Java 언어를 이용한 객체이동시스템의 설계 및 구현

전 병 국† · 이 근 상† · 최 영 근††

요 약

분산 컴퓨팅의 발전으로 인한 분산 객체 컴퓨팅(distributed object computing) 기술은 네트워크(network) 상의 동일 및 이기종 시스템간 분산처리 능력을 개선하였다. 그러나 분산 객체 컴퓨팅 기술조차도 분산 컴퓨팅의 근본적인 문제인 다수의 클라이언트(client)와 서버(server)에 의한 많은 요구와 네트워크 과부하 문제 등은 해결되지 않고 있다.

본 논문은 이러한 문제를 해결하기 위한 방안으로 자바(Java) 언어를 이용하여 객체이동시스템을 설계하고 구현한다. 이를 위해, 먼저 객체의 상태와 코드를 유지하는 이동객체 모델에 대한 특성을 제안한다. 구현된 객체이동시스템은 이동객체가 특정 노드(node) 등으로 이동되었을 때 해당 객체를 수용하며, 자동적으로 수행 가능한 가상 장소를 지원한다. 따라서, 기존의 분산 컴퓨팅이 갖고 있는 문제들에 대한 해결과 동형/이기종 간에도 객체 이동시 객체 이주 투명성을 제공한다.

Design and Implementation of an Object Migration System Using the Java Language

Byung-Kook Jeon† · Keun-Sang Yi† · Young-Keun Choi††

ABSTRACT

Distributed Object Computing, owing to the development of distributed computing, has improved the performance of distributed processing conducted between homogeneous and heterogeneous systems in network. However, it has failed to solve fundamental problems such as network overload and enormous requests demands by servers and clients.

In this paper, we propose to design and implement an Object Migration System that uses the Java language to tackle the mentioned problems. As the first step of the implementation of the system, we justify the characteristics of the mobile object model that keeps codes and states of an object. Implemented Object Migration System would accept objects being migrated to a specific node and support the virtual place in which objects could be executed automatically. Therefore, the Object Migration System we suggest could not only solve problems imposed to traditional distributed computing but also offer transparency of object migration between homogeneous and heterogeneous systems.

1. 서 론

분산 컴퓨팅(distributed computing) 기술과 객체 지향 기술의 발전은 인터넷(Internet) 상의 동일 기종간의 분산처리뿐만 아니라, 이기종간에도 분산 컴퓨팅이 가능한 분산 객체 컴퓨팅(distributed object computing)으로 발전되었다. 이를 대표하는 OMG의 CORBA 명세에 따른 분산 객체 컴퓨팅은 실행 단위를 객체로 분리하고, 객체간의 메시지 통신을 통해 객체간 상호 운용성(interoperability)을 보장하고 있다. 그러나, CORBA에서 제안한 분산 객체 컴퓨팅도 단지 네트워크를 통

† 준 회 원 : 광운대학교 대학원 전자계산학과
†† 정 회 원 : 광운대학교 전자계산학과 교수
논문접수 : 1998년 6월 1일, 심사완료 : 1998년 11월 4일

간 이기종간의 상호 운용성만을 해결한 상태이며 기존의 분산 컴퓨팅처럼 다른 노드(node)에 서버(server) 역할을 하는 코드(code), 즉 서버 객체(object)가 존재해야 한다. 대부분의 서버 객체의 존재 및 정상적인 활동이 보장되지만 그렇지 못할 경우 수많은 클라이언트(client)의 요구에 응할 수 없는 문제를 일으키게 된다. 또한 데이터 전송 및 접속 유지에 따른 네트워크 오버헤드를 해결하지 못하고 있다. 이러한 문제를 해결하는 방안중 하나로서, 최근의 Web과 분산 컴퓨팅을 결합한 Web 기반 분산 컴퓨팅 기술이 있다. Web 기반 분산 컴퓨팅이란 HTTP(Hyper Text Transfer Protocol)상에서 실행 코드(execution code)를 이동시키는 방법[15]이다. 그러나, 이 방법은 간단한 데이터를 보내기 위해 데이터의 양보다 더 많은 양의 코드를 가져와야 하는 Web의 근본적인 문제가 있기 때문에 분산 객체 컴퓨팅의 문제 중 접속 유지에 따른 네트워크 오버헤드 문제만을 다소 해결하는데 불과하다. 현재 이런 분산 객체 컴퓨팅의 근본적인 문제점인 서버 객체의 존재 보장과 실행 코드의 다운로드에 따른 네트워크 과부하를 해결하고자 많은 연구가 진행 중에 있다[1, 5, 11, 15, 17].

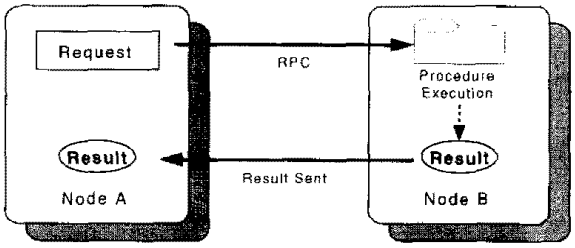
본 논문에서는 제기된 분산 객체 컴퓨팅의 단점을 해결하기 위한 방법으로 객체 단위의 실행 모듈을 네트워크를 통해 다른 노드로 자율적으로 이동시키며, 이동된 객체가 자율적(autonomous)으로 수행되기 위한 기반 환경을 제공하기에 해당 각 노드에 서버 코드가 필요없는 시스템을 설계 및 구현한다.

본 논문의 구성으로, 2장에서는 객체 이동에 관한 기존 연구들을 분석한다. 3장에서는 객체 이동을 지원하는 객체 이동 시스템을 설계 및 기능을 제안하며, 4장에서는 이에 대한 구현 및 실례를 보인다. 끝으로 5장에서는 앞으로의 연구 방향을 논한다.

2. 객체이동(Object Mobility)

2.1 분산 시스템

기존의 분산 컴퓨팅은 각자의 역할을 명확하게 구별하는 모델인 클라이언트와 서버 개념을 이용해 왔다. 하지만 클라이언트/서버 모델은 서버가 제공하는 서비스만을 제공받을 수 있는 단점이 있으며, 클라이언트와 서버간의 통신인 메시지 전달(message passing)을 사용하기 위해 네트워크상의 정확한 주소(address)와 시스템 동기화(synchronization) 기능을 반드시 제공해야만 하는 단점이 있다[1]. 이런 클라이언트/서버 모델은 저수준 프로토콜을 사용하는 피어-투-피어(peer-to-peer) 방식과 현재 대부분의 클라이언트/서버 모델에서 사용하는 RPC 기반 모델이 있다. 또한 최근들어 주목받기 시작한 각기 다른 시간대와 성능을 가진 시스템들을 비동기적으로 연결하는 방식인 MOM (Message-Oriented Middleware)이 있다. 그중 클라이언트/서버 모델에 RPC개념을 적용한 모델은 이런 문제를 해결하기 위해 클라이언트가 지역 함수 호출(local function call)과 같은 방법으로 서버의 서비스를 받을 수 있게 하므로, 데이터 전송을 줄여 네트워크 오버헤드를 줄이고 네트워크 투명성을 제공하고 있다(그림 1). 그러나, RPC 모델 또한 클라이언트의 수가 증가하거나 서버의 서비스가 클라이언트의 요청에 적합하지 않을 때는 서버의 성능저하와 서비스 부재 문제가 발생하게 된다.



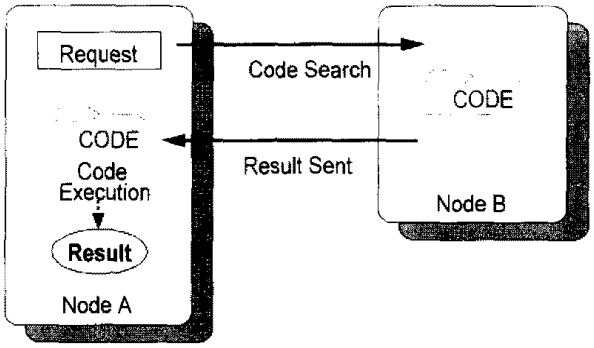
(그림 1) RPC 모델
(Fig. 1) An RPC Model

이와 같은 문제를 해결하는 방법 중 하나로서 OMG의 CORBA 명세에 따른 클라이언트/서버 시스템이 있다. CORBA는 클라이언트/서버 개념에 재사용성, 상속과 캡슐화(encapsulation)의 객체 지향 이론을 도입하고, 보안과 인증 문제를 소켓(socket) 대신에 사용자 레벨 쓰래드(user level thread) 인터페이스(interface)를 제공하는 분산 객체 컴퓨팅 환경을 제시하고 있다[11]. 이 방법은 클라이언트 객체가 서비스 제공자, 즉 서버가 위치하는 곳의 부프로그램(sub-program) 객체를 구동시키기 위해 서버의 정확한 주소나 구동전 동기화가 필요하지 않는 객체 위치 투명성을 제공하고 있다. 또한 최근에는 실행가능한 부프로그램 객체를 원격 노드(remote node)로 이동시켜 컴퓨팅하는 연구가 활발하게 진행되어 CORBA의 범용성으로 인한 서비스 제공의 제약과 완벽하지 못한 상호 운영성의 문

제를 해결하고 있다[2, 5, 9, 15]. 그러나, 부프로그램 객체 자신이 다른 노드로 이동이 가능해도 이동된 노드의 시스템 자원을 보안 문제로 원활하게 이용하지 못하는 단점이 있다. 따라서, 객체를 이동시키고, 이동된 객체를 구동하는 방법과 구동된 객체가 시스템 자원을 동적으로 할당받는 등의 환경 즉, 이동 객체가 시스템에서 활동할 수 있는 가상 장소(place)를 제공해 해결이 가능하다[11, 12, 16].

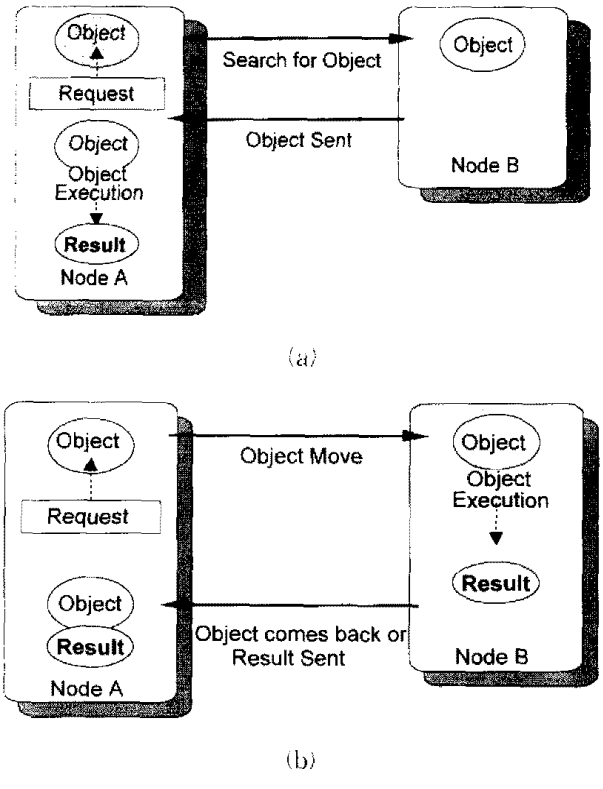
2.2 이동객체 시스템

객체 이동은 객체가 실행되는 환경에 따라 분류할 수 있다[5, 9, 14, 16]. 가장 간단한 모델인 RPC 모델은 (그림 1)과 같이 제어를 가진 객체가 동일 노드에서 실행되고 다른 노드의 실행 모듈을 제어해 결과만을 돌려 받는 방법이 있다. 이 방법은 객체가 이동되기보다는 제어를 이동하는 방법이다. 대표적으로 RPC를 이용한 클라이언트/서버가 있다. 두 번째 모델은 (그림 2)처럼 특정 노드에서 코드를 찾아서 코드만을 자신의 노드에 이동시킨 후 실행하는 방법이다. 이 방법은 객체의 상태를 유지할 수 없고 다시 실행되는 노드에서 새롭게 초기화(initialize)를 시켜야만 하는 것으로 컴퓨팅의 연속성 문제를 가지고 있다. Web 기반의 자바(Java)의 애플릿(Applet)이 대표적이다.



(그림 2) 코드를 다운(down load)받아 실행하는 방법
(Fig. 2) The executing mechanism downloading a code

또 다른 모델은 (그림 3)의 (a)같이 실행에 필요한 객체를 다른 노드에서 이동시켜 자신의 노드에서 실행시키는 방법이다[9, 10, 13]. 이때 이동되는 객체는 실행중이므로 객체의 상태와 값을 이동된 후에도 유지해야만 하며 다른 노드에서 실행중인 객체를 찾는 또 다른 객체를 필요로 한다. 이러한 방식은 프로세스 이주기법(process migration technique)이라고도 하며 객체의 상태와 값을 유지하지 않아도 되는 모델은 자바의 애플릿이 있다. 다음 (그림 3)의 (b)는 실행객체(running object)가 자신의 상태와 실행에 대한 결과(result)를 가지고 이동되며 실행하는 것을 나타낸다[11]. 이러한 방법은 객체의 이동시와 결과값의 전송시만 네트워크를 이용하여 결과적으로 네트워크 오버헤드를 줄일 수 있다. 또한 컴퓨팅 객체가 다른 노드를 이동해 가며 실행되기 때문에 (그림 3) (a)와 같이 이동을 알려줄 노드 A의 또 다른 객체를 필요로 하지 않는다. 결국 클라이언트/서버 모델에서 단점으로 지적되어온 서버의 성능 저하 및 서비스 부재문제를 해결하면서 네트워크 오버헤드를 줄이고 있다[3, 6, 7, 8, 11]. 따라서 본 논문에서는 (그림 3) (b)와 같은 모델을 적용하여 객체의 이동과 객체가 활동할 수 있는 장소를 제공하는 시스템을 설계, 구현하고자 한다.



(그림 3) (a) 실행되는 객체를 이동시키기 위해 또 다른 객체를 필요로 한다.
(b) 객체가 다른 노드로 이동해 결과 또는 결과와 함께 원래 노드로 복귀한다.
(Fig. 3) (a) Need of another object to migrate the running object
(b) Object is to be migrated, return with the results only or both results and itself.

3. 객체이동시스템 설계 및 구성

본 논문에서는 객체의 상태와 코드를 유지하는 이동객체 모델에 대한 특성과 이동객체가 특정 노드 등으로 이동되었을 때 객체를 수용하며 자동적으로 수행 가능한 장소를 제공하는 객체이동시스템(object migration system)을 설계하고 각 모듈에 대한 기능을 제안한다.

3.1. 이동객체 모델

이동객체는 본 논문에서 구현된 객체이동시스템으로 이동될 수 있도록 실행 코드와 실행에 필요한 데이터 및 실행상태 등 여러 정보를 가지고 있다. 본 논문의 이동객체 모델은 다음과 같은 특성을 만족한다.

1) 이동객체 : 이동객체는 불변해야 하며, 이미 다른 노드에 존재할 수도 있고 재 전송될 수 있다. 본 논문에서는 이동객체 코드를 클래스 단위로 적용하였으며 이동을 위해 바이트 단위의 코드 스트림(stream)으로 자동 변환되어 목적지에 전송된다.

2) 객체상태 : 객체 초기화 혹은 실행 중에 요구되는 상태(state)를 갖는 정보로서, 객체 이주후에도 목적지에서 실행될 때 해당 객체에 필요한 정보이다. 여기서 필요한 정보란 객체가 임의 노드에서 실행된 후 다른 노드로 이동하여 실행될 때, 이전 노드에서의 해당 객체 실행 결과를 갖고 있는 정보는 이동된 현재 노드에 있는 객체에 필요한 정보이다.

3) 동기화 기법 : 이동객체는 자신만의 실행 쓰레드를 갖기 때문에 이동객체는 동기적 또는 비동기적으로 실행된다.

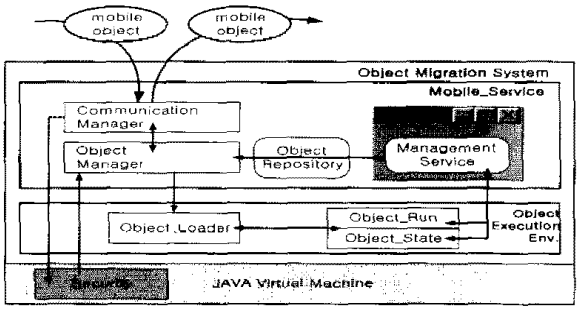
4) 병렬 실행 : 서로 다른 이동객체는 병렬로 작업을 수행하기 위해 서로 다른 목적지로 이동될 수 있다. 이동객체는 작업을 공유할 수 있도록 복사될 수 있어 개별 실행이 가능하며, 이를 위해서는 복사된 이동객체들의 실행 결과를 수집하는 작업이 필요하다.

5) 비연결성 : 이동객체는 네트워크를 점유하여 계속 연결될 필요가 없다. 즉, 네트워크 과부하가 발생하는 기존 분산처리시스템과는 달리 객체 이동시와 결과 전송 시에만 네트워크가 설정되므로 전반적인 네트워크 부하를 줄인다.

3.2. 객체이동시스템

제안된 객체이동시스템은 3.1에서 제시한 특성을 갖는 이동객체를 위해 입력/출력 기능이 있는 추상적인 장소를 제공하여, 객체가 해당 노드의 시스템 자원을 이용할 수 있는 환경으로 그림 4와 같다. 또한 제안된 시스템은 각 노드에 하나 이상 존재할 수 있으며, 여러 노드에 객체이동시스템이 있을 경우 이를 객체이동시스템 영역이라 한다.

객체이동시스템을 구성하고 있는 최소 구성 요소는 (그림 4)와 같이 그래픽 기반의 사용자 인터페이스(GUI) 모듈과 객체실행환경 모듈, 객체이동관리 모듈로 구성되어 있으며, 객체이동시스템간의 투명성을 위해 저장소(repository)를 구현해 객체 이름 투명성(object naming transparency)을 제공한다. 이는 이동객체가 객체이동시스템으로 이동되었을 때, 실행 환경을 제공하고 또다른 노드로의 이동 기능을 제공하는 것으로 각 모듈의 기능은 다음과 같다.



(그림 4) 객체이동시스템
(Fig. 4) Object Migration System

3.2.1. 사용자 인터페이스(GUI) 모듈

사용자 인터페이스는 객체이동시스템에 적용시키기 위한 이동객체의 명령(예를 들면, 객체를 적재(load)하여 라우팅(routing) 스케줄을 구성한 후, 이동(dispatch)시키거나 회수(retract), 저장(save))을 수행한다. 또한, 이 모듈은 사용자에게 현재 이동객체 상태를 알 수 있는 정보창을 제공하며, 현 이동객체의 상태와 목적지 노드 위치 및 실행되는 객체이동시스템 주소에 대한 정보 등을 제공한다.

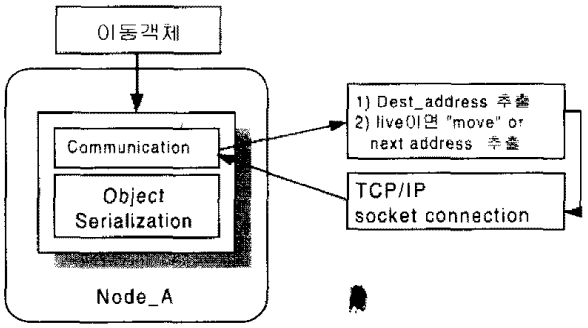
3.2.2 객체이동관리 모듈

네트워크를 통해 객체를 이동시키기 위해 자바 언어에서 제공하는 시리얼라이제이션(serialization) 기법[7]을 사용한다. 시리얼라이제이션은 자바 언어에서 제공하며 보안 기능도 갖고 있으므로, 보안 수준은 자바

의 시리얼라이제이션 기능에서 제공하는 것을 따른다.

객체이동관리 모듈은 이동객체가 이동 및 도착을 위해 TCP/IP 기반 네트워크 연결을 수행하며, 연결된 후 원격지의 이동객체와 객체정보는 시리얼라이즈 모드로 변환되어 해당 목적지에 도착한다. 이들은 목적지에서는 대응하는 디시리얼라이즈 모드로 변환하므로써 객체 이동을 종료하고, 동시에 네트워크 자원을 해제하여 네트워크 부하를 제거한다.

또한, 이동객체와 객체이동시스템간의 통신 서비스를 위한 부가적인 통신 기능으로서, 이동객체의 목적지가 연결 설정이 불가능에 대한 다음 목적지로의 전환(hop) 기능을 제공한다. 즉, 기본적 통신 모듈은 TCP/IP의 소켓 연결을 기반으로 이동될 객체이동시스템 간에 연결 설정을 수행한다. 그러나, 사용자가 지정한 목적지의 객체이동시스템이 없거나 준비가 안되었을 경우 자동적으로 해당 목적지로 이동할 수 없음을 사용자에게 알려주면서 다음 목적지를 찾아 계속 수행할 수도 있다. 이와 같은 원리에 따라, (그림 5)는 임의의 노드 A의 객체이동시스템에 객체 이동에 관한 내부 표현을 도식적으로 나타낸 것이다.



(그림 5) 객체이동시스템에서 객체 이동
(Fig. 5) Moving an object in the Object Migration System

3.2.3 객체실행환경 모듈

객체실행환경 모듈은 객체이동관리 모듈을 경유해 이동객체가 도착하였을 때, 실행되는 논리적 장소이다. 객체실행환경 모듈이 제공하는 장소는 이동객체가 실행시 필요한 자원, 즉 다른 이동객체 정보 등을 포함한 여러 각 객체에 대한 상태정보를 담고 있는 상태정보 저장소를 제공할 뿐만 아니라 자바 가상머신 환경하에서 자바 플랫폼(platform)이 허용하는 시스템 자원이나 환경, Database 접근 등의 자원을 제공한다. 이 모듈이 제공하는 상태정보 저장소는 객체가 객체이동시스템을 떠날 때까지 유지 관리된다. 또한, 이동객체가 객체이동관리 모듈을 통해 떠난 때 해당 정보는 자동적으로 소멸되게 하여 시스템 자원을 효율적으로 유지한다. 저장소에는 객체시스템의 주소와 각 객체 식별자 등을 담고 있는 이름 서비스 기능을 담당하는 별도의 객체정보 저장소가 있다.

객체실행환경 모듈 내에는 이동객체가 자동적으로 구동되어 실행되거나 이동을 위한 준비작업을 지원하기 위한 Object_Loader 모듈과 Object_Run 모듈이 있다.

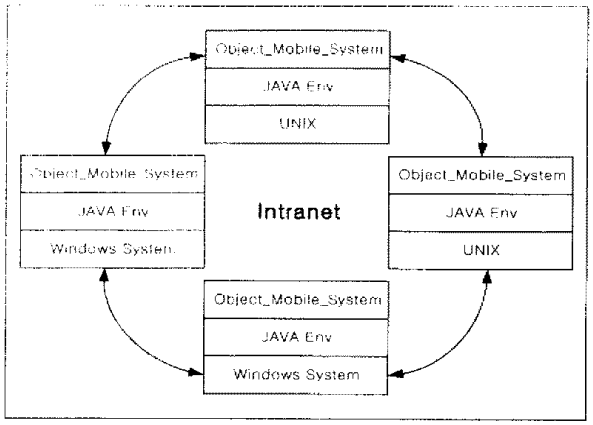
- Object_Loader : 이 모듈은 자바 Class Loader 클래스를 변형한 것으로서, 바이트 코드 스트림의 이동객체를 해당 시스템에 등록과 저장, 또 자동 실행될 수 있는 코드로 unmarshal 작업을 수행한다.
- Object_Run : Object_Loader 모듈에 의해 unmarshal된 자동 실행 가능한 이동객체는 이전 상태정보를 이용하여 재생성 또는 리인스턴스를 수행한다. 이는 자바 Reflection[6] 기법을 이용하여 각 객체에 대한 실행상태를 보관 유지한다.

4. 객체이동시스템의 구현 및 실행 예

자바 언어는 객체 이동에 필요한 정교한 네이밍 스킴을 가지고 있지 않고 애플릿으로 구성된 클라이언트/서버 모델 이상의 보안은 제공하지 못하는 단점이 있다. 하지만 자바는 네트워크 기능을 지원하며, 객체 이동에 필요한 보안 기능은 자바 컴파일러와 가상머신상에서 지원하고 있으므로 본 논문에서는 자바 애플릿을 인증하는 시스템의 보안 수준만을 고려하였다. 따라서, 제안된 객체이동시스템을 구현하는 언어로는 자바를 사용하였으며, 해당 각 모듈과 서브모듈들은 완전히 독립적이며, 이동객체도 자바 언어로 작성된 클래스(class) 객체이다.

4.1. 객체이동시스템 구현 환경

본 논문의 객체이동시스템은 (그림 6)과 같이 동형 또는 이형 환경하에 있는 자바 가상머신을 기반으로 구현하였다. 즉, 현재 실험의 대상인 환경은 인트라넷(Internet) 망에 연동되어 있는 UNIX 기반 서버들과 Windows 95, NT 기반의 PC와 서버들로 (그림 6)과 같이 구성되어 있다. 이중, 각 서버 혹은 PC들은 하나 이상의 객체이동시스템이 적재될 수도 있다.



(그림 6) 객체이동시스템 구현 환경
(Fig. 6) An environment implementing the object migration system

4.2. 객체이동시스템 구현 단계와 실례

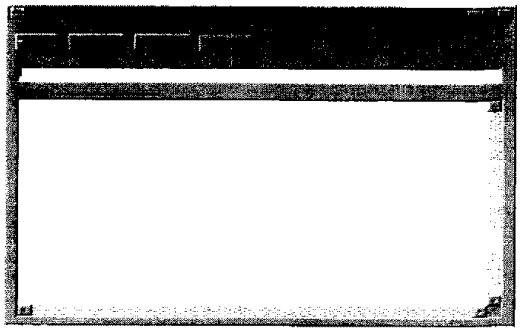
본 논문의 이동객체는 객체이동시스템에 이동되어 실행되어야 하므로, 이동객체와 객체이동시스템간의 인터페이스가 필요하다. 따라서, 사용자는 이동객체가 수행하여야 할 작업을 코딩할 때에 다음 구문 형식에 맞게 자바 프로그램을 작성하고, 컴파일 하여 클래스 파일을 생성하여야 한다. 여기서 begin 메쏘드의 argument는 원래의 자바 언어 main부분에서 외부 인수를 전달받기 위한 것에 상응하며, 제안된 객체이동시스템의 이동객체는 main을 begin 메쏘드가 대신한다.

```
public interface Object_Runnable {
        public void begin(String[] arguments);
}
```

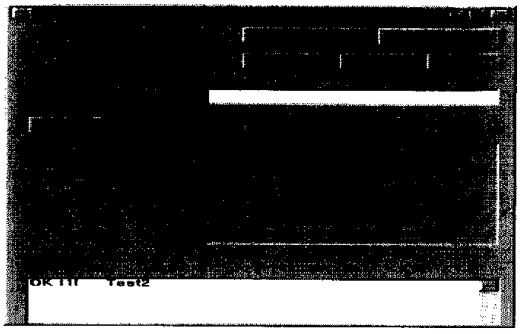
다음은 생성된 이동객체에 대해 객체이동시스템이 제공하는 각 단계별 구현에 관한 시나리오 및 각 모듈의 역할과 동작을 기술한 것이며, 그에 따른 실례를 첨가하였다.

단계 1. 사용자 인터페이스에서 컴파일된 이동객체를 load하기 위해, (그림 8)처럼 객체를 선하하고, Routing Address에 목적지의 IP와 포트번호를 넣어 Add 버튼을 눌러 라우팅 스케줄을 설정한다.

단계 2. 단계 1에서 load된 이동객체에 명령을 하기 위해 (그림 7)의 초기 화면에서 해당 명령 버튼을 클릭 한다. 예를 들어, Dispatch 버튼을 클릭 한다.



(그림 7) 초기 GUI 화면
(Fig. 7) Initial GUI Window

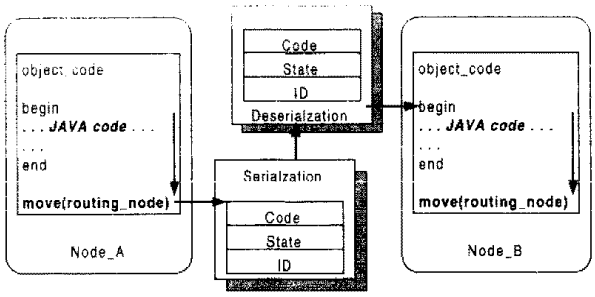


(그림 8) load 화면
(Fig. 8) Load_Module GUI window

단계 3. Dispatch 명령에 의해 이동객체는 객체이동관리 모듈 환경 하에서 지원을 받는다. 객체이동관리 모듈은 먼저 이동객체의 라우팅 정보를 엑세스하여 통신 모듈에 의한 대상 목적지와의 소켓 연결을 설정한다.

단계 4. 설정된 소켓에 따라 객체이동관리 모듈은 (그림 9)와 같이 이동객체를 시리얼라이제이션하고 목적지에서의 객체이동관리 모듈은 이를 디시리얼라이제이션한다.

단계 5. 객체실행환경 모듈에서는 디시리얼라이제이션된 이동객체를 초기 사용자 명령에 따라 적절한 동작을 행한다. 예를 들면, 위처럼 Dispatch 명령일 경우, 이동객체가 현 시스템에 이미 있는가를 객체 저장소에 등록 확인 작업을 하고, Object_Loader 모듈과 Object_Run 모듈을 통해 이동객체를 자바 가상머신 상에서 실행한다(그림 9). 이때, 객체 상태를 재 설정하기 위한 정보가 있으면, 이동객체는 생성자 인수를 통해서 재 초기화된다.

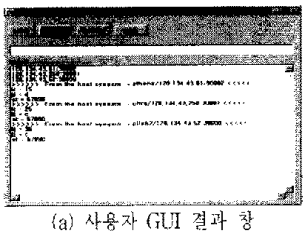


(그림 9) 이동객체 마샬링(marshalling)과 실행
(Fig. 9) Marshalling and executing an mobile object

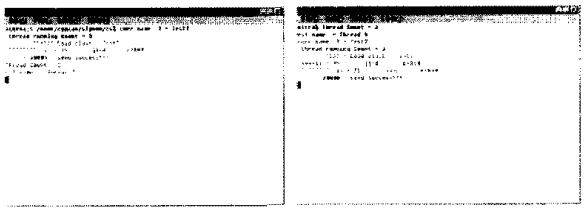
단계 6. 단계 5에서 실행된 이동객체는 결과 및 상태 정보는 해당 저장소에 보관하고, 단계 8의 (그림 10)의 (a)의 GUI 창으로 결과를 즉시 알려준다.

단계 7. 단계 6을 수행한 후, 이동객체는 다음 목적지로 이동을 위해 단계 3~단계 7을 반복적으로 계속 수행하여 임무를 완수한 후, 이동객체는 자동적으로 소멸 혹은 되돌려진다.

단계 8. 첫 번째 목적지로 이동되어 수행된 이동객체에 대한 정보가 (그림 10)의 (a)와 같은 초기 화면의 GUI 창에 나타나 사용자에게 알려주며, 또한 다음 목적지로 이동한 객체의 수행 결과도 즉시 출력된다. 여기서 (그림 10)의 (b)는 이동객체가 각 서버에 도착한 후의 서버 창에서의 실행 결과이다.



(a) 사용자 GUI 결과 창



(b) 각 서버에서 실행된 이동객체 결과

(그림 10) Dispatch된 이동객체가 각 서버에서 자동으로 실행된 결과
(Fig. 10) The executing results of autonomous an mobile object on the servers

5. 결론 및 향후 과제

본 논문에서 설계하고 구현된 객체 이동을 위한 시스템은 다음과 같은 기능을 가지고 있다.

① 동형 및 이기종간에도 객체 이동시 객체 이주 투명성을 가진다.

② 객체가 이동되는 장소를 제공하고, 필요에 의해 객체의 실행을 중단할 수 있으며, 상태와 함께 저장하는 기능이 있다.

③ 객체 이동을 관리하기 위한 객체 저장소에 객체에 관한 정보를 보관한다.

그러므로, 본 논문에서 구현한 객체이동시스템은 분산 객체 컴퓨팅의 문제를 해결하였으며, 이는 자율성을 가지는 에이전트(agent)의 이동을 위한 시스템 설계와 구현에 기본이 된다. 따라서, 제안된 객체이동시스템의 성능과 기능을 확장하여 이동에이전트를 위한 시스템 연구/개발로의 확장이 필요하다.

참 고 문 헌

[1] Agha, G., "A Model of Concurrent Computation in Distributed systems," MIT Press, Cambridge, MA, 1987.

[2] Baharat, K. A., Cardelli, L. "Migratory Applications," Proc. of the 8th Annual ACM Symp. on UIS Tech., Novemer 1995.

[3] Bernd M., "Mobility in Persistent Object Systems," Ph.D. thesis, Dept. of CS, Hamburg Univ., Germany, May 1996.

[4] Cardelli, L., A Language with distributed scope. Computing Systems, MIT Press, 1995.

[5] Dejan Milojicic. "Mobile Agents," 1996.

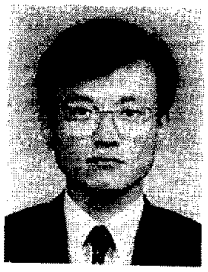
[6] Gosling, "The Java Language : A White Paper," Sun co., 1994.

[7] Java Object Serialization Spec., http://chat subo.javasoft.com/current/serial/index.html.

[8] Java-To-Go, http://ptolemy.eecs.berkeley.edu/~wli/group/java2go/java-to-go.html.

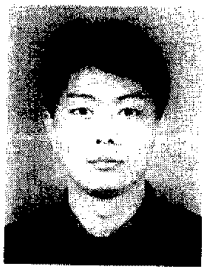
[9] Joseph W., "bots and other internet beasties," Samsnet, 1996.

[10] Milojicic, D., et al., "Task Migration on Top of the Mach Microkernel," Proc. of the third USENIX Mach Symposium, April 1993.
[11] OMG, "Mobile Agent Facility Interoperability Facilities Specification(MAF)," OMG.
[12] TCDocument ORBOS/1997-10-05, http://www.opengroup.org/~dejan/maf/draft10.
[13] Rousch, E., Campbell, R., "Fast Dynamic Process Migration," Int'l. Conf. on Distributed Computing System, 1996.
[14] Steengaard, B., Jul, E., "Object and Native Code Thread Mobility," Proc. of the 15th SOAP, December 1995.
[15] Sun Microsystems, Remote Method Invocation, 1996. http://chatsubo.javasoft.com/current/rmi.
[16] Vitek, J., Tschudin, C., "Mobile Objects Systems : Towards the Programmable Internet," Springer Verlag, April 1997.
[17] 이근상, 전병국, 유황빈, 최영근, "CORBA 객체의 이주를 위한 Object Transfer의 설계와 구현", 한국정보처리학회 춘계 학술발표 논문집 제3권 제1호, 1996.



전 병 국

e-mail : jeonbk@sky.wonju.ac.kr
1985년 광운대학교 전산과 졸업(이학사)
1991년 광운대학교 대학원 전산과 졸업(이학석사)
1991년~1993년 KINITI 연구원
1993년~현재 원주대학교 사무자동화과 조교수
1992년~현재 광운대학교 대학원 전자계산학과 박사과정
관심분야 : 이동에이전트, 분산처리, 전자상거래



이 근 상

e-mail : ksyi@cs.kwangwoon.ac.kr
1993년 광운대학교 전산과 졸업(이학사)
1995년 광운대학교 대학원 전산과 졸업(이학석사)
1995년~현재 광운대학교 대학원 전자계산학과 박사과정
관심분야 : 이동에이전트, 분산처리, 전자상거래

최 영 근

e-mail : ygchoi@daisy.kwangwoon.ac.kr
1980년 서울대학교 사범대학 수학교육과(이학사)
1982년 서울대학교 대학원 계산통계학과(이학석사)
1989년 서울대학교 대학원 계산통계학과(이학박사)
1983년~현재 광운대학교 전자계산학과 교수
1997년~현재 광운대학교 전자계산원 원장
관심분야 : 병렬 컴파일러, 병렬 프로그래밍 언어, 객체지향 프로그래밍 언어, 객체지향 분산 컴퓨팅