論文
3-2-8

에이젼트 관리 방안 연구
(A Study of Agent Management Scheme)

이 길 흥*
(Kil-Hung Lee)

요 약

에이젼트를 이용한 응용의 증가와 함께, 이의 효율적 운영을 위한 관리 방안이 필요해졌다. 본 논문에서는 소프트웨어 에이젼트의 동작을 모니터링하고 작업을 제어하는 관리 방안에 대해 연구하였다. 에이젼트를 효과적으로 관리하기 위하여 에이젼트의 동작 특성을 조사하고, 관리에 필요한 관리정보를 SNMP SMI 표준에 따라 정의하였다. 구현 방안은 많은 에이젼트를 관리하기 위해, 에이젼트 매스터를 이용하여 동작을 관리하는 방식으로 하였다. 본고에서 정의한 에이젼트 관리정보를 자체 제작한 에이젼트 시스템에 구현하여 이를 적용하여 실험한 결과, 효과적인 에이젼트의 동작 모니터링 및 관리가 가능해짐을 알 수 있었다.

ABSTRACT

As the number of agent application grows, agent management scheme is required for the efficient manipulations of it. In this paper, we studied an agent management scheme for monitoring and control of the working of an agent. For efficient management of an agent, we investigated the characteristics of the agent, and defined management information according to the specification of the SNMP SMI standard. Implemented scheme uses agent master system for the efficient management of many agent applications. We applied the defined management scheme using agent system that we developed in our laboratory, and found that our scheme was efficient for the monitoring and management of the agent applications.

1. 에이젼트 관리 개요

인터넷의 확장과 소프트웨어 개발 방법의 발달로 에이젼트를 이용한 응용 프로그램의 비중이 증가하고 있다. 에이젼트는 소프트웨어 사용자의 위임을 받아 자율적으로 작업을 수행하는 프로그래밍 개체이다. 이러한 동작 방식은 전통적인 분산 컴퓨팅에 대한 또 다른 대안으로서, 새로운 컴퓨팅 패러다임을 형성하고 있다. 에이젼트의 특징은 자율성과 상황에의 반응성, 스스로의 학습과 협력 및 이동 능력 등이다. 필요한 기능을 적시에 필요한 장소에서 동작시킬 수 있는 에이젼트의 기능을 효과적으로 이용하면 네트워크 트래픽의 감소와 업무의 효율을 기할 수 있다. 앞으로 에이젼트의 기능은 계속 발전할 것이며 에이젼트는 소프트웨어 기술의 중요한 위치를 차지할 것이다.[1]

에이젼트는 플레이스 (place)에서 존재한다. 플레이스는 에이젼트가 동작하는 환경으로서 접근제어

* 정회원 : 서울산업대학교 컴퓨터공학과 전임강사

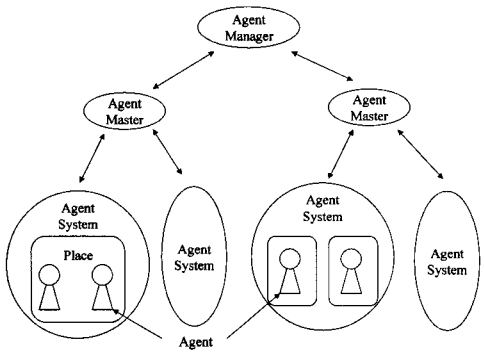
논문접수 : 2002. 1. 30.
심사완료 : 2002. 2. 20.

※ 이 논문은 서울산업대학교 학술문화연구재단 학술연구비 지원에 의하여 연구되었음.

같은 기능을 제공하는 컨텍스트 (context)이며, 하나의 에이전트 시스템 (agent system) 내에는 하나 이상의 플레이스가 존재할 수 있다. 에이전트 시스템은 에이전트를 생성하고, 실행하며, 전송하고, 종료시킬 수 있는 플랫폼이다. 하나의 호스트 (host)에는 하나 혹은 그 이상의 에이전트 시스템을 가질 수 있다. 에이전트의 위치는 에이전트가 위치한 플레이스와 에이전트 시스템의 주소를 포함하고 있다. 같은 관리체계를 갖는 에이전트 시스템 그룹은 하나의 영역 (region)으로 표현된다. 에이전트는 플레이스를 이동하면서 임무를 수행하는데, 하나의 플레이스에 둥지를 틀고 생성되어 활동을 시작한 다음 소멸되거나 다른 플레이스로 이동할 수 있다.[2]

이러한 에이전트 기술을 이용하여 전자상거래나 검색 기술 등 여러 서비스에 적용 가능하나, 특히 라우팅과 망 관리 서비스에 응용하는 사례가 늘고 있다. [3]에서는 MAF (Mobile Agent Facility Specification)의 정의에 따라 MIB (Management Information Base)을 정의하여 자체의 분산 망 관리 시스템이 적용하였다. SNMP (Simple Network Management Protocol) 관리자는 Agent API를 통해 에이전트를 생성 및 제어하고, MAF MIB과 DISMAN (Distributed Management) MIB을 통해 필요한 망 관리 서비스를 수행한다.[4] [5]에서는 에이전트 서비스를 망 관리 서비스와 통합하기 위하여 에이전트와 SNMP 에이전트간에 정보 교환을 위해 필요한 인터페이스를 정의하고, 에이전트 기술을 응용하여 효과적으로 망 관리 서비스가 수행됨을 보였다. 또한, 에이전트를 제어하기 위한 관리 정보를 정의하였다. SNMP 에이전트와 이동 에이전트간은 같이 존재하거나 RMI (Remote Method Invocation)를 통해 정보를 교환한다.

[6]에서의 관리정보 구조는 에이전트를 관리하기 위해 에이전트마다 SNMP 에이전트가 존재하는 방식으로 정의된다. 하지만 에이전트의 수가 많아지면 이들 에이전트마다 SNMP 에이전트를 두고 관리하는 방식은 관리 노드 수의 증가로 관리자에게 부담을 줄뿐만 아니라. 에이전트가 있는 시스템의 부하를 가중시킨다. [3]의 경우는 RMI를 통해 하나의 SNMP 에이전트가 지역 내의 다수의 에이전트를 관리하나, 관리 정보가 충분치 않아서 에이전트를 효과적으로 관리하기가 어렵다. 따라서, 본 연구에서는 많은 수의 에이전트를 효과적으로 관리하기 위하여 한 지역 혹은 같은 커뮤니티의 에이전트를 에이전트 매스터를 통해 관리하는 구조를 정의하고, 에이전트 매스터에게 필요한 충분한 관리 정보 구조를 정의하였다. 그리고, 에이전트 매스터와 에이전트간의 교환 메시지와 클라이언트 서비스 실행 구조를 정의하였다.



[그림 1] 구현 에이전트 관리 구조

[Fig. 1] Implemented agent management architecture

[그림 1]에서와 같이 에이전트를 관리하고자 하는 에이전트 관리자는 에이전트 매스터에 설치되어 있는 SNMP 에이전트를 통해 지역의 에이전트를 관리한다. 에이전트 관리자는 지역의 여러 에이전트 정보를 SNMP 에이전트를 통해 받아 모니터링하고, 새로운 관리 정보를 SNMP 에이전트를 통해 설정함으로써 망 관리 기능을 수행한다. 모든 에이전트 시스템에 SNMP 에이전트를 동작시키는 것은 에이전트 시스템의 부하를 유발하므로, 에이전트 마스터에게 SNMP 에이전트를 동작시키고 필요한 관리 기능은 에이전트 마스터를 통해 이루는 방식으로 동작시킨다. 에이전트 매스터와 에이전트간은 소켓 통신을 통해 관리 기능이 전달된다.

2. 에이전트 관리 MIB

2.1 관리 정보 데이터베이스

에이전트를 관리하기 위해서는 관리에 필요한 MIB을 미리 정의하여야 한다. 관리자는 MIB의 정보를 읽어서 망의 상태를 파악하고, 필요 시 관련 MIB의 정보를 적절한 값으로 설정 혹은 변경하여 에이전트의 제어를 수행할 수 있다. 관리 MIB의 정보는 SNMP 프로토콜을 통해 SNMP 관리자와 SNMP 에이전트 사이에 교환된다. 관리 MIB에 형식에 대한 정의는 SNMP 버전 1에서 정의한 RFC (Request For Proposal) 1157과, SNMP 버전 2에서의 RFC 1902, 그리고 SNMP 버전 3에서의 RFC 2553이 있다. 본 고에서는 SNMP 버전 1과 버전 2의 정의에 따라 관리 정보를 정의하였다.

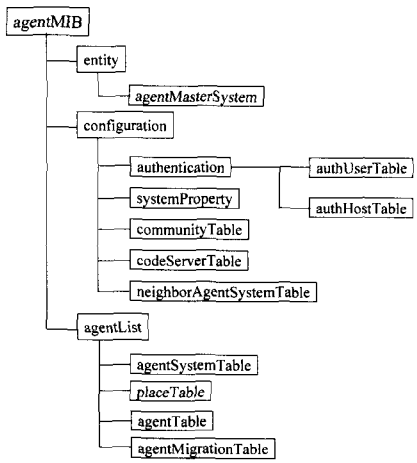
에이전트를 관리하기 위해 필요한 관리정보는 OMG (Object Management Group)에서 에이전트의 상호 연동에 관한 MAF 스펙 [7], FIPA (Foundation for Intelligent Physical Agents) [8]에서 이질적인 소프트웨어 에이전트간의 상호 연동을 위해 정의한 FIPA-OS 규정을 기초로 했다. MAF에서는 같은 언어로 만들어진 에이전트간에 상호 작용을 위해 에이전트 전송, 클래스 전송, 에이전트 관리에 대한 표준을 마련하고 있다. 이는 여러 벤더에 의해 만들어진 에이전트나 에이전트 응용간의 상호 연동에 필요한 최소한의 기능을 표준화한 것이다.

FIPA는 에이전트의 통신에 관계된 것을 주로 다루는 것으로서 애플리케이션, 추상 구조, 에이전트 관리, 에이전트 통신, 에이전트 메시지 전송 등 5가지에 대한 표준을 마련하고 있다. FIPA에서 정의한 에이전트의 관리는 에이전트의 생성, 등록, 위치지정, 통신, 이동이나 소멸 등의 생명주기 관리에 대해 다룬다.[9]

2.2 에이전트 관리 MIB 정의

관리 정보용 MIB은 Object Identifier (oid)의 관리 번호로 분류한다. MIB의 oid 값은 보통 트리 구조로 이루어지는데, 최상위 oid 값은 iso (1) 이다. iso 하부에 org (3), dod (6), internet (1), private (4), enterprise (1) 순으로 연결되고, 구현한 에이전트 MIB은 enterprise 하부에 snut (12012), network (1), agent (1) 로 이루어진다.

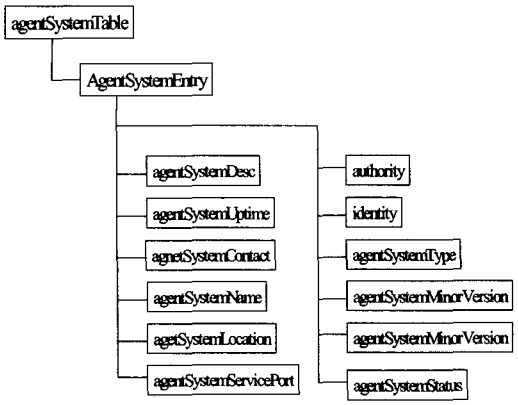
에이전트를 관리하기 위하여 정의한 관리 MIB을 [그림 2]에 보였다. agentMIB 모듈은 하부에 entity, *authentication*, *agentList* 등 3 개의 그룹으로 이루어진다. 각각의 그룹 밑에는 필요한 스칼라 정보 혹은 테이블로 구성되어 있다.



[그림 2] 에이전트 관리 MIB
[Fig. 2] Agent management MIB

우선 에이전트 매스터의 일반적인 시스템 정보에 대한 관리 MIB이 필요하다. 에이전트 시스템의 이름과 관리자, 연락 정보, 시작 시간 등을 정의할 필요가 있다. 이러한 정보는 MIB-II의 System 그룹에 이미 정의되어 있는 것도 있지만, 에이전트 마스터 시스템에 대한 보다 정확하고 자세한 정보를 위해 추가로 entity 그룹 밑의 agentMasterSystem 그룹으로 정의하였다.

[그림 3]에서의 agentSystemTable은 SNMP 에이전트에 등록한 에이전트 시스템 정보를 갖는 테이블이다. 에이전트 시스템 테이블 정보는 에이전트 시스템 이름, 주소, 시스템 종류, 등록 시간, 서비스 포트, 플레이스 포인터, 서비스 패스워드 등이다.



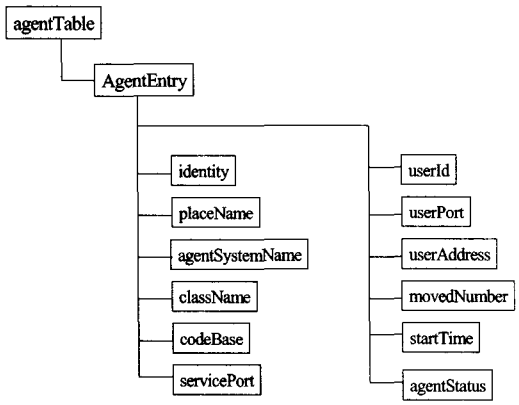
[그림 3] agentSystemTable 관리 정보

[Fig. 3] agentStstemTable information

agentSystemUptime은 에이전트 시스템이 처음 등록한 이후 경과된 밀리 초 단위의 시간이고 agentSystemRecentContactTime은 최근에 에이전트 시스템과의 통신 시간 정보를 갖는다. 등록확인 메시지를 교환했던 시간이나, 서비스 실행 시 통신이 이루어진 시간 정보가 있게 된다. authority는 에이전트 시스템을 엑세스할 때의 사용 아이디이고, identity는 에이전트 시스템의 ID 정보이다. agentSystemType 정보는 에이전트 시스템의 종류에 대한 정보이다. MAF 표준에서는 aglets(1), MOA(2), AgentTcl(3), NonAgentSystem(0) 등의 값이 미리 정의되어 있다. agentStstemMajorVersion, agentSystemMinorVersion은 에이전트 시스템의 버전 정보를 갖는다. agentSystemStatus는 에이전트 시스템의 현재 동작 상태 정보이다. servicePort는 에이전트 시스템의 TCP/UDP 서비스 포트 번호이다. 이 값은 에이전트 시스템이 실행할 때 시스템으로부터 동적으로 할당받은 포트 번호이다.

placeTable은 에이전트 시스템에 존재하는 플레이스의 정보를 갖는다. 플레이스의 이름, 주소, 서비스 종류, 서비스 포트 등의 내용을 담는다. 따로 플레이스를 지정하지 않을 경우, 디폴트 플레이스를 말하는 것으로 정의한다. 따라서, 에이전트 시스템 등록 시에 플레이스에 대한 언급이 없을 경우에는 자동으로 디폴트 플레이스 등록을 하는 것으로 간주한다. placeName은 플레이스 이름이다. placeLocation은 플레이스가 있는 에이전트 시스템의 정보를 갖는다.

agentTable은 에이전트 마스터 시스템의 관리 하에 있는 영역 내이서 생성된 에이전트의 정보를 갖는다. 여기에는 에이전트의 이름, 아이디, 서비스 이름, 버전, 능력, 요청자 정보, 시작 시간, 관리자 정보, 호스트 주소, 서비스 포트번호, 작업상태 정보 등의 내용이 있다. 등록된 에이전트의 수만큼의 열이 테이블에 나타날 것이다. 각 열은 세션 아이디로 구분한다. 세션 아이디는 서비스 포트 정보로 이루어진다.



[그림 4] agentTable 관리 정보

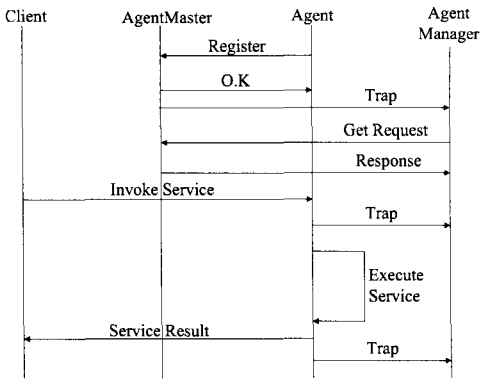
[Fig. 4] agentTable management information

[그림 4]에서 identity는 에이전트의 ID로서, 이 값은 전체 도메인에서 유일해야 한다. 본 고에서는 이 값을 클래스 이름, 버전, 시작 시각, 호출자 아이디로 정의하였다. placeName은 에이전트가 위치한 플레이스 정보, agentSystemName은 에이전트가 위치한 에이전트 시스템 이름, className은 패키지명을 포함하는 클래스 이름, codeBase는 코드가 위치한 클래스의 기준 위치 URL 정보를 갖는다. servicePort는 서비스를 호출한 시스템의 TCP/UDP 서비스 포트 번호이다. 이 값은 에이전트 시스템이 호출자에게 정보를 보낼 때 사용하는 값으로 에이전트 이동 후 재접속에 필요하다.

agentTable의 내용은 에이전트가 이동할 때에는 agentMigrationTable의 한 열로 이동한다. agentMigrationTable에서의 migrationNumber 항은 이동 순서를 의미한다. 맨 처음 에이전트가 플레이스 A에서 생성되면 agentTable에 하나의 열이 새로 생성되고 migrationNumver의 값은 1이 될 것이다. 이 에이전트가 플레이스 B로 이

동하면, 이 열은 migration 테이블로 복사되고, agentTable의 해당 열의 플레이스 값은 새로운 플레이스의 값으로 바뀌고, migrationNumber의 값은 2로 된다.

authentication 관리 정보 그룹은 2 개의 테이블로 구성되어 있다. authUserTable은 이용자가 접속할 때 인증에 사용하는 아이디와 패스워드 정보를 갖는 테이블이다. 클라이언트가 에이전트를 이용한 서비스를 수행하려고 할 때 인증에 사용된다. authHostTable은 에이전트 인증 시 특정 주소로부터 오는 서비스 요구는 주소를 인증으로 사용하여 서비스를 제공해주는 방식을 지원한다. 각각의 이용자와 호스트는 userLevel과 hostLevel의 값에 따라 서비스의 수준이 달리 적용될 수 있다.



[그림 5] 에이전트의 관리 동작 순서도

[Fig. 5] Sequence diagram of agent management behavior

3. 에이전트의 동작과 MIB

3.1 서비스 요청과 실행

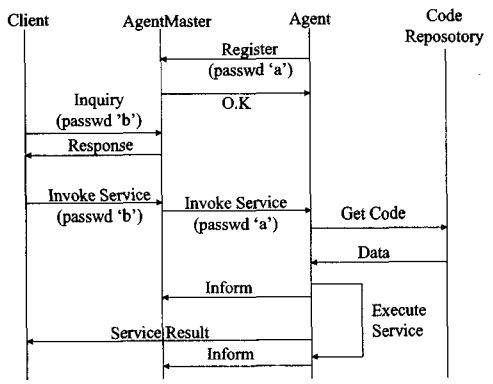
새로운 에이전트 시스템이 동작을 시작하면 우선 에이전트 매스터에게 등록을 한다. 에이전트 시스템은 등록 메시지에 플레이스 이름, 주소, 서비스 포트, 버전 정보, 능력, 접근 암호 등을 실어 에이전트 매스터에게 보낸다. 에이전트 매스터가 등록 메시지를 받으면, 기존의 리스트를 살펴 본 다음, 새로운 에이전트로부터 온 메시지의 경우 새로운 에이전트 시스템을 리스트에 추가하고, agentSystemTable에 하나의 열을 추가한다. 또한, 설정에 따라 에이전트 매니저에게 트랩 메시지를 보내서 매니저에게 알릴 수도 있다. 기존에 존재하는 에이전트 시스템으로부터 들어온 등록 메시지의 경우는 메시지 시간만을 업데이트 한다. 일정기간 메시지가 없는 경우 에이전트 리스트에서 해당 에이전트 시스템을 제거하고, 에이전트 등록 테이블에서 해당 에이전트 시스템의 정보를 갖는 열 정보를 제거한 다음 에이전트 관리자에게 보고한다.

클라이언트가 에이전트를 이용하여 새로운 작업을 수행하기 위해서는 먼저 에이전트 매스터를 통해야 한다. 에이전트 매스터에게 등록된 에이전트 시스템 정보를 요청하면 에이전트 매스터는 등록된 에이전트 시스템 정보를 보내준다. 클라이언트는 특정 에이전트를 선택하여 새로운 작업을 행할 수도 있고, 단지 서비스만을 지정하여 매스터에게 요청할 수도 있다. 두 가지 모두 매스터에게 요청 정보가 전달이 되고, 매스터 에이전트는 선택된 에이전트 시스템이나, 등록된 에이전트 시스템 중에서 하나를 선택하여 요청한 작업에 대한 수행 요구를 에이전트 시스템에게 보낸다. 클라이언트와 에이전트 시스템간에 새로운 세션이 성립되면, 매스터 에이전트는 에이전트 테이블에 새로운 열을 추가하여 새로이 시작된 에이전트 정보를 추가한다. 에이전트는 서비스를 수행하고, 서비스의 일시 정지나 다시 시작, 종료 등의 서비스의 상태 변화가 있는 경우 에이전트 매스터에게 상태 변화 보고를 한다. [그림 5]에 에이전트의 등록과 서비스 실행의 동작 순서를 시간과 메시지 다이어그램을 통해 보였다.

3.2 서비스 엑세스 제어

클라이언트가 에이전트 시스템을 이용하여 작업을 수행할 때는 이용자 아이디와 패스워드를 필요로 한다. 또한, 특정 에이전트 시스템에게 작업 요청을 하는 경우에는 그 에이전트 시스템의 패스워드를 알아

야 한다. 아이디와 패스워드 별로 권한이 따로 설정되어 있다. 클라이언트는 매스터 에이전트에게 접근하여 서비스를 요청하기 위해서는 아이디와 패스워드를 알아야 한다. 클라이언트가 매스터 에이전트에게 에이전트 시스템에 대한 정보 요청이나 서비스 요구 메시지를 보낼 때는 메시지 내에 이용자 아이디와 패스워드를 포함시켜야 한다. 매스터 에이전트는 클라이언트로부터 들어온 메시지에서 아이디와 패스워드 정보를 보고, 이를 이용자 인증 테이블에서의 항목과 비교한다. 이용자 항목에서 해당 정보가 존재하고, 요구하는 서비스의 수준이 테이블에 정의되어 있는 서비스 수준에 합당할 때는 메시지에 요구된 서비스를 자신이 직접 수행하거나 합당한 에이전트 시스템에게 서비스 수행 메시지를 넘겨준다. 에이전트 시스템에게 서비스를 넘겨줄 때는 에이전트에게 보내는 메시지에 에이전트 시스템이 요구하는 패스워드 정보를 포함시켜 넘긴다. 이 패스워드 정보는 에이전트 시스템이 등록을 할 때 포함되어 있던 정보로서 이 정보는 매스터 에이전트의 등록 테이블에 담겨 있다. 클라이언트의 서비스 요구 메시지의 아이디와 패스워드가 적절치 못한 경우 매스터 에이전트는 서비스를 수행하지 않거나 에이전트 시스템에게 서비스 요구를 전달하지 않는다.



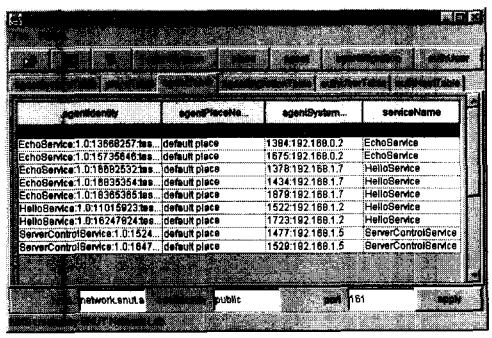
[그림 6] 에이전트 시스템의 서비스 코드 실행 순서도

[Fig. 6] Sequence of executing service code of agent system

[그림 6]에서 에이전트가 authority를 'a'로 등록한 이후에, 이용자의 서비스 요구를 authority 'b'를 통해 에이전트 매스터에게 의뢰하여 에이전트 서비스를 실행하는 순서이다. 에이전트 매스터는 에이전트 고유의 authority로 에이전트 시스템에게 서비스 요청을 전달하여 이용자가 직접 에이전트 시스템으로 엑세스하지 못하도록 한다.

4. 구현 및 실험

실험에 이용된 에이전트는 자체 제작한 자바 에이전트 시스템이다. 최근의 이동 에이전트 시스템은 대부분 자바 기반이고, 네트워크 플러그 앤 워크의 기능을 제공하는 지니 기반의 에이전트 개발도 관심을 끌고 있다. [10] SNMP 에이전트는 AdventNet [11] SNMP 관리자와 에이전트 개발 툴과 MG-Soft [12]의 관리 툴을 이용하였다. 처음에 매스터 에이전트가 부팅이 되면, 서버 소켓 포트 12012를 열고 에이전트 시스템의 등록을 기다린다. 또한, SNMP 에이전트를 기동하고, 에이전트 매스터의 관리정보 테이블을 초기화한 다음 트랩을 통해 에이전트 관리자에게 보고한다.



[그림 7] 에이전트 관리자 화면

[Fig. 7] The scene of agent manager

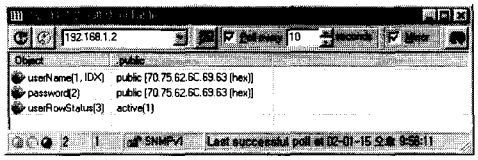
이후 에이전트 시스템이 시작을 하면 에이전트 마스터에게 등록을 수행한다. 에이전트 마스터는 등록된 에이전트 시스템이 대한 테이블의 한 열을 추가하고, 플레이스의 내용에 따라 플레이스 테이블에도 등록되는 플레이스의 개수만큼 한 열을 추가한다. 등록 시 플레이스에 대한 정보가 없으면 디폴트 플레이스를 하나 등록하는 것으로 한다.

클라이언트는 자바 애플리케이션 혹은 JSP (Java

Server Page)를 통해 에이젼트 매스터에 접속한다. 에이젼트 매스터에 등록된 에이젼트 시스템을 조사하여 원하는 에이젼트 시스템을 선택하고, 선택한 에이젼트 시스템에게 필요한 서비스를 호출한다. JSP를 통해 에이젼트 매스터에 연결되는 경우, 이용자 인터페이스는 애플릿을 통해 클라이언트로 다운로드 된다.

애플릿과 에이젼트간에 서비스 세션이 열리고 에이젼트로 서비스 프로그램의 URL (Uniform Resource Locator) 정보와 파라미터가 보내진다. 에이젼트는 URL을 분석하여 네트워크의 서비스 프로그램 소스로부터 코드를 획득하고 에이젼트 생성 후 서비스를 시작한다. 이후 서비스 상태정보 및 결과 데이터가 에이젼트 매스터를 통하여 애플릿으로 보내진다. 에이젼트를 통한 서비스를 실행 후에 세션이 종료하면, 에이젼트 마스터는 에이젼트 정보를 로그파일에 저장한다.

이용자가 에이젼트 시스템에게 필요한 서비스를 호출하는 경우 새로운 에이젼트가 생성되고, 에이젼트 매스터의 에이젼트 테이블에 생성된 에이젼트의 정보가 새로 추가된다. 이 테이블은 에이젼트가 소멸될 때까지 테이블에 존재하게 되고, 상태 변경 시에는 상태 정보만이 바뀐다. 에이젼트가 이동하는 경우에도 에이젼트의 플레이스 정보와 에이젼트 시스템의 정보만 바뀔 뿐 테이블의 열은 그대로 존재하게 된다. 에이젼트가 소멸되면 해당 에이젼트의 agentTable 정보는 agentMigrationTable로 이동한다.



[그림 8] authUserTable 뷰

[Fig. 8] authUserTable view

에이젼트 이용자가 엑세스할 때 필요한 인증 정보는 authUser 테이블과 authHost 테이블이고 이 내용은 에이젼트 관리자가 설정한다. [그림 8]은 authUserTable과 authHostTable을 엑세스하여 보인 것이다. 이 정보는 에이젼트 관리자가 설정하는 값이다.

5. 결 론

에이젼트 MIB을 정의하고 에이젼트 매스터를 통해 에이젼트 서비스의 상태를 모니터링하고 관리하는 방안을 제시하였다. 에이젼트 매스터를 통해 관리하는 방안은 에이젼트의 관리를 효율적으로 할 수 있게 해 주었다. 하나의 에이젼트 매스터를 통해 다수의 에이젼트를 관리하고, 에이젼트 관리자는 다수의 에이젼트 매스터를 통해 많은 수의 에이젼트 관리가 가능해진다. 에이젼트 관리 정보 MIB은 에이젼트 시스템 그룹과 에이젼트 시스템, 플레이스, 그리고 에이젼트 테이블로 구성된다. 정의 MIB을 자체 구현한 자바 에이젼트와 AdventNet의 SNMP 에이젼트 툴 킷을 이용하여 구현하고 에이젼트 매니저를 통해 효과적으로 관리됨을 실험을 통해 알아보았다. 앞으로 에이젼트의 안전한 동작 환경을 제공하기 위해서 암호화 및 보안 관리 부분을 보완해야 할 것이다.

※ 참고문헌

[1] Green Shaw, Hurist L., Nangle, B., Cunning gham, P. Somars, F., Evans, R., Software A gents: A review, http://www.cs.tcd.ie/research _groups/aig/iag/ toplevel2.html.

[2] Danny B. Lange and mitsuru Oshima, Progr amming and Deploying Java Mobile Agent with Aglets, Addison Wesley, 1998.

[3] Patricia Cuesta Rivalta, Mobile Agent Mana gement, Thesis for Master of Engineering at Dept. of Systems and Computer Engineerin g, Carleton University, October 2000.

[4] Jrgen Schnwlder et al., "Building Distributed Management Applications with the IETF Scr ipt MIB", *IEEE Journal of Selected Areas i n Communications (JSAC)*, vol. 18, No. 5, pp. 702-714, May, 2000.

[5] Pagurek, B., Wang, Y., White, T., "Integrati on of Mobile Agents with SNMP: How and Why", *Network Operations and Management Symposium 2000, (NOMS '00)*, IEEE/IFIP, p p. 609-622.

[6] http://nms.estig.ipb.pt/masif_mib/index.jsp

[7] GMD FOKUS and IBM, Mobile Agent Faci lity Specification V1.0, Jan. 2000.

[8] http://www.fipa.org

[9] XC00023H, FIPA Agent Management Specif ication, 2001.10.3

[10] 김진홍 외, JMoblet : Jini 기반의 이동 에이 전트 시스템, 정보처리학회 논문지 B, Vol. 8-B, No. 6, Dec 2001, pp. 641-650.

[11] http://www.adventnet.com

[12] http://www.mg-soft.com

이 길 홍

1989년2월 연세대학교 전자공학과 졸업(공학사)

1991년2월 연세대학교 대학원 전자공학과 졸업(공학석사)

1991년-1995년 LG정보통신 연구소네트워크그룹

1999년8월 연세대학교 대학원 전기컴퓨터공학과 졸업 (공학박사)

2000년5월-현재 서울산업대학교 컴퓨터공학과 전임강사