객체 복제를 통한 이동 에이전트의 병렬 이주 방식 설계

김 광 종† · 이 연 식††

요 약

대부분의 이동 에이전트는 순차적인 노드 이주 방식에 의하여 다수의 이동 에이전트 시스템들로 이주된다. 하지만, 이러한 이주방식에서는 호스트의 결점이나 장애 등과 같은 문제가 발생하였을 경우, 이동 에이전트가 무한 대기나 고아 상태에 빠지므로 노드들 사이의 네트워크 소요시간이 증가하기 때문에 실제 다른 분산 기술들을 사용한 것만큼의 기대효과를 얻기가 어렵다. 따라서 이러한 문제를 해결하기 위한 연구들이 진행되어 왔지만 대부분 수동적 라우팅 테이블을 기반으로 노드의 이주를 결정하거나 문제가 발생된 호스트를 선회하는 방법이므로 실제 전체적인 네트워크 소요시간을 감소시키기 위한 연구는 아직 미흡하다. 본 논문에서는 이동 에이전트가 이동 에이전트 시스템들로 이주시 네트워크 소요시간을 감소시키기 위하여 네이밍 에이전트의 메타-테이블에 등록된 구현 객체 정보를 기반으로 능동적 라우팅 테이블을 설계한다. 또한 사용자의 키워드에 대하여 메타-테이블에서 일치하는 객체 참조자의 정보와 수에 따라 다수의 에이전트 객체를 복제한다. 복제된 객체는 이동 에이전트 시스템들로 병렬 이주되며, 최소의 네트워크 소요시간을 제공한다.

Design of Parallel Migration Method of Mobile Agents Using an Object Replication

Kwang-jong Kim† · Yon-sik Lee††

ABSTRACT

Most mobile agents are migrated to many mobile agent systems by the sequential node migration method. However, in this case, if some problems such as host's fault or obstacle etc. happened, mobile agent falls infinity wait or orphan states. Therefore, it is difficult to get an expectation effect as use of other distribution technologies because the required time for networking between nodes increases. And so, many researches have been performed to solve this problems. However, most of methods decide node migration based on passive routing table or detour hosts which have some problems. Actually, the researches for reducing the total required time for networking are insufficient yet. In this paper, to reduce the required time for networking of mobile agent we design an active routing table based on the information of implemented objects which are registered in the meta-table of naming agent. And also, for user's keyword, we propose an replication model that replicates many agent object according to the information and number of object references corresponding to meta-table. Replicated objects are migrated to mobile agent systems in parallel and it provides minimized required time for networking.

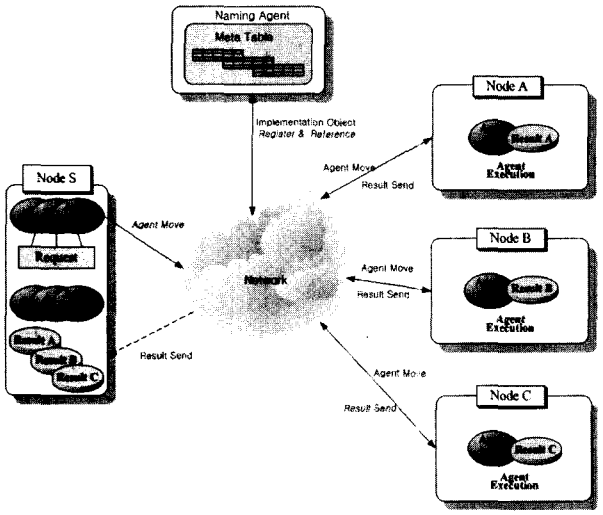
키워드 : 이동 에이전트(Mobile Agent), 네이밍 에이전트(Naming Agent), 메타테이블(Meta-Table), 병렬 이주(Parallel Migration), 복제(Replication)

1. 서 론

최근 분산 환경은 분산 객체 컴퓨팅(distributed object computing) 기술로 인해 네트워크상의 동일 및 이기종 시스템간 분산처리 능력이 향상되었으나 여전히 네트워크 트래픽과부하 문제를 안고 있다[1, 2]. 따라서 이를 해결하기 위해 새로운 패러다임인 이동 에이전트(mobile agent)에 대한 요구가 급증되고 있다[3, 4]. 기존에 클라이언트/서버 구조에 기반을 둔 RPC나 멀티 에이전트의 경우, 원격 통신에 의해 작업을 수행했던 반면 이동 에이전트 시스템은 에이전트 코드 자체가 서버로 직접 이동하여 주어진 작업을 수행하는 구조를 갖음으로써 네트웍 트래픽을 감소시킬 수 있다[5-8]. 그러나 기존 이동 에이전트는 수동적 라우팅 스케줄에 따라 순차적으로 이주하므로 인해 해당 노드에서 발생할 수 있는 호스트의 결점이나 장애 등으로 이동 에이전트가 무한 대기상태나 고아(Orphan)상태에 빠져 사용자에게 부여받은 작업의 지연 및 네트워크 소요시간이 증가한다. 그러므로 이동 에이전트를 이용하여 개발한 분산 시스템이 기존의 접근방식에 비해 성능이 좋은 지의 여부는 아직도 의견이 분분하다. 즉, 이동 에이전트는 노드간의 통신 횟수, 전송 데이터의 양, 에이전트의 크기, 네트워크 상태 등의 요소에 따라 매우 다양한 성능을 보이지만 분산 시스템의 전체 성능에 큰 영향을 주는 요소는 이동 에이전트의 이주 방식에 있다[2, 9, 10].

† 준 회 원 : 군산대학교 대학원 컴퓨터정보과학과
†† 종신회원 : 군산대학교 컴퓨터정보과학과 교수
논문접수 : 2003년 5월 20일, 심사완료 : 2003년 11월 22일

본 논문에서는 기존 이동 에이전트의 이주 방식으로 인한 문제와 네트워크 소요시간을 감소시키기 위한 방법으로 객체 복제를 통한 이동 에이전트의 병렬 이주 방식을 제안한다. 제안한 이주 방식은 이동 에이전트가 이주할 분산 노드들에 대해 동적으로 라우팅 스케줄을 지정하는 네이밍 에이전트(naming agent)와 구현 객체 정보를 등록하고 관리하는 메타테이블(meta-table)을 설계하고, 메타 테이블에 등록된 구현 객체와 키워드 정보를 기반으로 이주할 노드의 객체를 복제하는 알고리즘을 제시한다. 그리고 객체 복제에 의한 병렬 이주 모델을 설계 및 구현한다. (그림 1)은 본 논문에서 제안한 병렬 이주 모델의 수행구조이다.



(그림 1) 병렬 이주 수행구조

본 논문의 구성은 다음과 같다. 먼저 2장에서 이동 에이전트의 이주 전략과 관련된 기존의 연구를 살펴보고, 3장에서는 병렬 이주 모델의 구조 및 통신과정과 네이밍 에이전트의 구조를 보이며, 객체 복제를 위한 방법과 구현 객체, 키워드를 관리하는 메타테이블의 알고리즘을 제시하며, 4장에서는 제안된 병렬 이주 방식과 기존의 순차적 이주 방식의 네트워크 소요시간 실험을 통해 결과를 분석하며, 마지막 5장에서 결론 및 향후 연구방향을 제시한다.

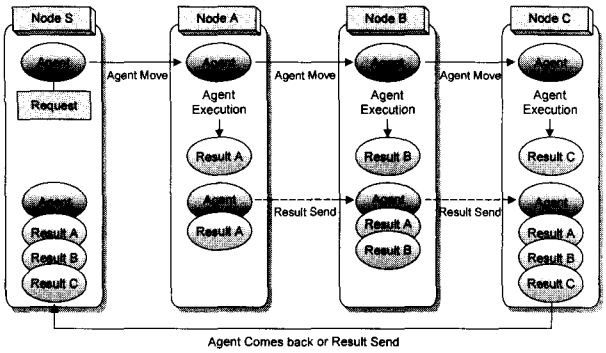
2. 관련 연구

기존에 개발된 대부분의 이동 에이전트 시스템들은 이동 에이전트의 이주 전략과 관련하여 특별한 이주 보장 정책을 지원하지 않거나 이를 지원한다 해도 대부분 통신망 혹은 노드 결점에 대해 이주 신뢰성을 보장하기 위한 이주 정책들만을 제공하고 있다[2-4, 11-16]. 기존 이동 에이전트 시스템들이 에이전트의 이주 보장을 제공하지 않는다는 것은 이주 지원을 위해 이상적인 조건, 즉 통신망이나 노드의 결점이 없는 경우를 가정하고 있거나 발생한 결점에 대처할 수 있는 적절한 프로토콜만을 제공할 뿐 이주 신뢰성을 보장하기 위한 별도의 이주 정책은 지원하지 않는다는 것이다.

현재까지 연구된 대부분의 이주 보장 정책들은 이동 에이전트가 통신망에 연결된 다수의 이동 에이전트 시스템들로 이주할 때 발생할 수 있는 통신망 혹은 노드 결점에 대처할 수 있는 이주 정책[2-4, 9, 17-20]들이다. 통신망 결손이나 노드 장애 또는 데이터베이스 등 정보 서비스의 부재시 이동 에이전트는 무한 대기나 고아 상태에 빠질 수 있고 혹은 파괴되어 쓰레기로 처리될 수 있다[9]. 따라서 이동 에이전트의 이주시 발생할 수 있는 통신망 결손 및 노드 장애에 대처하기 위해, JAMAS[9]에서는 경로 재조정과 후위 복구 기법을 제안하였고, [9, 17]에서는 방문하는 노드의 결점에 대처하기 위한 프로토콜을 제시하였다. 또한, Mole [20]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 노드에 이동 에이전트를 복사함으로써 투표(Voting)와 선택(Selection) 프로토콜을 이용한 효율적인 이주 기법을 제공하지만, 작업(Worker) 노드를 제외한 모든 관찰(Observer) 노드간의 상호 감시와 통신 기능이 필요하며, 통신망 접속에 결점이 없는 것을 가정하고 있다.

한편, 이동 에이전트의 이주 노드에 대한 라우팅 스케줄 지정과 관련하여 대부분의 이동 에이전트 시스템은 Aglets [12-14]에서와 같이 이동 에이전트가 고정된 순서대로 노드를 방문하고 방문한 노드에서 검색한 데이터를 계속 누적시키면서 이동하는 단순 구조를 가지고 있다. 이는 복잡하고 다양한 사용자 요구를 처리하는 에이전트의 작업 수행에 있어 이주 노드를 재지정해야 하는 불가피한 상황이나 통신망 혹은 노드의 결점과 같은 특정한 문제 발생시 동적으로 에이전트의 이주를 수행할 수 없음으로써 에이전트 이주시 발생할 수 있는 여러 가지 요인들에 대해 능동적으로 대처할 수 없다. 또한, 이동 에이전트가 방문해야 하는 노드의 수나 검색되는 데이터의 양이 조금이라도 많은 시스템에서는 이동 에이전트의 이주 시간 비용이 많이 소요되어 이동 에이전트를 사용하는것 자체가 비효율적일 수 있다. (그림 2)는 고정된 라우팅 스케줄에 따라 분산된 노드들을 이주하며 작업을 처리하는 기존 이동 에이전트의 이주 구조이다.

이와 같이 현재에는 이동 에이전트의 이주 전략과 관련하여 통신망 혹은 노드의 결점이 발생하였을 때 이주 신뢰성을 보장하기 위한 연구가 주류를 이루고 있으며, 최소의 네트워크 소요시간을 갖도록 이동 에이전트의 이주 계획을 세우는 연구는 아직 미흡하다.



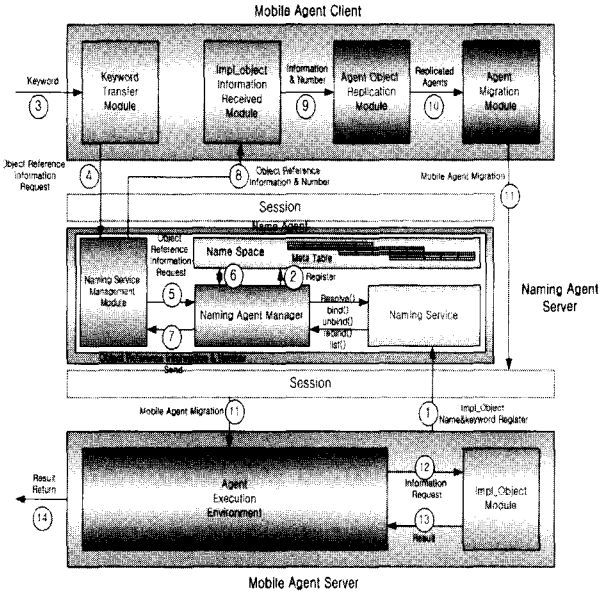
(그림 2) 기존 이동 에이전트의 이주 구조

3. 병렬 이주 모델

3.1 모델의 구조

이동 에이전트의 가장 큰 특징은 에이전트가 사용자를 대신하여 분산된 노드들을 순회하며 작업을 처리한다는 것이다. 이러한 특성을 통해 이동 에이전트의 순회 작업 처리 시간이 전체 에이전트 프레임웍(Framework)의 성능이나 효율성 향상 문제와 직결되어 있음을 알 수 있다. 에이전트의 순회 작업 처리 시간은 에이전트의 구조나 이주 정책에 따른 노드 간 이주 방법이 큰 영향을 미친다. 따라서 효율적인 이동 에이전트 시스템 개발을 위해서는 이동 에이전트를 어떻게 구성할 것인가와 어떠한 이주 방법을 통해 에이전트를 이주시킬 것인가가 매우 중요하다. 그러므로 병렬 이주 모델은 분산된 노드로의 이주시 네트워크 소요시간을 감소시키기 위한 에이전트 프레임웍(Framework)이다. 기존의 이동 에이전트 시스템에서는 대부분 이동 에이전트의 각 노드 이주를 위해 사용자에 의한 수동적 라우팅 스케줄 지정 방식을 사용하고 있으나, 제안된 모델은 네이밍 에이전트에 의해 관리되는 네임 스페이스의 메타테이블 이용하여 능동적으로 라우팅 스케줄을 지정함으로써 노드를 순회하지 않고 직접적인 노드로의 이주를 통하여 순차적 이주시 발생되는 문제와 네트워크 소요시간을 최소화할 수 있도록 하며, 각 복제된 에이전트는 호출 모듈만을 포함하므로 네트워크의 부하를 최소화하여 효율적인 에이전트의 이주가 이루어질 수 있도록 지원한다.

병렬 이주 모델 구조는 (그림 3)에서와 같이 사용자의 키워드를 전달받아 이주 노드에 대한 위치 투명성 및 초기 라우팅 스케줄을 지정하는 네이밍 에이전트 서버(Naming Agent Server), 네이밍 에이전트 서버내 네이밍 에이전트에서 획득한 정보를 기반으로 객체를 생성 및 복제하는 이동 에이전트 클라이언트(Mobile Agent Client), 복제된 이동 에이전트가 이주하여 사용자의 요청 작업을 수행할 이동 에이전트 서버(Mobile Agent Server)로 구성된다. 따라서 병렬 이주 모델에서의 이동 에이전트는 네이밍 에이전트 서버와 이동 에이전트 클라이언트와의 상호 협력을 통해 객체를 복제하여 분산된 각 노드로 이주되며, 안정적이고 신속한 정보의 능동적인 서비스를 지원한다.



(그림 3) 병렬 이주 모델의 구조

이동 에이전트 클라이언트 내에 에이전트 객체를 복제하기 위해 구성된 모듈은 사용자의 키워드를 네이밍 서비스 관리모듈에게 전달하는 전송 모듈(transfer module), 네이밍 에이전트 서버의 네임 스페이스내의 메타테이블에 등록된 키워드와 일치하는 객체 참조자의 정보를 검색한 후 객체 참조자 정보와 수를 전달받는 수신 모듈(received module), 획득된 객체 참조자 정보와 수를 기반으로 객체의 생성 및 복제를 실행하는 복제 모듈(replication module), 복제된 객체를 분산된 각 노드로 이주시키는 이주 모듈(migration module)과 같이 네 개의 모듈로 구성되어 있으며, 이러한 각 모듈의 기능은 다음과 같다.

첫째, 전송 모듈은 사용자의 의해 전달된 키워드를 구현 객체의 객체 참조자의 정보와 수를 얻기 위해 먼저, 네이밍 서비스 관리자로 사용자가 입력한 키워드를 전송한다. 이때 구현 객체의 참조자를 얻기 위해, 위 (그림 3)에서와 같이 네이밍 에이전트 관리자는 네이밍 서비스의 reslove() 메소드를 호출하고, 네임 스페이스의 메타테이블을 검색한 후 일치하는 키워드에 대한 객체 참조자 리스트의 정보와 수를 다시 네이밍 서비스 관리모듈에 전달하면, 전달된 객체의 정보를 수신 모듈에게 전달한다.

둘째, 수신 모듈은 네이밍 서비스 관리 모듈에서 전달된 객체 참조자의 정보와 수를 확인한 후, 에이전트 객체의 복제를 위해 정보를 복제 모듈로 전달한다.

셋째, 복제 모듈은 수신 모듈로부터 전달된 객체 참조자의 정보를 기반으로 먼저 원본 에이전트 객체를 생성하며 속성 값을 True로 설정하며, 이렇게 원본 에이전트 객체가

생성된 후, 원본 에이전트 객체를 복제하기 위해 컬렉션(collection)을 생성한다. 생성된 컬렉션으로부터 객체 복제를 위해 복제의 수를 초기화하고 객체 참조자 정보를 비교하여 에이전트 객체를 복제한다. 이때, 복제시 조건은 동일한 키워드의 정보를 갖는 객체 참조자의 리스트의 수 만큼 전부를 복제하게 된다면 객체 복제에 따른 비용과 시간에 관한 문제가 발생한다. 그러므로 3.3.2절에서 제시한 키워드에 의한 메타데이터 구조에서 hit count에 따른 복제 수의 임계값을 설정하여 복제 수를 제한하면, 복제의 비용과 시간을 줄일 수 있으며 각 노드에서 동일한 키워드에 대한 정보를 사용자에게 신속하게 전달한다.

넷째, 이주 모듈은 복제된 에이전트 객체들을 네트워크를 통해 이동시키기 위해 TCP/IP 기반의 네크워크 연결을 수행하며, 연결된 후 원격지로 이동될 객체와 정보는 직렬화 모드로 변환되어 해당 노드의 분산 시스템에 도착한다. 이들은 목적지에서 대응하는 역직렬화 모드로 변환됨으로써 객체의 이동을 종료하고, 동시에 네트워크 자원을 해제하여 네트워크 부하를 제거한다. 또한 제안된 모델에서 에이전트 객체는 각 이동 에이전트 서버에 하나 이상 존재 할 수 있으며, 각 서버에 존재하는 에이전트 객체는 큐에 대기하여 스케줄링에 의해 자신의 실행 순서를 기다린다.

3.1.1 모델의 통신 과정

병렬 이주 모델에서의 통신 과정은 각 구현 객체에 관련된 정보 등록, 구현 객체 정보의 요청과 전달, 객체의 생성 및 복제, 복제된 객체의 이주 과정으로 구분된다. 앞서, 3.1절에서 보인 (그림 3)의 ①~② 과정은 병렬 이주 모델의 흐름을 구성하는 각 시스템들의 기동시 각각의 시스템 내에 포함된 구현 객체(Impl_Object Name&Keyword Register)들을 네이밍 서비스를 지원하는 네이밍 에이전트 서버의 네이밍 에이전트에 등록하는 과정이다. 등록된 구현 객체 정보들은 이동 에이전트의 이주시 원격 시스템들 간의 연결이 가능하도록 위치 투명성을 제공하며, 이동 에이전트가 분산된 각 노드들을 이주하며 작업을 처리할 수 있도록 지원한다.

한편, ③~⑭ 과정은 사용자의 요구에 대한 병렬 이주 모델에서의 각 모듈 간 작업 처리 흐름으로 그 과정은 다음과 같다.

③ 먼저, 클라이언트 브라우저를 통해 사용자가 요구하는 정보들을 검색하기 위한 키워드를 입력받는다.

④ 브라우저를 통해 입력된 검색 키워드(Search Keyword)가 키워드 전송모듈(Keyword Transfer Module)에 전달되면, 키워드 전송모듈은 네이밍 서비스 관리 모듈(Naming Service Management Module)에게 키워드를 전달한다.

⑤ 네이밍 서비스 관리모듈(Naming Service Management Module)은 전달받은 키워드를 네이밍 에이전트 관리자(Naming Agent Manager)에게 키워드를 전달한다.

⑥ 네이밍 에이전트 관리자(Naming Agent Manager)는 전달된 키워드를 기반으로 객체 참조자 리스트(이하 라우팅 테이블)를 획득한다. 이 때, 라우팅 테이블은 검색 키워드와 네이밍 스페이스의 메타 테이블(Meta Table) 정보와의 비교를 통해 해당 객체 참조자들이 추출된다.

⑦~⑧ 획득한 객체 참조자의 정보와 수를 구현 객체 정보 수신모듈(Impl_Object Information Received Module)에 전달한다.

⑨ 객체 참조자 리스트로부터 획득한 정보와 수를 에이전트 객체 복제모듈(Agent Object Replication Module)로 전달하면 에이전트 객체를 생성 및 복제한다.

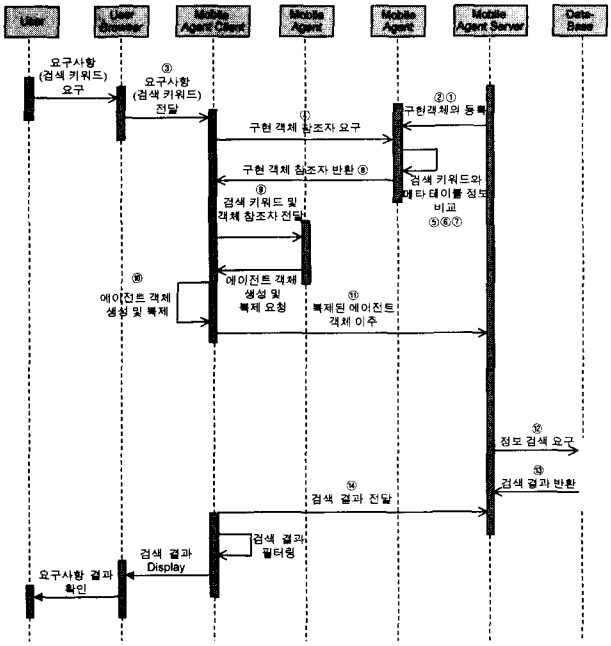
⑩ ⑨의 과정이 실행된 후, 복제된 에이전트 객체들은 에이전트 이주모듈(Agent Migration Module)로 전달된다.

⑪ 전달된 각 에이전트 객체들은 각각 다른 라우팅 스케줄 갖으며, 분산된 노드로 이주된다.

⑫ 이주된 에이전트 객체는 이동 에이전트 서버의 구현 객체 모듈(Impl_Object Module)과 연동되어 작업 실행 모듈의 흐름에 따라 데이터베이스 핸들러(Database Handler)를 이용하여 데이터베이스에 접근한다.

⑬ 데이터베이스 핸들러는 데이터베이스를 검색한 후, 해당 검색 결과를 반환한다.

⑭ 반환된 결과를 사용자에게 전달한다.

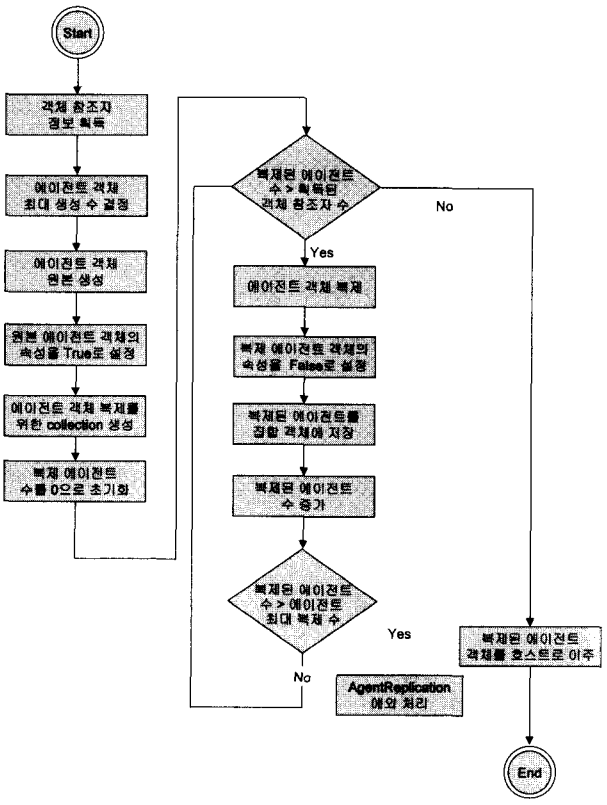


(그림 4) 병렬 이주 모델의 통신과정

(그림 4)는 사용자로부터 요구사항을 입력받아 이를 처리하고자 네이밍 에이전트의 메타테이블내에 객체 참조자 정보의 요청 및 반환과, 이러한 정보와 수를 통해 복제된 에이전트 객체가 이주하는 병렬 이주 모델의 통신 과정을 시퀀스 다이어그램으로 나타내었다.

3.1.2 객체 복제 알고리즘

병렬 이주 모델은 기존 이동 에이전트의 순차적 이주 방식으로 인한 문제와 네트워크 소요시간을 감소시키기 위한 모델이다. 제시된 (그림 5)는 객체 복제에 위한 알고리즘이다.



(그림 5) 객체 복제 알고리즘

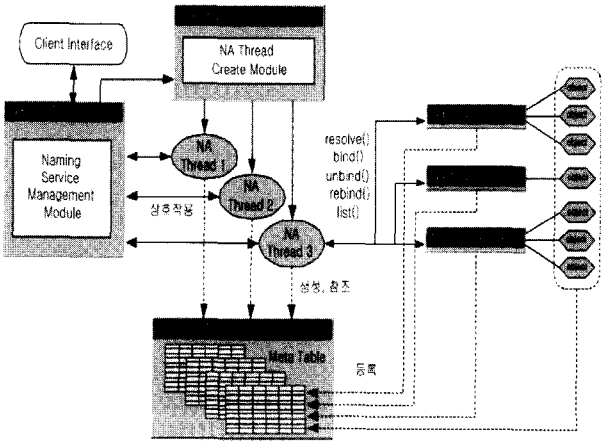
이동 에이전트 클라이언트에서 사용자의 키워드를 네이밍 에이전트 서버에게 보내면 네이밍 에이전트의 메타테이블을 검색한 후 일치하는 키워드에 대한 객체 참조자의 정보와 수는 이동 에이전트 클라이언트의 수신 모듈로 전달된다. 이렇게 전달된 객체 참조자의 정보와 수는 에이전트 객체 복제의 수를 결정하게 되고, 에이전트 객체의 원본을 생성하며 복제를 위한 컬렉션 생성과 더불어 객체 복제를 실행하게 된다. 복제 실행과정에서 복제 에이전트는 속성을 False로 설정하고, 복제된 에이전트의 수가 에이전트 최대 복제 수를 넘기면 에이전트 복제 예외처리를 실행한다. 복제 에이전트 객체 수는 설정한 임계값에 일치하게 되면 복제를 중지하고 에이전트의 이주를 위한 준비를 실행하며 분산된 각 이동 에이전트 시스템으로 이주된다.

3.2 네이밍 에이전트 서버

기존의 이동 에이전트 시스템에서는 이동 에이전트의 각 노드 이주를 위해 사용자에 의한 수동적 라우팅 스케줄 지정 방식을 사용하고 있으나, 제안된 모델에서의 네이밍 에이전트 서버는 서버내 네이밍 에이전트가 노드 이주를 위한 위치 투명성을 보장하는 능동적 라우팅 테이블을 지원한다. 이러한 네이밍 에이전트는 Naming Agent Impl, Naming Agent Manager, Name Space, Namimg Service로 구성되어 있다.

3.2.1 네이밍 에이전트 설계

네이밍 서비스는 클라이언트가 구현 객체를 찾기 위해 구현 객체의 이름을 사용하여 구현 객체의 객체 참조자를 얻을 수 있는 분산 객체 서비스이다. 장점으로는 사용자가 구현 객체의 이름을 알고 구현 객체가 등록된 네이밍 서비스의 객체 참조자나 IOR(Interoperable Object Reference)을 안다면 구현 객체와 쉽게 연결할 수 있으며, 등록된 구현 객체를 트리 형태의 네이밍 그래프로 관리한다. 이러한 트리 구조를 이루는 요소는 파일 시스템에서 디렉토리와 같은 네이밍 컨텍스트와 파일이름과 같은 구현 객체 이름으로 구성된다. 따라서 구현 객체는 이름과 객체의 쌍으로 구성된 고유의 이름을 네이밍 서비스에 등록하고 클라이언트는 객체의 이름으로 구현 객체를 참조한다. 네이밍 서비스는 분산된 객체의 객체 참조자를 쉽게 얻을 수 있도록 객체 참조자를 한곳에 모아 이용 가능하게 하여 분산 시스템 환경에서의 객체에 대한 위치 투명성을 보장한다[24]. 그러므로 본 논문에서는 병렬 이주를 위한 검색에 사용되는 키워드들을 저장하기 위한 네임 스페이스를 생성하여 객체 참조자와 계층적 검색 키워드들을 저장한다. (그림 6)은 네이밍 서비스별로 쓰레드 할당을 통해 객체 참조를 유지하고 관리하는 네이밍 에이전트의 구조를 나타낸다. 여기서, 계층적 검색 키워드는 데이터베이스의 테이블명과 주요 필드명, 테이블 관계 정보 등에 대한 데이터이다.

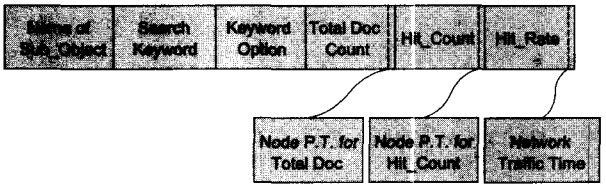


(그림 6) 네이밍 에이전트의 구조

Naming Agent Impl은 네이밍 에이전트 시스템으로부터 확장되어 네이밍 서비스에 대한 접근을 담당하는 실질적인 구현이며, 네이밍 에이전트 매니저(Naming Agent Manager)는 구현 객체의 등록 요청에 따라 네이밍 에이전트 스레드(Thread)를 생성한다. 네이밍 에이전트 스레드는 등록 에이전트에 대한 메타데이터를 생성하고 네이밍 에이전트와 네이밍 서비스의 연결을 담당한다. 기존의 네이밍 서비스에서는 등록된 구현객체의 객체 참조자를 획득하기 위해 해당 구현객체의 이름을 알고 있어야 하는 문제가 있었으나, 본 논문에 사용된 네이밍 에이전트는 새로운 네임 스페이스의 추가 생성을 통해 사용자에 의해 요청된 검색 키워드만으로도 구현 객체의 객체 참조자를 얻을 수 있다. 또한, 검색 키워드 정보를 통해 보다 효율적이고 신속한 검색 서비스를 제공할 수 있다.

3.2.2 키워드에 의한 메타데이터 구조

네이밍 에이전트는 키워드 정보에 의한 메타데이터를 제공한다. 키워드별 메타데이터의 각 필드는 노드 이주와 객체 복제의 정보를 제공한다. 검색 키워드를 통한 접근 방식의 메타데이터는 검색 키워드를 입력하여 다수의 객체 참조자를 얻은 후 적중률을 비교하여 임계값에 의해 객체 복제 수를 결정하도록 한다. (그림 7)은 검색 키워드에 따라 구현 객체의 이름과 검색 키워드, 해당 문건의 총 수, 검색된 문건의 수 등으로 구성된 검색 키워드 별 메타데이터 구조를 나타낸다.

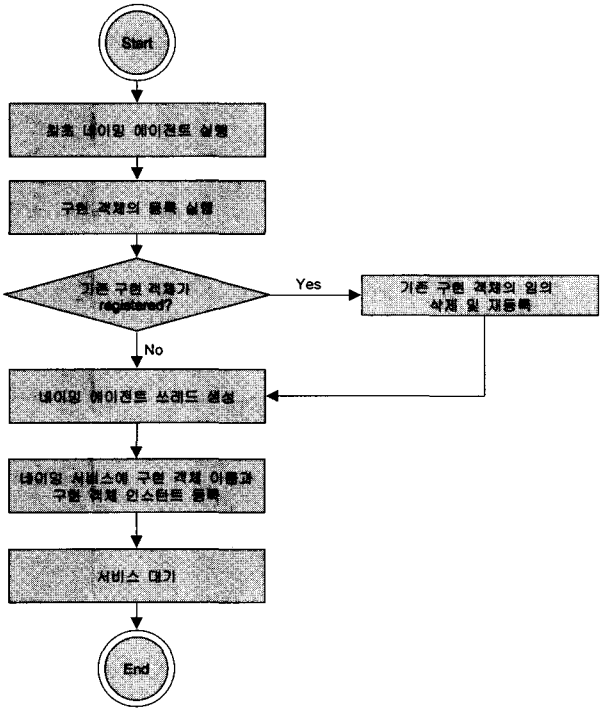


(그림 7) 키워드에 의한 메타데이터 구조

검색 키워드 별 메타데이터 필드 중, 전체 문건 및 적중 문건에 대한 노드 처리 시간, 이주된 서버에서 클라이언트 측으로 보내어지는 적중된 다수의 문건에 대한 네트워크 시간 등은 노드 이주의 또 다른 주요 정보로 이용될 수 있다. 그러나 외부 요인의 간섭으로 객관적이고 정확한 시간을 알기 어렵기 때문에 본 논문에서는 고려되지 않았다. 이 메타데이터의 이용은 단지 사용자에 의해 동적으로 검색 키워드만 주어지면 된다. 네이밍 에이전트에 사용자가 입력한 검색 키워드만을 전달하고, 메타데이터 테이블 내에 이미 등록된 모든 구현 객체들의 검색 키워드를 비교한 후 일치하는 키워드를 가진 모든 구현 객체의 이름을 추출한다. 추출된 구현 객체의 이름들을 통해 구현 객체 이름을 통한 접근 방식에 사용된 메타데이터 테이블로부터 각각 해당 네이밍 서비스의 객체 참조자를 획득한다.

3.2.3 메타데이터 생성

구현 객체의 등록에 의해 메타데이터는 생성된다. 따라서 구현 객체의 위치 정보를 관리하는 네이밍 에이전트는 가장 먼저 실행되어 각 에이전트의 요구에 따라 에이전트 이름을 등록하고 생성된 메타데이터의 각 필드를 초기화한다. (그림 8)은 메타데이터 생성 알고리즘을 나타낸다.



(그림 8) 메타데이터 생성 알고리즘

네이밍 에이전트는 각 구현 객체의 위치 정보를 관리하므로 제일 먼저 실행되어 구현 객체 이름을 등록하고 관계한 메타데이터의 각 필드를 초기화 한다. 이 때 네이밍 에이전트는 같은 이름으로의 에이전트 등록을 방지하고 중복을 제거하여 분산 객체의 위치에 대한 신뢰성을 제공해야 한다. 따라서 구현 객체의 이름 충돌 시 예외 기능을 이용하여 기존 구현 객체의 삭제 및 재등록 과정을 수행한다. 동시에 발생하는 구현 객체의 등록 요구에 대해 네이밍 에이전트는 스레드별로 생성한다. 생성된 스레드들은 상호 동기화 시켜 잘못된 내용에 대한 read, write를 방지한다. 등록되는 이동 에이전트 서버의 구현 객체는 키워드별 메타테이블에 해당 구현 객체를 등록하고 등록된 구현 객체에 대한 키워드별 메타데이터의 각 필드는 초기화된다. 구현 객체에 대한 키워드별 메타데이터 생성은 구현 객체가 실행모듈을 가지고 있으며 그 실행 결과에 따라 이동 에이전트의 키워드에 따른 노드 이주에 중요한 정보를 제공하기 때문이다. 모든 구현 객체는 고유한 이름을 갖고 이름별 메타테이블에 등록된 후 각 구현 객체의 상호 접근을 제공하게 된다.

4. 실험 및 평가

본 장에서는 3장에서 설계한 병렬 이주 모델을 기반으로 에이전트의 네트워크 소요시간 분석은 순차적 이주와, 제안된 병렬 이주 방식에 대한 실험을 통해 두 시스템에서의 네트워크 소요시간을 측정한 후 이를 비교한다. 이러한 비교, 분석을 위해서 이동 에이전트 개발 툴로 IBM Japan research group에서 개발한 Java Aglet 개발환경인 Aglet Workbench[14]를 사용하였고 또한, 제안된 모델에서의 에이전트 네트워크 소요시간 측정 실험은 전체 네트워크의 대역폭이 100Mbps이고, 노드 간 거리가 15m~20m인 네트워크 환경하에서 수행하였다. 본 논문에서 실험한 환경은 <표 1>과 같다.

〈표 1〉 이주 시간 측정 실험 환경

기 종	CPU	Memory	OS	DBMS	IP
Pentium Ⅳ	2.0GHz	512Mbytes	Windows 2000 Server	Oracle 8i	142
Pentium Ⅳ	1.2GHz	512Mbytes	Windows XP	Oracle 8i	144
Pentium Ⅲ	850MHz	512Mbytes	Windows 2000 Professional	Oracle 8i	143
Pentium Ⅲ	800MHz	384Mbytes	Windows 2000 Professional	Oracle 8i	148
Pentium Ⅱ	750MHz	256Mbytes	Windows 2000 Professional	Oracle 8i	146
Pentium Ⅱ	700MHz	256Mbytes	Windows 2000 Professional	Oracle 8i	153,152

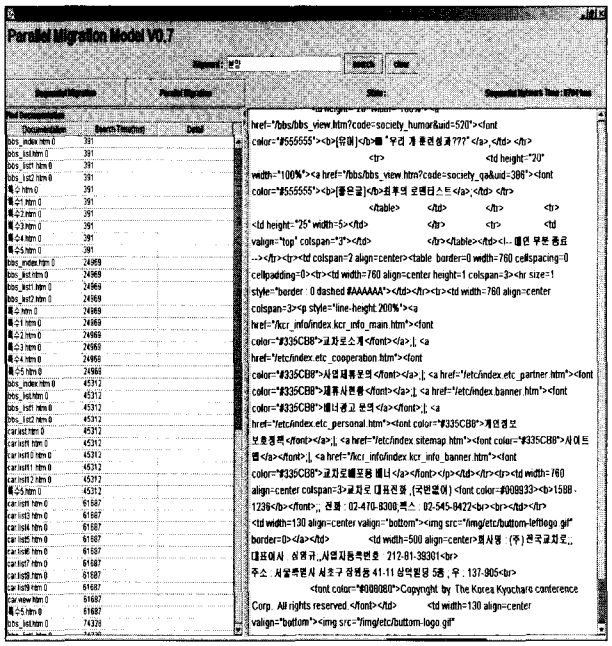
네트워크 소요시간 측정은 순수 에이전트 네트워크 소요시간을 측정하기 위한 실험이지만, 실제 순수 에이전트 네트워크 소요시간을 측정하는 자체가 환경적인 여건상 불가능함으로 실험 환경을 다음과 같이 가정하였다. 첫째, 균등 네트워크 환경이다. 둘째, 방문하는 노드의 수, 노드내의 문서의 수와 크기 등은 동일하다. 셋째, 이동 에이전트가 해당 호스트로 이주 후에는 호스트의 결점이나 장애가 발생하지 않는다.

4.1 순차적 이주 방식 따른 분석

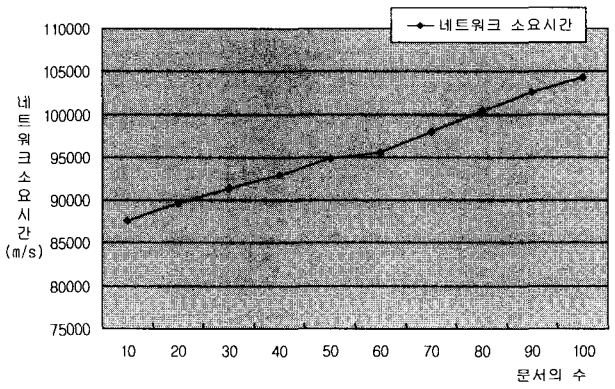
순차적 이주 방식에 따른 네트워크 소요시간 분석은 이동 에이전트 시스템의 각 노드로 이동 에이전트를 순차적으로 이주시켜 문서 검색의 네트워크 소요시간을 측정하며, 본 실험에서는 이동 에이전트 시스템으로의 이주 방식에 따른 차이를 최소화하기 위해, 동일한 라우팅 스케줄에 따라 네트워크 소요시간을 측정한다. 순차적 이주 방식의 경우, 이동 에이전트가 각 노드를 이주하며 작업을 처리한 결과를 포함하고 이주를 수행함으로 에이전트 실행 모듈의 크기가 커져 네트워크 소요시간이 많이 소요되며, 또한 작업 처리 결과량이 많아 에이전트 크기 증가로 네트워크 소요시간이 증가한다. (그림 9)는 사용자가 키워드를 입력하고 순차적 이주 방식을 선택하여 검색된 문서와 네트워크 소요시간을 나타낸 화면이다. <표 2>는 각 노드에 저장되어 있는 문서의 수에 따라 네트워크 소요시간을 측정한 것이다. (그림 10)은 <표 2>의 결과를 그래프로 표현하였다.

〈표 2〉 문서 수에 따른 네트워크 소요시간

문서의 수	10	20	30	40	50	60	70	80	90	100
네트워크 소요시간 (m/s)	87641	89609	91406	92969	94922	95641	97953	100406	102672	104281



(그림 9) 순차적 이주 방식에 따른 결과 화면



(그림 10) 네트워크 소요시간 측정 그래프

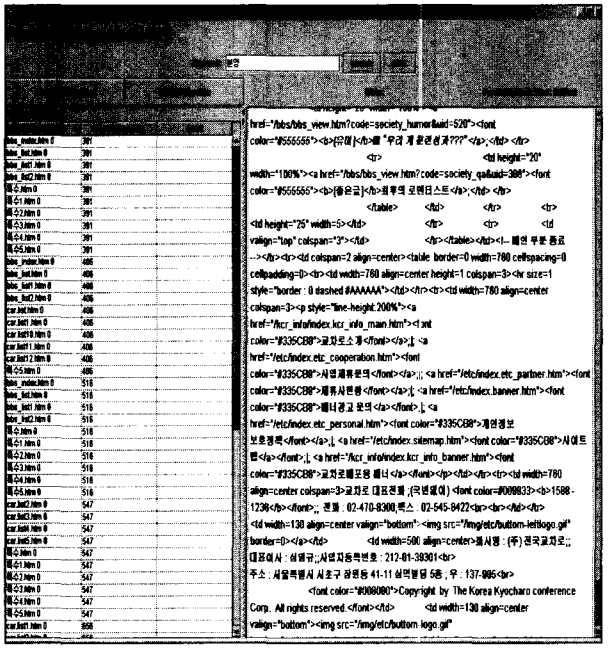
4.2 병렬 이주 방식에 따른 분석

병렬 이주 방식 따른 이주 소요시간 분석은 이동 에이전

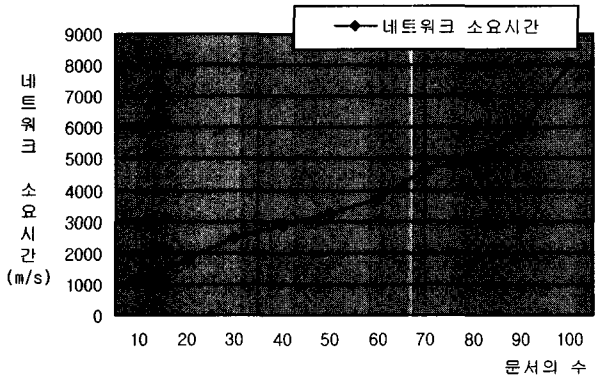
트 시스템의 각 노드로 이동 에이전트를 병렬로 이주시켜 문서 검색의 네트워크 소요시간을 측정한다. 순차적 이주

〈표 3〉 문서의 수에 따른 네트워크 소요시간

문서의 수	10	20	30	40	50	60	70	80	90	100
네트워크 소요시간 (m/s)	1182	1782	2500	2906	3219	3750	4672	5031	5984	8078



(그림 11) 병렬 이주 방식에 따른 결과 화면



(그림 12) 네트워크 소요시간 측정 그래프

방식의 경우, 고정적인 라우팅 스케줄에 따라 각 노드를 이주하며 작업을 처리함으로 에이전트의 크기가 커져 네트워크 소요시간이 증가하는 반면, 제안된 모델은 병렬로 이주됨에 따라 네트워크 소요시간은 앞서 측정한 순차적 이주 방식 보다 상당히 감소된다. 이는 실제 에이전트의 크기에 따른 트래픽의 영향을 받지 않기 때문에 순차적 이주 방식보다 네트워크 소요시간보다 적게 측정되는 것이다. (그림 11)은 병렬 이주 방식에 따른 결과 화면이다. <표 3>은 각 노드에 저장되어 있는 문서의 수에 따라 네트워크 소요시간을 측정한 것이다. (그림 12)는 <표 3>의 결과를 그래프로 표현하였다.

4.3 실험 결과

위와 같이 두 이주 방식의 실험에서 네트워크 소요시간을 비교하면 순차적 이주 방식은 <표 2>와 같이 각 노드에 분포되어 있는 문서를 검색하는데 문서의 개수에 따라 최저 87641(m/s) 에서 최고 104281(m/s)의 네트워크 소요시간이 소요되었다. 따라서 순차적 이주방식은 문서의 수가 증가하거나 문서의 크기가 변화함에 따라 결과가 더욱 누적되고, 에이전트의 크기 또한 증가하여 노드간의 네트워크 소요시간을 증가시킨다. 또한, 방문할 노드의 수가 증가할수록 더욱 더 많은 처리 결과를 누적시키므로 소요시간은 해당 노드 수에 비례하여 더욱 증가하게 될 것이다. 그러나 제안된 병렬 이주 방식은 순차적 이주 방식과는 달리 분산된 노드로 각각 개별적으로 이주되기 때문에 <표 3>과 같이 동일한 문서를 검색하는데 문서의 개수에 따라 최저 1182(m/s)에서 최고 8078(m/s)의 네트워크 소요시간이 소요되었다. 따라서 병렬 이주 방식은 순차적 이주 방식보다 31%정도의 네트워크 소요시간을 감소시킬 수 있었다. 하지만 제안된 이주 방식은 첫째, 노드의 수가 증가할수록 객체 복제의 수 또한 증가하므로 복제에 따른 비용 증가 문제와 둘째, 키워드에 의한 메타데이터의 hit_count에 의해 hit율이 높은 순으로 객체를 복제 후 이주시, 다른 사용자 요청으로 동일한 키워드에 대한 검색 서비스를 요구할 때 같은 hit_count 정보에 의해 동일한 라우팅 테이블이 생성되므로 특정 노드로의 병목현상이 발생할 수 있다. 그러므로 이러한 문제점을 해결하기 위해 복제에 따른 비용을 메타테이블에 등록된 해당 노드의 hit_count의 정보를 기준으로 복제 수의 임계값을 설정하여 제한한다면 동일한 정보의 검색에 따른 객체의 복제 수와 비용을 감소시킬 수 있으며, 병목현상의 발생을 해결할 수 있을 것이다. 하지만 이러한 문제점들에 대해서는 실험 환경상 본 논문에서 고려하지 않았다. 그리고 순차적 이주 방식의 문제인 호스트의 결점과 장애시 문제는 제안된 모델에서 이동 에이전트가 이주하기 전 처음 수행되어야 할 과정이 이동 에이전트 서버의 구현객체의 정보가 등록되어하므로 결점과 장애가 발생된 호스트는 네이밍 에이전트에 자신의 구현객체 정보를 등록하지 못하므로 해당 호스트로의 이주를 위한 능동적 라우팅 테이블이 생성되지 않는다. 따라서 이동 에이전트가 해당 호스트로 이주되지 않으므로 무한 대기상태나 고아상태가 발생하지 않는다.

5. 결론 및 향후 연구 과제

기존 이동 에이전트의 순차적 이주 방식으로 인해 발생하는 호스트의 결점이나 장애 등으로 인하여 이동 에이전트가 무한 대기상태나 고아상태에 빠져 네트워크 소요시간이 증가하기 때문에 분산 시스템의 전체 성능에 큰 영향을 주는 요소는 에이전트의 이주방식에 있다. 따라서 본 논문에서는 이동 에이전트의 네트워크 소요시간을 감소시키기 위해 객체 복제 통한 이동 에이전트의 병렬 이주 모델을 설계 및 구현하였다. 제안된 모델은 기존 이동 에이전트의 순차적 이주 방식으로 인해 발생하는 문제와 네트워크 소요시간을 감소시키기 위해 기존의 수동적 라우팅 스케줄 지정 방식에서 탈피하여 능동적인 라우팅 스케줄 지정과 객체의 위치 투명성으로 신뢰성을 보장하는 네이밍 에이전트를 설계하였으며, 이러한 네이밍 에이전트의 메타테이블을 기반으로 에이전트 객체를 복제하여 각각 개별 노드로 에이전트 객체를 병렬로 이주하는 방식을 제안하였다. 따라서 제안된 이주 방식은 분산 환경에서의 순차적인 이주 접근 방식과는 달리 에이전트 객체가 각각 개별 노드로 병렬 이주됨으로써 전체 노드로의 네트워크 소요시간을 감소와 호스트의 결점 및 장애의 문제를 극복할 수 있었다.

향후 연구과제로는 제안된 모델을 기반으로 하는 응용 시스템의 개발이 요구되며, 노드 수의 증가에 따른 객체 복제의 수 및 비용과 hit율에 의한 이주시 병목현상에 대한 연구가 지속적으로 수행되어야 한다.

참 고 문 헌

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

[2] 전병국, 최영근, "이동 에이전트를 위한 효율적인 이주 정책 설계 및 구현", 한국정보처리학회 춘계학술발표논문집, April, 1999.

[3] General Magic, "Odyssey, http://www.genmagic.com/agents/, 2000.

[4] ObjectSpace, "ObjectSpace Voyager, GeneralMagic Odyssey, IBM Aglets : A Comparison," VoyagerTM, 1997.

[5] K. A. Baharat and L. Cardelli, "Migratory Applications," Proceedingsof the 8th annual ACM symposium on user interface and software technology, November, 1995.

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

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

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

[9] 전병국, 이근상, 최영근, "Java 언어를 이용한 객체 이동시스템의 설계 및 구현", 정보처리논문지, 제6권 제1호, January, 1999.

[10] 권혁찬, 유우종, 김홍완, 유관종, "데이터 마이닝을 위한 이동 에이전트의 효율적인 이주전략", 정보처리논문지, 제7권 제5호, March, 2000.

[11] Robert S. G, "Agent Tcl : A flexible and secure mobile-agent system," TR98-327, Dartmouth Col., June, 1997.

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

[13] B. Venners, "The Architecture of Aglets," JavaWorld, http://www.javaworld.com/javaworld/jw-04-1997/jw-04-hood.html, 1997.

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

[15] S. Choy, T. Magedanz, "Grasshopper Technical Overview," IKV++ GmbH, 1999.

[16] IKV++, "A CORBA environment supporting Mobile Agent," IKV++ GmbH, 1999.

[17] David Ratner, Peter Reiher, Gerald J. Popek, and Geoffrey H. Kuenning, "Replication Requirement in Mobile Environment," Kluwer Academic Publishers, pp.525-533, 2001.

[18] Jason Maassen, Thlio Kielmann and Henri E. Bal, "Efficient Replicated Method Invocation in Java," ACM, pp. 88-96, 2000.

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

[20] Rushikesh K, Joshi. O, Ramakrishna. D, and Janaki Ram, "A Programming Model for Service Replication in Distributed Object Systems," Journal of PDC, pp.1-12, March, 1999.

[21] OMG, "Mobile Agent System Interoperability Facilities Specification," OMG TC Document orbos/97-10-05, 1997.

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

[23] 윤동식, 이병관, "객체 복제 기법에 의한 원격 접근 알고리즘", 정보처리논문지, 제7권 제3호, March, 2000.

[24] 임찬순, 이만희, 석우진, 변옥환, "안전한 네이밍 서비스 클라이언트의 설계 및 구현", 정보처리논문지, 제6권 제2호, October, 1999.

김 광 종

e-mail : kkjkim@kunsan.ac.kr

1993년 군산대학교 컴퓨터정보과학과(학사)

1999년 군산대학교 대학원 컴퓨터정보과학과(이학석사)

2003년 군산대학교 대학원 컴퓨터정보과학과(박사수료)

관심분야 : 이동 에이전트, 분산객체시스템, 능동 데이터베이스

이 연 식

e-mail : yslee@kunsan.ac.kr

1982년 전남대학교 전자계산학과(학사)

1984년 전남대학교 대학원 전자계산학과(이학석사)

1994년 전북대학교 대학원 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