결함 포용적인 이동에이전트 수행을 위한 지역기반 단계군 구성기법*

최성진[0] 백맹순 안진호 김차영 황종선
고려대학교 컴퓨터학과 분산시스템 연구실
{lotieye[0], msbak, jhahn, chayoung, hwang}@disys.korea.ac.kr

A Stage Construction Scheme based on a Region for Fault-tolerant Execution of Mobile Agent

Sung-Jin Choi[0], Maeng-Soon Baik, Jin-Ho Ahn, Cha-Young Kim, Chong-Sun Hwang
Distributed System Lab. Dept. of Computer Science & Engineering, Korea University

요 약

신뢰성 높은 이동 에이전트 시스템을 구성하는데 있어서 지속적인 이동 에이전트 연산을 보장하는 결함 포용기법은 중요한 고려사항이다. 이를 위해 많은 연구들이 단계군 구성에 기반한 이동 에이전트 수행에 대한 결함 포용 기법들을 제안하고 있다. 그러나 제안된 기법들은 단계군을 구성함으로써 에이전트 연산 실행에 대한 봉쇄 가능성을 감소시켰으나, 에이전트를 이주시키는 통신비용과 단계군 작업들에 대한 부하를 증가시켰다. 본 논문에서는 단계군내에 지역(region)적으로 다르게 분포한 실행장소(place)에 대해서 가짜 참여자(quasi-participant)를 두어 지역적으로 같은 곳에 모이게 하는 새로운 단계군 구성기법을 제안한다. 또한, 가짜 참여자와 실제 실행장소를 하나의 단계군으로 구성하기 위해 단계군내에 하위단계군(substage)을 두어 단계군을 구성하는 기법을 제안한다. 하위단계군은 가짜 참여자와 실제 실행장소의 작업을 분리하여 가짜 참여자로 인해 발생할 수 있는 추가 통신비용을 감소시킨다. 본 논문에서 제안하는 가짜 참여자와 하위단계군을 사용한 단계군 구성기법은 고장 자유(failure-free) 상태에서 단계군 작업들에 대한 수행시간을 단축시켜 단계군을 이용하여 결함을 포용하는 이동 에이전트의 전체 수행시간을 단축시킨다.

1. 서 론

이동 에이전트(mobile agent)는 이질적인 네트워크 환경에서 노드들을 옮겨 다니면서 사용자를 대신하여 자율적으로 주어진 일을 처리하는 프로그램이다[6]. 이동 에이전트 수행 중 통신 고장이나 노드고장이 일어나면 이동 에이전트의 연산은 부분적으로 혹은 전체적으로 손실되거나 수행이 봉쇄되는 결과를 초래한다. 따라서, 이동 에이전트 시스템(mobile agent system)을 개발하는데 있어서 결함 포용(fault-tolerance)은 기본적이면서도 중요한 고려사항이다[3].

이동 에이전트 환경에서 결함포용 기법은 대부분 이동 에이전트 실행장소의 복제를 이용한다. 복제를 기반한 결함 포용기법은 시간적 복제 기법(Temporal Replication Based Approach: TRBA)과 공간적 복제기법(Spatial Replication Based Approach: SRBA)으로 분류된다[3]. TRBA는 이전 단계가 에이전트를 저장하여 현재 단계가 고장이 나더라도 이전 단계에서 미리 저장되어 있는 에이전트를 다시 보내어 연산을 수행하는 기법이다[3]. SRBA는 한 단계에 단계군이라는 여러 실행장소를 두어 결함 포용을 해결하는 기법으로 만약 현재 작업자(worker)가 고장나면 관찰자(observer)들 중에서 새로운 작업자를 선출하여 연산의 지속성을 보장한다[2,3,4].

복제를 이용하는 기법 중 SRBA의 가장 큰 단점은 다음 단계군으로 이동 에이전트를 이주시킬 때 발생하는 추가 통신비용과 단계군 작업에 대한 부하이다. 추가 통신비용을 해결하기 위해 기존의 연구들은 이동 에이전트가 이주할 때 단계군들 간에 전달하는 메시지 수나 데이터 양을 줄이기 위한 기법을 제시하고 있지만, 단계군 작업으로 인하여 추가로 발생하는 수행시간에 대한 문제는 해결하지 못하고 있다.

본 논문에서는 SRBA가 기반하고 있는 단계군에 가짜 참여자(quasi-participant)를 두어 지역적으로 흩어져 있는 실행장소를 같은 곳에 분포시키는 새로운 단계군 구성기법을 제안한다. 또한, 단계군내에 하위단계군을 두어 가짜 참여자와 실제 실행장소들간의 작업을 분리하여 가짜 참여자로 인해 발생할 수 있는 추가 통신비용을 감소시키는 기법을 제안한다. 본 논문에서 제안하는 가짜 참여자와 하위단계군을 사용한 단계군 구성기법은 고장 자유 상태에서 단계군 작업들에 대한 수행시간을 단축시켜 단계군을 이용하여 결함을 포용하는 이동 에이전트의 전체 수행시간을 단축시킨다.

* 본 연구는 한국과학재단 목적기초연구(R01-2002-000-00235-0)지원으로 수행되었음.

2. 관련 연구

결함 포용을 위해 단계군을 구성하는 연구 중 Straßer와 Rothermel[1]은 단계군을 구성하는 실행장소들을 정규(regular)와 예외(exception) 노드로 나누어서 우선순위에 따라 단계군을 구성하는 기법을 제안했다. Pleisch와 Schiper[2]는 단계군을 구성하는 실행장소들을 같은 종류의 실행장소(iso-place), 다른 종류의 실행장소(hetero-place), 방관자 (hetro-place with witness)로 나누어서 단계군을 구성하는 기법을 제안했다. 비록 이들 연구들이 단계군들 간에 전달하는 메시지 수나 데이터 양을 줄이기 위한 기법으로 연속된 단계군들의 실행장소들이 가능한 많이 교차되도록 단계군을 구성하는 기법[1]이나 파이프라인 모델(pipelined model)[2]을 제안하고 있지만, 이동 에이전트의 수행시간과 관련된 단계군 작업, 특히 2PC를 사용하는 투표(voting), 동의(agreement) 프로토콜의 통신비용에 대한 부하를 줄이기 위한 기법은 제안하지 않고 있다. 또한, 지역적으로 분산되어 있는 노드들에 대한 단계군 구성기법도 제안하지 않고 있다.

3. 시스템 모델

본 논문이 가정하는 이동 에이전트 시스템은 이동 에이전트 a_i, 실행장소 p_i로 구성된다. 에이전트 a_i는 이동계획(itinerary)에 따라 노드와 노드사이를 옮겨다니며 일을 수행한다. 이때 노드에서는 에이전트가 수행할 수 있는 실행장소 p_i를 제공한다[6]. 이 실행장소는 에이전트에게 특정 서비스를 제공한다. 한 노드에 여러 실행장소가 있을 수 있다.

이런 에이전트 시스템들 중에서 같은 권한을 가진 에이전트 시스템의 집합을 지역 R_i 이라고 한다[6]. 본 논문에서는 이동 에이전트가 여러 지역으로 구성된 환경에서 옮겨다니면서 작업을 수행하는 걸로 가정한다.

또한, 본 논문에서 가정하는 고장에는 노드 고장, 에이전트 시스템 고장, 실행장소 고장, 에이전트 고장이 있다. 이런 고장은 고장 정지(failure-stop) 모델에 따라 고장이 일어나도 다른 구성요소에 어떠한 영향을 끼치지 않는다.

4. 지역에 기반한 단계군 구성기법

4.1 가짜 참여자를 이용한 단계군 구성기법

이동 에이전트가 각 노드에서 수행해야할 작업을 단계(step)라 하고 고장에 대비하여 각 단계를 여러 노드(또는 실행장소)로 구성한 것을 단계군 S_i라 한다[2,4,5]. 단계군을 구성할 때

이동 에이전트의 이동계획과 단계군을 구성하는 실행장소의 수에 따라 다른 단계군이 구성된다[1]. 이때 단계군을 이루는 실행장소는 서로 같은 지역(region)에 있을 수도 있고 서로 다른 지역에 있을 수도 있다. 즉, 다른 지역에 있으나 같은 서비스를 제공하는 실행장소들이 하나의 단계군으로 구성되거나, 다른 지역에 있으면서 서로 다른 서비스를 제공하는 실행장소들이 하나의 단계군으로 구성될 수도 있다. 그러나 다른 지역으로 구성된 단계군은 같은 지역에서 구성된 단계군에 비해 높은 통신비용이 발생한다는 문제점을 가지고 있다. 따라서 지역에 따라 단계군을 구성하는 기법이 필요하다.

지역에 따라 단계군을 구성하기 위해 본 논문에서는 가짜 참여자라는 새로운 개념을 사용한다. 가짜 참여자는 지역적으로 다르게 분포되어 있는 실제 실행장소를 대신하는 실행장소이다. 만약 단계군의 작업자가 고장나면 가짜 참여자는 임무를 종료하고 자기가 대신한 실제 실행장소에게 임무를 되돌리게 된다. 가짜 참여자는 단순히 실제 실행장소를 대신만 할 뿐 실제로 해당 서비스를 제공하지 않기 때문에 에이전트 코드와 실행 상태를 받지 않는다. 그래서 가짜 참여자를 두어도 에이전트 이주비용은 추가로 들지 않는다. 이런 가짜 참여자를 둠으로 해서 단계군 작업, 특히 2PC와 함께 동작하는 투표 프로토콜의 수행시간을 감소시킨다.

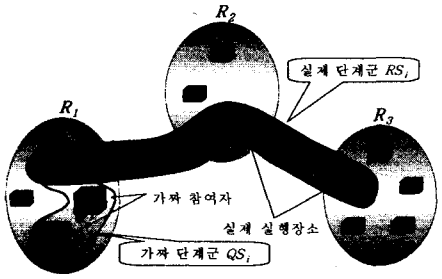

그림 1. 가짜 참여자로 구성된 단계군

그림 1은 실제 실행장소의 지역에 따라 가짜 참여자를 두어 단계군을 구성하는 예이다. 제안된 기법은 다른 지역에 분포하는 실행장소들(p_i^o와p_i^l)을 현재 작업자(p_i^w)가 있는 R_1로 모으기 위해 가짜 참여자(qp_i^o, qp_i^l)를 둔다. 따라서 실제 단계군 QS_i에 대하여 p_i^w, qp_i^o, qp_i^l 들로 이루어진 가짜 단계군 QS_i가 구성된다.

그림 2는 실행장소의 지역적 위치에 따라 실제 단계군에 대한 가짜 단계군을 구성하는 기법이다. S_i는 단계군 i를 구성하는 실행장소의 집합이며 $|S_i|$는 단계군 i를 구성하는 데 필요한 실행장소의 개수이다. N_i는 이동계획에 따라 단계군 i에 구성될 수 있는 모든 실행장소의 집합이다. p_i^w는 S_i의 작업자를 가리킨다. R_w는 작업자가 속한 지역에 위치한 실행장소들의 집합이다. P_i는 N_i에서 우선순위에 따라 선택된 실행장소의 집합이다. DP_i, SP_i는 각각 다른 지역, 같은 지역에 위치한 실행장소의 집합이다. QP_i는 DP_i에 대한 가짜 참여자 집합이다. PP_i는 $(R_w - P_i - QP_i)$ 집합에서 관리자의 정책에 따라 선택된 실행장소 집합이다. makeQuasiplace()는 같은 지역에 속한 실행장소 중에서 이전에 수행된 실행장소나 다음에 수행될 실행장소를 선택하거나, 이들이 없는 경우 같은 지역에 속한 임의의 실행장소를 선택하여 가짜 참여자를 만드는 함수이다. makeStage()는 단계군을 만드는 함수이다.

```
Do
QPi = ∅;
PPi = ∅;
Pi = choosebyPriority( Ni );
DPi = Pi - Rw ;
SPi = Pi - DPi ;
if ( Pi ⊈ Rw ) then
   QPi = makeQuasiplace( DPi );
fi;
if ( |Si| > |Ni| ) then
   PPi = choosebyPolicy( (Rw - Pi - QPi) );
fi;
Si = makeStage( QPi , SPi , PPi , piw);
oD;
```

그림 2. 실제 단계군에 대한 가짜 단계군 구성기법

4.2 하위단계군을 이용한 단계군 구성기법

가짜 참여자는 실제 실행장소를 대신하여 단계군 작업에 참여하나 실제 실행장소의 서비스를 제공할 수 없다. 따라서 가짜 참여자와 실제 실행장소의 단계군 작업은 분리된다. 그러나 가짜 참여자와 실제 실행장소는 한 단계에서 실행될 작업이므로 하나의 단계군으로 구성되어야 한다. 본 논문에서는 이런 성질을 만족시키기 위해 단계군 안에 있는 또다른 단계군, 즉 하위단계군이라는 새로운 개념을 사용한다.

단계군 S_i 내의 하위단계군 SS_i는 다음과 같은 특징을 갖는다.

- $SS_i \subseteq S_i$ and $p_i^w \in SS_i$
- $|SS_i| \leq |S_i|$

SS_i 는 가짜 하위단계군 QSS_i와 실제 하위단계군 RSS_i로 나뉜다.

- QSS_i if $\forall m,\ qp_i^m \in SS_i$
- RSS_i if $\forall m,\ qp_i^m \notin SS_i$

여기서 m은 $|QP_i|$이고 qp_i는 단계군 S_i에서 가짜 참여자를 말한다. RSS_i 내의 관찰자는 모든 단계군 작업에 완전히 참여하지만 QSS_i 내의 관찰자는 부분적으로 참여한다. 즉, QSS_i의 관찰자는 모니터링 과정에는 참여하지 않고 선출과정에는 부분적으로 참여한다. 하위단계군을 구성하는 실행장소는 동적으로 추가되거나 삭제될 수 있다. 만약 SS_i의 작업자가 고장이 나면 SS_i의 구성요소는 바뀌게 된다.

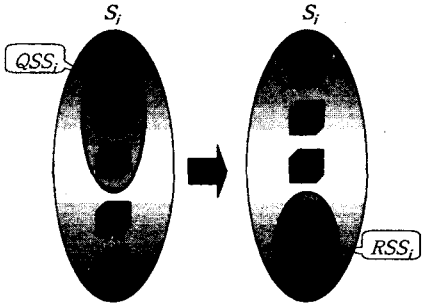

그림 3. 가짜 참여자와 하위단계군으로 구성된 단계군

그림 3은 가짜 참여자와 하위단계군을 두어 단계군을 구성한 예이다. S_i는 $p_i^w, qp_i^o, qp_i^l, p_i^o, p_i^l$로 구성되어 있다. S_i내에 p_i^w, qp_i^o, p_i^l를 QSS_i로 구성하고 만약 p_i^w가 고장이 나면 p_i^o p_i^l 중 우선순위가 높은 p_i^o를 새로운 작업자 p_i^{nw}로 선출한 다음 새롭게 RSS_i를 구성한다.

그림 4는 작업자가 고장나지 않은 경우와 고장난 경우 가짜 참여자와 하위단계군을 이용한 단계군 구성 기법이다. 여기서 k는 QSS_i 한계값(threshold)으로 단계군 구성 개수가 k개 이상이라면 QSS_i를 구성할 수 있다. makeSubStage()는 단계군내에 하위단계군을 만드는 함수이다.

```
Case 1: 작업자가 고장나지 않은 경우
Do
QPi = ∅;
PPi = ∅;
Pi = choosebyPriority( Ni );
DPi = Pi - Rw ;
SPi = Pi - DPi ;
if (( Pi ⊈ Rw ) & (k < |Si|)) then
   QPi = makeQuasiplace( DPi );
fi;
if ( |Si| > |Ni| ) then
   PPi = choosebyPolicy( (Rw - Pi - QPi) );
fi;
SSi = makeSubStage( QPi , SPi , PPi , piw );
if( DPi == ∅ ) then
   RSSi = SSi ;
   Si = makeStage( RSSi , piw );
else
   QSSi = SSi ;
   Si = makeStage( QSSi , DPi , piw );
fi;
oD;
```

그림 4. 가짜참여자와 하위단계군을 이용한 단계군 구성기법

```
Case 2: 작업자가 고장난 경우
Do
QPi = Ø;
PPi = Ø;
DPi = Pi - Rw ;
SPi = Pi - DPi ;
FPi = DPi + SPi ;
pi^nw = selectNewLeader( FPi );
if (k < | FPi | ) then
   if ( FPi ⊈ Rw ) then
      QPi = makeQuasiplace( DPi );
      SSi = makeSubStage( QPi , SPi , pi^nw );
   else
      SSi = makeSubStage( SPi , pi^nw );
   fi;
else
   SSi = makeSubStage( FPi , pi^nw );
fi;
if( DPi == Ø ) then
   RSSi = SSi ;
   Si = makeStage( RSSi , pi^nw );
else
   QSSi = SSi ;
   Si = makeStage( QSSi, DPi, pi^nw );
fi;
oD;
```

그림 4. 가짜참여자와 하위단계군을 이용한 단계군 구성기법 (계속)

4.3 수정된 단계군 작업 프로토콜

가짜 참여자와 하위단계군을 사용하여 단계군을 구성할 때 발생하는 추가 통신비용을 줄이고, 가짜 참여자가 없을 때 프로토콜에서 지원했던 연산의 정확히 한번 수행(exactly-once) 성질이나 봉쇄 없이 작업이 수행되는 성질을 유지하기 위해 단계군 작업 프로토콜은 수정되어야 한다. 다음은 가짜 참여자와 하위단계군을 둔 단계군 작업 프로토콜, 즉 모니터링, 선출, 투표 프로토콜의 수정사항이다.

· 모니터링 과정 : p_i^w는 $(DP_i + SP_i)$에게 메시지(alive message)를 보낸다. 하지만 QP_i에게는 보내지 않는다. 왜냐하면 QP_i는 DP_i를 대신할 뿐 실제 서비스를 제공하지 않기 때문이다. 모니터링 메시지 비용은 기존방식과 같다.

· 선출 과정 : $(DP_i + SP_i)$가 p_i^w 의 고장을 탐지하면 이들은 선출과정을 거쳐 p_i^{nw}를 선출한다. 이 때 p_i^{nw} 는 QP_i에게 메시지(destory message)를 보내게 된다. 이 메시지는 qp_i가 더 이상 투표과정에 참여할 필요가 없다는 것을 가리킨다. 이 때 qp_i 로부터 이 메시지에 대한 확인 응답이 올 때까지 p_i^{nw}는 작업을 시작하지 않는다. 왜냐하면 통신 고장에 의해 발생할 수 있는 중복 에이전트(duplicate agent)의 발생을 방지하기 위함이다. 즉, 수정된 프로토콜은 에이전트의 정확히 한번 수행 성질을 보장한다.

· 투표 과정 : SS_i 만 투표 과정에 참여한다. 동의 프로토콜에서도 이와 같이 SS_i 만 동의 과정에 참여한다.

5. 성능 평가

DP_i가 단계군 작업 프로토콜을 실행하는 실행시간(메시지 전송속도)을 $T_1, T_2, \cdots, T_n$이라고 가정한다. 여기서 n은 $|DP_i|$를 의미한다. 또한, QP_i의 실행시간을 $QT_1, QT_2, \cdots, QT_n$라 하자. 여기서 각 실행장소들이 처리하는 메시지 비용이 같다고 가정하면 DP_i와 p_i^w로 이루어진 단계군의 총 실행시간 TT와 QP_i와 p_i^w로 이루어진 단계군의 총 실행시간 TQT는 다음과 같다.

$$\left.\begin{aligned} TT &= m \times \sum_{i=1}^{n} T_i + \sum_{i=1}^{\lceil n/2 \rceil} T_i \\ TQT &= m \times \sum_{i=1}^{n} QT_i + \sum_{i=1}^{\lceil n/2 \rceil} QT_i \end{aligned}\right\} \quad m = \text{메시지전송개수} - 1$$

위 식은 찬성 투표가 과반수 되는 경우의 수 ${}_nC_r$ (여기서 $r = \lceil n/2 \rceil$) 중에서 실행장소 1부터 $\lceil n/2 \rceil$까지 최종 메시지를 보냈을 때를 나타낸다. 가짜 참여자를 사용하여 단계군을 구성할 때 성능상 이익 P는 다음과 같다.

$$P = (TT - TQT) \times l$$

여기서 l은 이동계획에서 가짜 참여자로 구성된 단계군의 수이다. 10~1000Mbps의 대역폭을 갖는 LAN에서 지연시간(latency)은 보통 1~10ms를 갖고 0.101~600Mpbs의 대역폭을 갖는 WAN이나 0.010~2Mbps 대역폭을 갖는 인터넷 환경에서의 지연시간은 100~500ms를 갖는다[7]. 그림 5는 이런 환경에서 지역적 위치를 고려하지 않은 단계군 구성기법과 제안된 단계군 구성기법들간의 단계군 작업(특히 투표과정)시에 발생하는 총 지연시간을 비교한 것이다. 한 단계군당 관찰자의 수에 따라 약 1000~6000ms 지연시간 차이가 있음을 알 수 있다.

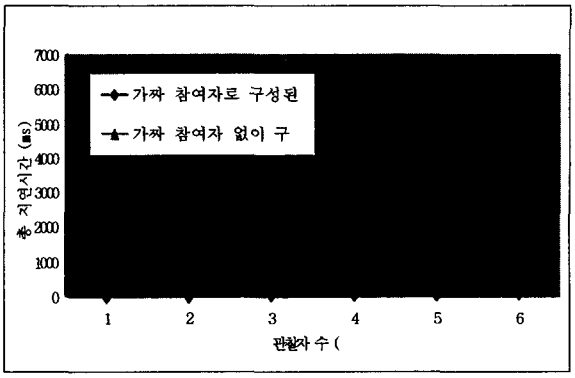

그림 5. 가짜 참여자로 구성된 단계군과 가짜 참여자 없이 구성된 단계군간의 지연시간 비교

6. 결론 및 향후 연구과제

본 논문은 SRBA가 기반하고 있는 단계군 구성기법에서 발생하는 단계군 작업들에 대한 부하를 줄이기 위해 실행장소들의 지역적 위치를 고려한 단계군 구성기법에 대해 제안했다. 제안된 기법은 지역적으로 다르게 분포한 실행장소를 대신하는 가짜 참여자를 두어 지역적으로 같은 곳에 분포시켜 이들 실행장소간에 일어나는 단계군 작업, 특히 투표와 동의 프로토콜의 수행시간을 줄였다. 또한, 단계군내에 하위단계군을 두어 가짜 참여자와 실제 실행장소를 하나의 단계군으로 구성하였고, 가짜 참여자와 실제 실행장소의 단계군 작업을 분리하여 추가 통신비용을 감소시켰다. 이렇게 가짜 참여자와 하위단계군으로 구성된 단계군은 고장 자유 상태에서 기존의 기법보다 단계군 작업의 수행시간을 단축시킨다.

앞으로 제안된 단계군 구성 기법을 기존의 이동 에이전트 시스템에 구현하여 실제 고장이 일어난 상황에서 지역적 위치를 고려한 단계군과 고려하지 않은 단계군 사이에 성능평가가 이루어질 것이다.

참고문헌

[1] M. Straßer and K. Rothermel, "Reliability Concepts for Mobile Agents", *International Journal of Cooperative Information Systems (IJCIS)*, Vol 7, No 4, pp. 355-382, 1998.

[2] S. Pleisch and A. Schiper, "Modeling fault-tolerant mobile agent execution as a sequence of agreement problems", In *Proc. of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pp. 11~20, 2000.

[3] S. Pleisch and A. Schiper, "Approaches to Fault-Tolerant Mobile Agent Executions", IBM Research Report 3333, 2001.

[4] K. Rothermel and M. Strasser, "A fault-tolerant protocol for providing the exactly-once property of mobile agents" In *Proc. of the 17th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pp. 100~108, 1998.

[5] F. Schneider, "Towards fault-tolerant and secure agentry", Invited paper, In *Proceedings of the 11th International Workshop on Distributed Algorithms*, 1997.

[6] Object Management Group, "Mobile agent system interoperability facilities specification", OMG TC Document orbos/97-10-05, 1997.

[7] G. Coulouris, J. Dollimore and T. Kindberg, "Distributed Systems Concepts and Design", Addison-Wesley, third edition, 2001.