JMoblet : Jini 기반의 이동에이전트 시스템

김 진 홍[†]·구 형 서[†]·윤 형 석[†]·안 건 태[†]·유 양 우[††]·이 명 준[†††]

요 약

Jini 구조의 네트워크 플러그 앤 워크(Network Plug and Work)기능은 분산 응용을 위하여 간단하면서도 유연한 네트워크 환경을 제공하고 있으며, 이를 통하여 이동에이전트 시스템의 동적인 등록 및 소재 파악의 기능과 에이전트의 활동에 유용한 서비스들의 동적 제공이 용이하게 지원될 수 있다. 본 논문에서는 Jini 기반 이동에이전트 시스템인 JMoblet 시스템에 관하여 기술하고 있다. JMoblet 시스템은 이동에이전트 시스템의 기본 기능인 에이전트 생성, 관리, 전송, 위치 파악 및 에이전트간의 통신 기능을 제공하고 있으며, 나아가 이동에이전트 시스템의 신뢰성을 위하여 예외 상황 처리 및 이동에이전트 시스템의 영속성 지원 기능을 제공하고 있다.

JMoblet : A Jini-based Mobile Agent System

Jin-Hong Kim[†]· Hyeong-Seo Koo[†]· Hyeong-Seok Yoon[†]· Geon-Tae Ahn[†]
Yang-Woo Yu[††]· Myung-Joon Lee[†††]

ABSTRACT

Jini architecture's Network Plug and Work provides simple and flexible network environment for distributed applications. Through the Jini technology, facilities for dynamically registering and locating mobile agent services can be easily supported, as well as the services useful for activities of mobile agents can be dynamically supported. In this paper, we describe a Jini-based mobile agent system named *JMoblet*, which provides the basic functions of a mobile agent system such as creation, control, transfer, location and communication among agents. To increase the reliability of the system, it also provides exception handling and persistence of the mobile agent systems for reliability.

키워드 : 이동에이전트(Mobile Agent), 이동에이전트 시스템(Mobile Agent System), Jini 기술(Jini technology), JMoblet 시스템(JMoblet system), 영속성(Persistence)

1. 서 론

인터넷의 사용이 보편화됨에 따라 인터넷을 통하여 정보제공 및 전자상거래 등의 편리한 서비스가 운영되고 있으며, 또한 무선 인터넷 기반의 이동 단말기를 이용한 서비스가 점차적으로 증가하고 있다. 이에 따라 빠르게 변화하는 사용자의 요구를 만족하면서도 편리하고 유용한 서비스를 제공하기 위하여 효과적인 분산 기술 및 분산 구조가 제시되어 왔으며, 그러한 노력의 일환으로서 사용자를 대신하여 자발적으로 행동하는 에이전트를 다른 시스템으로 이동시켜 작업을 수행하는 *이동에이전트* 기술[1, 2]과 *네트워크 플러그 앤 워크(Network Plug and Work)*라는 새로운 기능을 제공하는 *Jini* 기술[3]이 주목을 받고 있다.

*이동에이전트*는 사용자가 정의한 이동 경로를 따라 다른 여러 이동에이전트 시스템들로 자발적으로 이동하여 자신의 작업을 수행하고 그 결과를 자신의 사용자에게 보고하는 프로그램이다. 그리고 이동에이전트 시스템은 에이전트의 실행 환경과 에이전트의 생성, 이동, 자원 할당 그리고 자신이 생성한 이동에이전트의 상태를 제어하는 기본적인 기능들을 제공한다. 이동에이전트 기술은 많은 양의 데이터를 주고받는 서비스를 제공할 때 에이전트를 대상 시스템으로 이동시켜 실행함으로써 네트워크 트래픽을 줄이고, 서버의 기본 기능을 변경하지 않고 사용자의 다양한 요구사항을 지원할 수 있다. 이동에이전트 시스템을 이용한 서비스는 에이전트 프로그래밍에 의하여 결정되어지며, 고급화된 전자상거래 서비스, 전자 경매, 네트워크 관리, 과학적인 계산을 지원하는 분산처리 서비스 등에 적용될 수 있다.

이동에이전트 시스템에 대한 연구는 산업단체 또는 연구단체에서 최근 활발하게 진행되어 왔지만, 통일된 분산 기반구조를 채택하지 못하여 널리 사용되지 못하고 있다. 상이한 기

※ 본 연구는 정보통신부의 정보통신 우수시범학교 지원사업의 지원으로 수행되었음.
† 준 회 원 : 울산대학교 대학원 컴퓨터 · 정보통신공학부
†† 준 회 원 : 울산과학대학 컴퓨터정보학부 교수
††† 정 회 원 : 울산대학교 대학원 컴퓨터 · 정보통신공학부 교수
논문접수 : 2001년 10월 4일, 심사완료 : 2001년 12월 27일

반구조에서 작동하는 이동에이전트 시스템은 자신의 서비스를 사용자에게 편리하게 제공할 수 없으며, 사용자 또한 이동에이전트 시스템의 서비스를 용이하게 활용하기 힘들다.

1999년 1월에 발표된 Jini 기술은 실생활에서 사용하는 주변기기나 응용프로그램이 Jini 네트워크에 접속함과 동시에 동작하도록 하는 네트워크 플러그 앤 워크(Network Plug and Work)를 가능하게 하는 분산 패러다임으로 새롭게 자리잡고 있다.

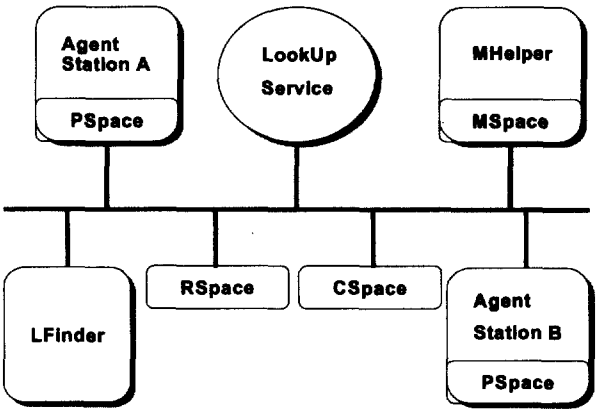
Jini의 Lookup & Discovery 서비스[4]는 Jini 서비스 형태인 이동에이전트 시스템의 동적인 등록 및 검색, 에이전트의 이동, 에이전트가 이용할 서비스의 동적인 등록 및 검색 등을 기본적으로 제공할 수 있으며, Jini의 여타 기능도 이동에이전트 시스템의 개발을 용이하게 하여준다.

본 논문에서는 이러한 Jini 기반 구조에서 동작하는 이동에이전트 시스템인 *JMoblet*에 대하여 기술한다. 개발된 JMoblet 시스템은 이동에이전트 시스템의 기능과 에이전트의 위치를 찾는 기능을 Jini 서비스 형태로 제공하여 이동에이전트 시스템의 서비스가 Jini 네트워크 하에서 자연스럽게 운용될 수 있도록 한다. 따라서, 일반적인 Jini 클라이언트에서 용이하게 Jini 서비스 형태인 JMoblet 시스템을 이용할 수 있다. 또한, 신뢰성 있는 이동에이전트 시스템을 위하여, JMoblet 시스템은 에이전트의 실행 시 발생하는 예외 처리 기능, 이동에이전트 시스템의 영속성(Persistence) 기능, 대상 이동에이전트 시스템 및 네트워크 문제로 인한 이동에이전트의 이동 불가 처리 기능, 그리고 이동에이전트 간의 통신 기능을 Jini 서비스인 *JavaSpace[5]*를 이용하여 제공하는 방법을 제시한다. Jini에서 제공되는 기본 서비스인 JavaSpace는 자바 객체를 직렬화하여 저장하고, 저장된 객체에 대하여 일정한 템플릿 형태로 탐색하고, 탐색된 객체를 가져오는 기능을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서는 JMoblet 시스템의 구조 및 각 모듈의 역할에 대해 설명하고, 3장에서는 이동에이전트 시스템의 신뢰성 지원을 위한 이동에이전트 시스템의 영속성 기능, 예외 상황 처리 기능, 그리고 에이전트 간의 통신 기능에 대하여 기술한다. 4장에서는 JMoblet 시스템의 구현에 대하여 알아본다. 5장에서는 Jini 기반 구조 및 최근 이동에이전트 시스템 구현 경향과 그들 시스템을 비교하며, 끝으로 6장에서는 결론 및 추후 연구 방향에 대하여 살펴본다.

2. JMoblet 시스템의 구조 및 기본 기능

JMoblet 시스템은 AgentStation, LFinder(Location Finder), MHelper(Moving Helper), RSpace(Resource Object Space), 그리고 CSpace(Communication Space)로 구성된다. JMoblet의 구성요소들은 Jini 서비스로 동작하며, 네트워크에 접속함과 동시에 다른 Jini 서비스나 사용자들이 사용할 수 있다. (그림 1)은 JMoblet 시스템의 각 구성요소를 포함한 구조를 나타내고 있다.



(그림 1) JMoblet 시스템의 구조

*AgentStation*은 이동에이전트의 실행 환경인 AgentPlace, 이동에이전트의 상태 추적 및 변경 기능을 제공하는 Agent-Monitor, 그리고 AgentStation의 자원을 관리하는 RManager로 구성되어 있다.

LFinder(Location Finder)는 AgentStation, AgentPlace, 그리고 이동에이전트의 위치정보를 관리하는 Jini 서비스이다. LFinder는 JMoblet의 구성요소와 이동에이전트에 대한 위치정보와 이름을 맵핑시킨 정보를 관리한다. 이동에이전트가 다른 AgentStation으로 이동하였을 때 LFinder에 저장되어 있는 위치정보 또한 변경된다.

MHelper(Moving Helper)는 이동에이전트가 목적지 Agent-Station으로 이동하지 못하는 경우, 이동 에이전트를 임시로 저장하고 관리한다. 일정한 간격으로 목적지 AgentStation으로 이동이 가능한지 파악하여 이동하려는 목적지 시스템이 작동할 때, MHelper는 이동에이전트를 목적지 AgentStation으로 이동시킨다.

RSpace(Resource object Space)와 CSpace(Communication Space) 각각 이동에이전트의 자원 접근 권한 정보를 가진 객체와 이동에이전트사이의 통신 객체를 저장하는 JavaSpace이다.

2.1 이동에이전트

2.1.1 이동에이전트 구조

JMoblet 시스템에서 이동에이전트의 속성은 에이전트 이름, 자신이 실행되는 AgentPlace, 에이전트 생성자, 에이전트 소유자, 에이전트 이동경로, 에이전트 자원등급, 현재 에이전트 위치정보이다. 이동에이전트는 Java의 직렬화(Serialization)를 통하여 네트워크를 통하여 전송될 수 있는 객체로 선언된다. 그리고 이동에이전트이름은 이동에이전트의 식별자로써 생성시간, 생성한 에이전트 시스템 이름, 작성자로 구성된다.

〈표 1〉 이동에이전트의 속성

속 성	의 미
Name	이동에이전트 이름(생성시간, 시스템 이름, 작성자)
AgentPlace	이동에이전트가 실행할 AgentStation 내의 AgentPlace 이름
Creator	이동에이전트를 생성한 사용자 이름(AgentStation의 이름)
Author	이동에이전트를 소유한 사용자 이름
Path	이동에이전트가 이동할 AgentStation의 경로
Resource Level	AgentStation의 자원을 접근할 접근 권한
Location	이동에이전트의 위치
run()	이동에이전트의 실행 루틴

2.1.2 에이전트의 이동(Mobility)

JMoblet 시스템에서 제공하는 에이전트 이동성은 약한 이동성(weak mobility)을 제공한다. 약한 이동성이란 에이전트의 상태 정보만을 전송하는 것을 말한다. 반면에 강한 이동성(strong mobility)은 에이전트 상태와 더불어 스택(stack)의 정보, 프로그램 카운터(program counter), 쓰레드 상태 정보 등 실행에 필요한 모든 정보를 전송하는 것을 말한다. JMoblet 시스템의 이동에이전트는 생성자의 요청에 의하여 이동하거나, 자신이 스스로 이동한다.

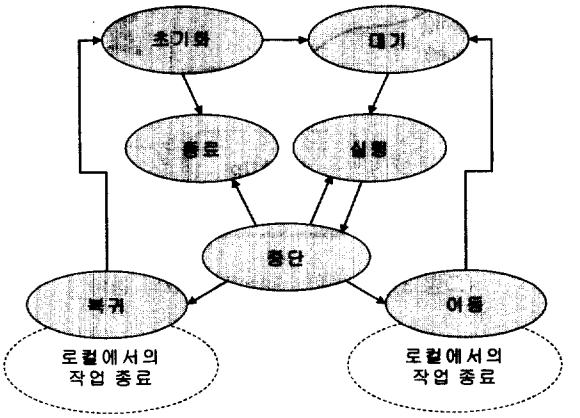
다음과 같은 과정으로 JMoblet의 에이전트는 목적지 AgentStation으로 이동한다.

① 이동에이전트는 목적지 AgentStation의 이름을 가지고 자신이 실행하고 있는 AgentStation에게 이동할 것을 요청한다.
② 요청을 받은 AgentStation은 이동에이전트를 직렬화(serialization)하고, LFinder(Location Finder)를 통하여 목적지 AgentStation의 이름으로 위치 정보(Location)를 알아낸다.
③ 요청을 받은 AgentStation은 목적지 AgentStation의 위치 정보를 가지고 Jini LookUp Service를 이용하여 목적지 AgentStation의 서비스 프락시를 다운받는다.
④ 목적지 AgentStation의 프락시를 통하여 목적지 AgentStation의 전송 함수를 호출하여 직렬화된 에이전트를 전송한다.
⑤ 목적지 AgentStation은 전송된 이동에이전트를 역직렬화(deserialization)하고 지정된 AgentPlace에 저장하고 실행시킨다.
⑥ LFinder에 이동한 에이전트의 위치정보를 변경한다.

2.1.3 에이전트의 생명주기(Life Cycle)

JMoblet 시스템의 이동에이전트는 생성 이후, 다음과 같은 상태로 변경하면서 자신의 작업을 수행한다. (그림 2)는 이동에이전트의 생명주기(상태변경)를 보여주고 있다.

- 초기화 : 생성된 직후, 실행되지 않은 상태이며, 수행할 작업을 부여받는 단계이다.
- 대기 : AgentPlace의 에이전트 저장소에 저장되어 시스템 자원을 할당받기 이전 상태이다.
- 실행 : AgentPlace에서 실행 중인 상태이며, 자신의 실행에 필요한 자원을 할당받는 단계이다.
- 중단 : 생성자 또는 실행 중인 AgentStation의 요구에 따라 실행에서 중단된 상태를 말한다.
- 종료 : 이동에이전트의 종료를 의미한다.
- 이동 : 직렬화되어 네트워크를 따라 이동되는 단계를 말한다.
- 복귀 : 생성된 AgentStation으로 이동하는 단계를 말한다.



(그림 2) JMoblet의 이동에이전트 생명주기

2.2 AgentStation의 구성 모듈

AgentStation은 이동에이전트 시스템으로서 이동에이전트의 실행환경을 제공하는 AgentPlace, 에이전트의 생명주기(Lifecycle)와 AgentPlace의 실행을 제어하는 AgentMonitor, 그리고 AgentStation의 시스템 자원을 제어하는 RManager(Resource Manager)로 구성되어 있다.

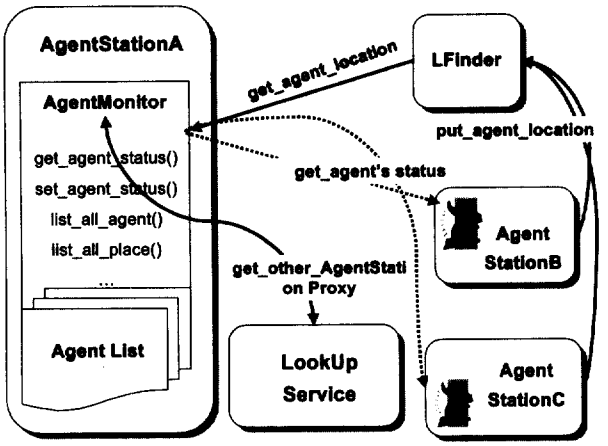
2.2.1 에이전트 플레이스(AgentPlace)

AgentPlace는 데몬(daemon) 형태로 독립적으로 수행되며, 이동에이전트의 실행 환경을 제공한다. AgentPlace는 에이전트를 저장하기 위하여 ArrayList를 사용한 저장소를 가지고 있으며, 이동에이전트의 생성 및 이동시 이 저장소에 추가 또는 삭제된다. AgentPlace는 AgentStation에서 독립적으로 여러 개가 존재할 수 있으며, 실행될 AgentPlace가 지정되지 않은 이동에이전트는 기본적으로 존재하는 DefaultPlace에서 실행된다.

2.2.2 에이전트 모니터(AgentMonitor)

AgentMonitor는 자신이 실행되고 있는 AgentStation의 AgentPlace 및 이동에이전트의 상태를 검색하고 제어하는 기능을 제공하며, 또한 자신의 AgentStation이 생성한 에이전트에 대한 원격 상태 검색과 제어를 할 수 있다. AgentMonitor는 데몬 형태로 AgentStation내에서 작업을 수행한

다. (그림 3)은 AgentMonitor의 메쏘드를 이용하여 Agent-Place 또는 이동에이전트의 상태를 알아보는 작동과정을 보여주고 있다. (그림 3)에서는 원거리에 있는 에이전트 상태를 알아보기 위해서 LFinder를 이용하여 에이전트가 실행 중인 AgentStation의 AgentMonitor를 이용한다.

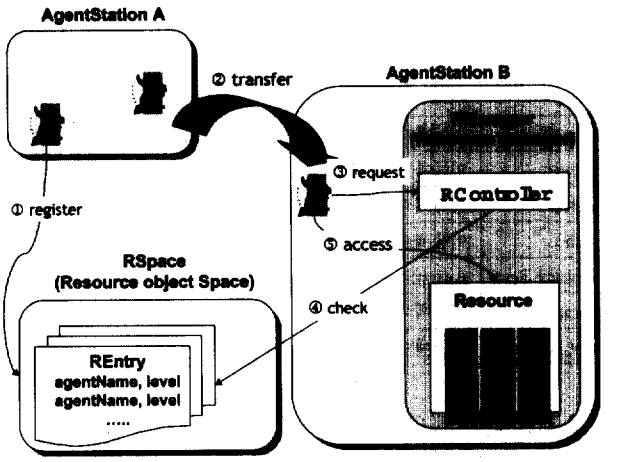


(그림 3) JMoblet의 AgentMonitor의 동작

AgentMonitor는 특정 이동에이전트의 상태를 검색하여 에이전트를 시작, 중단, 또는 재시작 시킬 수 있는 제어 기능을 지원한다. 예를 들어, 이동에이전트가 원격 AgentStation에서 수행 중에 시스템의 문제나 다른 요인으로 인해 수행이 중단되는 경우가 발생할 수 있다. 이때, 사용자는 LFinder와 AgentMonitor를 이용하여 이동에이전트의 상태를 실시간으로 파악하고 이동에이전트를 재실행시키거나, 종료시킬 수 있다.

2.2.3 RManager(Resource Manager)

RManager는 이동에이전트가 AgentStation의 자원을 접근하여 사용할 때, 에이전트의 자원 접근 레벨에 따른 자원 할당여부를 결정하는 기능을 제공한다. RManager는 Agent-Station의 자원 접근을 세 가지 클래스인 Admin_Service, User_Service, 그리고 Anony_Service을 통하여 접근할 수 있도록 하고 있다. 각 클래스는 자원등급에 맞추어 에이전트가 수행할 수 있는 저 수준의 API를 정의하고 있다. 그리고 시스템의 자원을 할당할 때 이용되는 이동에이전트의 자원 접근 정보는 이동에이전트 이름과 접근레벨의 정보를 가진 REntry 타입으로 RSpace에 저장된다. (그림 4)는 RManager가 이동에이전트가 요구한 자원의 할당 과정을 보여주고 있다. 이동에이전트는 계정에 따른 자원 등급을 가지고 이동에이전트 시스템에 도착한다. AgentStation은 이동에이전트의 자원 접근 레벨과 RSpace에 저장된 자원 관리 객체와 비교하여 등급이 같거나 작으면 그에 따른 자원을 할당하여 준다. 할당받은 자원 접근 객체를 통하여 이동에이전트는 AgentStation의 자원을 이용하여 실행할 수 있다.



(그림 4) RManager의 자원 할당 과정

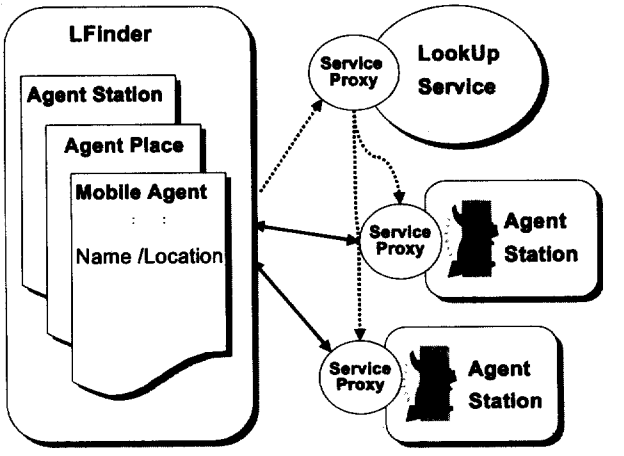
2.3 LFinder(Location Finder)

LFinder는 수행 중인 이동 에이전트 시스템과 플레이스, 그리고 이동 에이전트의 위치정보를 관리한다. JMoblet의 구성요소나 이동에이전트의 이름을 알고 있는 경우, LFinder는 각 이름에 해당하는 위치정보를 나타내는 문자열을 반환한다. 이 위치정보는 LookUp Service를 이용하여 실제 프락시를 얻을 때 사용된다.

위치정보는 문자열이며, jmoblet으로 시작하고 AgentStation 이름과 AgentPlace 이름으로 구성된다. 다음은 JMoblet 시스템에서 사용하는 위치정보를 나타내는 프로토콜이다.

jmoblet : //agentstation-name : port/agentplace-name/

AgentStation에 대한 위치정보는 AgentStation의 이름과 시스템이 운용되는 포트(port)가 " : "으로 표현되며, Agent-Place의 위치정보는 AgentStation 위치정보에 "/"를 구분자로 AgentPlace의 이름을 연결한 문자열로 기술된다. 그리고 이동에이전트의 위치정보는 자신이 실행되고 있는 Agent-Place의 위치정보와 같다.



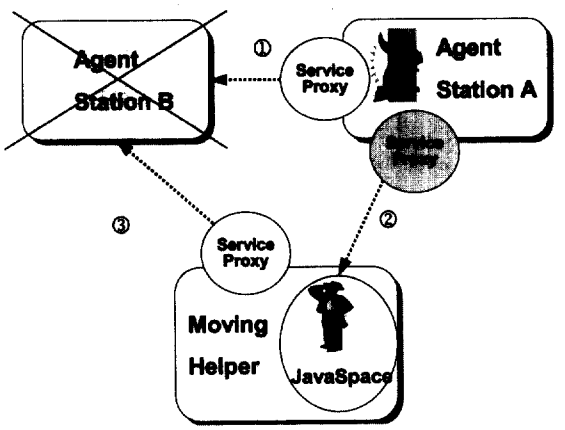
(그림 5) LFinder의 동작 과정

(그림 5)는 LFinder를 이용하여 다른 이동에이전트 시스템의 서비스 프락시를 얻는 과정을 보여주고 있다. 접근하고자 하는 AgentStation 이름으로 LFinder에서 위치정보를 얻어낸다. 그리고 LookUp Service를 통하여 AgentStation의 프락시를 얻어 RMI 통신을 한다.

2.4 MHelper(Moving Helper)

이동에이전트가 목적지 AgentStation으로 이동하지 못하는 경우, 이동에이전트는 MHelper의 MSpace(Missing Space)에 저장된다. MHelper는 목적지 AgentStation이 다시 작동할 때 저장된 이동에이전트를 이동시킨다. 이동에이전트는 AgentStation의 이름, 목적지 AgentStation의 이름, 그리고 이동 에이전트 객체를 인자로 가진 net.jini.core.entry.Entry 타입으로 MSpace에 저장된다.

(그림 6)에서 이동에이전트가 목적지 AgentStation으로 이동하지 못하는 경우를 처리하는 과정이다. 이동에이전트가 목적지 AgentStation의 문제나 네트워크의 문제로 이동할 수 없을 경우(그림 6-①), 이동에이전트는 MHelper의 MSpace에 임시로 저장(그림 6-②)된다. MHelper는 isAlive()를 이용하여 목적지 AgentStation으로 접속이 가능한 지를 일정 시간 간격으로 알아보고, 목적지 AgentStation과의 접속이 가능한 경우 이동 에이전트를 목적지 AgentStation으로 전송한다(그림 6-③).



(그림 6) MHelper의 작동 과정

2.5 에이전트간의 통신

JMoblet 시스템에서는 이동에이전트간의 통신 기능을 JavaSpace를 이용하여 지원하고 있다. 이동에이전트간의 통신을 위한 CSpace(Communication Space)는 전달하고자하는 에이전트, AgentStation, 그리고 메시지를 인자로 net.jini.core.entry.Entry 타입의 MEntry를 저장한다. <표 2>는 CSpace에 저장되는 MEntry의 속성을 나타내고 있다.

이동에이전트는 자신이 생성되거나 이동될 때, 에이전트 이름과 생성자, 그리고 소유자 정보를 가진 Entry 템플릿을 생성한다. 생성된 템플릿으로 자신이 메시지를 받을 AgentStation을 Listener로 등록한다. 특정 메시지가 CSpace에 저장되었을 때, CSpace는 메시지와 같은 템플릿으로 등록된 Listener에게 이벤트를 전달하여 메시지를 전달한다.

<표 2> 이동에이전트간의 메시지 형태

속 성		의 미
Sender_Agent		메시지를 보내는 에이전트 이름
Receiver_Agent		메시지를 받을 에이전트 이름(null : 그룹에 전달 시)
Agent_Creator		메시지를 전달받을 에이전트를 생성한 에이전트 시스템 이름
Agent_Author		메시지를 전달받을 에이전트를 소유하고 있는 사람 이름
Type	unique	특정 에이전트에게만 전달 방식 지정
	g_creator	Agent_Creator에서 기술된 에이전트 시스템이 만든 모든 에이전트에게 전달 방식 지정
	g_author	Agent_Author에서 기술된 에이전트 소유자의 모든 에이전트에게 전달 방식 지정
Message		전달될 메시지

3. 신뢰성을 위한 JMoblet 시스템의 기능

JMoblet에서는 이동에이전트 시스템이 가지는 기본적인 기능에 신뢰성 있는 이동에이전트 시스템을 위하여 이동에이전트 시스템의 상태 정보를 유지할 수 있는 영속성 기능, 그리고 에이전트 및 에이전트 시스템의 실행 시 발생 할 수 있는 예외 상황을 처리할 수 있는 기능을 제공한다.

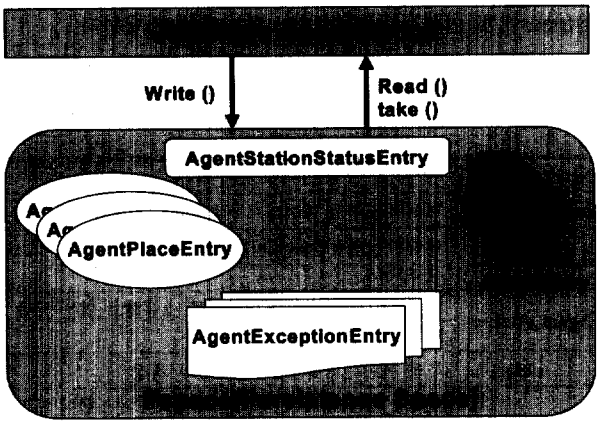
3.1 AgentStation의 영속성

이동에이전트 시스템은 시스템 다운이라는 최악의 상황에서도 이전의 상태로 복구하여 재 시작할 수 있는 기능이 제공될 필요가 있다. 그러므로 이동에이전트 시스템은 에이전트와 갱신된 시스템의 최근 정보를 영속적으로 유지하는 기능이 지원되어야 한다.

JMoblet 시스템은 AgentStation에서 실행되고 있는 이동에이전트, AgentPlace, 그리고 시스템의 정보를 PSpace(Persistence Space)이름의 JavaSpace에 저장한다. AgentStation은 다시 기동될 때 PSpace에 저장된 정보를 가지고 이전 상태로 복구될 수 있다. (그림 7)은 JMoblet 시스템에서 AgentStation의 영속성을 지원하기 위하여 PSpace에 저장되는 Entry의 종류 및 이를 다루는 JavaSpace의 오퍼레이션을 보여주고 있다.

실시간으로 변경되는 AgentStation의 정보에는 이동에이전트 상태에 대한 정보를 나타내는 AgentEntry, AgentPlace의 상태를 나타내는 AgentPlaceEntry, 그리고 AgentStation의 정상/비정상 종료를 표시하는 AgentStationStatusEntry가 있다. 또한 PSpace에서는 이동에이전트의 실행 시 발생하는 예외 상황에 대한 정보를 나타내는 AgentExceptionEntry

가 있다. <표 3>은 PSpace에 저장되는 각각의 Entry 정보가 저장 혹은 변경되는 시기와 삭제 시기를 나타내고 있다.



(그림 7) AgentStation의 영속성 지원

〈표 3〉 JMoblet의 AgentStation의 정보 저장 및 삭제 시기

	저 장 시 기	삭제시기
이동 에이전트	에이전트 생성 또는 이동 시 에이전트 상태 변경 시	에이전트 실행 완료 후
AgentPlace	AgentPlace 상태 변경 시 (AgentPlace 생성, 에이전트생성, 상태변경, 삭제)	AgentPlace 종료 시
종료상태	정상 종료 시	시스템 재시작 시
예외정보	에이전트 예외 발생 시	시스템 재시작 시

AgentStation이 재 시작할 때, 정상 또는 비정상인 종료 상태를 고려하여 다음과 과정을 거친다.

① AgentStation을 초기화한다.(Lookup Service, LFinder)
② PSpace에 있는 모든 AgentPlaceEntry를 이용하여 AgentPlace를 생성하여 실행한다.
③ PSpace에 있는 AgentStationStatusEntry를 참조하여 시스템의 정상/비정상 종료 판별한다.
- 정상 종료 시, PSpace에 저장된 모든 에이전트를 각각의 AgentPlace에서 실행시키다.
- 비정상 종료 시, 에이전트가 가진 자원 접근 등급에 따라 처리한다. 즉, 시스템 자원을 변경할 수 없는 에이전트는 바로 실행시키며, 시스템 자원을 변경할 수 있는 에이전트는 예외 상황을 발생시킨다(3.2.1절 참조).

3.2 JMoblet의 예외 처리 기능

JMoblet 시스템은 AgentStation의 비정상적인 종료 때 발생하는 예외 상황과 함께 목적지 AgentStation으로 이동할 수 없거나, 이동한 목적지 AgentStation의 자원을 사용하지 못하는 세 가지 예외 상황을 고려하여 처리한다. 발생된 예외 상황은 에이전트 시스템의 자원 접근 권한에 따라 처리되며, 발생된 예외 상황은 에이전트를 생성한 AgentStation에게 전달된다.

3.2.1 AgentStation의 비정상 종료 시 발생하는 예외 처리

이동에이전트 시스템의 비정상 종료에 따른 예외 상황은 시스템이 재 시작하면서 이전의 상태를 복구할 때 시스템의 상태를 변경할 수 있는 권한을 가진 에이전트에서 발생된다. 즉, 이전에 시스템의 정보를 변경하는 작업을 수행하던 중 시스템의 중단으로 인해 작업이 완료되지 않았기에 재시작 후 다시 자신의 작업을 수행한다면, 시스템의 정보를 중복해서 바꾸는 등의 문제를 초래할 수 있다. 이러한 경우, JMoblet 시스템에서는 이동 에이전트를 생성한 AgentStation에게 예외 상황을 통보한 후 에이전트를 종료시킨다.

3.2.2 AgentStation의 자원 접근 시 발생하는 예외 처리

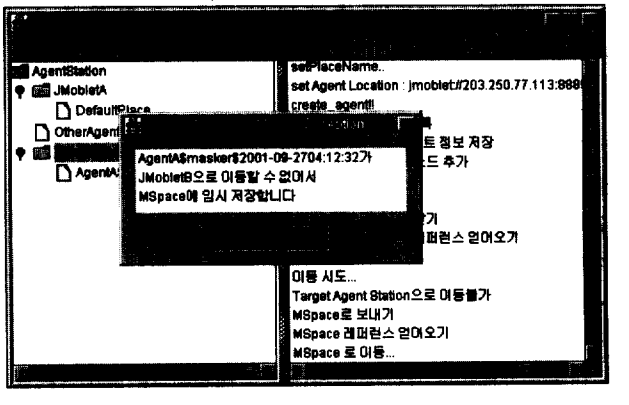
JMoblet 시스템은 AgentStation의 자원을 접근할 때 발생하는 예외 상황을 두 가지 경우로 나누어 처리하고 있다. AgentStation의 자원을 사용하여 시스템의 정보나 상태를 변경할 수 있는 권한을 가진 에이전트가 수행 중 자원에 접근하지 못하는 경우, 이동에이전트 시스템은 에이전트를 만든 생성 자에게 통보하고, 이전에 사용중인 자원을 반납하고 에이전트를 종료시킨다. 그리고 시스템의 자원에 영향을 미치지 않는 에이전트의 예외 상황은 에이전트를 생성한 AgentStation의 응답에 따라 재실행 또는 종료하도록 하고 있다.

다음은 에이전트를 생성한 AgentStation에게 보낼 메시지를 처리하는 함수를 나타내고 있다.

- ExMessage report_Exception(Name agent_name, ResourceName resource_name, string message) : 이동에이전트가 특정 자원을 접근 시, 발생하는 예외 상황에 대한 메시지를 에이전트를 생성한 AgentStation에게 전달한다.

3.2.3 이동에이전트의 이동 시 발생하는 예외 처리

이동에이전트가 목적지 AgentStation의 문제 혹은 네트워크의 문제로 인해 이동을 수행하지 못하는 경우, JMoblet 시스템은 이동에이전트를 MHelper의 JavaSpace에 임시로 저장해 두고 실시간으로 목적지 AgentStation으로의 이동가능



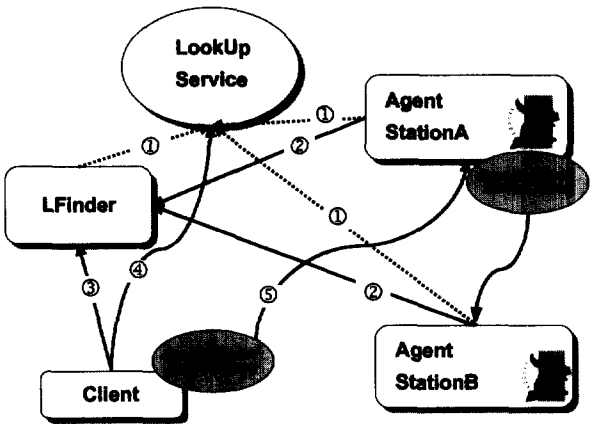
(그림 8) 이동 불가 예외 상황 처리

여부를 확인한 후 이동을 시도한다. 이동에이전트는 Missing-AgentEntry 형태로 JavaSpace에 저장된다. (그림 8)는 이동에이전트가 AgentStation으로 이동하지 못할 때 발생하는 예외 상황을 생성한 AgentStation에게 보고한 모습을 보여주고 있다.

4. JMoblet 시스템 구현 및 실행

4.1 JMoblet 시스템의 동작

다음은 JMoblet 시스템에서 지정된 경로를 따라 이동하여 자신의 작업을 수행하는 이동에이전트에 대하여 기술한다. 사용자에 의하여 생성되는 간단한 이동에이전트는 지정된 AgentStation으로 이동하여 자신의 작업을 수행한다. (그림 9)는 AgentStation의 기동 및 등록 과정과 생성된 이동에이전트가 이동하는 과정을 보여주고 있다.



(그림 9) JMoblet의 이동에이전트 이동 과정

1) AgentStationA, AgentStationB 그리고 LFinder는 기동시 Jini 서비스로써 LookUp Service와 LFinder에 등록된다(그림 9-①②).
2) 클라이언트는 자신이 생성한 이동에이전트를 AgentStationA에 이동시키기 위하여 LFinder에서 AgentStationA의 이름으로 위치 정보를 알아낸다(그림 9-③).
3) AgentStationA의 위치 정보를 가지고 LookUp Service에서 AgentStationA의 프락시를 가져온다(그림 9-④).
4) AgentStationA의 프락시를 이용하여 생성한 이동에이전트를 AgentStationA에 전송한다(그림 9-⑤).

4.2 클라이언트 프로그램

(그림 10)은 클라이언트에서 이동에이전트를 생성하여 목적지 AgentStation으로 이동하기 위한 프로그램이다.

클라이언트는 먼저 LookUp Service를 이용하여 LFinder의 프락시를 얻는다. 그리고 LFinder를 이용하여 목적지 AgentStation의 위치정보를 구하고, 이를 이용하여 목적지 AgentStation의 프락시를 얻는다. 마지막으로, 생성된 이동에이전트를 목적지 AgentStation의 transfer( )를 통하여 이동시킨다.

```
public class JMobletClient
{
  static LookupLocator lookup ;
  static ServiceRegistrar serviceRegistrar ;
  static JMobletLFinderService jmLFinder ;
  static JMobletAgentStationService jmAgentStationService,
       jmDestAgentStationService ;

  public static void main(String[ ] args) {
     // Get LookUp Service
     lookup = new LookupLocator(args[2]) ;
     // Get LookUp Service's ServiceRegistrar
     serviceRegistrar = lookup.getRegistrar( ) ;
     // Make LFinder's Attributes
     net.jini.core.entry.Entry aAttributes = new Entry[1] ;
     aAttributes[0] = new net.jini.lookup.entry.Name("JMoblet
     LFinder") ;
     template = new ServiceTemplate(null, null, aAttributes) ;
     // Get LFinder
     jmLFinder = (AgentStationService) registrar.lookup(template) ;
     // Get Destination AgentStation Location
     String destAgentStationServerLocation = jmLFinder.lookup
     ("AgentStationA") ;
     // Get Destination AgentStation
     jmDestAgentStationServer = getAgentStation(destAgent
     StationServerLocation) ;
     // Get a JMoblet's Agent and transfer it to the destination
     AgentStation
     Agent agent = new Agent("jMobletAgent," 10, path) ;
     jmDestAgentStationServer.trasnfer(agent) ;
  }
}
```

(그림 10) JMoblet의 클라이언트 프로그램

4.3 AgentStation 프로그램

(그림 11)은 AgentStation이 Jini 서비스로 등록하는 프로

```
public class JMobletAgentStation extends UnicastRemoteObject
       implements JMobletAgentStationService, ServiceIDListener,
       Serializable {

  public static void main(String[ ] args) {
     JMobletLFinderServer jmLFinder ;
     Entry[ ] aAttributes ;
     JoinManager joinmanager ;
     ...
     try {
        // Get LFinder's Reference
        jmLFinder = getLFinder( ) ;
        // Make AgentStationA's attributes
        aAttributes = new Entry[1] ;
        aAttributes[0] = new net.jini.lookup.entry.Name("JMoblet
        AgentStationA") ;
        // Make a AgentStation Object
        jmAgentStationA = new JMobletAgentStation("JMoblet
        AgentStationA) ;
        // Join JMobletAgentStationA into LookUp Service
        joinmanager = new JoinManager(jmAgentStationA,
                      aAttributes, jmAgentStationA, new
                      LeaseRenewal Manager( )) ;
        // Register JMobletAgentStationA to LFinder
        jmAgentStationALocation = makeLocation("JMoblet
        AgentStationA") ;
        jmLFinder.register("JMobletAgentStationA," jmAgent
        StationALocation) ;
        registerToLookupService("JMobletAgentStationA,"
        jmAgentStationALocation) ;
        ...
}
```

(그림 11) JMoblet 시스템의 AgentStation 프로그램

그램으로, 먼저 LFinder의 프락시를 구한다. 그리고 등록하려는 AgentStation을 생성하고, 이를 LFinder와 LookUp Service에 등록한다.

4.4 이동에이전트 프로그램

다음은 JMoblet 시스템에서 자신의 정보를 변경하는 이동에이전트에 대한 프로그램 코드이다.

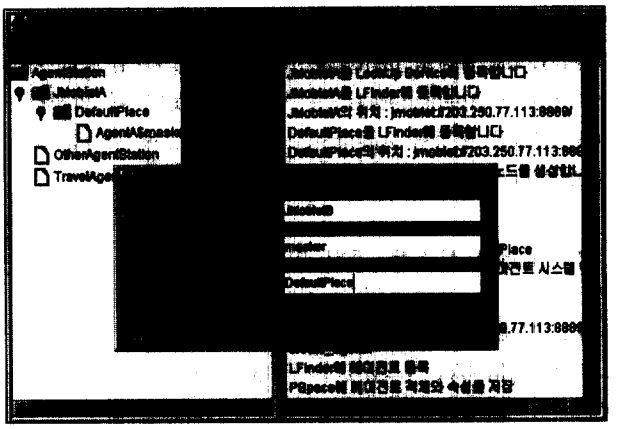
```
public class Agent implements Runnable, Serializable {
  public String agentName ;
  private int agentCounter ;
  private LinkedList agentPath ;
  ...
  public Agent(String name, int counter, LinkedList path) {
     this.agentName = name ;
     this.agentCounter = counter ;
     this.agentPath = path ;
  }
  public void run( ) {
     for(int I = 0 ; I < 10 ; I++)    // Change it's data
        agentCounter ++ ;
     // Move to the next AgentStation
     if(agentPath.size( ) != 0) {
        next = (String) agentPath.get(host) ;
        agentPath.remove(host) ;
        gotoNextHost(next) ;
     }
}
```

(그림 12) 간단한 이동에이전트 프로그램

(그림 12)는 에이전트 자신의 정보(agentCount)를 이동한 에이전트 시스템에서 증가시키는 간단한 에이전트 코드이다. 에이전트는 다른 이동에이전트 시스템으로 이동하였을 때, 쓰레드를 할당받아서 run( )을 실행한다. 연결리스트(linked list)로 표현된 에이전트 이동경로는 표현하고 있으며, 다음 이동할 에이전트 시스템의 이름을 추출하여 gotoNextHost( )를 호출하여 에이전트는 이동한다.

4.5 실행 결과

JMoblet 시스템은 이동에이전트의 생성과 이동 그리고 상태를 나타내는 기능을 위한 GUI를 제공하고 있다. AgentStation의 실행 모습과 기능을 (그림 13)에서 자세히 보여주고 있다. AgentStation에서 작동하는 AgentPlace와 각각의 AgentPlace에서 작동하는 에이전트들은 왼쪽 트리 패널에 나타나고, 각각에 대한 정보는 오른쪽 정보 패널에 나타난다. 그리고 AgentStation의 종료 및 재시작, 이동에이전트의 생성, 전송, 상태 변경의 에이전트 관리 기능과 AgentPlace의 생성, 삭제의 AgentPlace 관리 기능을 위한 툴바를 제공하고 있다. 또한 (그림 13)에서는 트리 패널에서 선택한 에이전트를 대상 AgentStation으로 이동하는 모습을 보여주고 있다.



(그림 13) JMoblet의 AgentStation의 실행 모습

5. 관련 연구

5.1 Jini 기반 분산 기술

Jini 기술은[3] 네트워크 상의 지능형 기기들이나 소프트웨어들이 Jini 네트워크에 접속과 동시에 서비스할 수 있는 기능(Network Plug and Work)을 제공하고, 사용자의 요구에 의한 서비스 요청(Services on Demand)을 처리할 수 있는 새로운 분산 기반 구조를 제공한다. Jini 기반 구조(Infrastructure)는 Jini 서비스를 자바의 객체로 저장하여 사용자의 요구에 따른 서비스를 관리하는 *서비스 관리자(Lookup Service)*, 사용자 및 서비스 제공자가 서비스 관리자의 위치를 자동적으로 파악하기 위한 *Discovery 프로토콜*, 그리고 서비스 제공자가 서비스 관리자에게 서비스를 등록하기 위한 *Join 프로토콜*로 구성된다. 그리고 분산 응용 프로그램의 개발을 위한 리스(Leases), 이벤트(Events), 그리고 트랜잭션(Transactions) 처리 프로그래밍 모델을 제공하여 분산된 자원의 효율적인 관리와 서비스간의 이벤트 및 트랜잭션 처리를 할 수 있다. JavaSpace는 분산 객체의 영속성과 데이터 교환 기능을 지원하는 Jini 서비스로서, 표준 인터페이스를 통하여 객체를 저장(write)하고 가져(take)올 수 있는 기능과 저장된 객체를 템플릿을 사용하여 탐색할 수 있는 기능 등을 제공한다. 또한 JavaSpace는 하나의 객체를 다중 사용자가 동시에 접근하려고 할 때 고려해야 할 동시성제어(concurrency control)를 지원한다.

5.2 타 이동에이전트 시스템 분석

현재까지 많은 이동에이전트 시스템이 개발되었다. 본 장에서는 JMoblet 시스템과 관련이 있는 Aglets[6], Concordia [7], Ajanta[8], 그리고 SMART[9-11] 시스템에 대하여 간략하게 살펴본다.

IBM의 *Aglets*은 IBM 도쿄 연구실에서 개발되었으며, "weak" 이동성[12]을 지원하는 Java 기반의 이동에이전트 시스템이다. Aglets은 call-back 기반의 프로그래밍 모델을 이용하여

에이전트에게 이벤트가 발생할 때마다 에이전트의 특정 메쏘드를 호출할 수 있다. 에이전트가 서버에 도착하면, 에이전트의 onArrival 메소드가 자동적으로 실행되는 것이 좋은 예라 할 수 있다. Aglets은 자신의 위치와 에이전트의 위치를 식별하기 위해 호스트 이름과 포트번호를 조합한 DNS 기반의 URL을 이용한다. Aglets은 비 동기적인 속성을 가진 "dispatch"와 동기적인 속성을 가진 "retract"을 이용하여 이동 기능을 지원하고 있다.

*Concordia*는 Mitsubishi 전자에서 Java를 이용하여 개발한 이동에이전트 시스템이다. Concordia는 Java의 직렬화와 클래스로딩 매커니즘을 이용하여 에이전트의 이동을 지원하지만, 쓰레드 레벨의 실행 상태 저장은 지원하지 않는다. Concordia에서 에이전트간 통신 기능은 비동기 이벤트 처리방식을 이용하여 지원되고 있으며, 신뢰성 있는 에이전트 전송과 서버나 에이전트의 복구 기능은 객체의 영속성 매커니즘을 적용하여 제공되고 있다. Concordia에서는 서버의 자원과 에이전트를 보호하기 위하여 사용자 ID를 이용하고, 악의 사용자로부터 에이전트와 시스템을 제한하기 위하여 암호화 프로토콜을 사용한다.

*Ajanta*는 미국의 미네소타대학에서 에이전트의 이동성을 지원하기 위해 Java 직렬화 매커니즘을 이용하여 개발된 이동에이전트 시스템이다. Ajanta 시스템에서 에이전트 코드는 에이전트가 명시한 서버로부터 동적으로 다운로드하여 사용할 수 있으며, 공개키 프로토콜을 이용하여 암호화 및 사용자 인증기능을 제공하여 에이전트 코드와 상태를 악의의 호스트나 에이전트로부터 방어할 수 있다. 에이전트의 이동 경로를 프로그래밍하기 위하여 이주 패턴(migration pattern)의 개념을 사용하였으며, 에이전트 이동성은 "weak" 속성을 지원한다.

*SMART*는 Java 언어를 이용하여 OMG MAF 명세를 따라 구현된 시스템으로 이기종간의 상호운용성을 지원하는 CORBA 기반의 이동에이전트 시스템이다. OMG MAF 명세에서 제시하고 있는 표준 인터페이스를 통하여 에이전트를 생성하고, 전송하고, 실행시키는 기능을 지원하고 있으며, 에이전트 및 이동에이전트 시스템의 위치를 파악하는 기능을 제공하고 있다. 또한 이동에이전트 시스템의 자원 접근 및 보안 정책을 지원한다.

〈표 4〉 이동에이전트 시스템의 기능 비교

기능 / 시스템	코드 이동성	시스템 자원 정책	영속성 지원	이동예외 처리기능	에이전트 간 통신	에이전트 제어
Aglet	weak	Yes	No	No	Yes	Yes
Concordia	weak	Yes	Yes	No	Yes	No
Ajanta	weak	Yes	No	Yes	Yes	Yes
SMART	weak	Yes	Yes	Yes	No	Yes
JMoblet	***weak***	***Yes***	***Yes***	***Yes***	***Yes***	***Yes***

<표 4>는 타 이동에이전트 시스템 및 JMoblet 시스템에서 제공하는 기능에 대하여 기술하고 있다. JMoblet 시스템은 에이전트의 자원접근 레벨에 따른 자원 클래스의 객체를 할당하여 자원을 접근하는 방법과 이동에이전트의 상태를 변경할 수 있는 제어기능을 제공하고 있다. 또한, JMoblet 시스템은 Jini 서비스로써 동작하는 이동에이전트 시스템의 신뢰적 기능으로 에이전트 시스템의 영속성, 이동예외처리, 에이전트간 통신 기능을 Jini 시스템에서 지원하는 기본 서비스인 JavaSpace를 기반으로 제공할 수 있는 방법을 보여 있다.

6. 결론 및 향후방향

이동에이전트 패러다임은 에이전트의 실행 코드와 관련 상태를 이동시켜 목적지 시스템에서 실행하는 기능을 통하여 인터넷 환경에서 특정 영역의 분산 애플리케이션을 개발하는데 유용하게 사용되고 있으며, 분산 서비스를 유연성 있게 지원하는 Jini 기술은 이동에이전트 시스템의 기본 기능을 구현하기에 적합한 분산 기반구조를 제공하고 있다. Jini에서 제공하는 Lookup & Discovery 서비스는 분산되어 있는 Jini 서비스 형태인 이동에이전트 시스템의 서비스를 동적으로 등록하고 등록된 서비스를 용이하게 찾을 수 있는 기능을 제공하며, 또한 이동에이전트를 자연스럽게 목적지 이동에이전트 시스템으로 이동시키고 그 위치를 추적할 수 있는 기능을 지원한다.

본 논문에서는 이러한 Jini 기반 구조하에서 Jini 서비스로 동작하는 이동에이전트 시스템인 JMoblet에 대하여 기술하였다. JMoblet 시스템은 에이전트 생성, 실행, 전송, 관리 등의 이동에이전트 시스템의 기능과 이동에이전트의 위치 파악 기능을 Jini 서비스 형태로 제공하고 있다. 그리고 Jini 서비스인 JavaSpace를 이용하여 신뢰성 있는 이동에이전트 시스템을 위한 기능을 제공하는 방법을 보여주고 있다.

JMoblet의 AgentStation은 이동에이전트의 실행환경을 제공하는 AgentPlace, 에이전트의 생명주기(Lifecycle)와 AgentPlace의 실행을 제어하는 AgentMonitor, 그리고 AgentStation의 시스템 자원을 제어하는 RManager(Resource Manager)로 구성되어 있다. JMoblet의 LFinder는 이동에이전트 시스템 및 동적으로 변화하는 이동에이전트의 위치를 추적할 수 있는 기능을 제공한다. 또한, JMoblet 시스템은 신뢰성 있는 서비스를 위하여 에이전트의 실행 시 발생하는 예외 상황 처리 기능, 이동에이전트 시스템의 영속성(Persistence) 기능, 그리고 이동에이전트 간의 통신 기능을 제공하고 있다.

현재의 JMoblet 시스템은 이동에이전트 및 이동에이전트 시스템간의 통신 방법을 JavaSpace를 통하여 제공하고 있지만, JavaSpace의 보안 기능이 취약함으로 인하여 안전한 통신 방법을 제공하지 못하고 있다. 앞으로 보안 기능이 구비된 JavaSpace를 개발하여 에이전트간의 통신을 안전하게 지원할 계획이며, 나아가 Jini의 리싱(leasing) 및 트랜잭션(tran-

saction) 기능을 활용하여 이동에이전트의 작업관리를 지원하도록 시스템을 확장할 계획이다.

참 고 문 헌

[1] N. M. Karnik and A. R. Tripathi, "Design Issues in Mobile -Agent Programming Systems," University of Minnesota, Dept. Computer Science and Engineering, IEEE, 1998.

[2] C. G. Harrison, D. M. Chess and A. Kershenbaum, "Mobile Agents : Are they a good idea?," Research Report, IBM, 1997.

[3] Jan Newmarch, "Jan Newmarch's Guide to JINI Technologies," http : //jan.netcomp.monash.edu.au /java/jini/tutorial/Jini.xml.

[4] Scott Oaks and Henry Wong, "Jini in a Nutshell," Oreilly Press, March 2000.

[5] Danny Ayers and Hans Bergsten, "Professional Java Server Programming," Wrox Press Ltd, August 1999.

[6] IBM Aglets. Available at URL : http : //www.trl.ibm.com /aglets/.

[7] Mitsubishi Concordia. Available at URL : http : //www.meitca.com/HSL/Projects/Concordia/.

[8] Minnesota Ajanta. Available at URL : http : //www.cs.umn.edu/Ajanta/.

[9] 유양우, 김진홍, 구형서, 박양수, 이명재, 이명준, "SMART : OMG의 MAF 명세를 지원하는 CORBA 기반의 이동에이전트 시스템", 한국정보처리학회논문지, 제8-C권 제2호, pp. 221-233, 2001.

[10] 유양우, 김진홍, 이명재, 박양수, 이명준, "SMART 이동 에이전트 시스템의 안전한 자원 접근 정책", 한국정보과학회 봄 학술발표논문집 Vol.27, No.1, pp.364-366, 2000.

[11] 유양우, 김진홍, 안건태, 문남두, 박양수, 이명준, "OMG MAF 명세를 지원하는 이동 에이전트시스템의 개발", 한국정보과학회 가을 학술발표논문집, Vol.26, No.2, pp.715-717, 1999.

[12] Alfonso Fuggetta, Gian Pietro Picco, Giovanni Vigna, "Understanding Code Mobility," IEEE Transaction On S/W Engineering, Vol.24, No.5, May, 1998.

김 진 홍

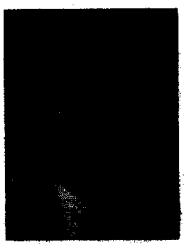
e-mail : karif99@dreamwiz.com

1999년 울산대학교 전자계산학과 졸업 (학사)

2001년 울산대학교 컴퓨터 · 정보통신공학부 졸업(석사)

2001년~현재 울산대학교 컴퓨터 · 정보통신 공학부 박사과정

관심분야 : 이동에이전트 시스템, 웹 프로그래밍, 분산객체 시스템, 생물정보학 등



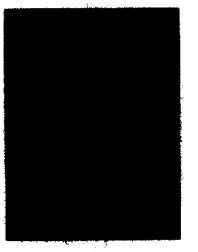
구 형 서

e-mail : masker@hanmail.net

2001년 울산대학교 컴퓨터정보통신공학부 졸업(학사)

2001년~현재 울산대학교 컴퓨터 · 정보통신 공학부 석사과정

관심분야 : 이동에이전트 시스템, 웹 프로그래밍, 생물정보학 등



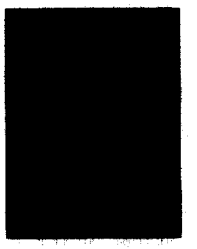
윤 형 석

e-mail : miracle@mail.ulsan.ac.kr

2001년 울산대학교 컴퓨터정보통신공학부 졸업(학사)

2001년~현재 울산대학교 컴퓨터 · 정보통신 공학부 석사과정

관심분야 : 이동에이전트 시스템, 네트워크 모니터링 시스템, 웹 프로그래밍, 생물정보학 등



안 건 태

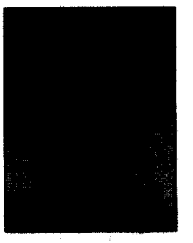
e-mail : java2u@lycos.co.kr

1999년 울산대학교 전자계산학과 졸업(학사)

2001년 울산대학교 컴퓨터 · 정보통신공학부 졸업(석사)

2001년~현재 울산대학교 컴퓨터 · 정보통신 공학부 박사과정

관심분야 : 이동에이전트 시스템, 그룹웨어, 웹 프로그래밍, 생물정보학 등



유 양 우

e-mail : soft@mail.ulsan-c.ac.kr

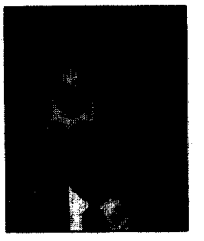
1995년 경일대학교 전자계산학과 졸업 (공학사)

1997년 울산대학교 전자계산학과 졸업 (공학석사)

2000년~울산대학교 전자계산학과 박사과정 수료

2000년~현재 울산과학대학 컴퓨터정보학부 근무(전임강사)

관심분야 : 이동에이전트 시스템, 그룹통신 시스템, 분산객체 프로그래밍 시스템 등



이 명 준

e-mail : mjlee@uou.ulsan.ac.kr

1980년 서울대학교 수학과 졸업(학사)

1982년 한국과학기술원 전산학과 졸업(석사)

1991년 한국과학기술원 전산학과 졸업(박사)

1982년~현재 울산대학교 컴퓨터 · 정보통신 공학부(교수)

1993년~1994년 미국 버지니아대학 교환교수

관심분야 : 프로그래밍언어, 분산 객체 프로그래밍 시스템, 병행 실시간 컴퓨팅, 인터넷 프로그래밍 시스템 등