연구논문집 제28권 제1호 pp. 131~141 울산과학대학
Journal of Research Vol. 28, No. 1, pp. 131~141, 2001
Ulsan College

견고한 이동 에이전트 시스템을 위한 영속성 및 예외처리 지원

유 양 우
컴퓨터정보학부

<요 약>

*이동 에이전트(mobile agent)*는 분산 응용 분야의 새로운 패러다임으로서, 최근에 이동 에이전트 플랫폼이 급증하고 있다. 이동 에이전트가 다른 컴퓨터로 이동하여 수행도중 에이전트 시스템이 비정상 종료되었을 때, 또는 자원 접근 권한이 없는 에이전트가 시스템 자원을 접근하고자 할 때 여러 가지 문제가 발생할 수 있다.

본 논문에서는 *MAF(Mobile Agent Facilities)* 명세의 표준 인터페이스를 구현하여 상이한 시스템간의 *상호운용성(interoperability)*을 지원하는 *SMART* 이동 에이전트 시스템을 개발하였다. SMART 시스템은 에이전트와 에이전트 시스템의 *영속성(persistency)*과 에이전트 *예외처리(exception handling)* 기능을 *자바스페이스(JavaSpace) 기술*을 이용하여 제공한다. 그 결과 보다 안정적이고 견고한 이동 에이전트 시스템을 구현하였다.

Supporting Persistency and Exception Handling for Robust Mobile Agent System

Yu, Yang Woo
School of Computer & Information

<Abstract>

A mobile agent is introduced as a new paradigm constructing distributed applications, and now a time applications using the paradigm are rapidly increased. However, various problems can be occurred when an agent of the application which is moved to another platform is anomaly shutdown, or when an agent which has not authority for a system tries to access resources of the system.

In this paper, we introduced a mobile agent system called SMART in order to resolve the problems. The SMART supports interoperability among heterogeneous systems by implementing standard specifications MAF proposed by OMG. In addition, it provides a persistency and exception handling using JavaSpace mechanism. From the facilities of SMART, developers can build mobile agent systems in more safe and robust manner.

키워드 : *이동에이전트, 이동에이전트 시스템, 상호운용성, 영속성, 자바스페이스.*

I. 서 론

네트워크 기술의 발달과 인터넷 사용자의 급속한 증가로 인해 최근 컴퓨터 프로그래밍의 패러다임에 큰 변화가 일어나고 있다. 특히, 이동 에이전트 기술의 보급으로 Aglets[1], Grasshopper[2], SMART와 같은 이동 에이전트 시스템이 개발되어 분산 시스템에서 에이전트가 사람을 대신하여 자율적으로 노드를 이동하며 작업을 수행할 수 있는 환경을 제공하게 되었다.

*이동 에이전트 시스템*의 기본적인 특성은 이동 에이전트를 인터넷상에 보내어 그들이 미리 설계된 경로를 따라 가거나 그들 자신이 모은 정보에 의해서 직접 동적으로 경로를 설정하여 돌아다닐 수 있도록 만든다[8]. 이들 에이전트들은 그들의 임무를 달성했을 때 그 결과를 사용자에게 전달하기 위하여 그들의 홈 사이트로 돌아오게 된다.

현재 개발된 다수의 이동 에이전트 시스템들은 그들의 구조와 구현이 상이하여 에이전트 기술의 빠른 확산과 상호운용성(interoperability)을 가로막고 있다. SMART 에이전트 시스템은 OMG(Object Management Group)에서 제안한 MAF(Mobile Agent Facilities)명세[3]의 표준 인터페이스를 구현하여 상호운용성을 지원하게 되었다.

한편, 이동 에이전트 시스템이 실행을 종료하고 다시 실행하는 경우 시스템은 자신의 이전 상태로 복원하여 재 시작될 필요가 있다. 또한 이동 에이전트 시스템의 비정상 종료나 다른 에이전트 시스템으로 이동하려는 에이전트가 수행 중에 발생할 수 있는 예외 상황을 효과적으로 처리할 수 있어야 한다.

본 논문에서는 SMART 시스템이 보다 안정적으로 운용되기 위하여 시스템의 상태를 저장하고 복원하는 영속성(persistence) 기능과 이동 에이전트가 수행 중에 발생하는 예외처리(exception handling)에 대하여 기술한다[22]. SMART 시스템의 상태와 이동 에이전트의 예외 상황 정보는 JavaSpace[4]에 저장된다. 시스템 정보의 저장소로 이용되는 JavaSpace는 자바 객체의 상태를 저장할 수 있는 효율적이고 간단한 메커니즘을 제공한다.

본 논문의 구성은 다음과 같다 2장에서는 이동 에이전트 시스템간의 상호운용성을 제공하는 SMART 시스템의 전반적인 구조와 각 모듈의 역할에 대해 설명하고, 3장과 4장에서는 SMART 에이전트 시스템의 영속성과 예외 처리 지원 방법에 대하여 기술한다[20]. 5장에서는 최근 에이전트 시스템들의 경향과 그들 시스템을 비교한다. 6장에서는 SMART 시스템에서 에이전트 생성, 이동, 그리고 추적 기능 등 시스템 구현 모습을 보여 준다. 끝으로 7장에서는 결론 및 추후 연구 방향에 대하여 살펴본다.

II. SMART 시스템의 소개

이동 에이전트 기술에서 가장 중요한 목적은 다양한 에이전트 시스템 사이의 상호운용성이다. 에이전트 전송과 클래스 전송, 그리고 에이전트 관리에서 표준화가 된다면 상호운용성은 더욱더 쉽게 이루어질 수 있다. 이를 위하여 OMG에서는 이종의 이동 에이전트 시스템에 대하여 상호운용 가능한 인터페이스를 정의하여 MAF 명세를 작성하였다[3]. SMART 시스템은 이러한 MAF 명세의 표준 인터페이스를 Java 언어로 구현한 CORBA 기반의 이동 에이전트 시스템이다[20].

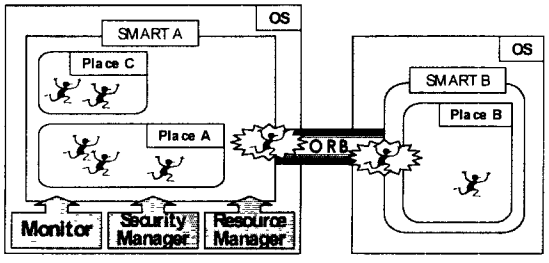

그림 2. SMART 시스템 구조

SMART 시스템은 그림 1과 같이 효율적인 이동 에이전트 시스템을 구축하기 위하여 에이전트를 실행시키고, 중지, 종료시키는 플레이스(place), 에이전트를 추적하고 감시하는 모니터(monitor), 에이전트의 안전한 전송을 위한 보안 관리자(security manager), 그리고 자원 관리자(resource manager) 등 네 개의 모듈로 구성되어 있다.

1. 에이전트(SMART Agent)

이동 에이전트는 한 에이전트 시스템에서 다른 에이전트 시스템으로 이동하여 자발적으로 행동하는 컴퓨터 프로그램이다. SMART 시스템에서 에이전트는 자바 쓰레드로 동작하며, 에이전트가 자신의 작업을 모두 완료하면 미리 설계된 경로를 따라 이동하거나, 그들 자신이 모은 정보에 의해 직접 동적으로 경로를 설정하여 돌아다닌다. 클라이언트에서 에이전트를 생성할 때 *create_agent()* 메쏘드를 이용한다.

2. 플레이스(Place)

플레이스는 쓰레드 형태로 에이전트가 실행하는 환경을 제공한다. 플레이스는 한 시스템에서 생성된 후 MAFFinder의 인터페이스인 register_place() 메소드를 이용하여 MAFFinder에 등록된다. 에이전트의 전송과정을 살펴보면, 목적지 에이전트 시스템의 참조를 구한 후 에이전트 객체를 직렬화(Serialization)하고, 목적지 에이전트 시스템의 receive_agent() 메소드를 호출함으로서 에이전트 전송은 완료된다. 목적지 에이전트 시스템에 도착한 에이전트는 인증 과정을 거치고 역직렬화(Deserialization)된 후 해당 플레이스에 저장되어 자원접근 등급에 따라 자원을 할당받아 실행하게 된다.

3. 모니터(Monitor)

SMART 시스템이 기동될 때, 모니터는 하나의 데몬 형태로 자신의 수행을 시작한다. 모니터의 주된 임무는 크게 이동 에이전트의 감시와 플레이스를 감시하는 기능을 제공한다. 또 다른 기능으로, 원거리에 있는 에이전트 시스템에게 특정 플레이스에 주어진 이름으로 에이전트를 만들 수 있도록 요청할 수 있다.

모니터는 플레이스 상태와 에이전트 상태를 파악하고, 플레이스에서 실행 중인 에이전트의 상태(running, suspended, terminated)를 제어할 수 있는 기능을 제공한다.

4. 보안 관리자(Security Manager)

OMG 의 MAF 명세에서는 에이전트가 에이전트 시스템의 기능을 호출하거나 다른 에이전트 시스템으로 이동하고자 할 때 네트워크 통신 보안의 품질에 대해 기밀성, 무결성, 인증, 재실행 탐지 등의 네 가지 요구사항을 제안하였다[3].

SMART 시스템의 보안 관리자는 *안전한 소켓 계층(Secure Sockets Layer : SSL)*을 기반으로 위의 네 가지 요구 사항을 구현하였다. SSL은 대칭 또는 비대칭적인 암호화 방법을 실제 사용할 수 있도록 작성한 산업 표준 프로토콜로서 기밀성과 데이터 무결성

그리고 클라이언트/서버의 상호 인증을 제공한다. SMART 시스템에서는 *Protekt 3.0 SSL* 패키지에서 제공하는 API를 이용하여 OMG MAF 명세에서 요구하는 세 가지 사항을 만족시켰다[5]. 네 번째 요구 사항인 재실행 탐지는 에이전트의 이름 속성에 계정과 권한 정보를 에이전트 시스템의 로그(log) 파일과 비교하여 구현하였다.

5. 자원 관리자(Resource Manager)

SMART 시스템의 자원에 대한 접근은 독자적으로 자원관리자(ResourceManager) 클래스를 구현하여 자원 접근 정책을 적용하고 있다. 자원관리자 추상클래스를 상속받아 서비스에 따른 세 가지 클래스 Admin_Service, User_Service, Gypsy_Service를 구현한다. 각 클래스는 자원등급에 맞추어 에이전트가 수행할 수 있는 모든 저 수준의 API를 정의하고 있다[20].

SMART 시스템에서 이동 에이전트가 가질 수 있는 자원 등급은 Admin, User, Gypsy 세 가지로 분류된다. 에이전트는 계정에 따른 자원 등급과 자신이 사용할 수 있는 서비스 객체의 메쏘드 정보를 가지고 에이전트 시스템에 도착한다. 에이전트 시스템에서는 이러한 정보를 원격 호스트에서 유지하고 있는 자원 관리 객체를 통해 자원 등급을 비교하여 등급이 같거나 작으면 그에 따른 자원을 할당하여 준다. 이 자원 관리 객체는 CORBA를 이용하여 구현하였으며, 저장하고 있는 정보는 계정과 그와 관련된 자원등급 그리고 사용하고자 하는 객체와 메쏘드 정보를 유지하고 있다. 이를 관리하는 관리자(Manager)는 계정에 따른 자원 등급을 상향시키거나 조절할 수 있는 권한을 가진다. SMART 시스템은 이동 에이전트가 사용할 수 있는 객체만을 사용하고 있는가를 감시하는 자원모니터(ResourceMonitor)가 있다 이러한 기능의 구현은 에이전트가 동적 클래스 로더를 이용할 때 그 정보를 알아낼 수 있다.

III. SMART 시스템의 영속성 지원

MAF 명세는 에이전트 생성, 이동 그리고 에이전트 시스템과 이동에이전트의 위치 추적 기능 등 상호운용성을 지원하기 위한 기본적인 기능에 대하여 기술하고 있다. 그러나 이동에이전트 시스템이 상호운용성을 지원하고 보다 안정적인 시스템으로 구축하기 위해서 추가적인 기능이 요구된다. 추가적인 기능으로는 에이전트와 에이전트 시스템의 영속성, 에이전트 예외처리 기능이 있다. 에이전트 시스템은 자신의 정보(실행 중인 에이전트, 플레이스)를 저장하고 종료하여, 종료 이전의 정보를 복구할 필요가 있다. SMART 시스템은 에이전트 시스템의 영속성과 예외를 발생한 에이전트를 저장하는 저장소로 JavaSpace[4]를 이용한다

JavaSpace는 분산 객체의 영속성과 데이터 교환을 지원하는 Jini[6]서비스로써, 표준 인터페이스를 통하여 net.jini.core. entry.Entry 형태의 객체를 저장(write)하고 구할(take) 수 있는 기능과 저장된 객체는 필드들이 탐색 패턴으로 설정된 템플릿을 사용하여 탐색 기능 등을 제공한다. SMART 시스템의 영속성은 분산 객체의 영속성을 지원하는 JavaSpace의 표준 인터페이스와 JavaSpace에서 객체의 존재 시간을 지정하는 리스(Lease) 개념을 이용하여 보다 쉽게 지원된다.

그리고, JavaSpace의 하부 기능을 제공하는 Jini는 네트워크상의 지능형 기기들이나 소프트웨어들이 동적인 상호 작용을 지원하는 기술이다. Jini는 접속과 동시에 각 기기들의 중재자 역할을 수행하는 연합체(Loopup Service)를 형성하고 이 연합체를 통하여 서비스를 등록하고 클라이언트로부터 요구된 서비스를 지원한다

SMART 시스템은 Jini 시스템상의 에이전트 시스템의 영속성을 위한 서비스로써 JavaSpace를 이용하고 있다.

1. 영속성 지원을 위한 시스템 구조

SMART 시스템은 영속성을 지원하기 위해 실행 중인 에이전트와 플레이스의 상태 정보를 그림 2.와 같은 구조로 저장한다.

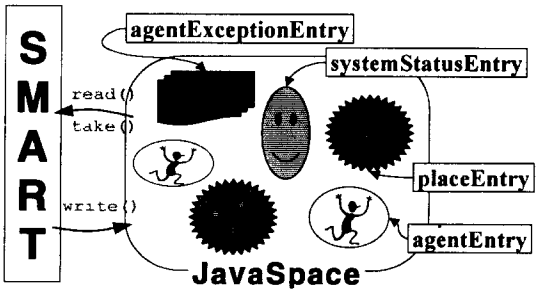

그림 3 SMART 시스템의 영속성 지원

시스템의 정보를 저장하기 위해 사용되는 JavaSpace는 SMART 시스템마다 존재하며, 표준 인터페이스를 통하여 net.jini.core.entry.Entry[6,7] 형태의 객체를 저장하고 복원한다. 즉, write() 메소드로 엔트리를 저장하고, read() 메소드로 주어진 템플릿과 일치하는 엔트리를 읽어온다. take() 메소드는 read() 메소드와 달리 엔트리를 읽어온 후 제거한다.

SMART 시스템은 현재 실행 중인 에이전트 정보를 agentEntry에, 플레이스의 정보를 placeEntry에 저장하고, 시스템의 종료 상태와 에이전트의 예외 정보를 각각 systemStatusEntry와 agentExceptionEntry에 저장한다. 표 1.은 JavaSpace에 시스템의 정보를 저장하고 삭제하는 시기를 보여준다.

표 1. 시스템 정보 저장 및 삭제 시기

	저장 시기	삭제시기
Agent	에이전트 생성 또는 이동 시 에이전트 상태 변경 시	에이전트 종료 시
Place	플레이스 상태 변경 시(플레이스 생성, 에이전트 생성, 상태변경, 삭제)	플레이스 종료 시
종료 상태	정상 종료 시	시스템 재시작 시
예외 정보	에이전트 예외 발생 시	시스템 재시작 시

2. 영속성 지원 구현

SMART 시스템은 재시작 시 다음과 같은 과정을 거치게 된다.

① 시스템 초기화(ORB, Naming Service, MAFFinder, Monitor)

② JavaSpace에서 Place를 가져와서 실행

③ JavaSpace에 저장된 systemStatusEntry 값을 참조하여 시스템의 정상/비정상 종료 판별

㉠ 정상 종료의 경우 : 모든 에이전트를 실행

㉡ 비정상 종료의 경우 : 자원 접근 등급에 따라 에이전트를 실행

- 시스템 자원을 접근하지 않는 에이전트는 바로 실행
- 시스템 자원을 접근하는 에이전트는 예외 발생

IV. SMART 시스템의 예외 처리 지원

일반적으로 에이전트 시스템에서 다음과 같은 예외상황이 발생할 수 있다.

- 이동 에이전트가 시스템의 자원에 대한 접근이 거부되는 경우

· 이동 에이전트가 목적지 에이전트 시스템을 찾지 못하는 경우
· 에이전트 시스템이 비정상 종료하는 경우

1. SMART 시스템의 예외 처리 방법

SMART 시스템은 발생한 예외 상황을 에이전트의 자원 접근 등급에 따라 처리한다. 자원 접근 등급은 이동 에이전트가 에이전트 시스템의 상태를 변경할 수 있는 권한을 판별하기 위해 사용된다. Admin과 User 자원 접근 등급을 가진 이동 에이전트는 실행 중인 에이전트 시스템의 상태를 변경할 수 있으며, Gypsy 자원 접근 등급을 가진 이동 에이전트는 에이전트 시스템의 상태를 변경할 수 없다.

SMART 시스템에서 예외 처리는 다음과 같다. 먼저 예외 상황을 접한 에이전트 시스템은 이동 에이전트를 생성한 에이전트 시스템에게 표 2.와 같이 자원 접근 등급(시스템 상태 변경 가능 여부)에 따라 예외 상황을 알려준다. 예외 상황을 보고 받은 에이전트 시스템은 에이전트의 재실행 또는 종료를 결정한다.

이동 에이전트가 목적지 에이전트 시스템을 찾지 못할 경우에는, 그림 3.과 같이 각 Region[3] 마다 존재하는 이동경로 관리자의 JavaSpace에 임시적으로 저장된다. 이 에이전트는 이동하려는 목적지 에이전트 시스템과 실행 중인 에이전트 시스템, 그리고 에이전트를 인자로 MissingAgentEntry 형태로 저장된다. 이동 경로 관리자는 JavaSpace에 저장된 에이전트를 목적지 에이전트 시스템으로 전송하는 기능을 한다.

표 2. SMART 시스템의 예외 처리

예외 상황	객체저장 위치	시스템 상태변경 가능 여부	예외 처리 방법
자원 접근 불가	에이전트 시스템의 JavaSpace	가능	에이전트를 생성한 에이전트 시스템에게 통보 후, 사용 중인 자원을 반납하고 에이전트 종료
		불가능	에이전트를 생성한 에이전트 시스템에게 통보 후, 응답 결과에 따라 에이전트 재실행 또는 종료
목적지 에이전트 시스템으로 이동 불가	이동경로 관리자의 JavaSpace	가능	에이전트를 생성한 에이전트 시스템에게 통보 후, 이동경로 관리자에게로 이동
		불가능	에이전트를 생성한 에이전트 시스템에게 통보 후, 이동경로 관리자에게로 이동
시스템 재시작	에이전트 시스템의 JavaSpace	가능	에이전트를 생성한 에이전트 시스템에게 통보 후, 사용 중인 자원을 반납하고 에이전트 종료
		불가능	에이전트를 생성한 에이전트 시스템에게 통보 후, 응답 결과에 따라 에이전트 재실행 또는 종료

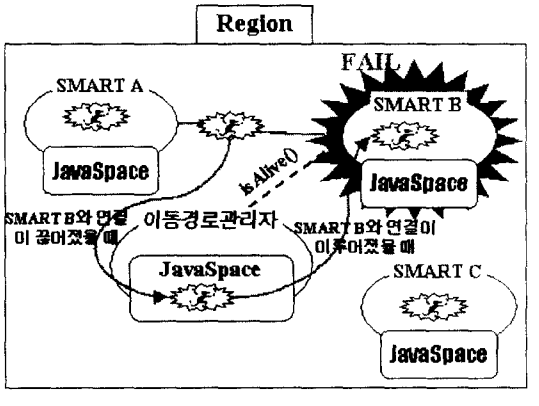

그림 4. 에이전트 전송 중 예외 처리

2. 예외 처리 구현

이동 에이전트가 목적지 에이전트 시스템으로 이동하지 못하거나, 시스템의 자원을 접근하지 못할 경우, SMART 시스템은 에이전트의 예외 상황을 에이전트를 생성한 에이전트 시스템에게 통보한다.

SMART 시스템은 MAF 명세[3]의 인터페이스에 다음과 같은 메소드를 추가하여 에이전트를 생성한 에이전트 시스템에게 예외 처리를 요청하였다.

```
· AgentExMessage report_Excpetion( in Name agent_name,
        in ResourceName resource_name,
        in string message, in MAFAgentSystem exception_catcher);
  // 에이전트는 목적지 에이전트 시스템의 자원 정보와 예외 상황을
  에이전트를 생성한 에이전트 시스템에게 전달한다
· boolean release_resource(in Name agent_name,
        in ResourceName resource_name),
  // 예외가 발생된 에이전트가 점유한 시스템 자원을 해제한다.
```

이동 경로 관리자는 ***isAlive()*** 메소드를 이용하여 이동 에이전트가 목적지 에이전트 시스템으로 접속이 가능한 지 여부를 일정 시간 간격으로 알아보고, 접속이 가능할 경우 JavaSpace에 저장된 에이전트를 목적지 에이전트 시스템으로 전송한다. MissingAgentEntry는 기본적으로 24시간의 Lease[4]로 설정되며, Lease를 초과하게 되면 JavaSpace에서 자동적으로 삭제된다. 그림 4.는 실행 중인 SMART 시스템에서 이동 에이전트의 예외 처리 과정을 보여주고 있다.

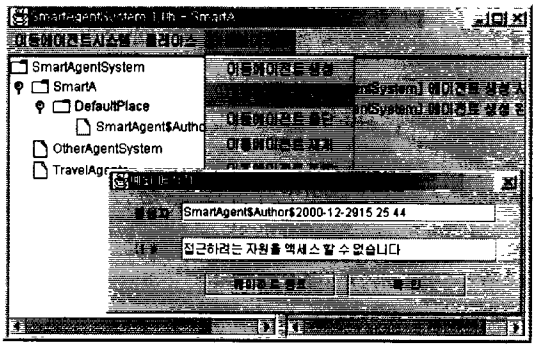

그림 5. SMART 시스템의 예외처리

V. 관련 연구

현재 이동 에이전트에 관한 연구는 활발히 진행되고 있다. 다양한 에이전트 시스템들은 에이전트의 이동성을 지원하기 위하여 제공하는 기법 또한 서로 다르다. 이 장에서는 지금까지 개발된 시스템들의 특징을 설명하고, 각 시스템에서 사용한 이동 에이전트의 이동성 기법에 대하여 비교한다.

IBM에서 개발한 에이전트 시스템으로는 *Aglets*이 있다[1]. 프로그래밍 언어는 Java를 이용하여 개발하였으며, 에이전트 이동성은 "weak" 속성을 가진다. Aglets API는 통신환경으로서 "context"의 개념을 제공한다. aglet의 context는 몇 가지 기본적인 서비스를 제공한다. Aglets은 두 개의 이주(migra tion) 오퍼레이션을 제공한다. 하나는 "dispatch"로서 독립적인 코드를 파라미터에 명시한 context로 이동시킨다. 이 기법

은 비 동기적인 속성을 가진다. 이와 대칭적인 오퍼레이션은 "re tract"로서 코드를 가져오고, retract가 실행되었던 context로 돌아오도록 요구하는데 사용된다. 이는 동기적인 속성을 가진다.

미 캘리포니아 대학에서 개발된 에이전트 시스템인 *Java-To-Go*이다[14]. 시스템 환경은 Java 프로그래밍 언어를 사용하여 개발하였으며, 에이전트 시스템의 기본적인 특성을 지원하는 비교적 간단한 시스템이다. 에이전트 이동성은 "weak" 속성을 가진다. 상호운용성은 지원하지 않는다. 특징은 에이전트 실행 환경인 "Hall"을 여러 개 수행할 수 있으며, 이를 관리하는 하나의 "Community" 서버에는 여러 개의 "Hall"이 수행된다. Community 서버는 또 다른 기능으로 에이전트 실행 시 필요한 클래스를 원격 에이전트 시스템으로부터 동적으로 가져올 수 있다.

*Ajanta*는 인터넷 상에서 이동 에이전트를 이용하여 애플리케이션을 프로그래밍 하기 위한 Java 기반의 시스템이다 [17]. 미네소타 대학에서 개발하였으며, 다양한 기술을 적용한 시스템이다. 먼저 에이전트가 서버의 자원을 액세스할 때, 프록시(proxy) 기반 제어 기법이 사용된다. 또한 에이전트의 순회 패스를 프로그래밍하기 위하여 이주 패턴(migration pattern)의 개념을 사용하였다. 에이전트 이동성은 "weak" 속성을 지원하고 상호운용성은 제공하지 않는다.

*Grasshopper*는 독일의 GMD Focus 연구소에서 개발한 에이전트 시스템으로써 시스템 환경은 Java 프로그래밍 언어를 사용하여 개발하였다[2]. 대표적인 특징은 OMG의 이동 에이전트 표준을 적용하여 구현한 첫 번째 에이전트 시스템이다. 에이전트 이동성은 "weak"속성을 가진다, 그리고 OMG MAF 명세를 따라 구현한 시스템이므로 이기종간의 상호운용성을 지원한다[2]. Grasshopper 시스템은 RMI, Socket, IIOP 등 다양한 통신 채널을 제공한다

표 3. 에이전트 시스템 비교

에이전트 시스템	상호 운용성	코드 이동성	보 안 정 책
Aglets	○	weak	Yes (MAC/MIC)
Java-To-Go		weak	No
Ajanta		weak	Yes (Proxy Object, RSA)
Grasshopper	○	weak	Yes (SSL/X.509 certificate)
SMART	○	weak	Yes (SSL, ResourceManager)

Ⅵ. SMART 시스템 실행 모니터링

SMART 시스템의 기동된 전체 모습과 기능을 그림 5.와 그림 6.에서 자세히 보여주고 있다. SMART 시스템은 자신의 IP주소, 포트번호, 시스템 이름, 그리고 CORBA 네이밍 주소 정보를 가지고 기동된다. 전체 프레임의 구성은 에이전트 시스템, 플레이스 그리고 에이전트에 해당하는 노드를 표

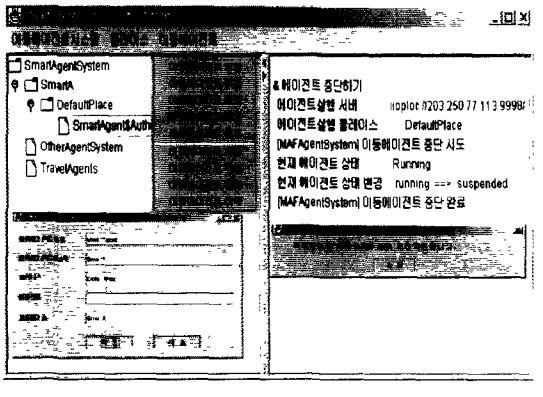

그림 6. 이동에이전트 생성과 이동

현한 트리 패널과 각각의 노드 정보를 나타내는 정보 패널, 그리고 메뉴바로 구성된다.

그림 5.는 이동에이전트 생성 메뉴를 통하여 에이전트 생성 기능과 에이전트 상태 정보, 에이전트에 대한 제어 정보를 정보 패널에 보여주고 있다.

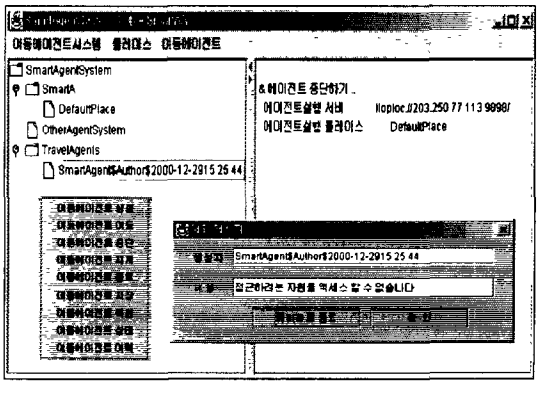

그림 7. 이동에이전트 정보 및 예외 처리

그림 6.은 이동 에이전트를 목적지 이동 에이전트 시스템으로 이동시키고, DefaultPlace 노드에서 TravelAgents노드로 이동됨을 보여 주고, 이 노드의 정보를 통하여 현재의 에이전트의 위치와 상태 정보를 알아 볼 수 있는 기능을 제공하고 있다. 그리고 이동시킨 이동 에이전트가 목적지 에이전트 시스템의 자원을 접근할 수 없는 상황을 알려주고 이를 처리하는 모습을 보여준다.

SMART 시스템에서 에이전트의 이름, 현 위치, 현 상태, 그리고 이동한 경로 이력 등의 에이전트에 대한 정보 보기 기능을 제공하고 있다.

VII. 결 론

에이전트 패러다임은 자신의 코드를 이동시켜 목적지 시스템에서 실행시키는 특성을 제공하는 관계로, 인터넷 환경에서 분산 애플리케이션을 개발하는데 현재 널리 이용되고 있다. 하지만 대부분의 이동 에이전트 시스템들은 매우 다른 구조로 구현되어 있어 이종의 시스템에서 생성된 에이전트의 실행이 지원되지 않고 있다. 또한 이러한 시스템들간의 차이는 에이전트 기술의 빠른 확산과 상호운용성을 가로막고, 에이전트 시스템의 적용을 어렵게 하고 있다.

본 논문에서는 서로 다른 에이전트 시스템간의 상호운용성을 위하여 OMG의 MAF 명세를 만족하는 SMART 시스템을 개발하였다. MAF 명세는 MAFAgentSystem과 MAFFinder 두 개의 인터페이스로 이루어져 있으며, MAFAgentSystem 인터페이스는 에이전트를 생성하고, 전송하고, 실행시키는 오퍼레이션을 정의한다. MAFFinder 인터페이스는 에이전트와 플레이스, 그리고 에이전트 시스템을 등록시키고, 등록된 것을 해제시키는 오퍼레이션 그리고 이들을 검색하는 오퍼레이션을 정의하였다. 이 두 개의 공통된 인터페이스를 이용하여 이기종간의 상호운용성을 지원할 수 있다. 그 결과 MAF 명세를 만족하는 이질적인 에이전트 시스템간의 통신이 가능하며, 다른 에이전트 시스템에서 생성된 에이전트도 수행할 수 있게 되었다.

SMART 시스템은 기능 별로 플레이스, 자원관리자, 보안관리자 그리고, 모니터의 네 개의 모듈로 이루어져 있다. 플레이스는 쓰레드 그룹을 이용하여 에이전트의 수행을 효율적으로 관리할 수 있으며, 에이전트 전송 규칙은 에이전트 생성 시 필요한 클래스의 이름 목록을 전송하고 에이전트 실행 도중에 필요한 클래스들은 "On-demand" 방법으로 수행하는 기법을 사용하여 더 적은 네트웍 트래픽을 제공한다. 보안관리자는 OMG MAF에서 요구하는 4가지 정책을

Protekt 3.0 SSL 패키지와 SMART 시스템의 로그파일을 기반으로 하여 간결하게 구현하였다. 그리고, 에이전트가 에이전트 시스템에게 자원을 요청 시 자원 관리자는 에이전트의 권한을 검사하여 자원을 할당하거나 거부한다. 모니터는 이동 에이전트와 에이전트를 실행시키는 플레이스가 정상적인 수행을 하고 있는지를 감시한다.

SMART 시스템의 견고성을 높이기 위하여 에이전트의 예외처리와 영속성(persistence) 기능을 제시하였다. 에이전트 시스템에서 일어날 수 있는 예외상황은 실행 중인 에이전트가 자신의 자원접근등급에 어긋난 자원을 액세스하고자할 때 발생할 수 있는 예외, 이동하려는 목적지 에이전트 시스템을 찾지 못할 때 발생하는 예외, 그리고 에이전트 시스템의 비정상 종료 시 발생하는 예외와 에이전트 시스템의 영속성을 JavaSpace를 이용하여 효율적으로 처리할 수 있는 기법을 제안하고 구현하였다.

SMART 시스템은 현재 에이전트 또는 에이전트 시스템들 간의 일대일 통신만을 지원하고 있다. 그리고 에이전트의 영속성을 위하여 JavaSpace를 이용하고 있다. 앞으로 이러한 기법을 더 발전시켜 서버의 실패나 네트웍의 분할(partition)이 발생하여도 안정된 서비스를 제공할 수 있는 통신 채널을 이용하여 에이전트간의 통신, 에이전트 시스템간의 통신을 지원할 예정이다[19].

참 고 문 헌

1) Mitsuru O, Guenter K., and Kouichi O., *"Aglets Specification 1.1 Draft"*, http://www.trl.ibm.com/aglets/spec11.html, 1998.9.

2) IKV++ GmbH, *"Grasshopper Technical Overview"*, 1999.2.

3) OMG Inc., *"Mobile Agent Facility Specification. V1.0"*, 2000.1

4) Sun Microsystems Inc., *"JavaSpace Service Specification V1.1"*, 2000.10.

5) Forge Information Technology, *"Protekt Encryption 3.0 Programming Guide"*, 1999.10.

6) Sun Microsystems Inc., *"Jini Technology Core Platform Specification"*, 2000.10.

7) Danny Ayers, Hans Bergsten, *"Professional Java Server Programming"*, Wrox Press Ltd, 1999

8) Alfonso Fuggetta, Gian Pietro Picco, Giovanni Vigna, "Understanding Code Mobility," IEEE Transaction On S/W Engineering, Vol.24, NO.5, May 1998.

9) Antonella Di Stefano, Lucia Lo Bello, Corrado Santoro, "Naming and Locating Mobile Agents in an Internet Environment," IEEE 0-7803-5784-1/99, 1999

10) T. Finin, Y Labrou, Y. Peng, "Mobile Agents Can Benefit from Standards Efforts on Interagent Communication," IEEE Communications Magazine, July 1998.

11) Neeran M. Karnik, "Security in Mobile Agent Systems," PhDS thesis, University of Minnesota, October 1998

12) Neeran M. Karnik, Anand R Tripathi, "Design Issues in Mobile Agent Programming Systems," University of Minnesota Minneapolis, June 1998

13) Krishna Sankar, "Java 1.2 Class Libraries Unleashed Vol I, II," Sams, 1999

14) William Li, David G Messerschmitt, Java-To-Go Mobile Agent System, University of California at Berkeley, 1998

15) Edwards, W. Keith, "Core JINI," The Sun Microsystems press Java Series, 1999

16) Gunter Karjoth, Danny B. Lange, Mitsuru Oshima, "Security Model For Aglets,"

IEEE Internet Computing, 1997.7.
17) Anand R. Tripathi, Neeran M. Karnik, Manish K. Vora, Tanvir Ahmed, and Ram D Singh, "Ajanta - A Mobile Agent Programming System," 1998.
18) C. Baumer, M. Breugst, S. Choy, T. Magedanz, "Release 1.2 Basics and Concepts," Grasshopper, February 1999.
19) 안건태, 문남두, 정현락, 유양우, 이명준, "JACE 그룹통신시스템을 이용한 신뢰성 있는 공유객체공간의 개발", 한국정보과학회 99 가을 학술발표논문집(III), pp.218-220, 1999.
20) 유양우, 김진홍, 이명재, 박양수, 이명준, "SMART 이동 에이전트 시스템의 안전한 자원접근 정책", 한국정보과학회 2000 봄 학술발표논문집(A), 2000, 4
21) 구형서, 김진홍, 유양우, 이명재, 이명준, "SMART 에이전트 시스템의 영속성 및 예외처리 지원", 한국정보과학회 2001 봄 학술발표논문집(A), 2001. 4
22) 유양우, 김진홍, 구형서, 박양수, 이명재, 이명준, "SMART: OMG의 MAF 명세를 지원하는 CORBA 기반의 이동 에이전트 시스템", 한국정보처리학회 논문집 제8-C권 제2호, 2001. 5.