효율적 이동 에이전트를 위한 객체 다중 복제 기법

김광종*, 최은실, 이연식
군산대학교 컴퓨터정보과학과
e-mail : kkjkim@kunsan.ac.kr

Multi-Replication of Objects for the Efficient Mobile Agent

Kwangjong Kim, Eunsil Choi, Yonsik Lee
Dept. of Computer Information Science, Kunsan National University

요 약

컴퓨터와 통신기술이 결합된 분산 컴퓨팅 환경에서 새로운 패러다임인 이동 에이전트는 통신망 점유와 과부하를 효율적으로 감소시켜 분산 시스템의 문제점들을 해결하고 있다. 그러나 이동 에이전트의 이동성 강조에 따른 지능성 결여와 다수의 이동 에이전트 시스템들로 이주시 발생하는 지연시간 문제, 및 호스트의 결점이나 장애 등으로 인하여 이동 에이전트가 무한 대기상태나 고아상태에 빠져 쓰레기로 처리될 수 있는 문제점들을 해결하기 위한 새로운 대안이 요구된다. 따라서 본 논문에서는 이동 에이전트 객체를 다중 복제하여 복제된 다수의 에이전트 객체를 여러 이동 에이전트 시스템으로 분산시킴으로써 이주시 발생하는 지연시간을 제거하고, 쓰레기로 처리될 수 있는 이동 에이전트들의 수를 줄임으로써 사용자에게 보다 빠르고 정확한 정보를 서비스할 수 있게 한다.

1. 서론

컴퓨터와 통신기술이 결합된 분산 컴퓨팅 환경이 출현하게 됨으로써 분산환경에 기초한 새로운 작업들과 서비스들이 등장하게 되었는데 그 예로 정보검색, 전자상거래, 원격교육, 이동컴퓨팅 등을 들 수 있다. 이에 따라 컴퓨터와 사용자 간에 보다 편리한 새로운 방식의 인터페이스에 대한 요구가 증가하게 되었다. 이러한 새로운 인터페이스에서는 사용자가 문제해결을 위한 구체적인 절차나 방법을 직접 기술하는 종래의 방식과는 달리 원하는 하나의 작업 목표만을 명시하면 프로그램 스스로 작업수행을 위한 세부계획과 절차를 세워 실행한 뒤 결과를 사용자의 요구에 맞게 제시해 줄 수 있다. 이와 같이 새로운 인터페이스를 제공할 수 있는 자율적 소프트웨어를 일반적으로 에이전트라 부른다. 최근 들어 다양한 분야에 걸쳐 이러한 응용 에이전트들이 개발되고 있는 실정인데, 특히 스스로 분산 네트워크 상의 여러 호스트들을 옮겨다니며 작업할 수 있는 이동 에이전트에 대한 관심이 높아져 가고 있다. 그러나 기존의 이동 에이전트 시스템들은 에이전트의 특성 중 이동성(mobility)만을 강조하여 지능성(intelligence)이 결여되어 있고, 이동 에이전트가 통신망에 연동된 다수의 이동 에이전트 시스템들로 이주할 때 발생하는 지연시간 문제와 호스트의 결점이나 장애 등으로 인하여 이동 에이전트가 무한 대기상태나 고아(orphan)상태에 빠져 쓰레기로 처리 될 수 있는 문제점들을 가지고 있다[1,2]. 따라서 이러한 이동 에이전트의 지능성 및 노드 이동에 따른 문제점들의 새

로운 대안이 요구된다. 본 논문에서는 다수의 이동 에이전트 시스템에 분산되어 있는 구현 객체가 네이밍 에이전트에 등록되어 유지 관리되므로 사용자의 요구사항에 대한 검색 키워드를 네이밍 에이전트의 메타데이터에서 일치하는 키워드의 구현 객체 수와 위치를 확인한 후 객체 참조자 정보를 이동 에이전트 클라이언트에 넘겨주게 되면 이동 에이전트 객체를 다중 복제하여 복제된 다수의 에이전트 객체를 여러 이동 에이전트 시스템으로 분산시킴으로써 이주시 지연시간을 제거하고, 쓰레기로 처리될 수 있는 이동 에이전트들의 수를 줄임으로써 사용자에게 보다 빠르고 정확한 정보를 서비스할 수 있게 한다.

본 논문의 구성은 2 장에서 기존 이동 에이전트의 수행 특성과 본 논문에서 제안하는 이동 에이전트의 구조를 제시하고, 3 장에서는 이동 에이전트 객체의 복제 방법과 알고리즘을 제안한다. 4 장에서는 CORBA를 기반으로 복제된 이동 에이전트의 통신 수행 과정을 보이고 5 장에서 결론 및 향후 연구방향에 대해서 기술한다. 본 논문은 객체 분산 환경에 OMG(Object Management Group)에 의해 표준화된 CORBA(Common Object Request Broker Architecture) 기반으로 연구하였다.

2. 이동 에이전트 수행 특성 및 제안 시스템 구조

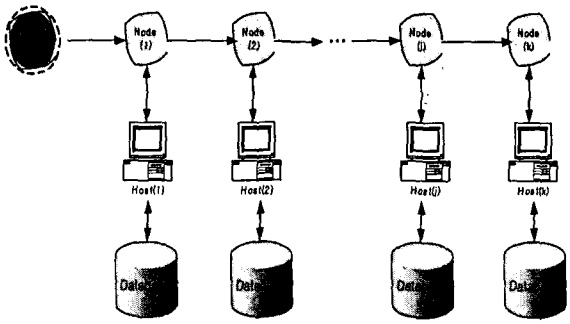
2.1 기존 이동 에이전트

기존의 이동 에이전트 시스템들은 이동 에이전트의 이주를 지원하기 위해 이상적인 조건 즉, 시스템이나 노드에서 결점이 없을 경우를 가정하고 있거나, 혹은 적절한 프로토콜을 제공하고 있다[8,9]. 그러나 만약 이동 에이전트가 적절한 라우팅 기법에 따라 해당 노드나 호스트들로 이동하고 있을 때, 이동 에이전트가 방문하고 처리하여야 할 임의의 노드가 고장 또는 접근 불가능할 경우 이동 에이전트의 이주시 문제가 발생한다. 예를 들면, 노드 결정시 이동 에이전트는 해당 노드에 접근하기 위해 해당 메시지 큐에 무한정 대기 하거나, 이동 시스템이 있는 호스트가 장애나 쓰레기 수집을 수행하여 이동 에이전트를 파괴하는 경우가 발생 할 수 있다[9]. 또한 올바르게 이주되었다 할지라도, 이동 에이전트가 자율적으로 실행하는 과정에서 해당 호스트에서 지원해야 할 자원에 대한 서비스 인터페이스가 없다면, 해당 에이전트는 오류를 범할 수 있으며, 이는 이동의 신뢰성을 저해하는 결과를 가져와 심각한 결과를 초래한다. 따라서 이러한 문제는 이동 에이전트가 목적지 노드에서 출발하여 임무를 완수한 마지막 노드까지 생명주기에도 영향을 미친다.

전형적으로 ORB는 참조가 없는 객체를 제거하기 위해 분산 쓰레기 수집을 하며, Voyager는 이동 에이전트의 생명주기를 위한 5 가지의 정책을 제공한다[3]. 또한 Mole은 이동 에이전트에 대한 고아 찾기와 성공적인 종료를 위한 쉐도우 프로토콜을 제공하고 있다[4]. 그리고 단순 프로토콜로써 트랜잭션 메세지 큐를 이용하는 방법이 있다[8,9]. 이 방법은 메시지 전달자가 메시지 큐에 메시지를 놓고 수신자는 가져오는 방식으로서 프로세스들간의 비동기적 통신을 제공한다. 그러나 이 방법은 에이전트 처리과정을 감시하는 기능이 없어 자율적인 이동 에이전트가 실행시에는 문제가 생긴다[8,9].

2.2 기존 이동 에이전트의 수행 특성

자율성을 가진 이동 에이전트는 라우팅 스케쥴에 의한 이주를 [그림 1]과 같이 수행한다. [그림 1]은 이동 에이전트가 수행하는 과정을 순차적으로 나타낸 것이다. 여기서 이동 에이전트가 Node(1) → Node(2) → … → Node(j) → Node(k) 순서로 방문한다고 할 때, 임의의 Node(i)와 Node(j)간의 통신망 장애가 발생하였다면 Node(j)의 호스트 Host(j)가 동작 중 일지라도 이주가 불가능한 상태가 된다.



[그림 1] 이동 에이전트의 이주 경로

이러한 경우 자율적인 이동 에이전트는 목적지까지 도달하기가 불가능해지며, 사용자 지시를 이전 노드에서 한없이 기다려야만 한다. 또한 이동 에이전트가 경로상에 통신망 장애는 발생하지 않았지만 Host(i)로 이주하여 실행하려 할 때, 해당 시스템 Host(i)에 장애가 발생하여 더 이상 서비스 지원이 불가능할 때 이동 에이전트는 무한 대기 상태로 되거나, 파괴되어 올바른 목적을 달성할 수 없다. 이와 같이 이동 에이전트는 이주시에 노드의 결점으로 인해 무한 대기 상태 및 쓰레기로 취급될 수 있으며, 만약 이동 에이전트가 Node(i)에서 Node(j)사이의 평균 이주시간이 n+1 개에서 노드의 결점이 없을 경우라고 가정한다면, 목적지 노드로 되돌아오는 전체 에이전트 지연시간(Agent

Delay Time)은 다음의 식(1)과 같으며,

$$ADT_n = \sum_{k=1}^{n} N_k = N_1 + N_2 + ... + N_n \quad ---(1)$$

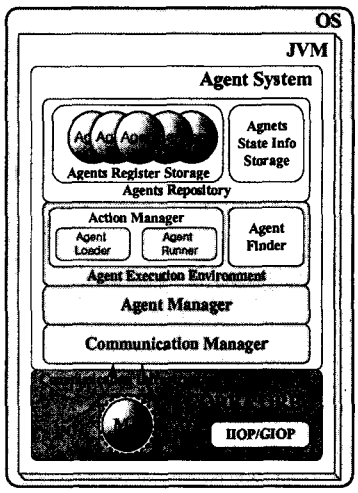
통신망 장애가 발생하거나 해당 호스트의 장애가 발생한다면 지연시간은 다음 식(2)와 같이 무한대로 빠지게 될 것이다.

$$ADT_n = \sum_{k=1}^{\infty} N_k = N_1 + N_2 + ... + N_n \quad ---(2)$$

또한 이동 에이전트는 사용자의 작업을 실행하기 위해서 실행 모듈을 가지고 이주해야만 하기 때문에 에이전트에 다양한 작업을 요구할 경우 실행 모듈이 커지게 되므로 이주시 네트웍 트래픽을 증가시키기도 한다. 따라서 본 논문에서는 이동 에이전트가 가지는 실행모듈의 증가, 통신망 장애, 서비스 지원의 부재 등으로 인한 지연시간의 증가, 무한 대기 상태 및 쓰레기로 처리 되어지는 이동 에이전트의 수를 줄이고, 이질적인 시스템 및 이기종 간의 어플리케이션 상에서도 상호간의 운용이 가능하고 객체의 투명성을 제공하는 CORBA 기반의 네이밍 에이전트를 통해 이동 에이전트 객체를 다중 복제하여 각각의 호스트에 분산시킴으로써 사용자의 질의에 대한 검색을 병렬적으로 수행하여 원하는 결과를 얻어 이동 에이전트의 오버헤드를 감소시킨다.

2.3 이동 에이전트의 구조

이동 에이전트 시스템은 에이전트의 이주나 주어진 작업 처리를 위한 리소스 제공 및 메시지 통신을 가능하게 하는 것으로[5,6,11], 다음 [그림 2]는 이동 에이전트 다중 복제 모델을 지원하는 에이전트 시스템의 기본 구조로써, 본 논문에서 제안한 이동 에이전트 모델 내의 객체의 다중 복제 작업을 수행하기 위한 기본 환경을 제공한다. 자바 가상 머신(JVM)은 통신망으로 전송된 자바 중간 코드인 바이트 스트림에 대한 원격 실행만 가능할 뿐 이동 자체를 지원하지 않음으로 에이전트의 이동성 지원을 위해 통신 관리자, 에이전트 관리자, 실행환경에서의 액션 관리자로 구성된 에이전트 이동 지원 모듈을 생성한다[12]. 통신 관리자는 에이전트의 출입이나 메시지의 송신/수신을 위한 접속 채널을 제공하고, 다수의 이동 에이전트들이 동시에 이동할 수 있도록 멀티캐스트와 다중 스레드를 지원한다. 에이전트 관리자는 전송되어온 에이전트나 기타 객체들의 안정성을 위해 사전에 에이전트 저장소에 등록작업을 수행한다.



[그림 2] 이동 에이전트 시스템의 기본 구조

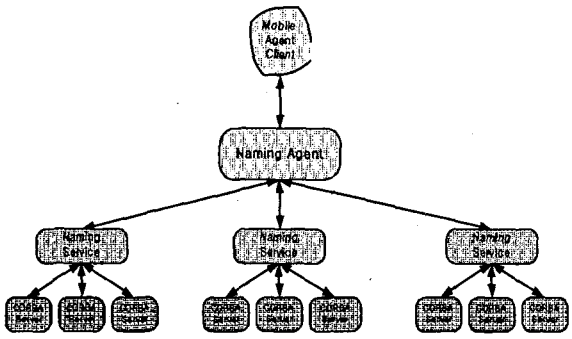
에이전트 실행 환경 모듈은 에이전트가 실제로 실행되는 논리적 장소로서, 바이트 스트림 형태의 에이전트를 자동실행 가능한 코드로 변환하여 메모리에 적재하는 에이전트 로더(loader)와 이전 상태 정보나 메시지 객체를 이용하여 멤버 필드에 대한 리인스턴스(re-instance)를 실행시키는 에이전트 러너(runner)로 구성된 액션 관리자와 외부로 이동된 에이전트의 위치를 관리하는 에이전트 파인더(finder)로 구성된다. 에이전트 저장소는 에이전트 등록 저장소와 상태 정보 저장소로 구성된다[12].

3. 이동 에이전트 객체 복제

3.1 이동 에이전트 객체의 복제 방법

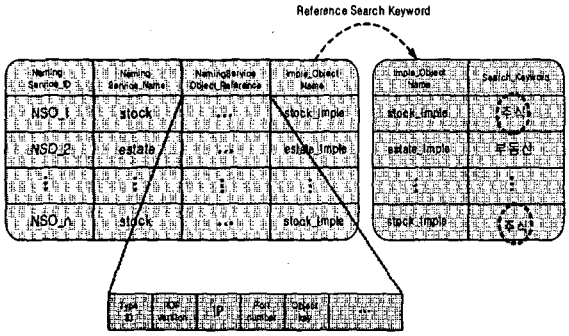
이동 에이전트의 객체의 다중 복제는 다수의 이동 에이전트 시스템들로 이주할 때 발생하는 지연시간이나, 호스트의 결점, 장애 등으로 인하여 이동 에이전트가 무한 대기상태이거나 고아(orphan)상태에 빠지는 에이전트의 수를 줄이고자 하는 것이다.

다음 [그림 3]은 네이밍 에이전트와 네이밍 서비스 관계를 나타낸 것이다. 네이밍 서비스는 네이밍 에이전트의 메타데이터로 유지 관리된다.

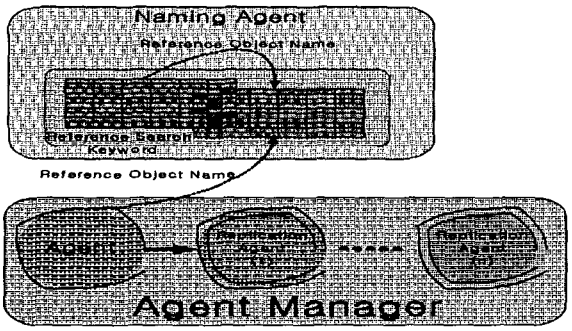


[그림 3] 네이밍 에이전트와 네이밍 서비스 관계

[그림 4]는 네이밍 에이전트의 메타데이터에서 키워드 검색시 동일한 키워드가 추출되는 것을 나타내며, [그림 5]는 네이밍 에이전트의 메타데이터로부터 구현 객체 참조자의 정보와 수를 얻어 사용자 질의에 일치하는 구현 객체와 이동 에이전트 객체 수를 일치시켜서 복제하는 것을 나타내고 있다.



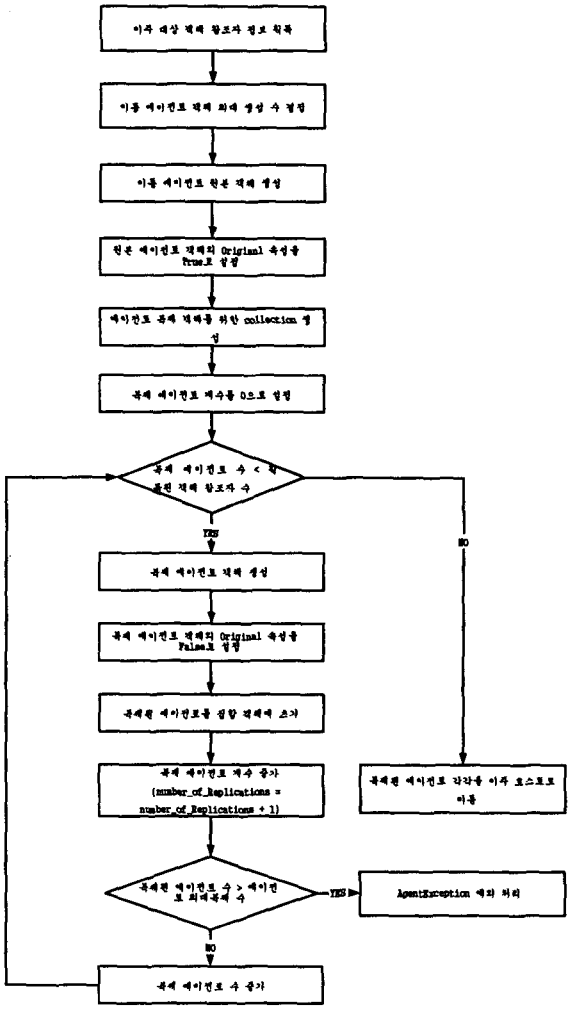
[그림 4] 네이밍 에이전트의 메타데이터 구조



[그림 5] 이동 에이전트 객체의 다중 복제

3.2 이동 에이전트 복제 알고리즘

이동 에이전트 객체 복제 알고리즘은 네이밍 에이전트에 등록되어 있는 구현 객체 참조자 수에 의해 결정되는데, 사용자 질의와 일치하는 구현 객체의 수와 에이전트 복제의 수를 일치시켜서 복제를 실행한다. 다음 [그림 6]은 이러한 복제과정의 흐름을 나타낸다.



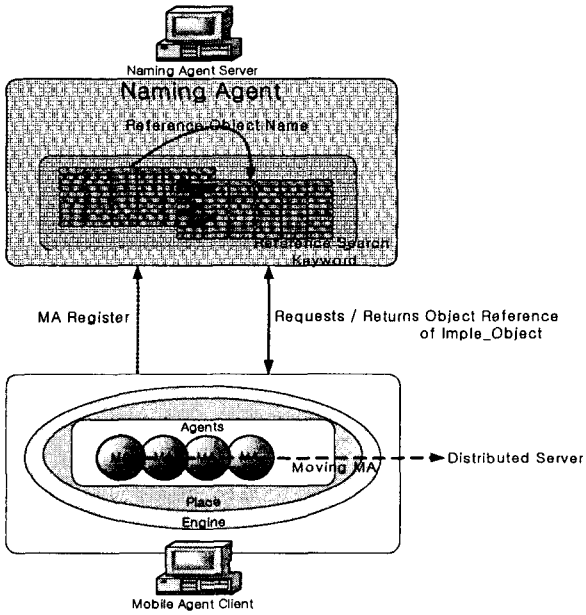
[그림 6] 에이전트 복제 과정

4. CORBA를 기반으로 복제된 이동 에이전트의 동작

다중 복제된 이동 에이전트 객체의 분산 모델은 CORBA를 기반으로 확장된 네이밍 에이전트와 상호 보완적 관계를 형성하여 서로 간에 협력함으로써 효율적이고 안정적인 정보 서비스를 지원하는 환경을 제공한다.

4.1 Mobile Agent Client와 Naming Agent간의 동작

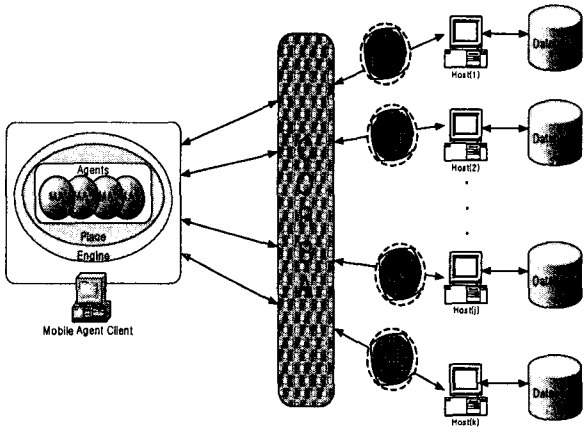
클라이언트 브라우저는 사용자 요구를 감지하여 네이밍 에이전트에 등록된 이동 에이전트 클라이언트의 이동 에이전트 위치를 확인하는 객체 참조자 정보를 얻은 후, 사용자의 키워드 정보를 이동 에이전트 클라이언트의 통신 관리자를 통해 에이전트 관리자에게 전달하고, 네이밍 에이전트의 메타데이터에서 사용자의 키워드와 일치하는 구현 객체의 수와 정보의 객체 참조자를 얻어와 에이전트 객체의 복제를 시작한다. 이러한 이동 에이전트 복제의 수는 네이밍 에이전트에 등록되어 있는 구현 객체 수에 의해 결정되는데, 사용자 질의와 일치하는 구현 객체와 에이전트 객체의 수를 일치시켜서 복제를 실행한다. [그림 7]은 네이밍 에이전트에 의해 유지 관리되는 객체 참조자의 정보를 요청/반환 받는 것을 나타내고 있다.



[그림 7] 네이밍 에이전트로부터 객체 참조자 정보의 요청/반환

4.2 Mobile Agent Client와 Multi Agent Server간의 동작

구현 객체의 수와 일치된 복제 이동 에이전트 객체는 분산되어 있는 이동 에이전트 서버에게 병렬적으로 이주되어 이동 에이전트 객체로부터 전달 받은 검색 키워드를 이용해 질의를 생성한 후, 이를 서버 어플리케이션에게 전달하여 데이터 검색을 수행한다. 이렇게 사용자에 의해 주어진 질의를 수행한 후, 검색의 결과를 사용자에게 전송하면 다수의 서버로부터 받은 중복된 정보를 필터링 과정을 거쳐서 사용자에게 제공한다. 그리고 만약 어떤 경로가 통신망의 장애 상태라든지, 호스트의 장애 등으로 인하여 서비스의 지원이 이루어지지 않는다면 그 에이전트는 Timestamp 이용하여 시간 제약에 따라 스스로 폐기 된다. [그림 8]은 구현 객체의 수에 맞게 복제된 이동 에이전트 객체의 분산을 나타낸 그림이다.



[그림 8] 복제된 이동 에이전트 객체의 분산

[그림 8]과 같이 이동 에이전트의 이주가 병렬적으로 이루어지기 때문에 노드 이주에 대한 지연시간은 제거되며 호스트의 장애나 통신망의 장애에 큰 영향을 받지 않으며, 실행 모듈 대신에 호출 모듈만이 이동하기 때문에 상대적으로 부하가 적다. 그러므로 사용자의 동일한 질의에 대해서 다수의 복제된 에이전트 객체가 다수의 서버에서 검색을 수행하므로 검색 성능과 시간적 측면에서 효율적이다.

5. 결론 및 향후 연구방향

본 논문에서는 분산 시스템의 새로운 패러다임인 이동 에이전트의 이동성 강조에 따른 지능성 결여와 다수의 이동 에이전트 시스템들로 이주시 발생하는 지연시간 문제, 혹은 호스트의 결점, 장애 등으로 인하여 이동 에이전트가 무한 대기상태이거나 고아상태에 빠져 쓰레기로 처리될 수 있기에 새로운 대안이 요구 되었다. 그러므로 이동 에이전트 객체를 다중 복제하여 복제된 다수의 에이전트 객체가 여러 이동 에이전트 시스템으로 분산되기 때문에 이주시 지연시간을 제거할 수 있고, 쓰레기로 처리될 수 있는 이동 에이전트들의 오버헤드를 줄이는 모델을 CORBA 를 기반으로 설계하였다. 따라서 사용자에게 보다 빠르고, 정확한 정보를 서비스할 수 있다.

향후 연구로는 이동 에이전트 클라이언트에서 1~n 개의 에이전트 객체를 복제하는데 있어서의 부하문제 및 메모리 낭비에 관한 부분과 설계한 이동 에이전트 객체의 다중 복제 기법 모델을 적용한 정보 검색 시스템의 개발이 요구되며, 기존의 이동 에이전트 시스템과의 정보를 검색하는 지연시간의 성능평가가 필요하다.

참고문헌

[1] K.A Bajarat, L. Cardelli, " Migratory Application" , Proc. of the 8th Annual ACM Symposium UIS Tech., Nov., 1995.

[2] J. Baumann, " A Protocol for Orphan Detection and Termination in Mobile Agent Systems" , TR-1997-09, Stuttgart Univ., July, 1997.

[3] ObjectSpace Voyager http://www.objectspace.comm, 2000.

[4] Bellavista, Antonio Corradi, Cesare Stefanelli, " A Mobile Agent Infrastructure for the Mobility Support" , Proc. of the 2000 ACM symposium, ACM Press, USA, pp.539-545, 2000.

[5] OMG, " Agent Technology Green Paper" , Agent Platform Special Interest Group, http://www.objs.com/agent/index.html, 2000.

[6] M.Hansson, " *Push Technology-The Next Big Thing?*" , " http:// www.tcm.hut.fi/Opinnot/Tik-10.350/Tehtavat/essays/push.html" , 1999.

[7] Vn Anh Pham and Ahmed Karmouch, " Mobile of software Agents: An Overview," IEEE Communication Magazine, pp.26-37, 1998.

[8] 전병국, 이근상, 최영국, " Java언어를 이용한 객체 이동 시스템 설계 및 구현" , 정보처리논문지, 6권, 1호, Jan., 1999.

[9] 전병국, 최영국, " 인트라넷상에서 자바 객체의 이동 시스템 설계 및 구현" , 정보과학회논문지(C), April, 1999.

[10] 이상호, " 웹과 CORBA 의 연동을 위한 Naming Agent" , 숭실대학교 대학원, 석사학위논문, 1998.

[11] 이승율, " CORBA 환경에서 이동 에이전트 시스템의 설계 및 구현" , 광운대학교 대학원, 석사학위논문, 1999.

[12] 김광종, 고현, 이연식," 분산 정보 서비스를 위한 CORBA 기반의 멀티 에이전트 모델 설계" ,정보처리학회춘계학술발표논문집, 2002.