네트워크 소요시간 최소화를 위한 이동 에이전트의 멀티캐스트 이주 모델 구현

김 광 종[†] · 고 현[††] · 김 영 자[†††] · 이 연 식[††††]

요 약

이동 에이전트는 호스트 간의 통신 횟수, 전송 데이터의 양, 에이전트의 크기, 네트워크 상태 등의 요소에 따라 매우 다양한 성능을 보이며 특히, 이주 방식은 분산 시스템의 전체 성능에 큰 영향을 준다. 대부분의 기존 이주 방식은 고정된 순서대로 호스트를 방문하고 방문한 호스트에서 작업을 수행한 후, 결과를 계속 누적시키면서 이주하는 단순 구조를 가지고 있다. 그러므로 호스트의 결점 및 장애, 서비스 부재 등과 같은 상황이나 방문해야할 호스트의 수가 많을 경우, 네트워크 소요시간이 증가되어 이동 에이전트를 사용하는 것 자체가 비효율적일 수 있다. 따라서 본 논문에서는 이러한 문제들을 해결하여 네트워크 소요시간을 최소화하기 위한 멀티캐스트 이주 모델을 설계 및 구현한다. 멀티캐스트 이주 모델은 분산된 서버의 위치 투명성 및 에이전트 객체 복제 정보를 제공하는 네이밍 에이전트와 호출 모듈만을 포함한 이동 에이전트 등과 같은 요소를 포함한다. 그리고 구현된 이주 모델을 검증하기 위해 프로토타입 시스템에 적용하여 기존 이주 방식과 비교 평가한다.

Implementation of Mobile Agent Multicast Migration Model for Minimizing Network Required Time

Kwang-jong Kim[†] · Hyun Ko[††] · Young Ja Kim[†††] · Yon-sik Lee[††††]

ABSTRACT

The mobile agent has very various performance according to the element of communication number of times between hosts, quantity of transmission data, agent's size, network state etc. specially, migration method is caused much effect in whole performance of distributed system. Most existing migration methods have simplicity structure that it moves doing to accumulate continuously result after achieving task by visiting host in the fixed order. Therefore, in case there are situation such as fault, obstacle, and service absence etc. This can be inefficient due to mobile agent increased network required time. In this paper, we design and implementation Multicast Migration Model for minimizing network required time by solving this problems. Multicast Migration Model includes components such as mobile agent including call module and naming agent, which provides object replication information and distributed server's location transparence. And we evaluate and compare with existing migration method applying prototype system to verify implemented migration model.

키워드 : 멀티캐스트(Multicast), 이동에이전트(Mobile Agent), 네이밍에이전트(Naming Agent)

1. 서 론

최근 분산 시스템은 분산 객체 컴퓨팅 기술로 인해 네트워크 상의 동종 및 이기종 시스템 간 분산처리 능력이 향상되었으나 여전히 데이터 전송 및 네트워크 지연과 대역폭의 오버헤드 문제를 안고 있다. 따라서 이러한 문제의 해결 대안으로 이동 에이전트가 주목 받고 있다[1, 2, 3, 4].

이동 에이전트는 네트워크로 연결된 독립적인 다수의 시스템을 하나의 유기적인 집합으로서의 기능을 수행하며, 현재 분산 시스템의 클라이언트/서버간의 종속적 관계 문제를 해결할 수 있는 새로운 분산처리 방안으로 고려되고 있다. 그러나 이동 에이전트는 호스트 간의 통신 횟수, 전송 데이터의 양, 에이전트의 크기, 네트워크 상태 등의 요소에 따라 매우 다양한 성능을 보이며 특히, 이주 방식은 분산 시스템의 전체 성능에 큰 영향을 준다[5, 6].

대부분 기존 이동 에이전트 시스템의 이주 방식은 이동 에이전트가 고정된 순서대로 호스트를 방문하고 방문한 호스트에서 작업을 수행한 후, 결과를 누적시키면서 이주하는 단순 구조를 가지고 있다. 이는 복잡하고 다양한 사용자 요구를 처리하는 에이전트의 작업 수행에 있어 이주 호스트를 재조정해야 하는 불가피한 상황이나 통신망 혹은, 호스트의 결점과 같은 특정한 문제 발생 시 동적으로 에이전트의 이주를 수행할 수 없음으로써 예견치 못한 여러 가지 요인들

※ 본 연구는 한국과학재단 특정기초연구(과제번호R01-2004-000-10946-0)지원으로 수행되었음
† 준 회 원 : 군산대학교 컴퓨터정보과학과 이학박사
†† 준 회 원 : 군산대학교 컴퓨터정보과학과 박사과정
††† 준 회 원 : 군산대학교 컴퓨터정보과학과 박사수료
†††† 종신회원 : 군산대학교 컴퓨터정보과학과 교수
논문접수 : 2005년 1월 7일, 심사완료 : 2005년 3월 8일

에 대해 능동적으로 대처할 수 없다. 또한, 호스트 결점이나 장애 및 서비스 부재 등으로 인해 발생하는 무한 대기상태 또는 고아(orphan)상태[7, 8]나 방문해야 할 호스트의 수나 결과의 양이 조금이라도 많을 경우, 네트워크 소요시간이 증가되어 이동 에이전트를 사용하는 것 자체가 비효율적일 수 있으며 작업 수행결과들과 실행 모듈이 함께 이주되므로 네트워크 트래픽 증가의 원인이 된다. 그러므로 기존 이주 방식이 아닌 새로운 이주 방식을 통해 에이전트 크기 증가 및 호스트의 결점이나 장애, 서비스 부재 등과 같은 상황에 능동적으로 대처하여 네트워크 소요시간을 최소화하기 위한 이주 모델이 요구된다.

본 논문에서는 분산 환경 하에서 기존 이주 방식에 따른 문제를 해결하고 네트워크 소요시간을 최소화하기 위한 방안의 멀티캐스트 이주 모델을 설계 및 구현한다. 멀티캐스트 이주 모델은 네이밍 에이전트(naming agent)의 메타 테이블에 등록된 분산된 서버 객체의 객체 참조자 정보를 이용하여 에이전트 객체를 복제함으로써 이루어진다. 복제된 에이전트는 네트워크 소요시간 감소를 위해 분산된 각 개별 호스트로 이주하여 사용자의 요청 작업을 수행한다. 따라서 이러한 이주 모델을 위해 에이전트 객체 복제를 위한 복제 메커니즘과 복제 정보를 제공하며, 능동적 작업 수행을 위해 분산된 호스트의 서버 객체 정보들을 관리하는 네이밍 에이전트, 실행 모듈과 결과 누적으로 인한 에이전트 크기 증가에 따른 네트워크 부하를 감소시키기 위해 호출 모듈만을 포함한 이동 에이전트를 구현하여 프로토타입 시스템에 적용하여 기존 이주 방식과 비교 평가한다.

본 논문의 구성은 다음과 같다. 2장 관련연구에서는 기존 이주 정책에 따른 이주 방식과 객체 복제 적용 기술에 대해 기술한다. 3장에서는 멀티캐스트 이주 모델 구조와 주요한 요소인 복제 정보를 제공하는 네이밍 에이전트 및 에이전트 객체 복제 메커니즘을 보이고 4장에서는 멀티캐스트 이주 모델과 기존 이주 방식을 프로토타입 시스템에 적용하여 성능을 평가한다. 마지막 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련연구

2.1 기존 이주 정책에 따른 이주 방식

기존 이동 에이전트 시스템들은 이주 지원을 위해 이상적인 조건 즉, 통신망이나 호스트의 결점이 없는 경우를 가정하고 있거나, 발생한 결점에 대처할 수 있는 적절한 프로토콜만을 제공하고 있다. 그러나 이동 에이전트가 라우팅 스케줄에 따라 해당 호스트로 이주하고 있을 때, 작업을 수행해야 할 임의의 호스트가 고장 또는 접근 불가능할 경우 이동 에이전트 이주에 관한 문제가 발생한다. 예를 들면, 통신망 결손이나 호스트 장애 또는 데이터베이스 등 정보 부재 시 이동 에이전트는 무한 대기나 고아 상태가 될 수 있으며 혹은, 파괴되어 쓰레기로 처리될 수 있다.

기존 연구에서는 이동 에이전트의 이주 시 발생할 수 있는 통신망 결손 및 호스트 장애에 대처하기 위해, JAMAS [6]의 경로 재조정과 후위 복구 방식을 제안하였고, [7]에서는 방문하는 호스트의 결점에 대처하기 위한 프로토콜을 제시하였다.

또한, Mole[7]은 이동 에이전트에 대한 고아 찾기와 성공적인 종료를 위해 그림자(shadow) 프로토콜을 제공하고 있으나, 이 프로토콜은 이동 에이전트가 이주 중에 결점이 발생한 경우에만 이를 찾아 처리할 뿐 에이전트의 이주 신뢰성은 보장하지 않는다. 무결점(fault-tolerance)을 지원하는 Mole은 전체 호스트에 이동 에이전트를 복사함으로써 투표(voting)와 선택(selection) 프로토콜을 이용한 효율적인 이주 기법을 제공하지만, 작업(worker) 호스트를 제외한 모든 관찰(observer) 호스트 간의 상호 감시와 통신 기능이 필요하며, 통신망 접속에 결점이 없는 것을 가정하고 있다.

그러나 대부분의 이동 에이전트 시스템들은 Aglets[8, 9, 10, 11]과 같이 이동 에이전트가 고정된 순서대로 호스트를 방문하고 방문한 호스트에서 작업 결과를 계속 누적시키면서 이주하는 단순 구조를 가지고 있다. 이는 복잡하고 다양한 사용자 요구를 처리하는 에이전트의 작업 수행에 있어 이주 호스트를 재조정해야 하는 불가피한 상황이나 통신망 혹은 호스트의 결점과 같은 특정한 문제 발생 시 동적으로 에이전트의 이주를 수행할 수 없음으로써 에이전트 이주 시 발생할 수 있는 여러 가지 요인들에 대해 능동적으로 대처할 수 없다. 그러므로 이러한 문제는 에이전트에게 부과된 작업의 실패 요인이 되고 작업시간을 증가시켜 이동 에이전트의 작업 대상인 전체 호스트에 대한 네트워크 소요시간이 증가하게 된다.

현재까지 연구된 대부분의 이주 정책과 관련한 이주 방식은 이동 에이전트가 통신망에 연결된 다수의 이동 에이전트 시스템들로 이주할 때 발생할 수 있는 통신망 혹은 호스트 결점에 대처할 수 있는 연구가 주류를 이루고 있으며[6, 9, 10, 11, 12, 13] 이주 방식은 순차적 수행 구조이므로 선행 문제들의 해결과 최소의 네트워크 소요시간을 갖도록 이동 에이전트의 이주 계획을 세우는 연구는 아직 미흡하다.

2.2 객체 복제 적용 기술

객체 복제에 대한 기술은 분산 환경에서 내부 호스트의 붕괴나 통신 링크 간의 단절이 예상되는 네트워크 상에 있는 객체들의 가용성을 높이기 위한 수단으로 그리드 컴퓨팅(Grid Computing), 분산 데이터베이스(Distributed Database), 그룹 통신(Group Communication) 등과 같은 여러 분야에서 사용되고 있다[14, 15, 16, 17].

객체 복제 기법을 사용하면 인터넷상에 존재하는 수많은 시스템들에 대하여 객체들을 분배하고 분배된 객체들의 동일성을 유지시켜 줄 수 있다. 또한, 이것은 로컬 객체에 대한 오퍼레이션만으로도 원격 객체에 적용이 가능하게 되어 효율적인 오퍼레이션을 가능하게 한다. 복제 기법을 구현한 객체 복제 시스템은 다중의 복사본(replica)들을 관리하고 중복된 객체들 간에 대하여 응용 개발자들에게 관리의 부담을 주지 않도록 하나의 이미지를 제공한다. 따라서 객체 복제

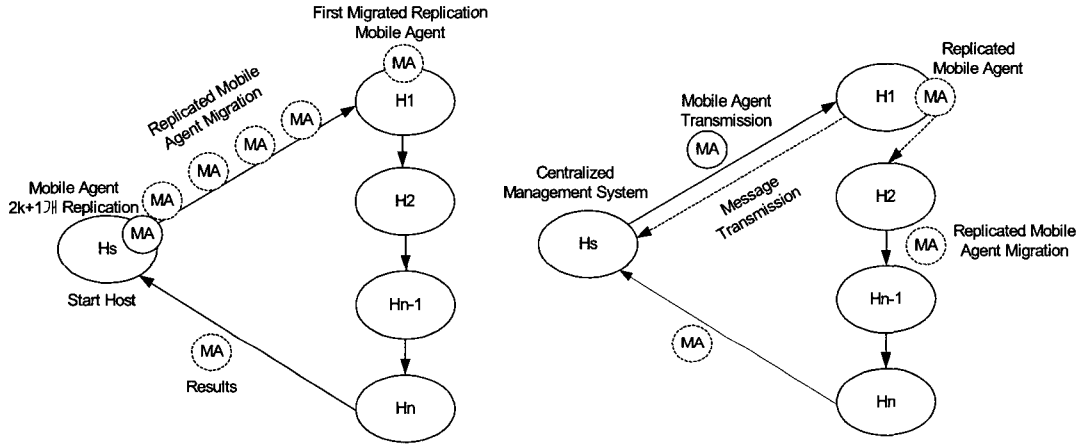
시스템의 가장 주된 기능은 객체에 적용된 모든 업데이트가 다른 분산 객체에 적용되고, 분배된 모든 객체도 동일성이 유지된다는 것이다.

그러나 최근 이러한 객체 복제에 대한 기술은 새로운 분산 패러다임인 이동 에이전트에 적용되어 이주 후, 수행 중인 에이전트가 시스템 오류로 인해 작업 수행이 불가능 상태일 경우, 이동 에이전트 복구를 위한 복제 기법[18, 19, 20], 중앙 집중형 관리 방식의 문제점을 보완하기 위한 복제 기법[21, 22], JAMAS[6]의 후위 복구 방식과 같이 이동 에이전트의 이주 신뢰성 보장에 관한 복제 기법 등에 사용되고 있다. (그림 1)은 이러한 기법들의 수행 구조를 나타낸 것이다.

(a)는 에이전트가 이주 수행 중 에이전트 오류, 플레이스 오류, 시스템의 오류로 손실되는 것을 방지하거나, 잘못된 오류 복구로 인한 에이전트의 중복 실행을 방지하기 위한 기법으로써 2k+1개의 에이전트를 복제하여 복제된 순서로 에이전트를 이주시키는 기법이다. 여기에서 k는 예기치 못한 k개의 오류를 말한다.

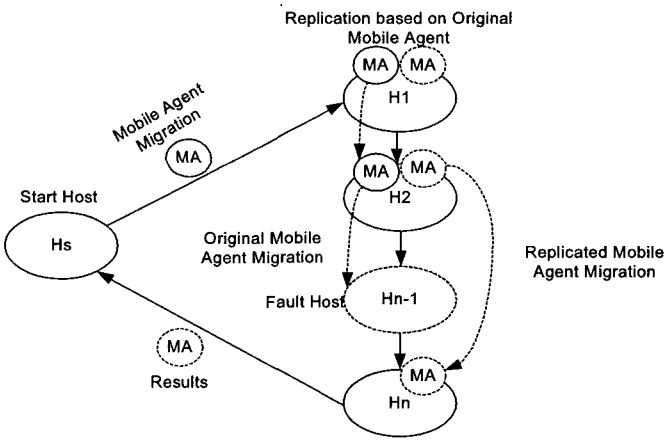
이러한 기법은 이동 에이전트를 k개의 오류에 견딜 수 있게 복제하여 제일 먼저 도착한 복제된 에이전트가 주 에이전트가 되어 작업을 수행하며 그 사이 나머지 복제 본들이 도착한다. 도착한 복제 본들은 주 에이전트의 이상 유무를 확인하고 오류가 발생되지 않았을 경우, 고정 에이전트의 동의를 얻어 다음 호스트로 이주하게 된다. 그러나 이 기법의 단점은 k개의 예기치 못한 오류를 예측하기 어렵다는 점과 에이전트의 크기가 크거나 네트워크 비용이 비싼 환경에서는 비효율적이라는 점이다.

(b)는 중앙 집중형 관리 방식의 관리 대상 호스트의 관리를 위한 기능이 관리자 호스트에 집중되어 확장성 및 관리 서비스의 동적인 변경이 어려운 문제가 있다. 그러므로 복제의 특성을 이용하여 관리 대상 호스트에서 에이전트의 작업이 지연될 경우 관리자 호스트로 새로운 에이전트를 파견하는 요청을 보내지 않고 작업 중인 에이전트의 복제 본을 생성하여 다음 관리 대상 호스트로 복제 본을 이주시켜 추가적인 메시지 전송 부하와 응답시간을 줄이는 기법이다. 그러나 에이전트의 복제로 인하여 발생되는 에이전트간의 통신 부하 및 관리자 호스트에서의 에이전트 제어에 관한



(a) 오류 손실로 인한 이동 에이전트 복구를 위한 복제 기법

(b) 중앙 집중형 관리 방식의 문제점을 보완하기 위한 복제 기법



(c) 이동 에이전트 이주 신뢰성 보장에 관한 복제 기법

(그림 1) 이동 에이전트 복제 기법

방법이 어렵다는 단점이 있다.

(c)는 이동 에이전트가 분산된 호스트로 이주 시 호스트의 결점이나 장애로 인해 에이전트가 무한대기 또는 고아상태가 되어 쓰레기로 처리되는 문제를 해결하려는 이동 에이전트의 이주 신뢰성 보장을 위한 기법이다. 이 기법은 원본 에이전트가 처음 호스트로 이주 시 원본 에이전트를 복제해 놓고 다음 호스트로 이주한다. 만약, 이주 후 호스트의 결점이나 장애가 발생되지 않았다면 이주 전 복제된 에이전트는 메시지를 통해 삭제 신호를 전달하여 삭제하게 된다. 하지만 이주된 원본 에이전트에게 응답이 오지 않을 경우, 해당 호스트 상에 문제가 발생한 것으로 간주하고 복제된 에이전트가 결점 및 장애가 발생된 호스트를 건너 뛰어 다음 호스트로 이주된다.

이 기법은 이동 에이전트의 신뢰성을 보장하긴 하지만 에이전트를 해당 호스트에 계속해서 복제해 놓고 이주하기 때문에 복제에 대한 복제시간과 이주 전 호스트와의 통신 문제가 발생한다. 그리고 무엇보다 중요한 이주 방식이 순차적 이주 수행 구조로써 에이전트의 크기 증가로 인한 네트워크 부하와 네트워크 거리만큼의 네트워크 소요시간이 증가한다는 점이다.

이와 같이 이동 에이전트와 관련된 복제 기법들은 이동 에이전트의 복구, 중앙 집중형 관리 방식, 이동 에이전트 이주 신뢰성에 관한 기법들이 연구되었다. 그러나 본 논문에서는 이러한 이동 에이전트의 복제 기법과는 다른 기존 이주 방식의 문제와 네트워크 소요시간을 최소화하기 위한 방안으로 새로운 이주 방식에 적용하는 다중 복제에 관한 것이다.

3. 이주 모델 설계

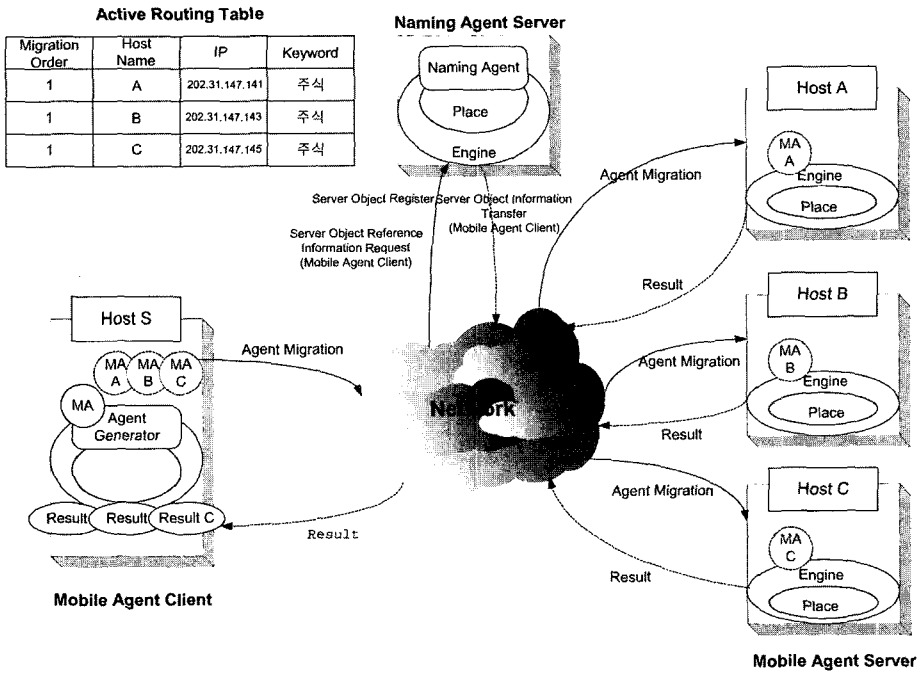
3.1 이주 모델의 구조

기존 이동 에이전트 시스템은 이동 에이전트의 순회 작업 처리를 위한 네트워크 소요시간과 처리 결과의 누적으로 인한 에이전트의 크기 증가가 전체 에이전트 시스템의 성능이나 효율성 향상 문제와 직결되어 있음을 알 수 있다. 즉, 이러한 문제는 에이전트의 구조나 이주 정책에 따른 이주 방법이 큰 영향을 미친다. 그러므로 이동 에이전트를 이용한 분산 시스템의 성능 향상을 위해서는 이동 에이전트를 어떻게 구성할 것인가와 어떠한 이주 방법을 통해 에이전트를 이주시킬 것인가가 매우 중요하다.

따라서 본 논문의 이주 모델은 네이밍 에이전트 서버내 서버 객체의 위치 정보를 관리하는 네이밍 에이전트에서 객체 참조자 정보를 얻고, 얻어진 객체 참조자 정보를 기반으로 에이전트 객체를 복제함으로써 이루어진다. 복제된 에이전트 객체들은 동일한 작업 수행을 위하여 각 호스트를 순차적으로 이주하지 않고, 개별적인 호스트로의 이주를 통하여 순차적 이주 시 발생되는 에이전트 크기 증가 문제와 네트워크 소요시간을 최소화한다. 이를 위해, 각 복제된 에이전트는 호출 모듈만을 포함하여 네트워크 트래픽을 감소시키며, 작업을 수행할 전체 호스트에 대한 능동적인 작업 수행이 이루어질 수 있도록 지원하며 네트워크 소요시간을 최소화한다. (그림 2)는 이주 모델의 구조이다.

3.1.1 이주 수행 과정

이주 수행을 위한 과정은 키워드 전송, 객체 참조자 정보



(그림 2) 이주 모델의 구조

수신, 에이전트 객체 복제, 에이전트 이주와 같이 네 개의 과정으로 구분되어진다. 각 과정의 역할 및 동작은 다음과 같다.

3.1.1.1 키워드 전송 과정

이동 에이전트의 출입 및 메시지의 송수신 기능을 담당하는 통신 관리자(Communication Manager)는 사용자의 키워드를 네이밍 에이전트 구현부의 네이밍 서비스 관리 모듈(Naming Service Management Module)에게 전달한다. 전송된 키워드는 네이밍 서비스 관리 모듈의 지시에 의해 네이밍 에이전트 관리자(Naming Agent Manager)가 스레드를 생성하고 생성된 스레드에 의해 네이밍 서비스의 reslove() 메소드를 호출한다. reslove() 메소드의 호출을 통해 네임 스페이스내의 메타 테이블을 검색한 후, 일치하는 키워드에 대한 객체 참조자 리스트 정보를 다시 네이밍 서비스 구현부의 네이밍 서비스 관리 모듈에 전달한다.

3.1.1.2 객체 참조자 리스트 정보 수신 과정

네이밍 서비스 관리 모듈은 전달 받은 키워드를 이용하여 네이밍 서비스의 네임 스페이스 내 메타 테이블에 등록된 키워드와 일치하는 객체 참조자 정보를 검색한 후, 검색된 객체 참조자 리스트 정보를 다시 통신 관리자에게 보내면, 객체 참조자 리스트 정보는 확인 과정을 거쳐 원본 에이전트 객체의 생성과 복제를 위해 에이전트 관리자(Agent Manager)로 전달된다.

3.1.1.3 에이전트 객체 복제 과정

객체의 생성 및 복제를 실행하는 에이전트 관리자는 네이밍 서버로부터 전달된 객체 참조자 리스트 정보를 기반으로 먼저, 원본 에이전트 객체를 생성하며 원본·복제에 대한 식별값을 True로 설정한다. 원본 에이전트 객체가 생성되면, 원본 에이전트 객체를 객체 참조자 리스트의 수만큼 복제를 실행한다. 이 때, 전달 받은 모든 객체 참조자 리스트의 수만큼 복제한다면 검색 적중률과 관련하여 객체 복제에 따른 비용과 시간에 관한 문제가 발생한다. 이는 최근에 자주 이용된 정보가 다시 재사용될 수 있다는 관점 하에 자주 이용되지 않는 정보를 가진 호스트로 이주할 에이전트 객체를 복제하는 것은 검색을 통해 얻어진 정보의 정확도에 비해 효율적이지 못하다는 점에 근거한다. 따라서 객체 참조자 리스트 정보 중 Hit_Count의 속성에 따라 복제 수를 결정하여 복제에 따른 비용과 시간을 줄이고 최적의 검색 적중률을 제공한다.

3.1.1.4 에이전트 이주 과정

복제된 에이전트 객체들을 네트워크를 통해 이주시키기 위해 TCP/IP 기반의 네크워크 연결을 수행하며, 연결된 후 원격지로 이동될 객체와 정보는 직렬화 모드로 변환되어 해당 호스트의 시스템에 도착한다. 이들은 목적지에서 대응하는 역직렬화 모드로 변환됨으로써 객체의 이주를 종료하고, 동시에 네트워크 자원을 해제하여 네트워크 부하를 제거한다. 또한, 에이전트 객체는 각 이동 에이전트 서버에 하나 이상 존재 할 수 있으며, 각 서버에 존재하는 에이전트 객체는 큐에 대기하며 스케줄링 된다.

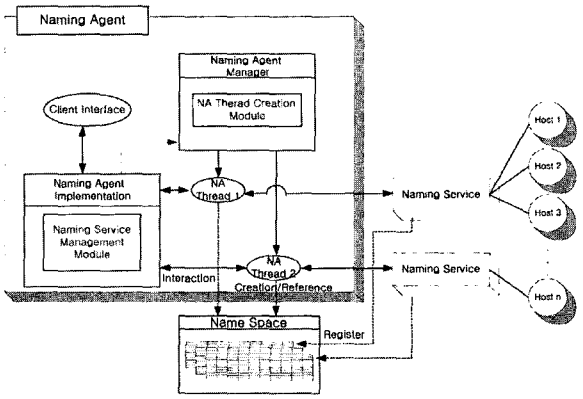
3.2 복제 정보 제공을 위한 네이밍 에이전트

멀티캐스트 이주 모델을 위해서는 분산된 서버 객체의 객체 참조자 정보를 제공하는 네이밍 에이전트가 필요하다. 따라서 서버 객체의 객체 참조자 정보를 위한 네이밍 에이전트 서버의 네이밍 에이전트 구조와 수행과정을 보이며, 이동 에이전트의 이주를 위해 키워드 및 적중 문건의 필드 정보로 구성된 메타데이터 구조와 각 키워드 정보에 따른 서버 객체 등록 요청 시 네이밍 에이전트에 의한 메타데이터의 생성 과정을 기술한다.

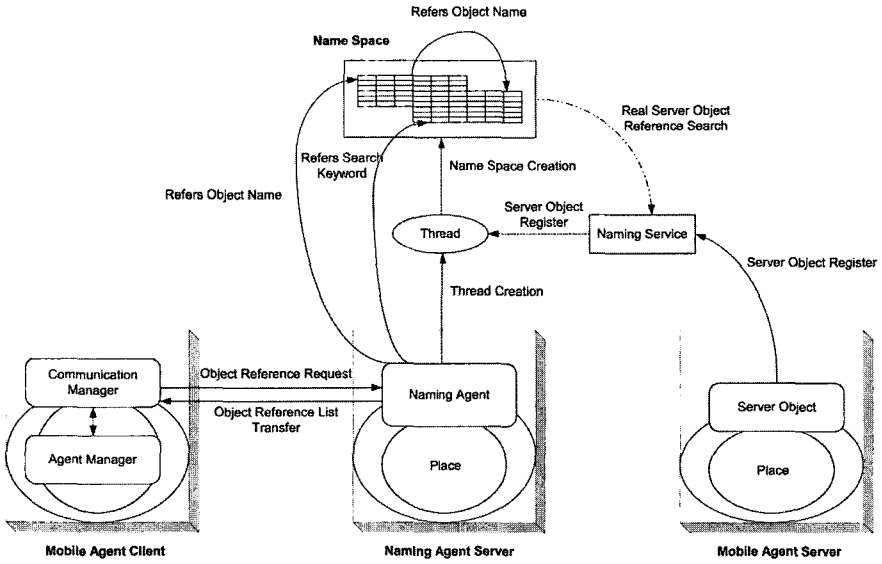
3.2.1 네이밍 에이전트 구조

기존 이동 에이전트 시스템에서는 이동 에이전트의 각 호스트 이주를 위해 사용자에 의한 수동적 라우팅 스케줄 지정 방식을 사용하고 있으나, 제안 이주 모델에서는 네이밍 에이전트 서버내 네이밍 에이전트가 호스트 이주를 위한 위치 투명성을 보장하는 라우팅 테이블을 지원한다. 이러한 네이밍 에이전트는 네이밍 에이전트 구현부(Naming Agent Implementation), 네이밍 에이전트 관리자(Naming Agent Manager), 네이밍 에이전트 스레드(Naming Agent Thread), 네임 스페이스(Name Space)로 구성되어 있다[24].

네이밍 에이전트 관리자는 서버 객체의 등록 요청에 따라 네이밍 에이전트 스레드를 생성하여 해당 네이밍 서비스와 연결을 통해 각 서버 객체의 이름과 객체 참조자를 rebind() 메소드에 의해 메타 테이블에 등록하고, 네이밍 에이전트 스레드의 네임 스페이스를 생성하여 등록된 서버 객체에 대한 메타 테이블을 생성한다. 그리고 네이밍 에이전트는 네이밍 서비스와의 연결을 통해 각 네이밍 서비스에 이동 에이전트 서버 객체들의 정보를 등록 및 관리하고 이동 에이전트 클라이언트의 요청에 대해 서버 정보를 반환함으로써 통합된 네이밍 서비스의 기능을 제공한다. (그림 3)은 네이밍 서비스별로 스레드 할당을 통해 객체 참조를 유지하고 관리하는 네이밍 에이전트의 구조를 나타낸다.



(그림 3) 네이밍 에이전트 구조



(그림 4) 네이밍 에이전트의 수행과정

Naming Service Object_ID	Name of Naming Service	Object Reference of Naming Service	Name of Sub_Object

(a) 서버 객체 이름을 통한 접근 방식의 메타데이터 구조

Name of Sub_Object	Search Keyword	Keyword Option	Total Document Count	Hit Count	Hit Rate

Host Processing Tme for Total Doc. | Host Processing Time for Hit Count | Network Traffic Time

(b) 키워드를 통한 접근 방식의 메타데이터 구조

(그림 5) 메타데이터 구조

네이밍 에이전트는 클라이언트 요청에 따라 스레드를 생성하여 네이밍 서비스에 등록되는 정보들을 관리하기 위한 테이블 형태의 네임 스페이스를 갖고, 네임 스페이스에 네이밍 서비스로부터 받은 분산된 서버 객체 정보를 중간 형태의 메타데이터로 바꾸어 보관한다. 이러한 각 스레드의 메타데이터 참조를 통해 클라이언트의 요청 시 어떤 네이밍 서비스에 클라이언트가 요구하는 서버 객체가 등록되었는지를 파악한 후, 이동 에이전트 서버 객체의 이름을 해당 스레드가 관리하는 네이밍 서비스에 요청하여 처리 후, 해당 서버 객체의 객체 참조자 정보를 클라이언트에 반환한다.

네이밍 에이전트는 네이밍 서비스에 호스트 정보와 서버 객체의 이름을 등록하고, 추가로 검색에 사용되는 키워드들을 저장하기 위한 새로운 네임 스페이스를 생성하여 분산된 서버의 객체 참조자와 계층적 검색 키워드들을 저장한다. 여기서, 계층적 검색 키워드는 데이터베이스의 테이블명과 주요 필드명, 테이블 관계 정보 등에 대한 데이터이다. (그림 4)는 네이밍 서비스에 서버 객체의 등록, 키워드 저장 및 객체 참조자 정보의 요청·반환의 네이밍 에이전트 수행과정을 나타낸다.

기존 네이밍 서비스에서는 등록된 서버 객체의 객체 참조자를 획득하기 위해 해당 서버 객체의 이름을 알고 있어야 하는 문제가 있었으나, 위의 네이밍 에이전트는 새로운 네임 스페이스의 추가 생성을 통해 사용자에 의해 요청된 키워드만으로도 객체 참조자를 얻을 수 있다.

3.2.2 메타데이터 구조

네이밍 에이전트는 네이밍 서비스별로 스레드를 할당하여 각 네이밍 서비스에 접근할 수 있도록 스레드별 네임 스페이스를 생성한다. 네임 스페이스는 네이밍 서비스와 해당 네이밍 서비스에 등록된 서버 객체의 접근정보를 갖는 메타데이터의 집합이다. 메타데이터는 (그림 5)의 (a), (b)와 같이 두 개의 테이블 형태로 분류된다.

(그림 5) (a)는 CORBA의 기존 네이밍 서비스에서 지원하던 서버 객체의 이름을 통한 접근 방식에 사용된 테이블 형태이고, (b)는 제안된[24] 구조로써 서버 객체의 이름과 검색 키워드 등의 정보로 구성되어 키워드를 통한 접근 방식을 지원하는 테이블의 형태이다. 여기서, 검색 키워드는 데이터베이스의 테이블명과 주요 필드명 등에 대한 정보이

다. 그러므로 네이밍 에이전트의 메타데이터 두 구조 중 서버 객체 이름을 통한 접근 방식의 메타데이터는 네이밍 서비스 객체의 식별을 위한 고유키와 네이밍 서비스 이름, 네이밍 서비스를 접근하기 위한 객체 참조자, 그리고 네이밍 서비스에 등록된 서브(Sub) 네이밍 서비스의 객체 이름으로 구성된다.

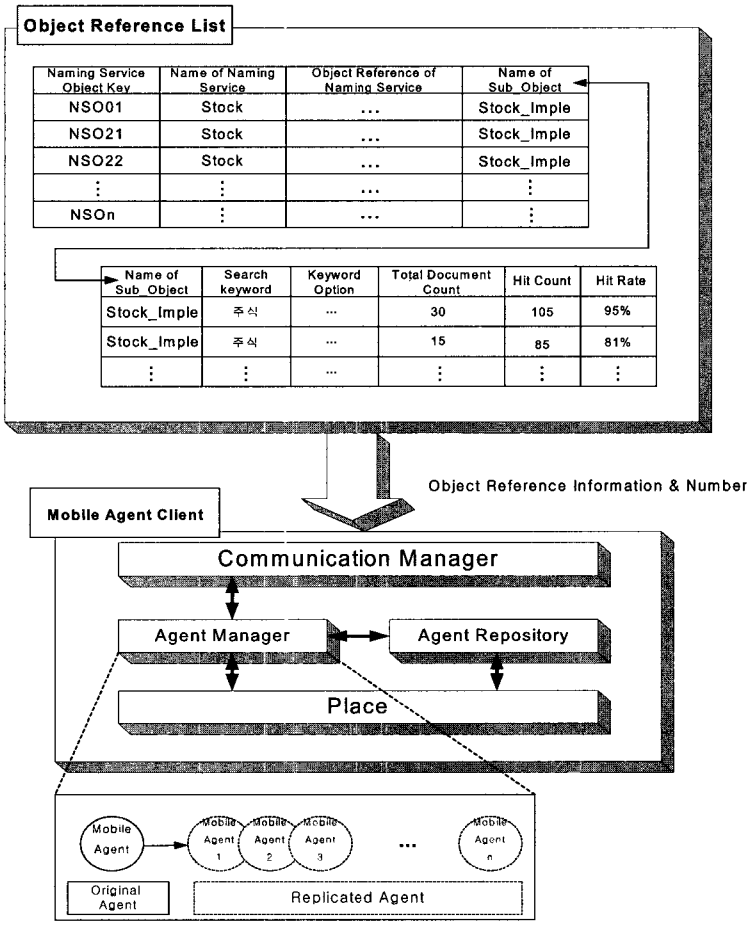
3.3 객체 복제 메커니즘

이동 에이전트에 적용된 복제 기술에 대해 2.2절에서 살펴보았듯이 에이전트 복제에 관한 기술들은 이동 에이전트 복구를 위한 복제, 중앙 집중형 관리 방식의 문제 해결을 위한 복제 그리고 이동 에이전트 이주 신뢰성에 관한 복제 기술들이었다. 그러나 이 중, 이동 에이전트의 이주 방식과 관련된 복제 기법은 (그림 1)(c)의 기법이다.

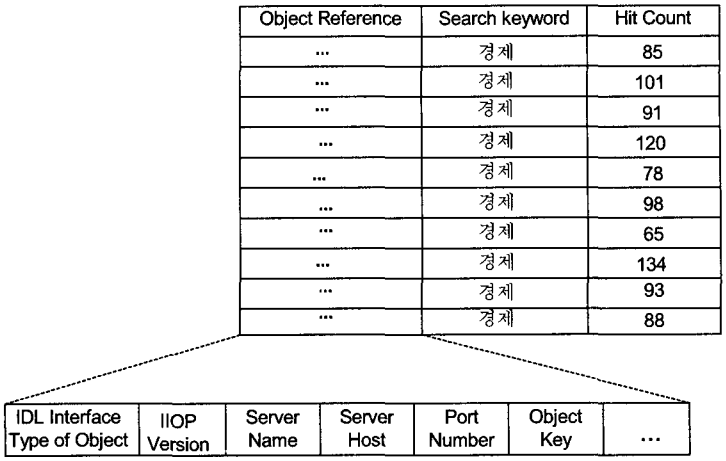
이 기법은 에이전트를 해당 호스트에 계속해서 복제해 놓고 이주하여 이주된 호스트가 결점 및 장애가 없을 시에는 이주 전, 호스트에 복제해 놓은 에이전트를 삭제하기 위해 통신을 수행한다. 하지만 통신망에 장애가 발생되었을 경우에는 이주 전 호스트의 삭제 명령을 수행하지 못하며 이주 방식이 순차적 이주 수행 구조로써 에이전트의 크기 증가로 인한 네트워크 부하와 네트워크 거리만큼의 네트워크 소요시간이 증가한다는 점이다. 그러나 본 논문에서는 이와는 다른 분산된 서버의 객체 참조자 정보를 이용하여 에이전트 객체를 복제하고, 복제된 에이전트를 이주시켜 기존 이주 방식 문제와 네트워크 소요시간을 최소화하기 위해 새로운 이주 수행 구조인 멀티캐스트 이주 모델에 이를 적용한다.

에이전트 객체 복제[23]는 사용자가 요구하는 작업을 수행함에 있어 각 호스트로 이주를 위해 분산된 호스트 위치 정보를 관리하는 네이밍 에이전트 서버의 네이밍 에이전트로부터 이주 호스트들의 객체 참조자 정보를 받아 이를 기준으로 복제를 실행한다. 이와 같이 객체 다중 복제 메커니즘을 이용할 경우 에이전트 객체의 가용성 및 기존 이주 수행 구조에서 네트워크 소요시간과 에이전트 크기 증가에 따른 문제를 최소화할 수 있다. (그림 6)은 에이전트 객체 복제 구조이다.

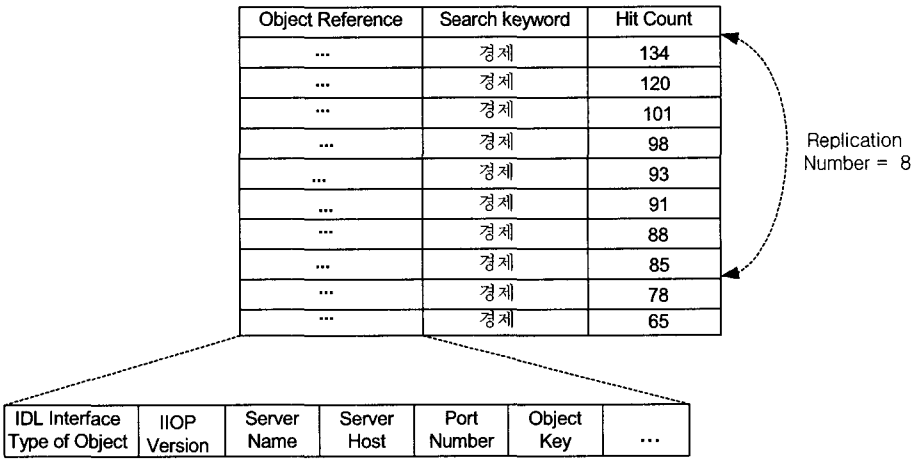
(그림 6)과 같이 다중 복제는 네이밍 에이전트의 객체 참조자 리스트 정보를 획득함으로써 이루어진다. 먼저, 사용자의 키워드 입력을 통해 네이밍 에이전트를 호출할 메시지 객체를 생성하고 생성된 메시지 객체는 네이밍 에이전트 구현부의 관리자를 호출하게 된다. 호출된 관리자는 네이밍 에이전트 스레드 생성 명령을 내리면 이러한 스레드를 통해 해당 네임 스페이스의 메타테이블을 검색하여 해당 객체 참



(그림 6) 에이전트 객체 복제 구조



(a) Hit_Count 기준 내림차순 정렬 전



(b) Hit_Count 기준 내림차순 정렬 후

(그림 7) Hit_Count 기준 복제 수 결정 과정

조자 리스트 정보를 획득한다.

획득된 객체 참조자 정보는 통신관리자(Communication Manager)의 메시지 관리자(Message Manager)를 거쳐 에이전트 관리자(Agent Manager)에게 전달되면 에이전트 생성기(Agent Generator)가 원본 에이전트 객체를 생성하여 에이전트 등록 저장소(Agent Register Repository)에 저장해 둔다. 그런 다음 전달된 정보를 Hit_Count를 기준으로 에이전트 객체를 복제한다. 이렇게 Hit_Count를 기준으로 복제의 수를 제한하는 것은 사용자가 요청하는 동일한 작업에 대해 전체 호스트로의 작업 수행을 위해 에이전트를 복제하게 된다면 복제에 대한 비용과 부하가 걸리기 때문이다. 이러한 Hit_Count를 기준으로 복제의 수를 결정하는 방법에 대해서는 다음 절에 기술하였다.

3.3.1 객체 복제 수 결정을 위한 Hit_Count

멀티캐스트 이주 모델을 위해서는 에이전트 객체에 대한 복제가 요구된다. 하지만 획득된 모든 객체 참조자 리스트 정보에 대한 복제는 복제에 대한 부하와 비용을 증가시킨다. 따라서 네이밍 에이전트로부터 획득한 객체 참조자 리스트의 정보 즉, 동일한 작업 정보를 보유하고 있는 모든 호스트에 대해 복제를 수행한다면 복제에 대한 부하와 비용을 감당하기 어려우며 부득이하게 동일한 요청 작업을 수행함에 있어 전체 호스트에 대해 이동 에이전트를 복제하여 이주 시킨다는 것은 비효율적일 것이다.

그러므로 키워드를 이용한 메타데이터의 Hit_Count를 기준으로 복제를 수행하는데 이때, 클라이언트는 네이밍 에이전트로부터 획득되어진 객체 참조자 리스트를 Hit_Count를 기준으로 내림차순 정렬하여 이중 80%에 해당하는 서버 객체의 객체 참조자를 기준으로 복제를 수행한다. 이렇게 복제에 대한 비율을 정하는 것은 사용자 키워드의 빈도수에 따른다. 만약, 키워드의 빈도가 낮은 경우는 비율을 낮추고, 빈도가 높은 경우는 비율을 높인다. 따라서 이러한 방법은 복제에 대한 부하와 비용을 감소시키며 또한, 동일한 작업 수행 요청에 대한 일관성을 유지시킨다. (그림 7)은 객체 참

조자 리스트를 내림차순 정렬하여 복제 수를 결정하는 과정이다.

(그림 7)(b)와 같이 복제의 수가 결정되면 생성된 원본 에이전트를 기반으로 에이전트 객체를 결정된 복제 수에 맞게 복제한 후, 에이전트를 해당 호스트로 이주시킨다.

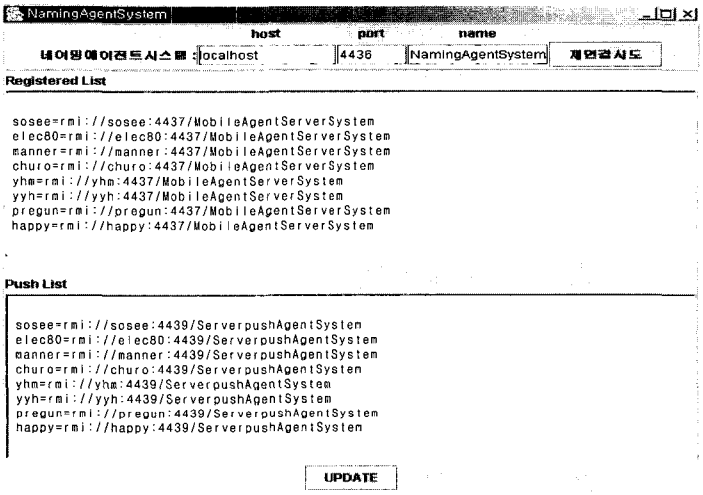
4. 실험 및 평가

4.1 실험환경

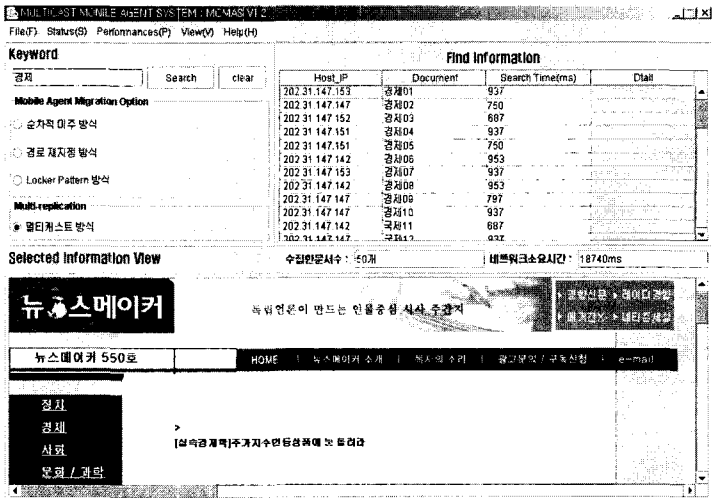
실험환경은 네트워크에 연동된 한 개 이상의 호스트를 대상으로 각 에이전트가 해당 호스트에서 제공하는 정보에 접근하여 수집한 후, 사용자에게 전달하는 것이며 시스템 사양은 <표 1>과 같다. 또한, 구현 환경은 에이전트 시스템들 간의 상호운용성 지원을 위한 OMG의 MAF 명세와 CORBA 3.0 Spec.에 기반한 OrbixWeb 3.2와 프로그램 개발 도구인 JBuilder 7.0과 JDK 1.4.2를 사용하였다.

〈표 1〉 실험 환경 및 시스템 사양

Host ID	IP Address	Port Number	Operating System	CPU
happy	202.31.147.141	4437	Windows 2003 Server	P-Ⅳ 2.0GHz
elec80	202.31.147.147	4437	Windows 2000 Professional	P-Ⅳ 2.8GHz
manner	202.31.147.143	4437	Windows 2000 Professional	P-Ⅲ 1.7GHz
sosee	202.31.147.152	4437	Windows XP	P-Ⅳ 2.4GHz
yhm	202.31.147.146	4437	Windows 2000 Professional	P-Ⅲ 1.2GHz
yyh	202.31.147.150	4437	Windows 2000 Professional	P-Ⅳ 2.4GHz
Pregun	202.31.147.157	4437	Windows 2000 Professional	P-Ⅲ 1.2GHz
Churo	202.31.147.153	4437	Windows 2000 Professional	P-Ⅳ 2.4GHz



(그림 8) 네이밍 에이전트의 호스트 등록상황



(그림 9) 프로토타입 시스템 구동 화면

먼저, 이동 에이전트를 이주시키려면 각 호스트의 위치 정보를 알아야 한다. 왜냐하면 인터넷 같은 분산 환경에서 에이전트를 무작위로 이주 시킬 수 없기 때문이다. 따라서 멀티캐스트 이주 모델에서는 이동 에이전트의 이주를 위해 네이밍 에이전트의 메타테이블을 검색하여 분산된 서버 객체 참조자 정보를 얻는다. 이러한 정보는 각 이동 에이전트를 이주시킬 호스트의 위치와 네트워크에 연동된 각 호스트가 동작 중인 상태를 알려준다. 만약 호스트의 결점 및 장애가 발생시에는 해당 호스트가 네이밍 에이전트에 등록되지 않으며 (그림 8)의 아래에 위치한 Update 버튼을 누르면 등록이 갱신된 호스트의 정보를 확인할 수 있다.

네이밍 에이전트는 이동 에이전트의 능동적 작업 수행을 위해 분산된 호스트를 관리하며 (그림 8)같이 각 호스트들을 등록한다. 이러한 기능은 이동 에이전트 이주를 위한 목적지 호스트에 대한 위치 투명성과 동작 상태를 제공한다.

이와 같이 네이밍 에이전트에 등록된 각 호스트로 이동 에이전트가 이주하여 정보를 수집한 후, 결과를 가져오기 위한 작업을 수행하기 위해서는 프로토타입 시스템을 구동시켜야 한다. (그림 9)는 멀티캐스트 방식과 기존 이주 방식의 실험을 위해 구현한 프로토타입 시스템이다.

4.2 성능평가

분산 시스템에서의 성능 평가는 균등 네트워크 상의 동일한 호스트 프로세싱 시간을 가정하고 사용자 요구에 대한 응답인 네트워크 소요시간만을 평가하였으며, 실험은 전체 네트워크의 대역폭이 100 Mbps이고, 호스트 간 거리가 15m ~ 20m인 네트워크 환경 하에서 수행하였다. 그리고 선택된 호스트에서 적중된 문건에 대한 각각의 자료 처리 시간은 동일하다. 또한, 이주 방식에 따른 차이를 최소화하기 위해 동일한 라우팅 스케줄에 따라 이주를 수행한다. 이러한 가정들은 성능 평가 시 외부적인 문제를 배제하고 순수하게 키워드에 따른 성능을 추출해내기 위한 것이다.

멀티캐스트 이주 방식은 에이전트 객체를 다중 복제하여 각 해당 호스트로 이주시켜 네트워크 소요시간을 측정한다. 네트워크 소요시간은 각 해당 호스트로 이주된 여러 개의 이동 에이전트 중에서 결과 값을 가장 늦게 반환한 이동 에이전트의 도착시간에서 출발시간을 뺀 시간이며, 이는 여러 개의 이동 에이전트가 전체 호스트에서 결과를 가지고 되돌아오는 시간이 각기 다르기 때문에 마지막으로 도착한 이동 에이전트가 전체 호스트를 대상으로 한 작업이 끝나는 시점이 되기 때문이다. 실험은 8개의 호스트를 대상으로 10회 반복하여 수행하였으며 또한, 10회 반복한 네트워크 소요시간의 편차를 구하여 이동 에이전트가 신뢰성 있게 실행되고 있음을 나타내었다. 이에 대한 실험의 결과는 <표 2>와 같다.

순차적 이주 방식[2, 3, 5, 19]은 각각의 호스트를 순차적으로 방문하여 실행모듈과 수집된 결과를 포함하여 이주를 수행한다. 이에 대한 실험 결과는 <표 3>과 같다.

〈표 2〉 멀티캐스트 이주 방식의 네트워크 소요시간

(단위:ms)

실험회수 \ 호스트 수	1	2	3	4	5	6	7	8
1	316	5297	9422	11257	15587	20124	26924	35045
2	309	4985	8750	10895	16895	19856	28561	39875
3	250	5093	8954	12001	14985	18569	28145	36520
4	266	5078	8781	10024	14532	18782	29110	36247
5	281	4984	8922	11854	15864	19530	28620	39581
6	250	5078	8750	11203	16025	17568	27569	38659
7	281	4969	8812	10115	15548	18960	29036	39457
8	234	4969	8906	12689	14580	20542	29854	37452
9	312	5031	8750	11805	15829	19963	30457	40112
10	234	4984	9078	10706	16980	18634	30894	41257
평균	273	5047	8913	11255	15683	19253	28917	38421
표준편차	32	100	210	851	842	904	1239	2005
최대편차	82	328	672	2665	2448	2974	3970	6212

〈표 3〉 순차적 이주 방식의 네트워크 소요시간

(단위:ms)

실험회수 \ 호스트 수	1	2	3	4	5	6	7	8
1	3211	21000	37500	50254	85980	125876	205485	322215
2	3210	20940	33910	59865	89562	126988	215482	328541
3	2500	17500	34840	58756	87245	134502	210256	330215
4	2810	16240	35980	57965	87451	130255	225482	345860
5	2660	15640	38300	56540	85245	140222	223652	354581
6	3150	14680	36950	59995	88212	140655	214587	338546
7	2660	17940	38300	60540	90154	139952	215036	335421
8	2960	19680	38750	51256	91754	128455	238754	345178
9	2340	20320	36890	60478	89059	139852	235487	354580
10	2567	15620	38750	59854	90457	134452	201253	354781
평균	2807	17956	37017	57550	88512	134121	218547	340992
표준편차	313	2391	1667	3795	2063	5897	12226	11839
최대편차	871	6320	4840	10286	6509	14779	37501	32566

〈표 4〉 경로 재조정 이주 방식의 네트워크 소요시간

(단위:ms)

호스트 수 / 실험회수	1	2	3	4	5	6	7	8
1	2915	18060	37589	55502	88854	135641	202541	355482
2	3642	21000	36892	56845	89562	145210	215412	356200
3	2475	15750	37548	57421	87548	132568	213564	364510
4	3102	18060	38542	56852	85421	142050	201245	342514
5	3051	18093	39512	56895	86547	145215	235487	332585
6	3102	17409	39522	58002	88451	135481	235840	345215
7	3091	17703	38665	57454	87451	138745	236589	365487
8	3221	15435	37225	59684	89520	139852	235458	332012
9	2750	16086	39001	59852	87451	145261	225481	339854
10	2856	17703	39215	57842	88025	157841	215487	357841
평균	3021	17530	38371	57635	87883	141786	221710	349170
표준편차	308	1586	980	1321	1296	7281	13925	12393
최대편차	1167	5565	2630	4350	4141	25273	35344	33475

〈표 5〉 Locker Pattern 방식의 네트워크 소요시간

(단위:ms)

호스트 수 / 실험회수	1	2	3	4	5	6	7	8
1	1140	12885	16775	20542	29856	41458	53250	70110
2	1064	14760	16025	21450	28754	39887	54110	69825
3	1248	11955	13675	23541	29654	40568	53268	69520
4	1060	12900	15625	22541	29250	41587	54215	70256
5	1016	12180	13675	21014	28998	41552	53250	71548
6	1002	12885	16025	21563	28568	40998	53689	70364
7	936	12420	16025	21548	29754	42008	53774	71254
8	1060	11955	12900	22140	28456	41552	53225	69885
9	1004	11010	16000	22365	29851	41845	54100	69554
10	1156	12885	14850	23652	29880	40881	54258	70158
평균	1069	12584	15158	22036	29302	41234	53714	70247
표준편차	91	976	1308	1018	570	649	438	672
최대편차	312	3750	3875	3110	1424	2121	1033	2028

경로 재조정 이주 방식[6, 7, 8, 12]은 임의의 호스트 장애 및 결점이 발생하였을 때 이주의 신뢰성을 보장하는 방법이다. 따라서 실제 라우팅 스케줄상의 호스트에서 장애가 발생하였을 경우, 이동 에이전트는 더 이상 이주가 불가능한 상황이 되므로 경로를 재조정하여, 재조정된 호스트로 이주하여 작업을 수행한다. 이에 대한 실험 결과는 <표 4>의 결과와 같다.

Locker Pattern 이주 방식[20, 21, 22]에서는 각 호스트에서 수집된 결과는 클라이언트로 전송하고, 이동 에이전트 자신은 다음의 호스트로 실행모듈만을 가지고 이동하는 구조를 갖는다. 이에 대한 실험 결과는 <표 5>에 나타내었다.

이와 같이 본 논문에서는 세 가지 기존 이주 방식과 멀티캐스트 이주 방식을 8개의 호스트를 대상으로 작업을 수행하여 동일한 키워드에 대한 네트워크 소요시간을 산출하고 이를 평가 및 분석하였다. <표 6>은 멀티캐스트 이주 방식과 기존 이주 방식과의 평균 네트워크 소요시간이다.

첫 번째, 순차적 이주 방식은 에이전트의 실행모듈과 각 호스트에서의 결과를 포함하여 이주하기 때문에 에이전트 크기가 증가한다. 따라서 이러한 에이전트의 크기 증가는 네트워크 트래픽 부하를 유발하며 결국, 전체 호스트로의

〈표 6〉 멀티캐스트 이주 방식과 기존 이주 방식의 평균 네트워크 소요시간

(단위:ms)

이주 방식 \ 호스트 수		1	2	3	4	5	6	7	8
기존 이주 방식	순차적	2807	17956	37017	57550	88512	134121	218547	340992
	경로 재조정	3021	17530	38371	57635	87883	141786	221710	349170
	Locker Pattern	1069	12584	15158	22036	29302	41234	53714	70247
제안 이주 방식	멀티캐스트	273	5047	8913	11255	15683	19253	28917	38421

순회 작업을 위한 네트워크 소요시간이 증가한다. 또한, 호스트 수에 따라 상대적으로 이주해야할 거리는 길어지며 결과 전달에 소요되는 시간이 증가한다. 그러므로 네트워크 소요시간이 멀티캐스트 이주 방식보다 상당히 증가됨을 볼 수 있다.

두 번째, 경로 재조정 이주 방식은 에이전트의 이주 신뢰성을 보장하는 방법이다. 하지만 호스트 결점 및 장애가 발생되었을 경우, 경로 재설정과정에 따른 라우팅 테이블의 재설정 시간과 이동 에이전트의 호스트에 대한 이주의 신뢰성 보장 측면이란 점을 배제한다면 순차적 이주 방식과 동일하다.

세 번째, Locker Pattern 이주 방식은 순차적 이주 방식과 경로 재조정 방식보다는 네트워크 소요시간이 상당히 감소됨을 볼 수 있다. 이것은 방문할 호스트의 수에 따라 결과가 누적되어 이주 시 에이전트 크기가 증가하여 네트워크 트래픽을 유발시키는 순차적, 경로 재조정 방식과는 달리 각 호스트에서 수집된 결과는 클라이언트로 전송하고, 에이전트 자신은 다음 호스트로 실행모듈만을 가지고 이주하는 구조를 갖기 때문이다. 그러나 멀티캐스트 이주 방식과는 네트워크 소요시간에 있어서 그리 큰 차이는 나타나지 않았다. 이는 실험환경이 소규모 네트워크 환경이고, 호스트간의 네트워크 거리가 짧고, 비균등 네크워크 환경이 아니기 때문이다. 하지만 네트워크의 거리와 대 규모 네트워크 환경에서는 네트워크 소요시간이 상당한 차이를 보일 것이다.

이와 같이 기존 이주 방식과의 네트워크 소요시간을 비교 평가해 본 결과 멀티캐스트 이주 방식은 분산된 각 호스트의 서버 객체 정보를 획득하여 에이전트 객체를 다중 복제하여 각 개별 호스트로 이주되므로 네트워크 거리에 대한 네트워크 소요시간을 감소시켰으며 또한, 결점이나 장애가 발생된 해당 호스트는 자신의 서버 객체를 네이밍 에이전트에 등록할 수 없으므로 해당 호스트에 대한 라우팅 테이블을 생성하지 않는다. 따라서 이동 에이전트의 이주 신뢰성과 해당 호스트에 대한 작업 신뢰성을 확인할 수 있었다.

5. 결론 및 향후 연구과제

본 논문에서는 기존 이동 에이전트 시스템의 이주 방식 문제와 네트워크 소요시간을 최소화하기 위해 객체 다중 복제 기반의 멀티캐스트 이주 모델을 설계 및 구현하였다. 따라서 구현된 모델은 통신 관리자, 에이전트 관리자, 에이전트 등록 저장소, 에이전트 상태 정보 저장소, 에이전트 생성기, 에이전트 이주기, 플레이스 등의 요소를 포함하며 멀티캐스트 이주를 지원한다.

이러한 모델을 위해 분산된 서버의 객체 참조자 정보를 획득하기 위한 방법으로 서버 객체의 이름뿐만 아니라 키워드만으로도 접근이 가능한 네이밍 에이전트와 멀티캐스트 이주 방식을 위해 이동 에이전트 객체를 복제하는 다중 복제 메커니즘의 구조와 복제 수를 제한하는 방법에 대해 제시하였다. 따라서 멀티캐스트 이주 모델은 에이전트 객체에 대해 각 호스트의 위치 투명성과 네이밍 에이전트에 등록된 서버 객체들의 정보를 통해 에이전트 객체가 복제되어 각 개별 호스트로 이주되므로 분산 처리가 가능한 컴퓨팅 환경을 제공하며 네트워크 소요시간을 감소시킨다. 이러한 제안된 이주 방식을 검증하기 위하여 기존 이주 방식인 순차적, 경로 재조정, Locker Pattern 방식을 프로토타입 시스템 적용하여 동일한 라우팅 테이블과 키워드를 기준으로 이동 에이전트를 이주시켜 분산된 8개의 호스트에서 정보 수집 작업을 수행한 후, 결과를 가져오는 네트워크 소요시간을 실험하였다.

실험 결과, 순차적 이주 방식은 네트워크 소요시간이 2807~340992(ms), 경로 재조정 이주 방식은 3021~349170(ms), Locker Pattern 이주 방식은 1069~70247(ms)의 시간이 소요되었으며 멀티캐스트 이주 방식은 273~38421(ms)와 같이 소요되었다. 이와 같이 멀티캐스트 이주 방식은 네트워크 거리에 대한 네트워크 소요시간을 감소시켰으며 또한, 결점이나 장애가 발생된 해당 호스트는 자신의 서버 객체를 네이밍 에이전트에 등록할 수 없으므로 이동 에이전트의 이주 신뢰성과 해당 호스트에 대한 작업 신뢰성을 확인하였다.

향후 연구과제로는 본 논문에서 고려하지 않은 네트워크 지연 시간, 호스트의 프로세싱 시간 등의 문제와 호스트 수의 증가에 따른 객체 복제 수를 제한하기 위해 Hit_Count뿐만 아니라 Hit_Rate과 Total_Document Count 등도 포함하여 임계값 설정할 수 있도록 하는 문제를 종합적으로 고려하여 멀티캐스트 이주 모델에 대한 성능을 평가하는 연구가 수행되어야 한다. 또한, 대규모 네트워크 환경에도 적용이

가능하도록 확장하는 연구가 필요하다.

참 고 문 헌

[1] Ahmed Karmouch, *"Mobile Software Agents for Telecommunications"*, IEEE Communication Magazine, Vol. 10, pp.24-25, 1998.

[2] V. A. Pham and A. Karmouch, *"Mobile software Agents: An Overview"*, IEEE Communication Magazine, Vol.11, pp.26-37, 1998.

[3] P. Bellavista, A. Corradi and C. Stefanelli, *"A Mobile Agent Infrastructure for the Mobility Support"*, Proceedings of the 2000 ACM Symposium, ACM Press, USA, pp.539-545, 2000.

[4] J. W. Baek, G. T. Kim and H. Y. Yeom, *"Timed Mobile Agent Planning for Distributed Information"*, ACM AGENTS, 01', 2001.

[5] D. B. Lange and M. Oshima, *Programming and deploying Java Mobile Agents with Aglets*, Addison Wesley Press, 1998.

[6] 전병국, 이근상, 최영근, *"Java 언어를 이용한 객체 이동시스템의 설계 및 구현"*, 정보처리논문지, 제6권, 제1호, pp.124-137, February, 1999.

[7] J. Baumann, *"A Protocol for Orphan Detection and Termination in Mobile Agent Systems"*, TR-1997-09, Stuttgart University, 1997.

[8] K. Rothermel and M. Strasser, *"A Fault-Tolerant Protocol for Providing the Exactly-Once Property of Mobile Agents"*, Proceeding of the 17th IEEE SRDS98, 1998.

[9] B. Venners, *"The Architecture of Aglets"*, JavaWorld, http://www.javaworld.com, 1997.

[10] J. Bradshaw, *Software Agents*, AAAI Press/MIT Press, Menlo Park, Cal., 1996.

[11] IBM, *"The Aglets Workbench", http://www.trl.ibm.co.jp/aglets*, 2002.

[12] A. Fedoruk and Ralph Deters, *"Improving Fault-Tolerance by Replicating Agent"*, AAMAS'02, pp.737-744, 2000.

[13] J. Vitek and C. Tschudin, *"Mobile Object Systems: Towards the Programmable Internet"*, Springer-Verlag, April, 1997.

[14] 윤동식, 이병관, *"객체 복제 기법에 의한 원격 접근 알고리즘"*, 정보처리논문지, 제7권 제3호, pp.873-884, October, 2000.

[15] N. Adly, *Management of Replicated Data in Large scale System*, Cambridge University, 1995.

[16] Bo Li, *"Content Replication in a Distributed and Controlled Environment"*, Journal of PDC, pp.229-251, June, 2000.

[17] Yuri Breitbart and Henry F. Korth, *"Replication and Consistency in a Distributed Environment"*, Journal of PDC, pp.26-69, September, 1999.

[18] E. Gendelman, L. F. Bic and M. B. Dillencourt, "An Application-Transparent, Platform-Independent Approach to Rollback-recovery for Mobile Agent Systems", Proc. of the 20th Int. 1, conf. on Distributed Computing Systems, 2000.

[19] H. Peine and T. Stolpmann, *"The Architecture of the Ara Platform for Mobile Agent"*, Proc. of the 1st International Workshop on Mobile Agents(MA'97), Berlin, Germany, April, 1997.

[20] M. Strasser and K. Rothermel, *"System Mechanism for Partial Rollback of Mobile Agent Systems"*, Proc. of the 20th Int. 1, Conf. on Distributed Computin Systems, 2000.

[21] K. Hosoon and et al, *"An Intelligent Mobile Agent Framework for Distributed Network Management"*, Globecom' 97 Phoenix, AZ. Nov. 3-8, 1997.

[22] D. Hagimont and L. Ismail, *"A Performance Evaluation of the Mobile Agent Paradigm"*, Proc. OOPSLA' 99, Int, Conf. on Object-Oriented Programming, Systems and Applications, November, 1999.

[23] 김광종, 이연식, *"객체 복제를 통한 이동 에이전트의 병렬 이주 방식 설계"*, 정보처리학회논문지D, 제11-D권 제2호, pp.351-360, April, 2004.

[24] 김광종, 이연식, *"네이밍 에이전트의 메타데이터를 이용한 멀티 에이전트의 협력 및 호스트 이주 기법"*, 정보처리학회논문지D, 제11-D권 제1호, pp.345-354, April, 2004.

김 광 종

e-mail : kkjkim@kunsan.ac.kr

1993년 군산대학교 컴퓨터정보과학과 (이학사)

1999년 군산대학교 컴퓨터정보과학과 (이학석사)

2004년 군산대학교 컴퓨터정보과학과 (이학박사)

관심분야 : 에이전트, 분산객체시스템, 능동 데이터베이스

고 현

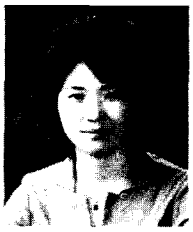
e-mail : khyun001@hanmail.net

2001년 군산대학교 컴퓨터정보과학과 (이학사)

2003년 군산대학교 컴퓨터정보과학과 (이학석사)

2004년 군산대학교 컴퓨터정보과학과 박사과정

관심분야 : 에이전트, 분산객체시스템, 멀티미디어 데이터베이스



김 영 자

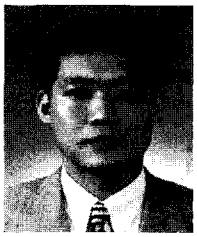
e-mail : tiny89@kopo.or.kr

1993년 군산대학교 컴퓨터정보과학과 (이학사)

2000년 순천대학교 컴퓨터정보과학과 (이학석사)

2004년 군산대학교 컴퓨터정보과학과 박사수료

관심분야 : 에이전트, 분산객체시스템, CDN



이 연 식

e-mail : yslee@kunsan.ac.kr

1982년 전남대학교 전자계산학과(이학사)

1984년 전남대학교 전자계산학과(이학석사)

1994년 전북대학교 전산응용공학전공 (공학박사)

1995년~1997년 군산대학교 교무부처장

1997년~1998년 University of Missouri 교환교수

1999년~2001년 군산대학교 전자계산소 소장

1986년~현재 군산대학교 컴퓨터정보과학과 교수

관심분야 : 번역기 이론, 객체지향시스템, 능동시스템, 지능형 에이전트