작업을 하기 위하여, 노드 중에 코디네이터(Coordinator)를 두어 하부 작업을 다른 노드에 할당하거나 다른 노드들을 관리하는 등의 관리자 일을 담당한다[1, 2]. 시스템 장애나 통신 상의 장애로 인한 코디네이터의 부재 시 나머지 노드들이 동적으로 코디네이터를 선출하는 알고리즘(Election Algorithm)을 통하여, 다른 가능성이 높은 노드를 코디네이터로 선출하여 시스템의 상태와 기능을 복귀시키게 된다[3]. 분산 시스템에서의 대표적인 코디네이터 선출 알고리즘으로는 Bully Algorithm[4], Invitation Algorithm[4], Ring Algorithm[5]과 같은 알고리즘이 존재한다. 본 논문에서는 노드 시스템과 네트워크 장애의 실제 상황을 고려하여 만든 우리의 알고리즘[6]을 제안하고, 다른 알고리즘들과의 성능 비교를 해보도록 한다.

본 논문의 구성은, 다음 장에서 기존 알고리즘에 대한 설명을 하며, 3장에서는 제안하는 알고리즘에 대한 소개와 비교를 한다. 4장에서는 여러 가지 상황에 대한 성능을 각각의 알고리즘에 따른 비교를 하고 5장에서 결론을 맺는다.

2. 기존의 코디네이터 선출 알고리즘

2.1. Bully 알고리즘

Bully 알고리즘에서는 노드들에게 정하여진 우선순위가 있다. 높은 우선순위의 노드가 코디네이터의 역할을 담당하며, fail 시 다음 우선순위를 가진 노드가 코디네이터의 역할을 수행하게 된다. 즉, 기존 코디네이터의 fail 시 나머지 노드들이 각각 자신보다 높은 우선순위의 노드의 생존을 확인한다. 만일 자신보다 우선 순위가 높은 노드가 살아있다면 순복을 하고 선출됨을 포기한다. 한편, 자신이 그 중 가장 높은 우선순위를 가졌다면, 그 노드가 코디네이터가 되어 다른 낮은 우선순위를 가진 노드들에게 자신이 코디네이터가 되었다고 확인시키는 과정으로 이루

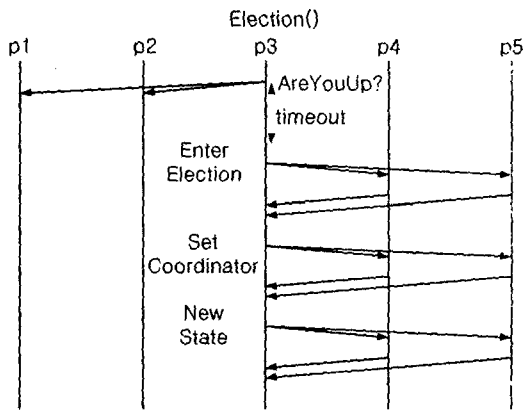

그림 1 Bully-코디네이터 선출 알고리즘

어진다. 선출과정은 그림1과 같다. 또한 일반 노드가 시스템 fail 상태에서 recovery 상태로 전이 될 경우나 새로운 노드가 그룹에 추가되는 경우 다시 선출 알고리즘을 실행하여 그 그룹에 포함되게 하였다.

2.2. Bully 수정 알고리즘

Bully 알고리즘에서 고려하지 못한 네트워크 장애, 즉 Hub의 장애 등의 통신 분리 경우는 여러 코디네이터들이 발생을 하게 된다. 우리는 Bully 수정 버전으로, 한 코디네이터가 자신이 아닌 다른 코디네이터를 발견하는 경우 Bully의 선출 알고리즘을 재수행하여 한 코디네이터만을 선출하게 하였다.

2.3. Invitation 알고리즘

Invitation 알고리즘은 네트워크 장애를 고려하여 고안되었으므로 다수의 코디네이터들의 존재를 허용한다. 각 코디네이터는 고유의 그룹을 가지고 서로 간의 초청(invitation)과 합병(merge)을 통해서 다른 그룹들을 계속적으로 합병시켜서 결국에는 하나의 코디네이터와 하나의 그룹을 만든다. 이 합병(merge) 알고리즘은 그림2에서 보인다. 이 알고리즘에서는 코디네이터의 fail 발견 시 해당 그룹의 노드들은 모두 코디네이터가 되어 각자 자신의 고유한 그룹을 만들고 초청과 합병을 시작하게 된다. 또한 일반 노드가 fail 상태에서 recovery 상태로 전이되는 경우나 새 노드가 추가되는 경우 역시 해당 노드는 자신이 코디네이터인 고유의 그룹을 만들게 되고 초청과 합병을 시작하게 된다.

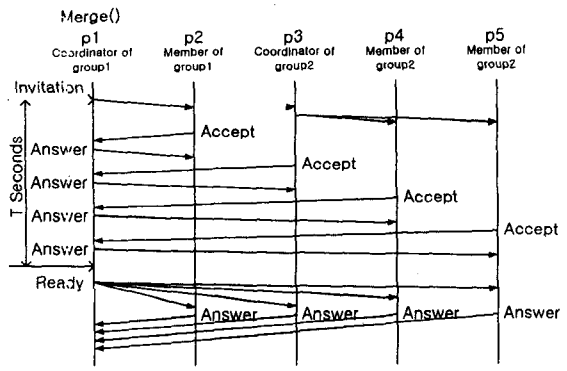

그림 2 Invitation-그룹 합병 알고리즘

3. 새로운 코디네이터 선출 알고리즘

3.1. 기존 알고리즘의 문제점

Bully 알고리즘은 신뢰성 있는 네트워크를 바탕으로 하고 있으며, 선출 과정이 단순하다는 장점을 가지고 있으나 fail 노드의 recovery와 복수의 코디네이터의 존재 등의 상황을 처리하기 위해서 불필요한 선출과정을 거치게 된다는 단점을 가지고 있다.

Invitation 알고리즘은 네트워크 분리로 인한 복수의 코디네이터와 그룹의 존재를 허용하는 장점이 있으나, 기존의 코디네이터가 fail된 경우 그룹 구성원 각각이 새로운 코디네이터와 그룹을 생성함으로 불필요한 다수의 그룹이 생성되어 병합에 걸리는 시간이 항상 일정 시간(그림2에서 T)이상이 소요된다는 단점을 가지고 있다.

3.2. 코디네이터 선출 알고리즘 Pseudo Code

새로 제안하는 알고리즘에서 선출 과정은 Bully와 동일한 알고리즘을 사용한다. 즉, 모든 노드들은 일련의 번호가 있으며, 높은 우선순위의 노드가 fail이면 다음으로 높은 우선순위를 가진 노드가 코디네이터의 역할을 수행하게 된다.

이외에도, 네트워크의 분리 등의 문제로 인하여 복수의 코디네이터와 그룹이 존재하는 경우, 하나로 합병하는 알고리즘과 fail 상태에서 회복한 노드를 다시 그룹에 포함시켜 새로운 코디네이터에게 복속되도록 하는 합병 알고리즘을 추가 시켰다. 여기서는 서로가 알고 있는 코디네이터의 ID를 비교하여 합병의 주도자가 결정된다. 다른 방법으로는 서로 가지고 있는 구성원의 개수를 사용 할 수 있다. 이 알고리즘은 [6]에서 소개되었으며 그림3은 이를 도식화한다.

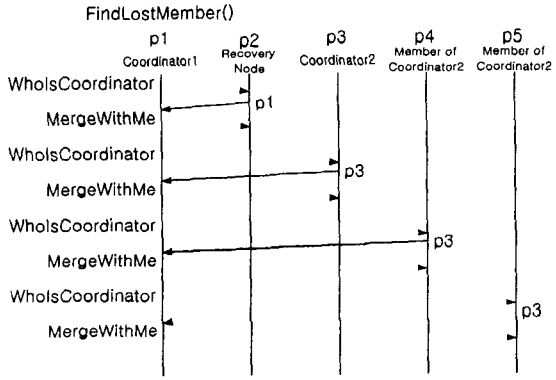

그림 3 새로운 알고리즘-Recovery 노드, 그룹 합병 알고리즘

3.3. 기존 알고리즘과의 비교

정상적인 상황에서의 선출 알고리즘은 Bully와 같은 방법을 사용하고 있지만, recovery 노드를 그룹에 포함시키거나 다수의 코디네이터의 발견 시 다시 선출 알고리즘을 거치지 않고 그룹 합병 알고리즘을 적용하므로, Bully 와는 달리 항상 우선순위가 높은 노드가 코디네이터가 되지 않는다. 이때 낮은 우선순위의 코디네이터가 높은 우선순위의 recovery 노드를 자신의 그룹에 영입할 수 있다.

Invitation 알고리즘에서의 불필요한 다수의 그룹

과 코디네이터의 발생은 Bully의 선출 알고리즘을 적용시키므로 개선하였다. 또한, 다른 그룹과의 병합의 경우는, 우리의 알고리즘에서는 멤버에 대한 정보를 이용하여 직접적으로 개별 노드와 통신을 하기 때문에 Invitation 알고리즘의 경우와 같이 어떤 노드에서 Accept 메시지를 받을지 기다리는 소요 시간(T)이 없어지게 되어 노드들간에 즉각적인 조치를 취할 수 있다.

4. 실험결과

동종의 노드들로 구성된 분산 시스템 상에서, (1) 코디네이터 fail시 선출 알고리즘을 실행하여 새 코디네이터를 선출하기까지 소요시간, (2) 네트워크 분리로 인해 여러 개로 나뉜 그룹들이 하나로 합쳐지기까지의 소요시간, (3) fail된 노드가 다시 recovery 되어 그룹에 포함되는 시간을 측정 비교 하였다.

그림4에서와 같이 새 코디네이터 선출 소요시간이 Bully와 Invitation 알고리즘보다 우리의 알고리즘이 좋은 성능의 결과를 보였고, 노드 개수 증가에 비하여 선출 시간의 overhead 가 줄어든 것을 볼 수 있다.

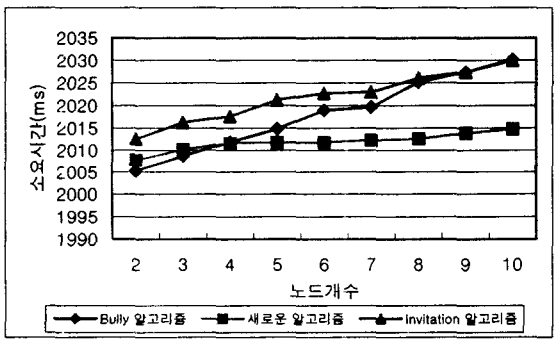

그림 4 새로운 코디네이터 선출 시간

네트워크 분리로 인해 생긴 여러 그룹 병합시간은 그림 5에서 나타난다. Invitation 알고리즘은 일정 시간(T=1초) 이상을 소요하며, 이에 반해 Bully와 우리의 알고리즘은 비슷하게 좋은 결과를 보였다.

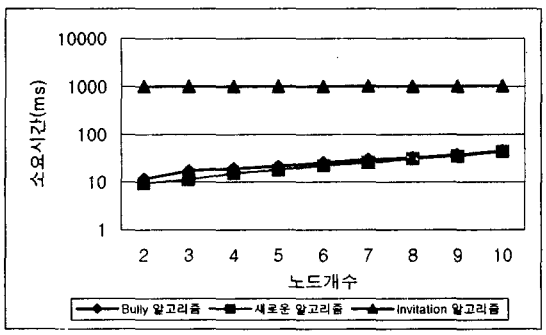

그림 5 네트워크 분리로 인해 생긴 여러 그룹 병합시간

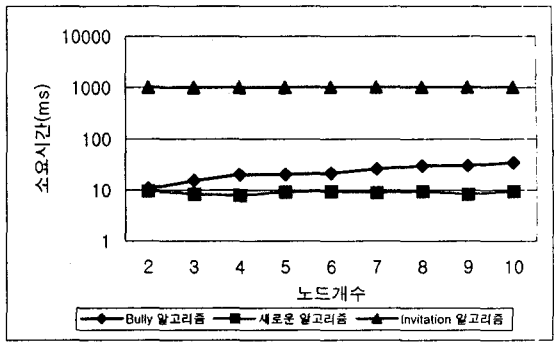

그림 6 Recovery 노드 포함시간

그림6의 Recovery 노드 포함시간의 측정 결과에서, Invitation 알고리즘은 T=1초 이상의 시간을 소요하고 있으며, Bully는 노드의 개수에 따라 소요시간이 상대적으로 증가하였다. 우리 알고리즘에서는 fail된 멤버 목록을 관리하여, Recovery 노드를 발견하였을 경우 다시 선출 알고리즘을 실행하지 않고 단순히 그룹에 포함시키므로 노드의 개수 증가에도 불구하고 일정한 소요 시간을 얻었다.

5. 결론

본 논문에서는 기존의 코디네이터 선출 알고리즘에서 신뢰성 있는 네트워크를 기반으로 한 Bully 알고리즘과 문제가 존재하는 네트워크에서 사용 가능한 Invitation 알고리즘의 장점을 결합하고 단점을 개선하여 보다 안정성이 있는 효과적인 코디네이터 선출 알고리즘을 제안했으며 실험을 통해 다른 알고리즘들과의 성능 비교와 이상 상황에서의 대처 능력을 비교하여 제안하는 알고리즘이 새로운 코디네이터 선출시간과 네트워크 분리로 인해 생긴 그룹 병합시간에서는 Bully와 비슷한 소요시간을, Recovery 노드 포함시간에서 다른 알고리즘에 비해 상당한 좋은 성능의 결과를 보였다.

참고 문헌

[1] Andrew S. Tanenbarum, Maarten van Steen, "Distributed Systems principles and Paradigms", Prentice Hall, 2002

[2] George Coulouris, Jean Dollimore, Tim Kindberg, "Distributed Systems Concepts and Design", Addison Wesley, 2001

[3] Randy Chow, Theodore Johnson, "Distributed Operating Systems & Algorithms", Addison Wesley, 1997

[4] H. Garcia-Molina, "Elections in a distributed computer system", IEEE Trans. on Computers, C-31(2):48-59, 1982

[5] Chang, E.G. and Roberts, R. "An improved algorithm for decentralized extrema-finding in circular configurations of processors.", Comms. ACM, Vol. 22, No.5, pp.281-3, 1979

[6] 김기, 이동훈, 최은미, "분산 시스템에서의 효과적인 코디네이터 선출 알고리즘", 한국정보과학회 추계학술대회, pp. 382-4, 2003