SecureJMoblet : Jini2.0 기반의 안전한 이동에이전트 시스템

유 양 우[†] · 문 남 두[††] · 이 명 준[†††]

요 약

이동에이전트는 네트워크 상의 여러 노드들을 자발적으로 이동하는 동적인 개체이다. 자바의 Jini 프레임워크는 분산 네트워크 프로그래밍을 위한 주요한 기능을 제공함으로써, 이동에이전트 시스템의 개발을 용이하게 한다. 하지만, Jini1.0 서비스는 안전한 원격통신을 위한 보안성이 취약하여 이를 이용한 이동에이전트 시스템의 개발은 근본적인 제약점을 가지고 있다. 본 논문에서는 *Jini2.0* 기반의 안전한 이동에이전트 시스템인 *SecureJMoblet*에 대하여 기술한다. SecureJMoblet은 Jini2.0 위에서 이동에이전트 시스템의 기본 기능인 에이전트 생성, 전송, 제어 기능을 제공한다. 또한, 안전한 JavaSpace 서비스를 제공하기 위하여 개발된 *SecureJS*를 이용하여 에이전트 객체를 안전하게 저장하기 위한 객체 저장소와 이동에이전트 간의 안전한 통신 기능을 지원한다.

SecureJMoblet : Secure Mobile Agent System based on Jini2.0

Yang-Woo Yu[†] · Nam-Doo Moon[††] · Myung-Joon Lee[†††]

ABSTRACT

Mobile agents are autonomous and dynamic entities that can migrate among various nodes in the network. Java's Jini framework facilitates mobile agent system development, providing key features for distributed network programming. However, due to the security weakness, Jini1.0 service has a fundamental limitation on developing mobile agent systems which support secure remote communications. In this paper, we describe a *Jini2.0*-based secure mobile agent system named *SecureJMoblet*. On the top of Jini2.0, the system provides basic functionalities of a mobile agent system such as creation, transfer and control. In addition, with the *SecureJS* developed for secure JavaSpace service, SecureJMoblet supports a secure object repository and a reliable communication among mobile agents.

키워드 : 이동에이전트(Mobile Agent), 이동에이전트 시스템(Mobile Agent System), Jini2.0, SecureJMoblet, SecureJS

1. 서 론

인터넷 기술은 빠르게 발전하고 있으며 이를 활용할 수 있는 분야도 더욱더 다양해졌다. 인터넷을 통하여 정보제공 및 전자상거래 등의 편리한 서비스가 운영되고 있으며, 또한 장소나 시간에 구애받지 않고 생활 속에서 자연스럽고 편리하게 컴퓨터를 사용할 수 있는 유비쿼터스(Ubiquitous) 컴퓨팅[1]에 대한 관련 연구가 현재 활발히 진행되고 있다. 이에 따라 빠르게 변화하는 사용자의 요구를 만족하면서도 편리하고 유용한 서비스를 제공하는 이동에이전트 기술은 유비쿼터스 컴퓨팅환경에서 응용서비스를 제공하는 주요 기술로 사용될 수 있다. 이러한 이동에이전트 시스템을 개발하는데 있어서 개발자에게 분산네트워크 프로그래밍을 위한 주요기능을 제공하는 Jini 기술[2, 3]이 주목받고 있다.

*이동에이전트*는[4, 5] 사용자가 지정한 이동 경로를 따라 다른 여러 이동에이전트 시스템으로 자발적으로 이동하여 자신의 작업을 수행하고 그 결과를 사용자에게 보고하는 프로그램이다. 이러한 기술은 네트워크 트래픽을 줄이고, 서버의 기본 기능을 변경하지 않으면서 사용자의 다양한 요구사항을 지원할 수 있는 장점을 갖는다. 이동에이전트 기술은 유비쿼터스 컴퓨팅 환경에서 필요한 정보를 얻고 교환하는 서비스를 제공하는 응용프로그램 개발에 적용될 수 있으며, 고급화된 전자상거래 서비스, 네트워크 관리 등의 다양한 분야에 적용되고 있다.

Jini는 다양한 이 기종(heterogeneous) 장치에서 유비쿼터스 컴퓨팅을 지원하기 위한 필수적인 기반구조와 미들웨어를 개발하는 플랫폼으로 이용될 수 있다. Jini 기술은 네트워크 상의 지능형 기기 또는 소프트웨어들이 Jini 네트워크에 접속과 동시에 서비스할 수 있는 기능을 제공하고, 사용자가 서비스를 요청하면 요청한 서비스를 찾아 처리를 할 수 있는 새로운 분산 기반 구조를 제공한다. Jini의 룩업서비스와 디스커버리 프로토콜[2]은 Jini 서비스로 동작하는 이동에이전트

※ 이 논문은 2004년 울산대학교의 연구비에 의하여 연구되었음.
† 준 회 원 : 울산과학대학 컴퓨터정보학부 교수
†† 준 회 원 : 울산대학교 대학원 컴퓨터정보통신공학부
††† 정 회 원 : 울산대학교 컴퓨터정보통신공학부 교수(교신저자)
논문접수 : 2004년 7월 27일, 심사완료 : 2004년 9월 30일

시스템의 동적인 등록 및 검색, 에이전트의 이동, 에이전트가 이용할 서비스의 동적인 등록 및 검색 등을 기본적으로 제공할 수 있다. 또한 Jini의 JavaSpace 서비스[2, 6]는 표준 인터페이스를 통하여 객체를 저장하고, 저장된 객체를 그 클래스의 템플릿을 이용하여 읽거나 가져오는 새로운 패러다임을 제공하고 있다. 하지만, 기존의 JavaSpace는 보안기능이 취약하여 누구나 객체를 저장하고 저장된 객체를 아무런 제약없이 접근할 수 있다. 그러므로 보안이 요구되는 정보를 교환하거나 보관할 경우 JavaSpace는 안전한 객체 저장소로서의 서비스를 제공할 수 없는 단점을 가지고 있다.

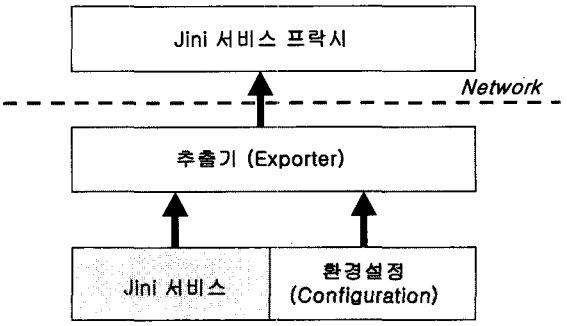
최근에 썬 마이크로시스템즈(Sun Microsystems)에서는 분산시스템 환경에서 자바코드의 이동성에 따른 보안요소를 충족시키는 Jini2.0을 제시하였다. 본 논문에서는 새로운 Jini2.0 기반에서 동작하는 이동에이전트 시스템인 SecureJMoblet에 대하여 기술한다. SecureJMoblet 시스템은 Jini2.0 보안모델을 적용하여 이동에이전트의 상호인증을 지원하며, JavaSpace 서비스에 대하여 접근제어 기능을 제공하는 SecureJS를 구현하였다. 그 결과, SecureJMoblet 이동에이전트 시스템은 안전한 JavaSpace 서비스를 지원하는 SecureJS를 이용하여 이동에이전트 상호간의 안전한 통신을 지원하며, 이동에이전트 객체의 안전한 저장소를 지원한다.

본 논문의 구성은 다음과 같다. 2장에서는 새롭게 추가된 Jini2.0 서비스와 JavaSpace 서비스에 대하여 소개하고, 다른 이동에이전트 시스템들의 특징을 간단히 설명한다. 3장에서는 안전한 JavaSpace 서비스를 제공하기 위하여 Jini2.0 기반의 SecureJS의 설계 및 구현에 관하여 자세히 설명한다. 4장에서는 SecureJS 서비스를 이용한 SecureJMoblet 이동에이전트 시스템의 구성과 주요기능을 소개한다. 그리고, 5장에서는 SecureJMoblet 시스템을 이용한 활용예제를 설명하며 시스템의 동작방식과 운영에 관하여 살펴본다. 끝으로 6장에서 결론 및 추후 연구방향에 대해 살펴본다.

2. 관련 연구

2.1 Jini2.0의 소개

Jini2.0 시스템 구조는 프로그래밍 모델, 하부구조 그리고 서비스들로 구성되어 있다. 이전 버전의 Jini 서비스는 룩업 서비스, 트랜잭션 관리, 리스, 이벤트처리, JavaSpace 서비스 등을 제공하였으며, Jini2.0은 이러한 서비스의 지원과 더불어 추가적으로 새로운 *프로그래밍 모델*과 *하부구조*를 추가시켰다[3]. 새롭게 추가된 프로그래밍 모델은 환경설정(Configuration), 추출기(Exporter), 프락시 준비기(ProxyPreparer)를 지원하며, 하부구조는 보안과 호출제약(Invocation Constraints) 그리고 JERI 등이 추가되었다[7, 8]. Jini2.0의 확장된 기능의 지원으로 기존의 보안성이 취약한 Jini 기반의 응용프로그램들에 대하여 강력한 보안요소와 보안의 요구사항에 맞는 다양한 환경설정을 제공하고, 클라이언트 측과 서버 측의 프로그래밍 모델을 단일화할 수 있게 되었다. 또한 JERI를 지원함으로써 다양한 전송계층을 사용할 수 있다는 큰 장점을 갖고 있다. (그림 1)은 Jini2.0의 새로운 프로그래밍 모델을 간략히 표현하고 있다.



(그림 1) Jini 프로그래밍 모델

Jini 기반의 응용 프로그램에서 클라이언트는 서비스를 받기 위해 룩업서비스를 이용하여 서비스프락시를 다운로드 받아야 한다. 그리고, 이 서비스프락시 내의 메소드를 호출함으로써 원하는 서비스를 이용할 수 있다. 대다수 응용프로그램들은 안전한 원격 통신을 위하여 다음과 같은 보안사항[3]을 요구한다.

- 클라이언트와 서버 사이의 상호인증
- 액세스 제어를 위한 권한 제한
- 무결성
- 암호화를 통한 기밀성

이러한 요구사항들은 안전한 데이터통신을 위하여 필요한 전형적인 보안 요구사항이다. 기존의 Jini1.0에서는 이러한 보안 요구사항을 지원하지 않았으며 보안은 개발자의 영역으로 남겨두었다. 하지만, Jini2.0 코델이 새롭게 출시되면서 위에서 제시한 보안요구사항을 모두 지원하며 추가적으로 코드무결성(code integrity)을 지원하고 있다.

Jini와 자바 RMI 모델에서, 서비스객체들의 데이터는 원격호출로 전송되지만, 그에 대한 코드 즉 서비스프락시는 원격호출과 달리 다운로드 해야한다. 그래서, 응용프로그램은 받은 객체 즉, 객체코드와 데이터들에 대하여 반드시 "trust"임을 검증해야한다. Jini2.0 보안모델에서는 응용프로그램이 원격호출 시, 다양한 요구사항을 명시할 수 있는 제약(constraints) 기반의 원격 호출 모델을 지원한다. 이러한 모델은 코드무결성을 보장하는데 사용되는 메커니즘으로 제공된다.

2.2 JavaSpace의 소개

JavaSpaces™ 기술은 자바 환경의 새로운 분산 컴퓨팅 모델로서 자바객체를 저장하고 접근할 수 있는 공간을 말한다. JavaSpace는 1980년대 Yale 대학에서 개발한 분산 애플리케이션 개발 도구인 Linda에서[9] 영향을 받았으며, Linda는 네

트워크 시스템으로 연결된 환경에서 병렬 분산 처리를 용이하게 구현할 수 있음을 보여주었다.

JavaSpace는 자바의 원격 객체 호출 시스템인 RMI(Remote Method Invocation)와 직렬화를 이용하며, 룩업(lookup) 서비스, 트랜잭션(transaction) 서비스 등 Jini서비스와 연계하여 분산처리를 위한 기능들을 제공함으로써 분산 시스템을 쉽게 구축할 수 있도록 한다.

기존의 분산 애플리케이션이 프로세스 간 통신 시 직접 메시지를 전달하거나 원격 객체의 메소드를 호출하는 방식으로 이루어지는 반면 JavaSpace를 이용한 통신 모델에서는 프로세스들이 직접 통신하지 않고 대신 JavaSpace라는 데이터 저장소를 통해 객체를 교환하게 된다.

JavaSpace를 이용한 프로그래밍 모델은 매우 간결하다. JavaSpace를 이용하고자 하는 응용프로그램은 Jini의 룩업(Lookup) 서비스를 이용하여 JavaSpace에 접근할 수 있는 서비스프락시를 다운로드 한다. 이후 서비스프락시를 이용하여 객체를 저장하고, 원하는 객체를 검색하여 그 객체를 읽거나 가져갈 수 있다. 이러한 패러다임은 객체를 공유하는 차원에서는 매우 유용하게 사용될 수 있다. 그러나 JavaSpace의 프락시를 룩업 서비스로부터 구하기만 하면 어떠한 목적의 응용프로그램도 JavaSpace 내의 공간에 객체를 저장하거나 가져갈 수 있기 때문에 이러한 객체정보들에 대한 접근 보안이 요구되는 분산 응용프로그램 개발에는 적합하지 않다.

따라서, 본 연구에서는 이동에이전트 상호간의 안전한 통신을 지원하기 위해 기존의 JavaSpace의 기능에 Jini2.0 보안요소를[3] 추가하여 안전한 JavaSpace 서비스를 제공하는 SecureJS 시스템을 개발하였다. 안전한 JavaSpace 서비스를 제공하기 위해 다음의 두 가지 사항을 고려한다. 첫째는 신원이 확인되고 허용된 클라이언트만이 룩업서비스로부터 JavaSpace의 서비스프락시를 구할 수 있도록 제한한다. 둘째는 JavaSpace에 객체를 저장할 때 객체에 접근할 수 있는 수신자를 지정하고, 지정된 수신자만이 객체에 접근할 수 있도록 제한한다. 이와 같이 개발된 안전한 JavaSpace 시스템은 AccessManager와 ObjectStore로 구성되어 있으며, 시스템에 관한 자세한 설명은 4장에서 논의할 것이다.

2.3 타 이동에이전트 시스템의 소개

분산 컴퓨팅 기술의 발전과 함께 현재까지 많은 이동에이전트 시스템이 개발되었다. 본 장에서는 SecureJMoblet 시스템과 관련이 있는 Aglets[10], Concordia[11], Ajanta[12], 그리고 SMART[5, 13]와 같은 이동에이전트 시스템에 대하여 간략하게 살펴본다.

IBM의 Aglets은 IBM 도쿄 연구실에서 개발되었으며, "weak" 이동성[14]을 지원하는 Java 기반의 이동에이전트 시스템이다. Aglets은 call-back 기반의 프로그래밍 모델을 이용하여 에이전트에게 이벤트가 발생할 때마다 에이전트의 특정 메소드를 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.

Concordia는 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, 악의적인 사용자로부터 에이전트와 시스템을 제한하기 위하여 암호화 프로토콜을 사용한다.

Ajanta는 미국의 미네소타대학에서 에이전트의 이동성을 지원하기 위해 Java 직렬화 메커니즘을 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.

SMART는 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 1>은 타 이동에이전트 시스템 및 SecureJMoblet 시스템에서 제공하는 기능에 대하여 기술하고 있다. SecureJMoblet 시스템은 에이전트의 자원접근 레벨에 따른 자원 클래스의 객체를 할당하여 자원을 접근하는 방법과 이동에이전트의 상태를 변경할 수 있는 제어기능을 제공하고 있다. 또한, SecureJMoblet 시스템은 Jini 서비스로서 동작하는 이동에이전트 시스템의 이동 예외처리, 에이전트간 통신 기능을 안전한 JavaSpace 서비스를 지원하는 SecureJS를 기반으로 제공할 수 있는 방법을 보여주고 있다.

〈표 1〉 이동에이전트 시스템의 기능 비교

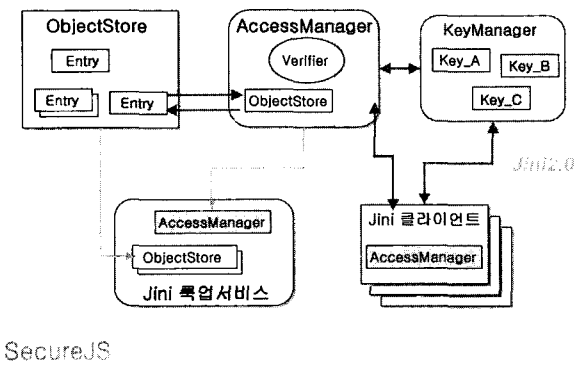
기능 / 시스템	코드 이동성	시스템 자원 정책	이동예외 처리기능	시스템 보안성	에이전트간 안전한 통신	에이전트 제어
Aglet	weak	Yes	No	Yes	No	Yes
Concordia	weak	Yes	No	No	No	No
Ajanta	weak	Yes	Yes	No	No	Yes
SMART	weak	Yes	Yes	No	No	Yes
Secure JMoblet	***weak***	***Yes***	***Yes***	***Yes***	***Yes***	***Yes***

3. 안전한 JavaSpace 서비스

JavaSpace는 분산 객체의 공유개념을 도입하여 누구든지 자바객체를 저장하고, 저장된 객체에 접근하여 읽거나 가져갈 수 있는 Jini 서비스이다. 기존의 JavaSpace는 보안성을 제공하지 않으므로 본 연구에서는 Jini2.0 보안모델과 프로그래밍 모델을 기반으로 보안성이 강화된 안전한 JavaSpace 서비스를 제공하는 *SecureJS*를 구현하였다[15]. 또한, SecureJMoblet 이동에이전트 시스템은 안전한 SecureJS를 이용하여 에이전트를 안전하게 저장하고 에이전트 간 통신기능을 제공할 수 있게 되었다.

3.1 안전한 JavaSpace의 보안정책

SecureJS는 *AccessManager*(접근관리자)와 *ObjectStore*(객체저장소) 그리고 *KeyManager*(키 관리자)로 구성되어 있다. ObjectStore는 JavaSpace 서비스를 제공하는 객체저장소이며, AccessManager는 인증된 클라이언트들에게 안전한 JavaSpace 서비스를 제공하는 데몬 형태로 동작하는 Jini2.0 서비스이다. 그리고, KeyManager는 AccessManager에서 올바른 수신자인지 여부를 검사할 때 사용하는 공개키들을 관리한다. 이러한 구성요소를 갖는 SecureJS는 분산 컴퓨팅 환경에서 객체를 안전하게 공유하고 저장할 수 있는 매우 유용한 기술을 제공하고 있다. (그림 2)는 SecureJS의 구조를 보여주고 있다.



(그림 2) SecureJS의 전체 구조

3.1.1 ObjectStore(객체저장소)

JavaSpace는 엔트리(net.jini.core.entry.Entry) 인터페이스를 구현한 자바객체를 저장하는 Jini 서비스이다. 엔트리는 java.io.Serializable을 확장한 인터페이스이며, 엔트리 객체의 모든 속성은 직렬화할 수 있는 객체 참조이어야 한다. 사용자는 JavaSpace에 엔트리를 구현한 객체를 저장할 수 있으며, 엔트리 객체의 템플릿을 이용하여 매칭되는 엔트리 객체를 구할 수 있다.

ObjectStore는 하나의 JavaSpace이며 Jini2.0 보안정책을 이용하여 JavaSpace에 접근할 수 있는 모든 권한을 Access Manager에게만 부여한다. 이러한 작업을 위해 ObjectStore는 Jini2.0 환경설정 파일을 작성해야 하며 (그림 3)에서 ①, ② 코드는 JavaSpace의 접근권한을 AccessManager에게만 부여하는 환경설정의 일부분이다.

```
Server.ObjectStore {
    private static AccessManagerKey = KeyStores.getX500Principal
                                ("AccessManager", caTruststore); ①
                    :
    AuthenticationPermission(AccessManagerKey, ObjectStoreKey, "connect"); ②
}
```

(그림 3) ObjectStore의 환경설정

ObjectStore에 저장되는 자바객체는 엔트리와 수신자 id 정보를 포함한다. 수신자 id는 저장된 엔트리에 대하여 올바른 수신자에게 정확히 요청한 객체를 검색하여 전달하기 위해 사용된다.

3.1.2 AccessManager(접근관리자)

일반적으로 클라이언트들은 JavaSpace 서비스를 이용하기 위하여 Jini 룩업서비스로부터 서비스프락시를 다운로드 받는다. 기존의 Jini 모델에서는 인증과정 없이 서비스프락시를 요구하는 모든 클라이언트들에게 이러한 프락시를 제공함으로써 서비스에 대한 보안성이 매우 부족하였다.

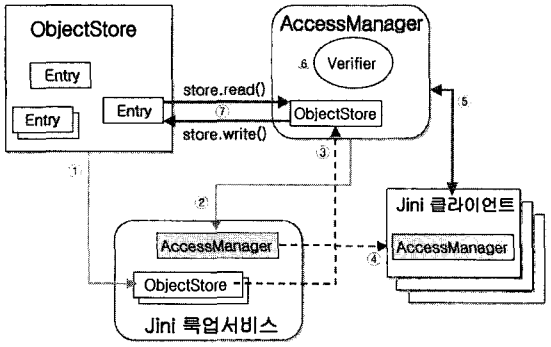
SecureJS는 JavaSpace 서비스에 대한 두 가지 보안사항을 적용하였다.

- JavaSpace에 접근할 수 있는 프로세스를 AccessManager로 한정시켜 클라이언트들 이 JavaSpace에 접근할 경우 AccessManager를 통해서만 접근할 수 있다.
- JavaSpace에 저장된 엔트리에 대하여 클라이언트들이 접근하고자 할 때, AccessManager는 적법한 수신자임을 검증하여 그 클라이언트에게 서비스를 제공한다.

SecureJS 시스템은 자바객체를 저장할 수 있는 공간으로 ObjectStore를 사용하고 있으며, ObjectStore에 대한 접근은 AccessManager만이 권한을 부여받아 서비스를 제공할 수 있다. 이는 (그림 3)의 환경설정 파일에서 지정한다. ObjectStore에 접근할 수 있는 인증된 주체를 "AccessManager"로만 한

정시켰다. 따라서, 클라이언트들은 안전한 JavaSpace의 서비스 이용하고자할 때, AccessManager를 통해서만 서비스를 제공받을 수 있다. 즉, 클라이언트들은 ObjectStore 서비스에 직접적으로 접근하여 사용 할 수 없다. 이러한 설계는 클라이언트들에게 안전한 JavaSpace 서비스를 제공하기 위한 새로운 보안모델로서 제시된다.

두 번째 보안사항은 JavaSpace에 저장된 엔트리에 대한 보안으로 인증된 클라이언트라 할지라도 그 엔트리를 접근할 수 있는 수신자가 아니라면 접근할 수 없도록 한것이다. 이는 **<올바른 수신자의 검증 알고리즘>**에서 자세히 설명할 것이다. (그림 4)는 AccessManager의 동작을 설명하고 있다.



(그림 4) AccessManager의 동작

① 자바객체를 저장하는 ObjectStore는 Jini 서비스로 동작하기 위해 룩업서비스에 자신의 서비스프락시를 등록시킨다. 그리고 자신을 접근할 수 있는 유일한 접근자로써 AccessManager를 지정한다.

② AccessManager 또한 클라이언트들이 자신의 서비스를 제공받을 수 있도록 룩업서비스에 서비스프락시를 등록시킨다.

③ AccessManager는 Jini2.0 보안모델을 이용하여 ObjectStore의 프락시를 다운로드한다. AccessManager를 제외한 다른 클라이언트들은 ObjectStore의 프락시를 다운로드 받을 수 없다.

④ 안전한 JavaSpace 서비스를 이용하기 위하여, 먼저 클라이언트들은 인증과정을 수행한 후, 룩업서비스를 통하여 AccessManager의 서비스프락시를 다운로드 받는다.

⑤ 클라이언트들은 다운로드 받은 AccessManager의 프락시를 이용하여 원격메소드를 호출한다.

⑥ AccessManager의 검증자(verifier)는 허가된 접근인지 여부를 검사한 후 적법한 클라이언트임이 검증되면 ObjectStore의 서비스프락시 내에 메소드를 호출하여 안전한 JavaSpace 서비스를 제공한다.

⑦ ObjectStore는 AccessManager에서 요구한 JavaSpace 서비스를 수행한다. 그리고 그 결과 값을 클라이언트에게 반환한다.

SecureJS는 기존의 JavaSpace에서 제공하는 오퍼레이션을 모두 지원하고 있으며, 클라이언트들은 정의된 오퍼레이션을 수행함으로써 안전한 JavaSpace 서비스를 제공받을 수 있다. SecureJS의 주요 오퍼레이션들은 AccessManager_Impl 클래스 내에 정의되어 있으며, 이러한 오퍼레이션을 이용하여 엔트리를 저장하고 검증된 수신자가 저장된 엔트리를 읽을 수 있도록 한다.

〈올바른 수신자의 검증 알고리즘〉

① 인증된 클라이언트는 엔트리**(entry)**를 SecureJS 내에 write() 메소드를 이용하여 저장할 수 있다. 일반적인 JavaSpace의 write() 메소드와는 달리 SecureJS 에서는 그 엔트리에 접근할 수 있는 수신자의 id**(receiver_id)**를 write() 메소드에 매개변수로 지정한다. 이는 올바른 수신자를 검증하기 위해 사용된다.

② 엔트리에 수신자를 지정하여 SecureJS의 **write(receiver_id, entry, txn, lease)** 메소드를 호출하여 엔트리를 저장한다.

③ 클라이언트가 read() 메소드를 이용하여 ObjectStore 내에 저장된 엔트리에 접근을 시도할 때, AccessManager는 저장된 객체를 서비스하기 전에 그 클라이언트가 해당 객체에 접근할 수 있는 올바른 수신자인지 검증해야 한다.

④ 먼저, 수신자는 자신이 올바른 수신자임을 증명하기 위하여 자신의 id를 개인키로 암호화시킨 값과 자신의 id를 read() 메소드의 매개변수로 전달한다.
예) **read(private_encrypt_id, id, id_tmpl, txn, lease)**

⑤ AccessManager는 수신자의 개인키로 암호화시킨 값과 수신자 id 그리고 KeyManager를 이용한 수신자의 공개키 정보를 이용하여 올바른 수신자임을 검증할 수 있다.

⑥ KeyManager에서 구한 수신자의 공개키를 사용하여 수신자의 개인키로 암호화된 값을 해독하여 id 값을 구한다. 이 값과 수신자가 보낸 id가 일치하면, 이는 올바른 수신자임이 검증된 것이다.

AccessManager는 AccessManager_Impl 클래스를 이용하여 구현된다. 다음 코드는 AccessManager_Impl 클래스의 write()와 read() 메소드를 이용하여 ObjectStore에 엔트리를 저장하고, 저장된 엔트리에 접근하기 위하여 올바른 수신자임을 검증하는 과정을 보여주고 있다.

클라이언트는 *write*() 메소드를 호출하여 ObjectStore에 엔트리를 저장한다. **receiver_id**인자는 엔트리에 접근할 수 있는 수신자의 id이다. 그리고 **entry** 인자는 ObjectStore 내에 저장하고자 하는 자바객체이다. (그림 5)는 AccessManager의 write() 메소드를 보여준다.

```
/* 객체를 저장하기 위하여 ObjectStore의 write() 메소드를 호출한다. */
  public Lease write(receiver_id, entry, txn, lease) {
     Id_Entry ie = new Id_Entry(receiver_id, entry);
     object_store.write(Id_Entry, txn, lease);
  }
```

(그림 5) AccessManager의 write() 메소드

ObjectStore에 저장된 엔트리에 접근하기 위하여 *read*() 또는 *take*() 메소드를 이용할 수 있다. AccessManager는 클라이언트가 이들 오퍼레이션을 호출할 때, 호출한 클라이언트가 올바른 수신자인지를 검증하고, 검증된 수신자라면 해당 오퍼레이션에 따른 ObjectStore 서비스를 제공한다. (그림 6)은 read() 메소드를 설명하고 있다.

```
/* 저장된 객체를 읽기 위하여 ObjectStore의 read() 메소드를 호출한다. */
  public Entry read(private_encrypt_id, id, tmpl, txn, lease) throws NotTrusted {
    boolean trust;

    ldap_pub_key = ldapCtx.search(id, PK_FILTER, constraints); ①
    trust = verifier(ldap_pub_key, private_encrypt_id, id);    ②

    if (trust) {
      Id_Entry id_tmpl = new Id_Entry(id, tmpl);               ③
      Id_Entry ie = object_store.read(id_tmpl, txn, lease);    ④

      return ie.get_entry();                                   ⑤
    }
          :
```

(그림 6) AccessManager의 read() 메소드

read() 오퍼레이션에서 첫 번째 인자 **private_encrypt_id**는 올바른 수신자임을 검증하기 위해 수신자 자신의 id를 수신자 개인키(private key)로 암호화시킨 값이다. 그리고, 두 번째 인자는 엔트리에 접근을 허용할 수신자의 **id**이다. **tmpl**인자는 수신자가 접근하고자하는 엔트리를 검색하기 위한 템플릿이다. read() 오퍼레이션의 동작은 다음과 같다.

① LDAP를[16] 이용하는 KeyManager에 접근하여 수신자의 id에 해당하는 수신자의 공개키를 얻어온다.

② *verifier*(*ldap_pub_key*, *private_encrypt_id*, *id*) 메소드를 이용하여 수신자가 올바른 수신자인지를 검사한다. *ldap_pub_key*인자는 KeyManager의 search() 오퍼레이션을 통하여 얻은 수신자의 공개키이며, *private_encrypt_id*는 수신자 id를 수신자 자신의 개인키로 암호화시킨 값이다. 그리고 id는 수신자 자신의 id이다.
: AccessManager는 수신자의 공개키를 가지고 수신자의 개인키로 암호화시킨 **private_encrypt_id**를 해독한다. 해독한 값이 read() 메소드의 두 번째 인자 id값과 일치하면 올바른 수신자임이 검증된 것이다.

③ **id_tmpl** 인자는 ObjectStore에 저장된 엔트리를 검색하기 위한 템플릿이다.

④ 템플릿에 정확히 매칭되는 엔트리를 ObjectStore의 read() 메소드를 호출하여 가져온다.

⑤ 검색된 엔트리에서 실제 수신자가 요구하는 엔트리만을 수신자에게 전달한다.

3.2 KeyManager(키 관리자)를 이용한 공개키 관리

SecureJS에서 AccessManager는 사용자의 신원을 인증하기 위한 방법으로 JDK에서 지원하는 DSA 보안 알고리즘[8]을 사용하며, 각 사용자들에 대한 공개키와 개인키를 구할 수 있다. 그리고 DSA 알고리즘을 이용하여 얻은 공개키는 LDAP를[16] 이용하는 KeyManager에서 관리된다. AccessManager와 클라이언트는 상호인증을 위해 KeyManager에서 관리되는 공개키를 사용한다.

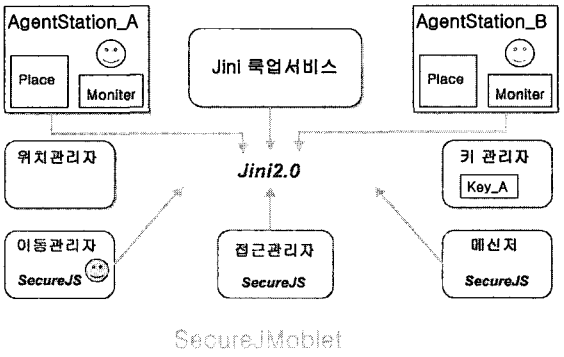
SecureJS에서 KeyManager는 공개키 관리를 위하여 LDAP 디렉토리를 사용한다. 디렉토리는 검색할 정보의 커다란 집합으로 볼 수 있다. 전화번호부와 같이 정보에 거의 변화가 없고, 매우 자주 검색되는 정보는 디렉토리의 영역이라고 볼 수 있다. LDAP는 국제 표준기구(ISO)와 국제 원거리 통신 연맹(ITU)에서 승인한 디렉토리 표준인 X.500 기술의 문제점을 극복하기 위해 고안되었다. LDAP는 인터넷의 TCP/IP 프로토콜을 사용할 수 있도록 설계되었으며, X.500과 비교하면 자원소모가 적고 기능이 단순화되었다. 네트워크 디렉토리를 특수한 데이터베이스로 생각한다면 LDAP를 이용하여 관련 서버와 서버 주변기기, 어플리케이션이나 문서의 위치를 파악하는데도 많은 도움을 주므로 그 기능을 크게 확장시켜 나갈 수 있다. KeyManager의 주요 기능은 다음과 같다.

- *바인딩(Binding)* : LDAP는 바인딩과 인증을 구별하지 않으며, 디렉토리 서버로 바인딩할 때 원하는 서버와 자신에 대한 정보(계정, 암호)를 함께 지정할 수 있다.
- *검색(Searching)* : 검색을 하려면 검색의 범위를 지정하여 *search*() 메소드를 호출해야 한다.
- *엔트리 추가(Adding Entries)*, *엔트리 변경(Modifying Entries)*, 엔트리 제거(Deleting Entries) : 디렉토리 구조에서 관리되는 각 엔트리를 추가, 변경, 삭제할 수 있다.

4. SecureJMoblet 이동에이전트 시스템

SecureJMoblet 시스템은 Jini2.0 프로그래밍 모델과 보안모델을 따르고 있으며, 하부 통신프로토콜은 JERI에서 지원하는 SSL 프로토콜을 사용한다. SecureJMoblet의 구성요소들은 Jini2.0 서비스로 동작하며, 상호 인증과정을 수행한 후 다른 Jini 서비스와 사용자들에 의해 사용될 수 있다. SecureJMoblet 시스템의 구성은 에이전트스테이션(AgentStation), 위치관리자(LocationManager), 이동관리자(MovingHelper), 접근관리자(AccessManager), 그리고 메신저(Messenger)로 구

성된다[17]. (그림 7)은 SecureJMoblet 시스템의 각 구성요소를 포함한 전체구조를 보여주고 있다.



(그림 7) SecureJMoblet 시스템의 구조

*에이전트스테이션*은 이동에이전트의 실행 환경인 에이전트플레이스(AgentPlace), 이동에이전트의 상태 추적 및 변경 기능을 제공하는 에이전트모니터(AgentMonitor), 그리고 에이전트스테이션의 자원을 관리하는 자원관리자(ResourceManager)로 구성되어 있다.

*위치관리자*는 에이전트스테이션, 에이전트플레이스, 그리고 이동에이전트의 위치정보를 관리하는 Jini 서비스이다. 위치관리자는 SecureJMoblet의 구성요소와 이동에이전트에 대한 위치정보와 이름을 맵핑시킨 정보를 관리한다. 이동에이전트가 다른 에이전트스테이션으로 이동하였을 때 위치관리자에 저장되어 있는 위치정보가 변경된다.

*이동관리자*는 이동에이전트가 목적지 에이전트스테이션으로 이동하지 못하는 경우, 이동 에이전트를 임시로 저장하고 관리한다. 일정한 간격으로 목적지 에이전트스테이션으로 이동이 가능한지 파악하여 이동하려는 목적지 시스템이 작동할 때, 이동관리자는 이동에이전트를 목적지 에이전트스테이션으로 이동시킨다.

*접근관리자*와 *메신저*는 각각 이동에이전트의 자원 접근 권한 정보를 가진 객체와 이동에이전트 간 통신을 위해 메시지 객체를 저장하는 JavaSpace로서 SecureJS를 이용한다.

4.1.1 이동에이전트 구조

SecureJMoblet 시스템에서 이동에이전트의 속성은 에이전트 이름, 자신이 실행되는 에이전트플레이스, 에이전트 생성자, 에이전트 소유자, 에이전트 이동경로, 에이전트 자원등급, 현재 에이전트의 위치정보이다.

이동에이전트는 Java의 직렬화(Serialization)를 이용하여 네트워크를 통하여 전송될 수 있는 객체로 선언된다. 그리고 이동에이전트의 이름은 이동에이전트의 식별자로서 생성시간, 생성한 에이전트 시스템 이름, 작성자로 구성된다. <표 2>는 이동에이전트의 속성을 보여준다.

<표 2> 이동에이전트의 속성

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

4.1.2 에이전트의 이동(Mobility)

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

다음과 같은 과정으로 SecureJMoblet의 에이전트는 목적지 에이전트스테이션으로 이동한다.

① 이동에이전트는 목적지 에이전트스테이션의 이름을 가지고 자신이 실행하고 있는 에이전트스테이션에게 이동시켜 줄 것을 요청한다.
② 요청을 받은 에이전트스테이션은 이동에이전트를 직렬화(serialization)하고, 위치관리자를 통하여 목적지 에이전트스테이션의 이름으로 위치 정보(Location)를 알아낸다.
③ 요청을 받은 에이전트스테이션은 목적지 에이전트스테이션의 위치 정보를 가지고 Jini 룩업(LookUp) 서비스를 이용하여 목적지 에이전트스테이션의 서비스 프락시를 다운로드 받는다.
④ 목적지 에이전트스테이션의 프락시를 통하여 목적지 에이전트스테이션의 전송 메소드를 호출하여 직렬화된 에이전트를 전송한다.
⑤ 목적지 에이전트스테이션은 전송된 이동에이전트를 역직렬화(deserialization)를 수행하고 지정된 에이전트플레이스에 저장한 다음 에이전트의 태스크를 수행시킨다.
⑥ 위치관리자에 이동한 에이전트의 위치정보를 최근의 것으로 변경한다.

4.2 에이전트스테이션의 구성 모듈

에이전트스테이션은 이동에이전트 시스템으로서 이동에이

전트의 실행환경을 제공하는 에이전트플레이스, 에이전트의 생명주기(Lifecycle)와 에이전트플레이스의 실행을 제어하는 에이전트모니터, 그리고 에이전트스테이션의 시스템 자원을 제어하는 자원관리자로 구성되어 있다.

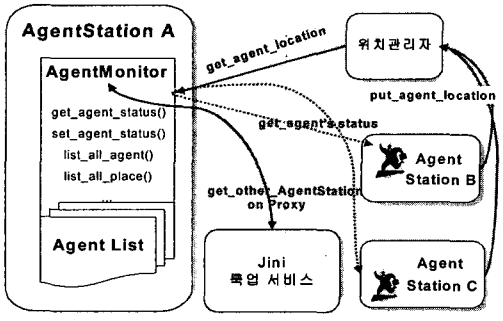
4.2.1 에이전트 플레이스(AgentPlace)

에이전트플레이스는 데몬(daemon)의 형태로 독립적으로 수행되며, 이동에이전트의 실행 환경을 제공한다. 에이전트플레이스는 에이전트를 저장하기 위하여 ArrayList를 사용한 저장소를 가지고 있으며, 이동에이전트의 생성 및 이동시 이 저장소에 추가 또는 삭제된다. 에이전트플레이스는 에이전트스테이션에서 독립적으로 여러 개가 존재할 수 있으며, 실행될 에이전트플레이스가 지정되지 않은 이동에이전트는 기본 에이전트플레이스로 사용되는 DefaultPlace에서 실행된다.

4.2.2 에이전트 모니터(AgentMonitor)

에이전트모니터는 자신이 실행되고 있는 에이전트스테이션 내의 에이전트플레이스 및 이동에이전트의 상태를 검색하고 제어하는 기능을 제공한다. 또한 자신의 에이전트스테이션이 생성한 에이전트에 대한 원격 상태검색과 제어를 할 수 있다. 에이전트모니터는 데몬 형태로 에이전트스테이션 내에서 작업을 수행한다. (그림 8)은 에이전트모니터의 메소드를 이용하여 에이전트플레이스 또는 이동에이전트의 상태를 알아보는 작동과정을 보여주고 있다. 원격에 있는 에이전트 상태를 알아보기 위해서 위치관리자를 이용하여 에이전트가 실행 중인 에이전트스테이션의 에이전트모니터를 이용한다.

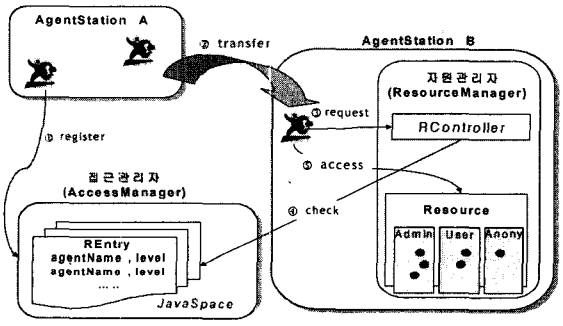
에이전트모니터는 특정 이동에이전트의 상태를 검색하여 에이전트를 시작, 중단, 또는 재시작 시킬 수 있는 제어 기능을 지원한다. 예를 들어, 이동에이전트가 원격 에이전트스테이션에서 수행 중에 시스템의 문제나 다른 요인으로 인해 수행이 중단되는 경우가 발생할 수 있다. 이때, 사용자는 위치관리자와 에이전트모니터를 이용하여 이동에이전트의 상태를 실시간으로 파악하고 이동에이전트를 재실행시키거나, 종료시킬 수 있다.



(그림 8) 에이전트모니터(AgentMonitor)의 동작

4.2.3 자원관리자(ResourceManager)

자원관리자는 이동에이전트가 에이전트스테이션의 자원을 접근하여 사용할 때, 에이전트의 자원 접근 레벨에 따른 자원 할당여부를 결정한다. 자원관리자는 에이전트스테이션의 자원 접근레벨을 *Admin_Service*, *User_Service*, 그리고 *Anony_Service*로 구별한다. 각 접근레벨의 자원등급에 맞추어 에이전트가 수행할 수 있는 API들이 제공된다. 그리고 시스템의 자원을 할당할 때 이용되는 이동에이전트의 자원 접근 정보는 이동에이전트 이름과 접근레벨의 정보를 가진 AccessEntry 타입으로 접근관리자(AccessManager)에 저장된다. (그림 9)는 자원관리자를 통해서 이동에이전트가 요구한 자원의 할당 과정을 보여주고 있다. 이동에이전트는 계정에 따른 자원등급을 가지고 이동에이전트 시스템으로 이동한다. 에이전트스테이션은 이동에이전트의 자원 접근 레벨과 접근관리자에 저장된 자원 관리 객체와 비교하여 그에 따른 자원을 할당하여 준다. 할당받은 자원 접근 객체를 통하여 이동에이전트는 에이전트스테이션의 자원을 이용하여 실행할 수 있다.



(그림 9) 자원관리자(ResourceManager)의 동작

4.3 위치관리자(LocationManager)

위치관리자는 수행 중인 이동 에이전트 시스템과 플레이스, 그리고 이동 에이전트의 위치정보를 관리한다. 사용하고자 하는 SecureJMoblet의 구성요소나 이동에이전트의 이름을 알고 있는 경우, 위치관리자를 이용하여 위치정보를 제공받을 수 있다. 이 위치정보는 룩업서비스를 이용하여 실제 프락시를 얻을 때 사용된다.

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

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

에이전트스테이션에 대한 위치정보는 에이전트스테이션의 이름과 포트(port)번호를 포함하며, 에이전트플레이스의 위치정보는 에이전트스테이션 위치정보에 "/"를 구분자로 에이전

트플레이스의 이름을 연결한 문자열로 표현한다. 그리고 이동에이전트의 위치정보는 자신이 실행되고 있는 에이전트플레이스의 위치정보와 동일하다.

4.4 이동관리자(MovingHelper)

이동에이전트가 목적지 에이전트스테이션으로 이동하지 못하는 경우, 해당 이동에이전트는 JavaSpace로 구현된 MSpace (Missing Space) 내에 저장된다. 이동관리자는 목적지 에이전트스테이션이 다시 작동할 때 저장된 이동에이전트를 이동시킨다. 이동에이전트는 에이전트스테이션의 이름, 목적지 에이전트스테이션의 이름, 그리고 이동 에이전트 객체를 인자로 가진 net.jini.core.entry.Entry 타입으로 MSpace에 저장된다.

이동에이전트가 목적지 에이전트스테이션으로 이동하지 못하는 경우는 목적지 에이전트스테이션 시스템 상의 문제나 네트워크의 문제로 이동할 수 없을 경우, 이동에이전트는 이동관리자의 MSpace에 임시로 저장된다. 이동관리자는 isAlive() 메소드를 이용하여 목적지 에이전트스테이션으로 접속이 가능한 지를 일정 시간 간격으로 검사하고, 목적지 에이전트스테이션과의 접속이 가능한 경우 이동 에이전트를 목적지 에이전트스테이션으로 전송한다.

4.5 에이전트간의 통신

SecureJMoblet 시스템에서는 이동에이전트 상호간 통신 기능을 제공하기 위하여 JavaSpace 서비스를 이용하고 있으며, 메시지 객체의 보안을 위하여 해당 메시지의 수신자만이 메시지 객체에 접근할 수 있는 안전한 JavaSpace 서비스를 제공하는 SecureJS를 기반으로 하고 있다.

이동에이전트 상호간 통신을 위한 메신저 서비스는 전달하고자하는 에이전트, 에이전트스테이션, 그리고 메시지를 인자로 하여 net.jini.core.entry.Entry 타입의 MsgEntry를 저장한다. <표 3>은 메신저의 JavaSpace에 저장되는 MsgEntry의 속성을 나타내고 있다.

〈표 3〉 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

이동에이전트는 자신이 생성되거나 이동할 때, 에이전트 이름과 생성자, 그리고 소유자를 인자로 한 Entry 템플릿을 생성한다. 생성된 템플릿으로 자신이 메시지를 받을 에이전트스테이션을 리스너(Listener)로 등록한다. 메시지가 JavaSpace에 저장되었을 때, JavaSpace는 메시지와 같은 템플릿으로 등록된 리스너에게 이벤트를 통지함으로써 메시지를 전달할 수 있다.

5. SecureJMoblet 시스템의 활용예제

이동에이전트를 이용한 분산 응용프로그램은 다양한 영역에서 활발하게 작성되고 있다. 특히, 전자상거래와 네트워크 관리 및 정보검색 분야에서 이동에이전트를 이용하여 효과적으로 운용하고 있다. 본 장에서는 SecureJMoblet 시스템을 이용한 간단한 전자경매서비스[18]의 개발에 대하여 설명한다.

SecureJMoblet 시스템을 이용한 경매서비스를 제공하기 위하여 AuctionService 클래스를 정의하였다. AuctionService 클래스는 아래의 <표 4>와 같이 경매서비스에서 요구되는 기본적인 메소드를 포함한다.

경매서비스에서 에이전트는 검색 에이전트(Searching Agent)와 입찰 에이전트(Bidding Agent) 두 종류로 구분된다. 클라이언트는 Jini의 룩업서비스를 이용하여 에이전트가 순회해야할 SecureJMoblet 기반의 경매시스템의 목록을 구한다.

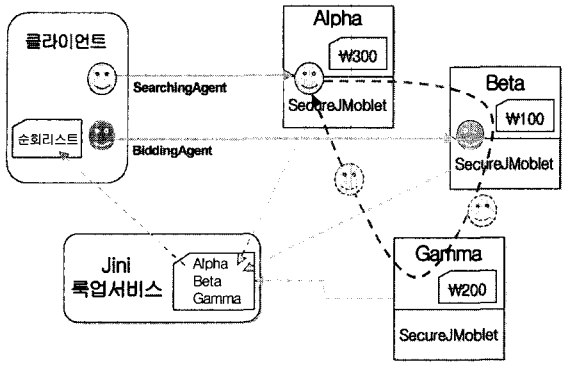
〈표 4〉 AuctionService 클래스의 주요메소드

메 소 드	설 명
register_goods	◦ 에이전트가 경매 물품을 등록
unregister_goods	◦ 경매 물품을 해제
lookup_goods	◦ 경매 DB 내의 경매 물품을 검색
bidding	◦ 입찰에이전트가 경매 가격을 제시
success_bid	◦ 경매 물품에 대한 낙찰

검색 에이전트는 클라이언트로부터 순회해야할 경매시스템의 목록을 받아 경로를 설정하고, 그 경로를 따라 시스템을 이동하면서 판매자가 등록한 상품 정보를 검색한다.

주어진 경로를 모두 순회한 후 검색한 정보를 가지고 사용자의 홈으로 되돌아온다. 입찰 에이전트는 판매자가 등록한 상품이 존재하면 최저가격을 제공하는 경매시스템으로 이동하여, 입찰 알고리즘에 따라 스스로 입찰을 수행한다.

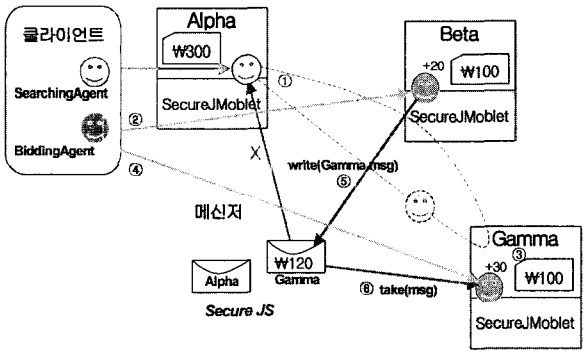
- *검색 에이전트* : **SearchingAgent** 옵션을 가지고 생성된 에이전트는 경매 시, 상품에 관련된 정보를 검색하고, 모니터하는 기능을 가지고 있다.
- *입찰 에이전트* : **BiddingAgent** 옵션을 가지고 생성된 에이전트는 클라이언트를 대신하여 입찰 에이전트가 직접 경매 물품에 입찰을 할 수 있는 권한을 가지고 있다.



(그림 10) 검색에이전트와 입찰에이전트의 동작

<메신저를 이용한 안전한 입찰 정책(Bidding Policy)>

물품을 구매하고자 하는 클라이언트는 검색 에이전트가 검색한 정보를 이용하여 최저가격을 제시한 경매시스템으로 입찰 에이전트를 전송한다. 실제적인 입찰을 하기 전에 검색 에이전트를 다시 한번 주어진 경로에 따라 에이전트를 전송하고, 검색된 값과 비교하여 입찰을 실시한다. 입찰가격과 설정은 이전의 <표 5>를 따른다. 하지만, 입찰을 수행한 후 동일한 물품이 현재 가격보다 작거나 같은 가격으로 검색이 되었다면, 클라이언트는 그 시스템으로 입찰 에이전트를 전송한다. 이러한 경우, 에이전트간 통신을 지원하는 메신저 서비스를 이용하여 개선된 경매서비스를 제공한다. (그림 11)은 메신저를 이용한 에이전트간의 통신과정을 보여준다.



(그림 11) 메신저를 이용한 에이전트간의 안전한 통신

① 검색에이전트는 경매시스템 Alpha→Beta→Gamma를 순회한 후 물품정보를 클라이언트에게 전송한다.

② 클라이언트는 최저가격을 제시한 Beta 경매시스템으로 입찰에이전트를 전송하고, 입찰에이전트는 주어진 입찰 알고리즘을 수행한다.

③ 검색에이전트에 의해 Gamma 시스템에서 새로운 물품이 동일가격 또는 그 이하의 가격으로 검색되었을 경우, 입찰에이전트들은 상호 메시지 통신을 수행하면서 사용자를 대신하여 최적의 경매환경을 실행할 수 있도록 SecureJMoblet 시스템은 지원하고 있다.

④ 클라이언트는 새로운 입찰에이전트를 생성하여 Gamma 시스템으로 전송한다.

⑤ Beta 시스템의 입찰에이전트는 입찰을 수행한 후, 그 가격정보를 메시지객체에 저장하고, 그 메시지 객체를 메신저에 저장하기 위하여 SecureJS의 인터페이스인 *write(Gamma_agent_id, msg, txn, lease)* 메소드를 호출한다. SecureJS 인터페이스는 3.1절에서 자세히 설명하고 있다.

⑥ Gamma 시스템의 입찰에이전트는 메신저 내에 자신에게 도착한 메시지가 있는지 여부를 검사한다.

⑦ Gamma_agent는 메신저 내에 메시지가 있다면 3.1절에서 소개한 <올바른 수신자의 검증 알고리즘>을 적용한다. Gamma_agent가 올바른 수신자임이 증명하기 위하여 *take(encrypt_agent_id, id, msg, txn, lease)* 메소드를 이용한다. 올바른 수신자임이 결정되면 메시지 내의 값을 확인한 후 입찰을 실시한다. 입찰가격 결정은 <표 5>에서 제시한 입찰가격 알고리즘을 따른다.

⑧ Gamma_agent는 메시지 내의 Beta 시스템의 가격보다 적으면 입찰에이전트가 입찰을 수행하고 그렇치 않으면 메신저를 검색하면서 기다린다.

경매물품에 대한 새로운 입찰 가격은 두개의 옵션(사용자설정, 자동설정)이 정의되어 있다. 아래의 <표 5>에서는 그 방법을 자세히 설명하고 있다. 새로운 입찰가격 C에서 ①은 사용자설정 가격이고, ②는 자동설정 가격을 나타낸다.

〈표 5〉 입찰가격 알고리즘

$C = Min(H + x,\ L)$	-----①
$C = H + (L-H)y/100$	-----②
C : 새로운 입찰가격	
x : 사용자정의 부가가격	y : 사용자 정의 백분율
L : 한계 가격	H : 현재 최고 입찰가격

6. 결 론

최근 장소나 시간에 구애받지 않고 생활 속에서 자연스럽고 편리하게 컴퓨터를 사용할 수 있는 유비쿼터스 컴퓨팅이 주목받고 있다. Jini는 다양한 이 기종 장치들 사이에서 유비쿼터스를 지원하기 위한 기반구조와 미들웨어를 제공하기 위한 플랫폼으로 적합하다. 또한 이동에이전트 패러다임은 이러한 환경에서 필요한 정보를 얻고 교환하는 응용서비스 개발에 이용될 수 있다.

이동에이전트는 에이전트의 실행 코드와 상태를 목적지 시스템으로 이동하여 목적지 시스템의 자원을 활용하여 자신의 작업을 수행하는 기술이다. 그리고, Jini는 이러한 이동에이전트 시스템의 기본 기능을 구현하기에 적합한 분산 기반구조

를 제공한다. 하지만, Jini1.0 서비스는 안전한 원격통신을 위한 보안성이 취약하여 이를 이용한 이동에이전트 시스템의 개발은 적합하지 않다. 기존의 Jini 서비스의 문제점을 해결하기 위하여 최근에 썬 마이크로시스템즈는 분산시스템 환경에서 자바코드의 이동성에 따른 보안요소를 충족시키는 Jini2.0을 제시하였다.

본 논문에서는 새로운 Jini2.0 기반에서 동작하는 이동에이전트 시스템인 SecureJMoblet에 대하여 기술하였다. 개발된 SecureJMoblet 시스템은 에이전트 생성, 실행, 전송, 관리 등의 이동에이전트 시스템의 기능과 이동에이전트의 위치 파악 기능을 Jini 서비스 형태로 제공하고 있다. 또한 신뢰성 있는 서비스를 위하여 에이전트의 실행 시 발생하는 예외처리 기능과 이동에이전트 간의 안전한 통신 기능을 제공하고 있다.

SecureJMoblet 시스템은 Jini2.0 보안모델을 적용하여 이동에이전트 또는 이동에이전트 시스템간의 상호인증을 지원한다. 그리고 에이전트 이동 시 원격통신에 사용되는 하부 통신 프로토콜을 보안 요구사항에 따라 다양하게 제공한다. 또한 보안성이 취약한 기존의 JavaSpace 서비스를 보완하여 저장된 객체에 대한 접근제어 기능을 구현하여 안전한 JavaSpace 서비스를 지원하는 SecureJS 서비스를 제공하고 있다.

SecureJMoblet 이동에이전트 시스템은 SecureJS를 이용하여 이동에이전트 객체를 안전하게 저장하기 위한 객체저장소와 이동에이전트 상호간의 안전한 통신 기능을 지원한다.

현재 SecureJMoblet 시스템을 이용한 전자경매서비스는 검색과 입찰과 같은 기본적인 기능만을 지원한다. 향후 전자경매서비스의 다양한 경매정책과 기능을 제공하고, SecureJMoblet 기반의 전자경매서비스를 웹 상에서 이용할 수 있도록 지원할 예정이며, 그 외 개발된 시스템을 효과적으로 활용할 수 있는 분산응용시스템을 개발할 계획이다.

참 고 문 헌

[1] W. Chan, "Using CoolBase to Build Ubiquitous Computing Applications," HP Technical Report, HPL-2001-215, 2001.

[2] Sun Microsystems, "Jini^{tm} Architecture Specification," Published Specification, http://java.sun.com/products/jini/2.0/doc/specs/html/jini-spec.html, 2003.

[3] Jan Newmarch, "Jan Newmarch's Guide to Jini Technologies," Manning Publications Co., 2003.

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

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

[6] Sun Microsystems, "JavaSpacestm Service Specification," Published Specification, http://java.sun.com/products/jini/2.0/doc/specs/html/jini-spec.html, 2003.

[7] Sun Microsystems, "Jini^{tm} Technology Starter Kit Overview v2.0," Published Specification, http://java.sun.com/developer/products/jini/arch2_0.html, 2003.

[8] Sun Microsystems, "Security enhancements for the Java2 SDK," http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html, 2003.

[9] G.P.Picco, A.L. Mruphy, and G-C. Roman, "Lime : Linda Meets Mobility," Ind.Garlan, editor, Proc. of the 21stInt. Conf. on Software Engineering, pp.368-377, 1999.

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

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

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

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

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

[15] 유양우, 이명준, "분산응용프로그램을 위한 안전한 JavaSpace", 한국정보과학회 춘계학술발표회, pp.352-354, 2003.

[16] 문남두, 안건태, 박양수, 이명준, "그룹통신을 이용한 견고한 LDAP 서버", 정보처리학회논문지C, 제10-C권 제2호, 2003.

[17] 김진홍, 구형서, 유양우, 이명준, "JMoblet : Jini 기반의 이동에이전트시스템", 정보처리학회논문지B, 제8-B권 제6호, pp. 292-312, 2001.

[18] 유양우, 구형서, 김진홍, 문남두, 이명준, "Jini 기반이 이동에이전트를 이용한 경매시스템", 한국정보과학회 춘계학술발표회, April, 2002.

유 양 우

e-mail : soft@mail.uc.ac.kr

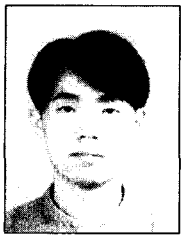
1995년 경일대학교 전자계산학과 (공학사)

1997년 울산대학교 대학원 전자계산학과 (공학석사)

2000년 울산대학교 대학원 전자계산학과 박사과정 수료

2000년~현재 울산과학대학 컴퓨터정보학부 전임강사

관심분야 : 분산객체 시스템, 이동에이전트 시스템, 웹기반 정보시스템 등



문 남 두

e-mail : dooya@ulsan.ac.kr

1997년 울산대학교 전자계산학과(공학사)

1999년 울산대학교 컴퓨터정보통신공학부 (공학석사)

2003년 울산대학교 컴퓨터정보통신공학부 (공학박사)

관심분야 : 그룹통신 시스템, 분산객체, CSCW, 웹 프로그래밍 등

이 명 준

e-mail : mjlee@ulsan.ac.kr

1980년 서울대학교 수학과(학사)

1982년 한국과학기술원 전산학과(석사)

1991년 한국과학기술원 전산학과(박사)

1982년~현재 울산대학교 컴퓨터정보통신공학부(교수)

1993년~1994년 미국 버지니아대학 교환교수

관심분야 : 웹기반 정보시스템, 프로그래밍언어, 분산 프로그래밍 생물정보학