계획 기능을 가진 지능형 이동에이전트 시스템

석 황 희[†] · 김 인 철[††]

요 약

대부분의 기존 이동에이전트 시스템들은 사용자나 혹은 프로그래머로 하여금 이동에이전트가 수행할 작업에 대해 매우 세부적인 행동 스크립트를 기술하도록 요구하며 이동에이전트는 실행 시에 단순히 이 고정 스크립트에 기술된 바대로 행동한다. 따라서 이동 에이전트 스스로 자율적으로 최종목표와 동적 상황에 맞는 계획을 수립하고 실행해나가는 것이 불가능하다. 이와 같은 기존 이동 에이전트 시스템의 제한점들을 극복하기 위한 한 가지 방법은 반응형 계획기를 포함한 지능형 이동 에이전트 시스템을 개발하는 것이다. 본 논문에서는 대표적인 반응형 계획 에이전트 구조인 JAM을 기초로 에이전트의 이동 모델과 에이전트간의 통신 모델을 설계하였다. 그리고 이 모델에 따라 에이전트의 이동과 통신을 위한 기본 동작들을 JAM에 추가 구현함으로써 스스로 계획을 세우고 실행하는 하나의 지능형 이동에이전트 시스템 IMAS를 개발하였다. IMAS 에이전트들은 기존의 이동에이전트들과는 달리 목표에 맞는 자신의 행동계획을 스스로 세울 수 있을 뿐만 아니라 동적인 환경변화에도 효과적으로 적응할 수 있다. 따라서 IMAS 에이전트들은 기존의 이동 에이전트들에 비해 보다 높은 유연성과 견고성을 보여줄 수 있다.

An Intelligent Mobile Agent System with Planning Capability

Hwang-Hee Seok[†] · In-Cheol Kim[††]

ABSTRACT

Most of conventional mobile agent systems require that the users or the programmer should give the mobile agent its detail behavioral script for accomplishing the given task. And during its runtime, such mobile agents just behave according to the fixed script given by its user. Therefore it is impossible that conventional mobile agents autonomously build their own plans and execute them in considering their ultimate goals and the dynamic world states. One way to overcome such limitations of conventional mobile agent systems is to develop an intelligent mobile agent system embedding a reactive planner. In this paper, we design both a model of agent mobility and a model of inter-agent communication based upon the representative reactive planning agent architecture called JAM. An then we develop an intelligent mobile agent system with planning capability, IMAS, by implementing additional basic actions for agent moves and inter-agent communication within JAM according to the predefined models. Unlike conventional mobile agents, IMAS agents can be able to adapt their behaviors to the dynamic changes of their environment as well as build their own plans autonomously. Thus IMAS agents can show higher flexibility and robustness than the conventional ones.

1. 서 론

네트워크상의 원격 호스트들 사이를 데이터뿐만이 아니라 코드와 상태정보까지 함께 가지고 이동하면서 업무를 처리하는 소프트웨어 에이전트를 이동에이전트라고 한다. 최근 들어, 다양한 분야에 걸쳐 이동에이전트에 대한 관심이 높아져 가고 있는데 이에 따라 이동에이전트를 개발할 수 있는 환경이 많이 개발되고 있으며 그 대표적인 예로, Telescript[5], Agent Tcl(Tool

† 준 회 원 : 경기대학교 전자계산학과 졸업(이학석사)
†† 정 회 원 : 경기대학교 정보과학부 교수
논문접수 : 2000년 8월 12일, 심사완료 : 2000년 11월 7일

Command Language)[2], Ara(Agent for Remote Action)[11], TACOMA(Troms And COrnell Moving Agents)[12], Aglets Workbench[9], Voyager[14], Odyssey[15], Concordia[13] 등이 있다. 또한, 이동에이전트를 이용한 응용시스템들도 여러 분야에 걸쳐 개발되고 있는데 이동 컴퓨팅 단말기, 네트워크 및 분산시스템 관리, 원격지 진단, 전자상거래, 인터넷 정보검색, 분산 시뮬레이션 그리고 가상기업 등이 대표적인 응용분야로 받아 들여 지고 있다.

대부분의 기존 이동에이전트 시스템들은 사용자나 혹은 프로그래머로 하여금 사전에 이동에이전트가 수행할 작업에 대해 매우 세부적인 행동 스크립트(script)를 기술하도록 요구하며 이동에이전트는 런타임(runtime)시에 단순히 이 고정 스크립트에 기술된 바대로 행동하는 제한적인 구조로 되어 있다. 따라서 에이전트 스스로 자율적으로 작업목표와 실행 당시 상황에 맞는 계획을 동적으로 수립하고 실행해나가는 것이 불가능하다. 런타임 이전에 사용자나 프로그래머가 작업목표를 성공적으로 달성할 수 있는 이러한 세부 행동 스크립트를 기술하기 위해서는 미리 에이전트의 동적 작업환경에 대한 완전한 정보와 모델을 가지고 있어야 한다. 하지만 사용자를 떠나 원격의 다양한 이종 호스트들을 순회하며 작업하는 대부분의 이동에이전트 적용환경에서는 네트워크 및 시스템 고장, 호스트 컴퓨터들의 내부 구성 변화, 과부하로 인한 적체 및 지연 현상 등으로 인해 발생 가능한 모든 상황을 미리 예측하기란 사실상 불가능하다. 따라서 불완전한 사전 정보를 바탕으로 작성된 고정 스크립트를 따라 행동하는 기존의 이동에이전트시스템에서는 실행 시에 예기치 못한 상황이 발생할 경우 상황 변화에 효과적으로 대응하기 어렵다는 문제점을 가지고 있다.

이와 같이 에이전트의 자율성(autonomy)과 반응성(reactivity), 그리고 견고성(robustness) 면에서 제한적인 기존 이동 에이전트 시스템의 문제점을 해결하기 위한 한 가지 방법은 반응형 계획기(reactive planner)[7, 8]에 기초를 둔 지능형 이동 에이전트 시스템을 개발하는 것이다. 일반적으로 반응형 계획기들은 계획생성 단계와 실행 단계를 완전히 분리된 순차과정으로 간주하는 전통적인 숙고형 계획기(deliberative planner)들과는 달리, 계획생성과 실행, 실세계 변화에 대한 감지와 그것에 따른 계획수정이 하나의 과정으로 통합되어 번갈아 수행됨으로써 환경변화에 민감하게 동적으로 반응할 수 있다는 장점이 있다. 이러한 특징을 가진 반응형 계획기를 기초로 이동기능과 통신기능을 갖춘 지능형 이동 에이전트 시스템을 구현하기 위해서는 에이전트의 이동과 통신에 관한 효과적인 모델을 세우고 이를 토대로 이동과 통신에 관한 기본 동작들을 구현하여 반응형 계획기내에서 제공할 수 있어야 한다. 이와 같은 지능형 이동 에이전트 시스템의 경우, 사용자는 특정 프로그래밍 언어나 스크립트 언어로 직접 이동 에이전트의 세부 행동 스크립트를 프로그래밍해야 할 필요가 없고 단지 에이전트의 작업 목표와 작업 영역지식(domain knowledge)을 일정한 표현법에 따라 선언적으로 입력해주면 에이전트 스스로가 목표달성에 필요한 행동 계획을 동적으로 수립하여 실행할 수 있게 된다. 그리고 이때 생성되는 이동 에이전트의 계획에는 사용자로부터 입력받은 영역지식 동작들뿐만 아니라 시스템에서 기본적으로 제공되는 이동 동작과 통신 동작들을 포함하게 된다. 또한 이동 에이전트 내부구조의 중심을 이루는 반응형 계획기는 계획 실행 중에도 지속적인 실세계 모니터링을 통해 남은 계획이 계속될 수 없는 상황변화를 감지하면 사용자의 직접적인 간여 없이도 즉시 새로운 계획생성이나 계획변경을 시도하게 된다.

본 논문에서는 대표적인 반응형 계획 에이전트 구조인 JAM[7]을 기초로 에이전트의 이동 모델과 에이전트간의 통신 모델을 설계하였다. 그리고 이 모델에 따라 에이전트의 이동과 통신을 위한 기본 동작들을 JAM[7]에 추가 구현함으로써 스스로 계획을 세우고 실행하는 하나의 지능형 이동에이전트 시스템 IMAS(Intelligent Mobile Agent System)을 개발하였다. IMAS는 사용자로부터 응용 영역 고유의 지식을 선언적으로 제공받을 수 있다면 손쉽게 다양한 응용 이동에이전트들을 개발할 수 있는 에이전트 개발도구로 이용될 수 있다. 그리고 이와 같이 IMAS를 이용하여 만들어지는 이동에이전트의 행위는 이동 동작과 통신 동작들을 포함한 계획들에 의해 실현된다. IMAS 이동 에이전트들은 기존의 이동에이전트들과는 달리 목표에 맞는 자신의 행동계획을 스스로 세울 수 있을 뿐만 아니라 동적인 환경변화에도 효과적으로 적응함으로써 높은 유연성(flexibility)과 견고성(robustness)을 나타낼 수 있다.

본 논문에서는 먼저 대표적인 반응형 계획에이전트 구조인 JAM[7]에 대해 간략히 소개하고, 이어서 IMAS

의 설계와 구현에 대해 상세히 설명한 뒤, 결론을 맺는다.

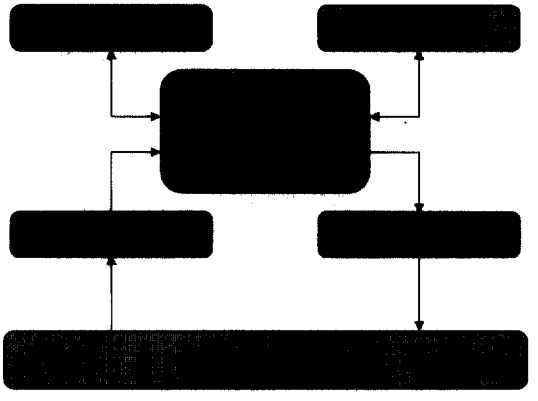
2. 반응형 계획에이전트 구조

2.1 기본 개념

일반적으로 계획기를 내부 핵심 요소로 채용하고 있는 에이전트 구조를 계획에이전트 구조라 부른다. 전통적인 인공지능 계획기 및 계획시스템은 계획생성(plan generation)과 계획실행(plan execution)이 완전히 별도의 독립적인 프로세스로 분리되어 있다고 가정한다. 즉 계획생성 단계에서는 계획기가 달성하고자하는 목표(goal)와 현재상태(current state), 가능한 동작(action)들을 기호논리로 표현한 뒤 이들을 바탕으로 목표를 달성할 수 있는 동작들의 순서를 결정하고, 계획실행 단계에서는 이렇게 생성된 계획을 별도의 실행기에 의해 실행하는 순차적 제어구조를 상정하였다. 이러한 전통적인 인공지능 계획기들을 일반적으로 숙고형 계획기라 부른다. 전통적인 숙고형 계획기들은 계획생성 단계에서 실세계에 대한 완전(complete)하고 정확한(correct) 모든 정보를 다 가지고 있다고 가정한다. 따라서 계획생성 중에 발생할 수 있는 환경변화나 계획생성 때 예측하지 못한 현상이 실행 중에 발생해도 이들에 대응할 수 있는 적절한 계획수정이나 재 계획이 신속히 이루어질 수 없다는 단점이 있다. 이러한 숙고형 계획기들에 반해 하나의 계획기내에서 계획생성과 계획실행, 그리고 계획수정을 동시에 병렬로 수행해가면서 환경변화에 신속하게 반응할 수 있는 계획기들을 일반적으로 반응형 계획기(reactive planner)라 부른다. 이러한 반응형 계획기로는 PRS(Procedural Reasoning System)[16]와 TouringMachine 그리고 InteRRaP 등이 있다.

2.2 JAM

JAM[7]은 Georgeff[8]에 의해 제안된 PRS 계획에이전트 구조에 몇 가지 기능을 확장하고 Java 언어로 재구현한 것이다. JAM[7]은 (그림 1)과 같이 5 가지 요소로 구성되어 있으며 이 5 가지 요소들의 상호 작용에 의해 외부 환경으로부터의 변화를 내부적으로 처리하는 행위와 그 결과를 다시 외부 환경으로 환원하는 행위를 반복적으로 수행함으로써 실세계와 반응하게된다.



(그림 1) JAM의 구성요소

(1) 실세계 모델(world model) : 현재 에이전트가 가지고 있는 실세계에 대한 정보를 기술해 놓은 것으로, 사실(facts) 또는 믿음(beliefs)을 다음과 같은 형식으로 기술한다.

```
FACTS :
   FACT robot_status "Ok"   ;
   FACT robot_position 10000 10000 0 ;
```

(2) 계획 라이브러리(plan library) : 에이전트 설계자에 의해 사전에 주어지는 추상 계획(abstract plan)들의 집합으로, 각 추상 계획은 다음과 같이 기술된다.

```
Plan :
{
   GOAL : [goal specification]
   NAME : [string]
   BODY : [procedure]
   <DOCUMENTATION : [string]>
   <PRECONDITION : [expression]>
   <CONTEXT : [expression]>
   <UTILITY : [numeric expression]>
   <FAILURE : [non-subgoaling procedure]>
   <EFFECTS : [non-subgoaling procedure]>
   <ATTRIBUTES : [string]>
}
```

(3) 인터프리터(interpreter) : 에이전트 설계자에 의해 초기 실세계 모델과 최종 목표, 그리고 이용 가능한 추상계획들의 집합이 선언적으로 주어지면 이들을 바탕으로 부속 목표들과 세부 계획들을 생

성하고 이들을 실행하는 역할을 한다.

(4) **의도 구조(intention structure)** : 최종 목표 달성을 위해, 에이전트가 현재 추구하는 부속 목표와 현재 실행 중인 세부 계획 상태를 나타낸다.

(5) **감지기(observer)** : 인터프리터의 추론 주기 사이 사이에 외부세계의 변화에 대한 감지 기능을 수행하며 다음과 같은 형식으로 기술한다.

```
OBSERVER :
   {
   RETRIEVE initialized $VALUE;
   WHEN : TEST (== $VALUE "False") {
      EXECUTE setShowAPL 1;
      EXECUTE setShowGoalList 1;
      EXECUTE setShowWorldModel 1;
      EXECUTE setShowIntentionStructure 1;
      UPDATE (initialized) (initialized "True");
   };
 }
```

다음의 예제는 위의 요소들을 이용한 하나의 완전한 JAM[7] 프로그램을 보이고 있다.

```
GOALS : ACHIEVE cycle_tested;
FACTS : test_done        "False";
        system_init      "False";
        cycle_number     0;
OBSERVER {
   RETRIEVE cycle_number $N;
   EXECUTE print "\n Cycle#" $N "\n";
   UPDATE (cycle_number $N) (cycle_number
                              (+ $N 1));
}
PLAN {
NAME : "Test CYCLE"
GOAL : ACHIEVE cycle_tested;
CONTEXT : FACT test_done "False";
BODY :
   EXECUTE print "\n Normal execution
                  started. \n";
   UPDATE (system_init) (system_init "True");
   WHILE : TEST (== 1 1)
   {
      EXECUTE noop;
   } ;
}
```

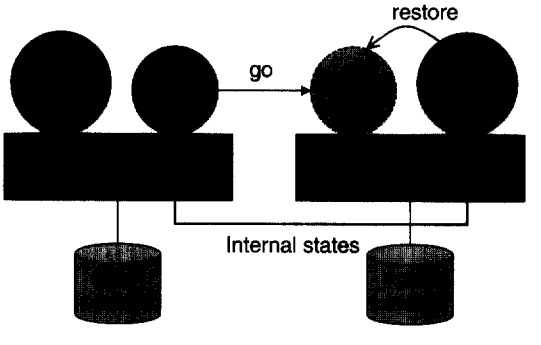
JAM[7]은 응용분야에 따른 확장을 용이하게 하기 위해 각 계획안에서 EXECUTE 명령문을 통해 수행할 수 있는 몇 가지 동작들을 기본적으로 제공하고 있을 뿐 아니라 이들 외에 필요한 다양한 동작들을 프로그래머가 Java 언어로 손쉽게 추가 정의할 수 있도록 하고 있다. JAM[7]에서 기본적으로 제공되는 중요한 기능 중의 하나는 수행중인 에이전트의 실행 상태를 그때그때 기록하는 체크포인터(check-point) 기능이다. 이 체크포인터 기능은 에이전트의 복제 및 복구, 그리고 이동에 중요한 기초를 제공한다.

3. 지능형 이동에이전트 시스템의 설계

앞서 설명한 바와 같이 본 연구에서는 자율적으로 계획을 생성하고 실행할 수 있는 지능형 이동에이전트 시스템을 개발하기 위해 반응형 계획기인 JAM[7]을 기초로 이동에이전트 시스템에 필수적인 에이전트의 이동과 통신 기능을 추가 구현하고자 한다. 이를 위해서는 에이전트 이동과 통신에 필요한 주요 구성 요소들과 이들간의 상호작용을 정의하는 구체적인 모델들의 정의가 선행되어야 한다. 따라서 본 장에서는 지능형 이동에이전트 시스템 IMAS의 기초가 되는 이동 모델과 통신 모델의 설계에 대해 설명한다.

3.1 에이전트의 이동 모델

본 절에서는 지능형 이동에이전트 시스템을 구현하기 위해 (그림 2)와 같이 장소(place), 이동에이전트(mobile agent), 이동 서버(mobility server)의 3 가지 기본 요소로 구성된 에이전트의 이동모델을 설계하여 기본적인 이동을 할 수 있게 하였다. 이 3 가지 구성요소의 기능은 다음과 같다.



(그림 2) 에이전트의 이동모델

(1) **장소(place)** : 하나의 장소는 분산 네트워크 상에서 이동 에이전트가 활동하는 하나의 가상 작업

영역(virtual work area)이다. 하나의 장소는 물리적으로 네트워크에 연결된 하나의 호스트 컴퓨터에 하나 이상 존재할 수 있다. 하나의 장소는 반드시 실행 중인 하나의 이동 서버를 포함해야 한다. 활동 중인 모든 이동 에이전트들은 현재 서로 다른 장소간에 이동 중이 아니라면 반드시 분산 네트워크 상의 하나의 장소에 머물러 있으며 작업을 수행한다.

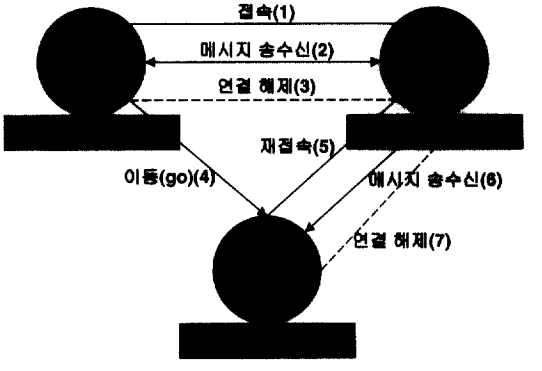
(2) **이동 서버(mobility server)** : 하나의 장소에 하나의 이동 서버가 활동하면서 자신이 속한 장소로 진입하려는 임의의 이동 에이전트가 있을 경우, 이 이동 에이전트의 진입(migration)을 도와주는 역할을 수행한다. 하나의 이동 서버는 호스트 이름(host name)과 포트 번호(port number)로 이루어지는 고유의 주소(address)를 가지며, 진입을 원하는 원격의 이동 에이전트와 소켓(socket) 연결을 통해 이동에이전트의 현재 내부상태 정보를 전송받아 현재의 장소에서 이 이동에이전트를 재생(restore)시켜 주는 역할을 수행한다.

(3) **이동에이전트(mobile agent)** : 하나의 이동 에이전트는 이동 서버의 도움을 받아 분산 네트워크 상의 여러 장소들을 옮겨 다니며 최초 출발지의 사용자(user)를 위해 유용한 작업을 수행하는 자율적인 소프트웨어이다. 이동 에이전트가 원격의 한 장소로 이동을 하기 위해서는 그 장소에 활동 중인 이동 서버의 물리적 주소를 이용하여 소켓 연결을 시도한 뒤, 그 소켓을 통해 자신의 내부 상태 정보를 직렬화(serialize)하여 전송한다. 목적지의 이동 서버의 도움으로 목적지 장소에서 재생되면 이동 직전의 수행 상태로 복귀하여 실행을 계속한다.

이동에이전트가 분산 네트워크상의 한 장소(place)로부터 원격의 한 목적지 장소(destination place)로 이동하는 단위 이동 과정의 세부단계는 다음과 같다. 우선, 이동하고자하는 목적지 장소에 이동 서버(mobility server)가 실행 중이어야 한다. 일단 이동 서버가 실행되면 정해진 포트(port)를 통해 임의의 한 이동에이전트의 내부 상태 정보(internal state information)가 도착하기를 기다리고 있다. 그러면 이동 에이전트(mobile agent)가 실행중인 계획 내에서 이동 행위(mobile action)인 Go를 실행한다. 이 이동 행위가 시작되면 먼저 목적지에 실행중인 이동 서버와 소켓(socket)으로 연결을 시도하고 이 연결이 성공하면 이 소켓을 통해 이동에이전트의 현재 내부 상태 정보를 직렬화(serialize)하여 전송한다. 마지막으로 목적지의 이동 서버가 소켓을 통해 이동에이전트의 내부 상태 정보를 수신하면 수신된 상태 정보와 함께 목적지 파일 시스템내의 이동에이전트를 위한 클래스 파일(class file)들을 이용하여 목적지 장소에서 이동에이전트를 재생시킨다.

3.2 에이전트간의 통신 모델

본 절에서는 지능형 이동에이전트 시스템을 구현하기 위해 (그림 3)과 같이 에이전트간의 통신모델을 설계하였다. 이것은 통신을 위한 기본 동작들, 메시지 서버 에이전트 그리고 메시지 클라이언트 에이전트로 구성되어 있으며 그 기능은 다음과 같다.



(그림 3) 에이전트간의 통신모델

(1) **메시지(message)** : 에이전트간의 통신은 기본적으로 임의의 문자열(character string) 형태의 한 메시지를 주고받는 것으로 이루어진다고 가정한다. 그리고 이러한 메시지 교환은 에이전트의 이동의 경우와 마찬가지로 서버측과 클라이언트측간의 소켓연결(socket connection)을 기반으로 이루어진다고 가정한다. EMAIL과 같이 비 연결방식의 비동기 메시지 교환도 가능하나 본 논문에서는 소켓연결을 통한 동기화된 메시지 교환방식만을 지원한다. 소켓연결을 통해서는 문자열이외에 다양한 멀티미디어 정보나 객체(object)를 나타내는 임의의 바이트 스트림(byte stream)도 전송할 수 있으나 본 논문에서는 에이전트간 표준통신언어인 KQML과 같이 가장 기본적인 문자열 형태의 메시지 전송만을 지원한다.

(2) **메시지 서버(message server)** : 메시지 서버는 분산 네트워크 상의 한 장소에 머물러 있으면서 이동 가능한 임의의 원격 클라이언트와 소켓을 통해 메시지를 교환한다. 이를 위해 하나의 메시지 서버 에이전트는 호스트 이름과 포트 번호로 구성된 고유 주소를 가지며, 이 주소를 이용해 연결을 청하는 다수의 클라이언트들과 통신한다.

(3) **메시지 클라이언트(message client)** : 지속적으로 한 장소에 머물러 있는 메시지 서버 에이전트와는 달리 메시지 클라이언트 에이전트들은 분산 네트워크상의 서로 다른 장소들을 옮겨 다니며 메시지 서버 에이전트와 통신 가능한 에이전트들이다. 메시지 클라이언트 에이전트들이 서로 다른 장소로 이동한 후에도 통신을 계속하기 위해서는 이동 전에 서버와의 연결을 해제하였다가 목적지에 도착 후 다시 서버와 소켓 연결을 해야 한다.

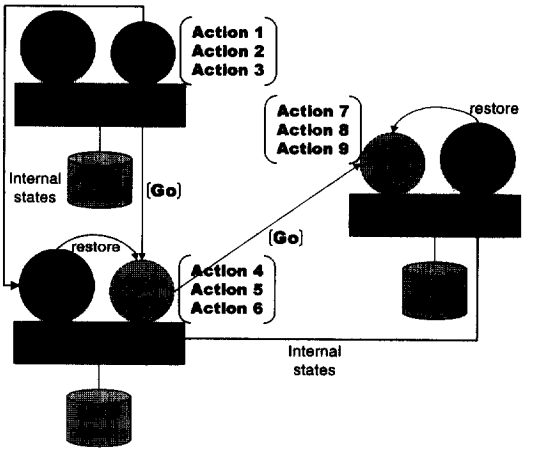
분산 네트워크상의 임의의 한 장소(place)에 있는 메시지 클라이언트 에이전트가 원격의 한 목적지 장소(destination place)에 있는 메시지 서버 에이전트와 통신을 하는 도중에 또 다른 원격의 목적지 장소로 이동하면서 이동에이전트간의 통신을 하기 위한 단위 통신 과정의 세부 단계는 다음과 같다. 우선, 장소 A에서 활동중인 한 메시지 클라이언트 에이전트가 원격의 한 장소 B의 메시지 서버 에이전트와 통신을 하기 위해 connectToAgentAsClient 동작을 수행함으로써 초기 소켓 연결(socket connection)을 요청한다. 이 연결이 이루어지면 소켓을 통해 메시지 서버 에이전트와 원하는 만큼 통신을 한다. 메시지 클라이언트 에이전트가 메시지 서버 에이전트와의 통신 도중에 새로운 원격의 장소 C로 이동하기 위해서는 기존 서버와의 소켓 연결을 해제한다. 그리고 해제 후 메시지 클라이언트 에이전트는 Go 명령을 이용하여 또 다른 원격의 목적지 장소 C로 이동한다. 이 이동이 성공적으로 이루어지면 다시 메시지 서버 에이전트와 통신을 위해 connectToAgentAsClient 동작을 수행함으로써 소켓 연결을 재 시도한다. 일단 소켓 연결이 다시 이루어지면 전과 같이 원하는 만큼의 통신을 한다. 더 이상의 통신이 필요하지 않거나 또는 다시 새로운 장소로 이동하기 위해서는 서버와의 소켓 연결을 해제한다. 이렇게 계속 이동하면서 에이전트간의 통신을 수행하기 위해서는 위와 같은 과정을 반복적으로 수행함으로써 이루어진다.

4. 지능형 이동에이전트 시스템의 구현

4.1 에이전트 이동 모델의 구현

본 연구에서는 3장에서 설계한 에이전트 이동 모델에 따라 각 호스트에 상주하며 에이전트의 이동을 돕는 별도의 이동서버(mobility server)를 Java로 구현하였다. 그리고 반응형 계획에이전트 구조인 JAM[7]에 이동을 위한 기본 동작인 Go를 추가 구현하여줌으로써 이 확장된 JAM[7] 구조를 바탕으로 이동에이전트(mobile agent)를 생성할 수 있도록 하였다.

다음의 (그림 4)는 서로 다른 세 가지 분산 호스들 간을 이동하며 작업하는 한 에이전트의 수행 시나리오를 예로 보여주고 있다. 이 수행 시나리오를 구현해줄 수 있는 JAM[7] 에이전트의 세부 계획은 Go 동작을 이용하여 <표 1>과 같이 표현된다.



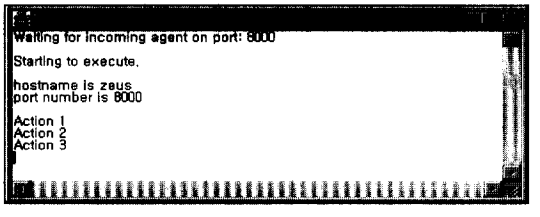
(그림 4) 한 에이전트의 이동 시나리오

〈표 1〉 에이전트의 이동 계획

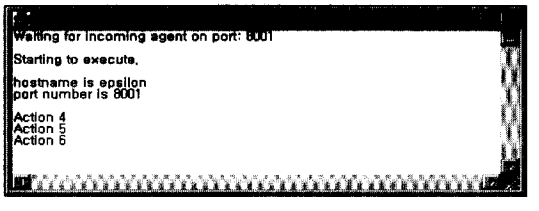
```
BODY :
                    . . . .
EXECUTE
        com.irs.jam.primitives.GetHostname.execute
        $hostname;
ASSIGN $port 8000;
EXECUTE print "hostname is " $hostname "\n";
EXECUTE print "port_number is " $port "\n";
EXECUTE print "Action 1\n";
EXECUTE print "Action 2\n";
EXECUTE print "Action 3\n";
// 주어진 호스트 이름과 포트 번호로 이동명령을 수행
//한다.
```

```
ASSIGN $hostname "epsilon.kyonggi.ac.kr"
ASSIGN $port (+ $port 1)
EXECUTE Go $hostname $port;
EXECUTE
        com.irs.jam.primitives.GetHostname.execute
        $hostname;
EXECUTE print "hostname is " $hostname "\n";
EXECUTE print "port_number is " $port "\n";
EXECUTE print "Action 4\n";
EXECUTE print "Action 5\n";
EXECUTE print "Action 6\n";
                . . . .
```

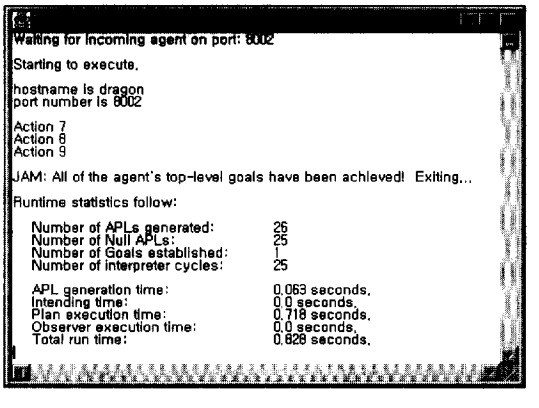
(그림 5), (그림 6), (그림 7)은 이 세부 계획에 따라 이동에이전트가 차례대로 각 원격 호스트를 옮겨다니며 실행한 결과를 보여준다.



(그림 5) 호스트 A에서의 수행



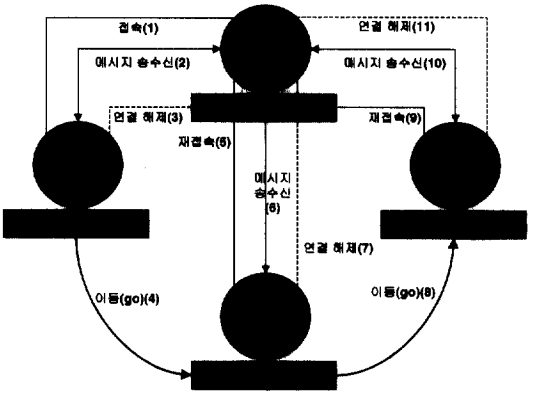
(그림 6) 호스트 B에서의 수행



(그림 7) 호스트 C에서의 수행

4.2 에이전트간 통신 모델의 구현

본 절에서는 3장에서 설계한 에이전트간의 통신모델에 따라 통신을 위한 기본 동작들인 connectToAgentAsServer, connectToAgentAsClient, sendMessage, recvMessage, disconnect 등을 JAM[7] 구조 안에 추가 구현하였다. 그리고 이 기본 동작들을 이용한 이동과 통신 계획을 세움으로써 메시지 서버 에이전트와 메시지 클라이언트 에이전트 역할을 수행하는 JAM[7] 에이전트들을 구현할 수 있게 하였다. (그림 8)은 에이전트간의 통신 시나리오의 한 예를 보여준다. 이 예는 장소 B에 위치한 이동 에이전트가 차례대로 장소 C와 D장소로 옮겨 다니며 장소 A에 위치한 다른 에이전트와 메시지를 주고받는 통신 시나리오이다. 이때 메시지 클라이언트인 이동 에이전트는 최초의 장소 B에서 메시지 서버 에이전트와 소켓 연결을 통해 통신을 하던 중 연결을 해제하고 다른 장소로 이동하여 다시 그 메시지 서버 에이전트와 재접속하고 통신을 하는 과정을 반복한다.



(그림 8) 에이전트간의 통신 시나리오

에이전트간의 통신을 위해 정의된 몇 가지의 기본 동작들은 다음과 같다.

(1) 메시지 서버의 연결 동작 : 네트워크 상의 한 장소에 머물러 있으면서 이동 가능한 임의의 원격 메시지 클라이언트의 소켓을 통한 연결 요청을 기다린다. 이를 위해 하나의 메시지 서버 에이전트는 호스트 이름과 포트 번호로 구성된 고유 주소를 가지며, 이 주소를 이용해 연결을 요청하는 다수의 메시지 클라이언트들과 통신한다.

(2) 메시지 클라이언트의 연결 동작 : 네트워크 상의

서로 다른 장소들을 옮겨 다니며 메시지 서버 에이전트와 통신 할 수 있는 에이전트로써 메시지 클라이언트 에이전트들이 서로 다른 장소로 이동한 후에도 이동한 메시지 서버 에이전트와 통신을 계속하기 위해서는 이동 전에 통신했던 서버와의 연결을 해제한 후 목적지에 도착하여 다시 그 서버와 소켓 연결을 하여야 한다.

(3) **메시지 보내기 동작** : 메시지 서버 에이전트나 메시지 클라이언트 에이전트를 통해 스트링(string) 형태의 메시지를 소켓을 통해 다른 장소의 메시지 서버 에이전트나 메시지 클라이언트 에이전트로 보내는 동작이다.

(4) **메시지 받기 동작** : 소켓을 통해 다른 장소의 메시지 서버 에이전트나 메시지 클라이언트 에이전트로부터 보내어진 스트링(string) 형태의 메시지를 받는 동작이다.

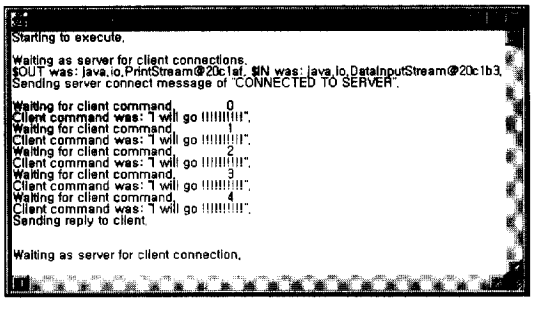
(5) **연결 해제 동작** : 더 이상 통신이 필요하지 않거나 또는 새로운 장소로 이동하고자할 때 수행하는 동작으로 현재 연결되어 있는 서버와의 소켓을 해제시키는 동작이다.

<표 2>는 에이전트간의 통신 계획을 구현한 것 중 메시지 클라이언트 에이전트가 갖는 계획의 일부를 보인 것으로, 메시지 서버 에이전트와의 반복 통신 후 접속을 해제하고 재접속 하는 과정을 보이는 것이다.

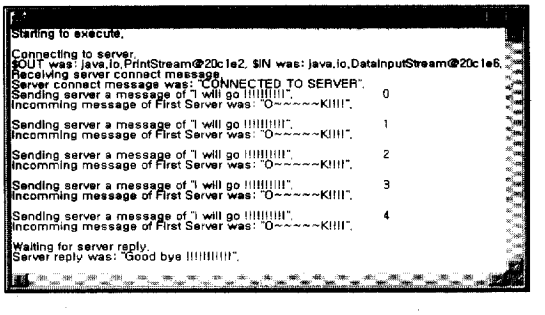
<표 2> 메시지 클라이언트 에이전트의 통신 계획

```
BODY :                    … … …
// 원하는 만큼 메시지 서버와 통신한다.
ASSIGN $x 0;
WHILE : TEST(< $x 5) {
 EXECUTE print "Sending server a message
    of \"I will quit !!!!!!!!!\"." " " $x "\n";
    EXECUTE
       com.irs.jam.primitives.sendMessage.execute
       $OUT "I will quit !!!!!!!!!";
    ASSIGN $x (+ $x 1);
};
EXECUTE print "Waiting for server reply.\n";
EXECUTE
   com.irs.jam.primitives.recvMessage.execute
   $IN $MSG;
EXECUTE print "Server reply was : \""$MSG"\"
   .\n";
// 다른 호스트로 이동하고자할 때, 해당 소켓연결
// 을 해제하고 입력스트림, 출력스트림을 모두 닫
// 고 비 접속상태로 된다.
EXECUTE
   com.irs.jam.primitives.disconnect.execute
   8000 "localhost" $socket $IN $OUT;
// 다른 장소로 이동한 후 서버로의 재접속을 시도
한다.
EXECUTE Go "epsilon" 8000;
EXECUTE
   com.irs.jam.primitives.connectToAgentAs-
   Client.execute 8000 "localhost" $socket $IN
$OUT ;              … … …
}
```

(그림 9)와 (그림 10)은 메시지 서버 에이전트와 메시지 클라이언트 에이전트를 수행시켜 이들간에 이루어진 통신 결과의 일부를 보여준다.



(그림 9) 메시지 서버 에이전트의 수행 결과



(그림 10) 메시지 클라이언트 에이전트의 수행 결과

5. 결 론

본 논문에서는 기존의 반응형 계획 시스템인 JAM [7]을 기초로 에이전트의 이동동작과 에이전트간의 통신동작을 확장 설계하고 구현함으로써 계획기능을 갖는 지능형 이동에이전트 시스템을 개발하였다. 이 시스템의 특징은 내부의 반응형 계획기를 바탕으로 추론을 통해 이동동작과 통신동작을 포함한 작업계획을 세우고 이것을 실행할 수 있으며, 실행 중 발생할 수 있

는 외부 환경변화에 신속히 대응하여 계획을 변경하거나 재 계획할 수 있다. 본 논문에서 개발한 지능형 이동에이전트 시스템은 응용 영역 고유의 지식만 제공할 수 있다면 손쉽게 다양한 응용 이동에이전트 시스템을 개발할 수 있는 편리한 개발도구로 이용될 수 있다. 반면에, 본 시스템의 이동 모델과 통신 모델은 몇 가지 제한점을 가지고 있다. 먼저 이동 모델 면에서 살펴보면, 첫째 동일한 장소로 동시에 진입하기 원하는 이동 에이전트들이 여러 개 존재할 때, 이들에 대해 하나의 이동 서버가 병행 서비스(concurrent service)를 제공할 수 없다는 점이다. 둘째는 이동 에이전트의 이동 동작 실행 중 직렬화(serialization) 단계나 재생(restoration) 단계에서 오류(error)가 발생했을 때 대처 방안이 제시되지 못함으로써 이동의 신뢰성(reliability)을 확보하지 못했다는 점이다. 통신 모델 면에서 살펴보면, 첫째, 이동 모델의 경우와 마찬가지로 메시지 서버의 병행 메시지 처리가 제공되지 못하고, 둘째, E-MAIL과 같은 비동기 방식의 에이전트간 통신을 지원하지 못하며, 셋째는 에이전트간에 교환되는 메시지가 일정한 문법과 의미론을 갖는 에이전트간 통신언어(inter-agent communication language)의 형태를 갖지 못했다는 점이다. 향후 이러한 제한점들을 해결하기 위한 연구들이 계속되어야 할 것이다.

참 고 문 헌

[1] C. G. Harrison, D. M. Chess, A. Kershenbaum, "Mobile Agents : Are they a good idea?," Technical Report, IBM T. J. Watson Research Center, 1995.

[2] Robert S. Gray. Agent Tcl : A transportable agent system. In Proceedings of the CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management(CIKM 95), Baltimore, Maryland, December, 1995.

[3] J. Baumann., F. Hohl. K. Rothermel., and M. StarBer., "Mole-Concepts of a Mobile Agent System," World Wide Web, Vol.1(3), pp.123-137, September 1998.

[4] Jaeho Lee, Edmund H. Durfee, and Patrick G. Kenny, Marcus J. Huber, UM-PRS : An Implementation of the Procedural Reasoning System for Multirobot Applications, Proceedings of CIRFFSS '94, pp.842-849, 1994.

[5] J. E. White, "Telescript Technology : Mobile Agents," General Magic White Paper, Appeared in Bradshaw, J., Software Agents, AAAI/MIT Press, 1996.

[6] J. E. White, "Mobile Agents," Software Agents, MIT Press and American association for Artificial Intelligence, 1997.

[7] Marcus J. Huber, "Jam Agents in a Nutshell", http://members.home.net:80/marcush/IRS, 1998.

[8] Michael Georgeff and Amy L. Lansky, Reactive Reasoning and Planning, In Proceedings of the Sixth National Conf. on Artificial Intelligence (AAAI-87), pp.677-682, Seattle, WA, 1987.

[9] Mitsuru Oshima, Guenter Karjoth, and Kouichi Ono, Aglets Specification 1.1, http://www.trl.ibm.co.jp/aglets/spec11.html, 1998.

[10] S. Hyacinth Nwana, "Software Agents : An Overview," Intelligent Systems Research, Advanced Applications & Technology Department, Knowledge Engineering Review, Vol.11, No.3, pp.205-244, October/November, 1996.

[11] H. Peine, and Stolpmann, T., "The Architecture of the Ara Platform for Mobile Agents," Proceeding of the First International Workshop on Mobile Agents(MA'97), Berline, Springer Verlag, LNCS 1219, pp.50-61, April, 1997.

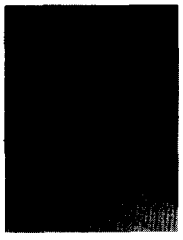
[12] Dag Johansen, Robbert van Renesse and Fred B. Schneider, Operating system support for mobile agents, Proceedings of the 5th. IEEE Workshop on Hot Topics in Operating Systems, Orcas Island, Wa, USA(4th-5th May, 1995), Published by : IEEE Computer Society, NY, USA, May 1995.

[13] Walsh, T., Paciorek, N., and Wong, D., "Security and Reliability in Concordia," Proceedings of the 31st Hawaii International Conference on Systems Sciences, VII : 44-53, January 1998.

[14] http://www.objectspace.com/products/voyager/index.asp

[15] http://www.generalmagic.com

[16] http://www.cs.bham.ac.uk/~sra/Thesis/index.html



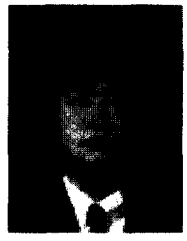
석 황 희

email : vinent2@kuic.kyonggi.ac.kr

1998년 경기대학교 전자계산학과 졸업(학사)

2000년 경기대학교 전자계산학과 졸업(이학석사)

관심분야 : 에이전트, 계획시스템, 분산 인공 지능 등



김 인 철

email : kic@kuic.kyonggi.ac.kr

1985년 서울대학교 수학과 졸업(학사)

1987년 서울대학교 전산과학과 졸업(이학석사)

1988년 서울대학교 중앙도서관 컴퓨터센터 연구조교

1995년 서울대학교 전산과학과 졸업(이학박사)

1991년 경남대학교 전산통계학과 전임강사

1995년 경남대학교 전산통계학과 조교수

1996년 RIACT AI Lab 객원연구원

현재 경기대학교 정보과학부 부교수

관심분야 : 계획시스템, 분산 인공 지능, 에이전트 등