에이전트 인터페이스 및 응용 서비스 개발

이 길 흥†

요 약

본고에서는 에이전트의 제어 및 관리를 위한 인터페이스 표준과, 각 인터페이스에 필요한 메시지를 구현하였다. 에이전트 인터페이스는 에이전트, 에이전트를 생성하고 활동 토대를 제공하는 에이전트 시스템, 서비스를 요구하는 클라이언트, 서비스를 중재하는 에이전트 매스터, 망 내의 에이전트 관리 기능을 수행하는 에이전트 관리자 사이의 메시지 교환으로 정의된다. 이러한 인터페이스 표준은 OMG의 MAF와 FIPA의 에이전트 표준을 반영하여 만들어 졌으며, 논문에서 정의한 인터페이스와 메시지를 이용한 응용을 구현하여 실험하였다. 실험 망은 컨텐츠를 분배해주는 망을 에이전트로 구현한 망으로서, 에이전트 인터페이스를 통해 이를 제어하고 관리하였다.

A Study on the Development of Agent Interface and Agent Application Service

Kil-Hung Lee†

ABSTRACT

In this paper, we defined agent interface standard and messages for the control and management of agent. Agent interface is defined by messages exchanged between each component of the agent system environment, such as agent, agent system that creates and controls the agent, client that requests the service, agent master that mediates the service, agent manager that performs management functions of the agent. Agent interface is defined after the MAF of the OMG and the agent standard of the FIPA. Experiments are done for the application using agent interface and messages of this paper. The test network was the content distribution network using agent service, and we controled and managed the test network through the agent interface.

키워드 : 에이전트(Agent), 인터페이스 및 메시지(Interface and Message), 제어 및 관리(Control and Management)

1. 서 론

인터넷의 발전으로 다양한 응용 서비스의 출현과 함께 이를 처리하는 다양한 응용 프로그램이 개발되어 필요한 서비스를 여러 방식으로 구현하게 되었다. 피어 투 피어 서비스의 개념과 소프트웨어 개발 방법의 발달로 에이전트를 이용한 응용 프로그램의 역할이 증대되고 있다. 에이전트는 소프트웨어 사용자의 위임을 받아 자율적으로 작업을 수행하는 프로그래밍 개체이다. 이러한 동작 방식은 전통적인 분산 컴퓨팅에 대한 또 다른 구현방식으로서, 새로운 컴퓨팅 패러다임을 형성하고 있다. 에이전트의 특징은 자율성과 상황에의 반응성, 스스로의 학습과 협력 및 이동 능력 등이다. 필요한 기능을 적시에 필요한 장소에서 동작시킬 수 있는 에이전트의 기능을 효과적으로 이용하면 네트워크 트래픽 감소와 업무의 효율을 기할 수 있다. 앞으로 에이전트의 기능은 계속 발전할 것이며 에이전트는 소프트웨어 기술의 중요한 위치를 차지할 것이다[1].

이러한 에이전트 기술을 이용하여 전자상거래나 검색 기술 등 여러 서비스에 적용 가능하고, 라우팅과 망 관리 서비스 등 통신망의 구조에 응용하는 사례가 늘고 있다[2]. [3]에서는 대규모의 네트워크를 효과적으로 모니터링하기 위해 에이전트를 이용하였다. 에이전트는 자신의 복제물을 생성하여 네트워크에 분산 배치하여 관리 기능을 분산화 함으로써 관리 트래픽을 줄이고 응답시간을 단축시켰으며, 고장에도 안정적으로 동작할 수 있는 환경을 제공함을 보였다. [4]에서는 에이전트 서비스를 망 관리 서비스와 통합하기 위하여 에이전트와 SNMP(Simple Network Management Protocol) 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 에이전트 간은 특수 API를 통해 바로 통신을 이루거나 RMI(Remote

† 정 회 원 : 서울산업대학교 컴퓨터공학과 교수
논문접수 : 2002년 9월 3일, 심사완료 : 2003년 6월 2일

Method Invocation)를 통해 정보를 교환한다.

OMG(Object Management Group)의 MAF(Mobile Agent Facility)나 FIPA(Foundation for Intelligent and Physical Agent)의 에이전트 표준은 에이전트의 생성이나 소멸 등에 관한 정보와 에이전트간의 통신에 비중을 두고 있다. 이러한 표준을 준수하면서 응용 환경에 맞는 시스템을 구현하는 방안은 옵션에 따라 다양한 형태의 모습을 가질 수 있다. 본 논문에서는 많은 수의 에이전트를 효과적으로 제어하고 관리하기 위한 시스템의 기본적인 컴포넌트의 구조와 기능, 인터페이스와 메시지 등을 정의한다. 기본적으로 에이전트 매스터의 중재를 통해 서비스가 이루어지고, 에이전트 관리자가 전체의 에이전트 시스템을 모니터링한다. 많은 수의 에이전트를 효과적으로 관리하기 위하여 한 지역 혹은 같은 커뮤니티의 에이전트를 에이전트 매스터를 통해 관리하는 구조이다. 2장에서 에이전트의 관리 표준에 대해 살펴보고, 3장에서 에이전트 인터페이스와 메시지를 정의한다. 4장에서 에이전트 제어 및 관리 시스템의 구현방안을 제시하고 에이전트를 응용한 컨텐츠 분배 망을 소개한 다음 5장에서 결론을 맺겠다.

2. 에이전트 관리 표준

소프트웨어 에이전트는 컴퓨터와 네트워크 안에 존재하면서 사용자의 작업을 수행하는 소프트웨어 컴포넌트이다. 이러한 에이전트는 이동 에이전트처럼 이동을 하면서 작업을 수행할 수도 있고, 한곳에 고정되어 수시로 필요한 기능을 부여받아 수행할 수도 있다. 제너럴 매직의 오딧세이(Odyssey), IBM의 에이질릿(Aglet)[5], 오브젝트 스페이스의 보이어져(Voyager)[6], 미쯔비시의 콘코디아(Concodia)[7] 등이 자바를 이용하여 만든 에이전트들이고, 자바 이외에도 Tcl[8], Scheme, Python 등 다른 언어를 지원하는 에이전트 기술이 연구되어오고 있으나 자바를 응용한 제품들이 가장 경쟁력 있으며 널리 받아들여지고 있다[9].

이동 에이전트는 복수의 컴퓨터간을 이동할 수 있는 프로그램이다. 보통 프로세스는 코드와 계산 도중의 결과를 유지하는 데이터, 그리고 현재 코드의 어느 부분을 실행하고 있으며 상태는 어떤가 등의 정보를 갖는 실행 컨텍스트 등 3 가지의 정보를 갖고 있다. 이동 에이전트 중에는 코드만 움직이는 이동 코드, 코드와 데이터가 같이 움직이는 협의의 이동 에이전트, 코드와 데이터, 실행 컨텍스트가 모두 움직이는 이동 프로세스(혹은 이동 컴퓨팅) 등으로 크게 구분된다. 앞에서 소개한 에이전트를 포함한 대부분의 에이전트들은 모두 코드와 데이터만 이동하는 협의의 이동 에이전트에 속한다고 할 수 있다.

다양한 종류의 에이전트 기술이 개발됨에 따라 이동 에이전트를 위한 기술의 표준화가 활발히 이루어지고 있다. CORBA(Common Object Request Broker Architecture)를 표준화하고 있는 OMG에서는 MAF라는 이동 에이전트를 위한 표준화 방안을 마련하였다[10]. MAF는 CORBA의 이동 에이전트용 설비로서 규격을 표준화하고 있는데, 이는 이동 에이전트 시스템에 대한 CORBA로의 외부 인터페이스를 중심으로 표준화한 것이다. MAF의 기본 구성은 에이전트, 플레이스(place), 에이전트 시스템, 통신하부구조 등으로 구성된다. 에이전트 시스템은 에이전트의 작성, 해석, 실행, 이동 등을 지원하는 플랫폼이다. 이 에이전트 시스템 내에 논리적인 장소를 나타내는 플레이스가 정의되고, 이 에이전트는 플레이스 중에 존재해서 플레이스 사이를 이동하며 서비스를 실행한다.

플레이스는 에이전트가 동작하는 환경으로서 접근제어 같은 기능을 제공하는 컨텍스트이다. 플레이스는 실행되기를 원하는 방문 에이전트를 위한 엔트리 포인트이며, 에이전트는 특정 위치에 관계없이 일정한 서비스를 플레이스로부터 제공받는다. 플레이스는 마치 에이전트를 위한 운영체제와 같은 것으로서 하나의 에이전트 시스템 내에는 하나 이상의 플레이스가 존재할 수 있다. 그러나 플레이스 그 자체가 에이전트를 실행시킬 수 있는 것은 아니다. 에이전트를 실행시키려면, 에이전트는 시스템 엔진 내부에 존재하여야 한다. 에이전트 시스템은 에이전트를 생성하고, 실행하며, 전송하고, 종료시킬 수 있는 플랫폼이다. 에이전트 시스템 엔진은 자바 가상엔진과 같이, 가상의 것이 아닌 실제적인 것이다. 하나의 호스트에는 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 영역으로 표현된다.

통신하부구조는 에이전트 시스템의 통신을 제공하기 위한 기반으로 이를 통해 RPC(Remote Procedure Call), 명명 서비스, 보안 서비스 등이 제공된다. 그러나 CORBA의 명명 서비스는 이동 에이전트에 대한 지원을 따로 수행하지 않으므로 MAF 전용으로 MAFFinder라고 하는 명명 서비스를 별도로 제공하고 그 인터페이스를 CORBA에 제공하는 구조를 취하고 있다.

FIPA는 IPA(Intelligent Physical Agents) 모델에 기초하여 에이전트 기능 모델이나 에이전트 간, 에이전트와 인간 사이에서의 통신시 필요한 프로토콜의 표준화가 이루어지고 있다. FIPA 표준은 크게 두 부분으로 이루어져 있다[11]. 파트 1은 에이전트 플랫폼과 에이전트 참조 모델을 정의한다. 파트 2는 에이전트의 통신 언어를 정의하였다. 에이전트 참조 모델은 DF(Directory Facilitator), AMS(Agent Management System), ACC(Agent Communication Channel)로 구성된다. DF는 에이전트에게 옐로우페이지의 기능을 제공한다. AMS와 ACC는 에이전트간의 통신을 지원한다. DF,

AMS, ACC 등은 기능을 정의한 집합으로서 하나의 에이전트에 의해 수행될 수도 있고, 각기 다른 에이전트에 의해 수행될 수도 있다.

원래 FIPA 표준의 목적은 다양한 컴퓨팅 환경에서 에이전트간의 통신 표준을 정의함으로써, 응용 에이전트간의 작업을 돕기 위한 표준으로서 출발했다. 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[12].

3. 에이전트 인터페이스 정의

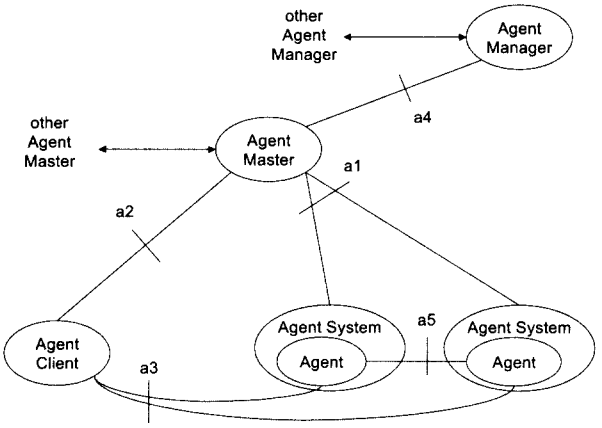
3.1 서비스 컴포넌트

에이전트 서비스를 실행하기 위해서는 에이전트 시스템, 에이전트 매스터, 클라이언트, 에이전트 관리자, 에이전트 등의 서비스 요소가 필요하다. 하나의 에이전트 시스템은 다수의 에이전트를 실행시키고 관리하며, 통신 서비스를 제공한다. 에이전트 시스템은 시작과 함께 자신을 에이전트 매스터에게 등록한다. 에이전트 매스터는 하나의 도메인 내에 있는 다수의 에이전트 시스템을 제어한다. 실행 가능한 에이전트 시스템의 상태와 능력을 항상 유지하면서, 에이전트의 전체 상태정보를 유지하고, 클라이언트와 에이전트 관리자 사이의 중개자 역할을 한다.

클라이언트는 원하는 서비스를 받기 위해 서비스에 필요한 에이전트 시스템을 선택하고, 에이전트 매스터를 통해 필요한 서비스를 요청한다. 에이전트 시스템은 클라이언트가 직접 선택할 수도 있고, 에이전트 매스터가 자동으로 지정할 수 있다. 에이전트 관리자는 에이전트 매스터를 통해 도메인 내의 에이전트의 활동과 상태를 모니터링하고 관리한다. 관리기능은 에이전트 매스터 내에 에이전트 관리를 위한 에이전트 관리정보를 통해 수행한다. 에이전트는 클라이언트의 요구에 의해서 에이전트 시스템에서 생성되고, 필요시 에이전트 시스템 사이를 이동하면서 서비스를 수행하는 개체이다. 서비스를 요청한 클라이언트와의 통신은 에이전트 시스템을 통해 수행하며, 필요 시 다른 에이전트와도 통신을 할 수 있다.

3.2 인터페이스 구조

에이전트의 효과적인 관리를 위해 에이전트 인터페이스를 정의한다. 에이전트의 인터페이스 정의는 교환되는 메시지의 정의로 이루어진다. 인터페이스의 정의는 MAF와 FIPA의 표준을 반영하여 정의한 것으로서 (그림 1)에 이를 보였으며, 각 인터페이스의 메시지는 <표 1>에 정의하였다. 에이전트 마스터는 FIPA의 DF 기능을 수행하고, 에이전트 시스템은 MAF의 플레이스와 에이전트 시스템, FIPA의 AMS 기능을 수행한다. 메시지는 FIPA의 표준 메시지와 유사한 기능을 수행하도록 최소한으로 정의하였다[12].



(그림 1) 에이전트 인터페이스

〈표 1〉 에이전트 인터페이스 메시지

인터페이스	메시지 종 류	메시지 방향	기 능
a1	Register	Agent System → Agent Master	에이전트 등록
	Unregister	Agent System → Agent Master	에이전트 탈퇴
	Service	Agent Master → Agent System	서비스 요구
	Status	Agent System → Agent Master	에이전트 상태보고
a2	GetList	Client → Agent Master	에이전트 정보요구
	List	Agent Master → Client	에이전트 정보보고
	Service	Client → Agent Master	서비스 요구
	Status	Agent Master → Client	서비스 상태보고
a3	Status	Agent → Client	상태 보고
	Service	Client → Agent	서비스 요구
a4	Get	Agent Manager → Agent Master	관리정보 요청
	GetNext	Agent Manager → Agent Master	다음관리정보 요청
	Set	Agent Manager → Agent Master	관리정보 설정
	Response	Agent Master → Agent Manager	관리정보 응답
	Trap	Agent Master → Agent Manager	이벤트 통지
a5	Inform	Agent → Agent	정보전달

3.3 인터페이스 메시지

a1 인터페이스는 에이전트 매스터와 에이전트 시스템간의 인터페이스이다. Register 메시지는 에이전트 시스템이 서비스의 시작을 알리기 위해 에이전트 매스터로 보내는 메시지이다. 이 메시지에는 에이전트 시스템 등록을 위한 에이전트 시스템의 식별자, 플레이스 식별자 등의 파라미터 값이 포함된다. Unregister 메시지는 에이전트 시스템이 에이

전트 서비스의 종료를 통보하는 메시지이다. 에이전트 시스템이 Unregister 메시지를 보내지 않고 서비스를 종료할 수도 있다. 이러한 서비스의 이상 종료는 에이전트 매스터가 Service 메시지를 보내어 에이전트 시스템이 Status 메시지로 응답하지 않을 때 검출될 수 있다. 클라이언트의 서비스 요구에 의한 에이전트 서비스의 요청은 Service 메시지에 의해 시작된다.

a2 인터페이스는 클라이언트와 에이전트 매스터간의 인터페이스이다. 클라이언트가 서비스의 요구에 필요한 정보와 함께 GetList 메시지를 에이전트 매스터에게 보내면, 에이전트 매스터는 이용자의 요구에 맞는 에이전트 시스템의 정보를 List 메시지와 함께 클라이언트로 보낸다. 클라이언트의 서비스 요구에 의한 에이전트 서비스의 요청은 Service 메시지에 의해 시작된다. 서비스 메시지에는 서비스의 이름, 프로그램 정보, 실행할 에이전트 시스템 정보 등이 포함된다. 클라이언트는 서비스의 상태를 Status 메시지를 통해 확인할 수 있다.

a3 인터페이스는 에이전트와 클라이언트간의 인터페이스이다. 이 인터페이스를 통해서 서비스의 상태와 결과 등을 교환한다. 보통 에이전트는 서비스중인 상태 정보를 에이전트 매스터를 통해 알린다. 클라이언트가 요구하였고 에이전트 시스템이 이를 허용하는 경우, 서비스의 중간 및 최종 결과 값은 클라이언트에게 직접 전달한다.

a4 인터페이스는 에이전트 관리자와 에이전트 매스터간의 인터페이스이다. 인터페이스를 흐르는 메시지는 관리를 위한 프로토콜인 SNMP 메시지이다. SNMP 메시지는 Get, GetNext, Set, Response, Trap 등의 다섯 개의 메시지로 구성된다. 메시지 내의 정보는 에이전트 관리를 위한 MIB (Management Information Base) 정보이다. 에이전트 관리자는 원하는 에이전트 시스템의 필요한 정보에 해당하는 MIB의 식별자와 값을 SNMP 프로토콜을 통해 직접 에이전트 시스템을 통하지 않고 매스터와 교환한다. 이는 에이전트 시스템에 망 관리 에이전트를 설치하지 않아 에이전트 시스템의 복잡성을 줄이고 에이전트 관리를 효율적으로 하기 위함이다. 중요한 에이전트 시스템의 경우 망 관리 에이전트를 설치하여 에이전트 관리자가 직접 에이전트 관리를 수행할 수 있다. 이러한 방안에서는 에이전트 및 에이전트 시스템이 대한 직접적이고 세밀한 제어 및 관리가 가능하다. a4 인터페이스에서 교환되는 에이전트 관리 MIB은 전체 MIB 트리에서 iso(1).org(3).dod(6).internet(1).private(4).enterprise(1).snut(12012).network(1).agent(1)의 하부에 위치하며 <표 2>와 같이 정의하였다[13].

a5 인터페이스는 에이전트간의 인터페이스이다. 에이전트간의 통신이 필요할 때 a5 인터페이스를 통해 바로 메시지 교환을 이룰 수 있다. 에이전트가 다른 에이전트의 위치 정보를 모르고, 에이전트는 수시로 이동할 수 있으므로 에이전트간의 정보의 교환은 에이전트 매스터를 통해 이루어지는 것이 보통이다.

〈표 2〉 에이전트 관리 MIB

에이전트 관리정보 그룹	에이전트 관리 정보	내 용
Entity	agentMasterSystem	에이전트 매스터 시스템 정보
Cnfiguration	authentication, codeServer, neighborAgentMasterSystem	환경 설정 정보
AgentList	agentSystem, place, agent	에이전트시스템과 에이전트 정보

4. 에이전트 응용 서비스 관리 시스템 구현

4.1 에이전트 응용 서비스의 전개 및 관리

에이전트 서비스를 통해 응용 서비스를 제공하는 시스템의 구성은 에이전트 매스터, 에이전트 시스템, 에이전트 관리자, 에이전트, 클라이언트로 구성된다.

에이전트 매스터는 지역 에이전트 시스템을 제어하고, 서비스 이용자인 클라이언트에게 에이전트 서비스를 제공하는 것을 중재하는 시스템이다. 또한, 에이전트 관리자에게 에이전트의 관리 인터페이스를 제공한다. 에이전트 관리자는 직접 에이전트를 관리하지 않고 에이전트 매스터를 통해 관리함으로써 에이전트 시스템의 복잡성을 덜고, 유연성과 확장성, 관리의 효율성을 이룰 수 있다. 에이전트 매스터에는 에이전트 관리에 필요한 에이전트 관리 정보가 구현되어 있다. 에이전트 관리자는 이 MIB을 읽어 서비스 모니터링을 수행하고, 필요한 값을 설정하여 에이전트 서비스 관리 기능을 수행한다.

클라이언트는 에이전트 매스터를 통해 필요한 서비스를 시작한다. 클라이언트가 요구한 서비스의 수행 코드는 클라이언트가 직접 제공할 수도 있고, 클라이언트가 서비스 저장소의 주소만을 제공하고 에이전트 시스템이 이를 직접 가져갈 수도 있다. 수행 코드를 메시지 내에 포함시켜 보내는 경우는 메시지의 크기가 커지고, 각각의 클라이언트가 서비스 코드를 관리해야 한다는 단점이 있다. 에이전트 코드를 다른 저장소에 위치시키고 위치 값을 전달하는 방법은 서비스 코드의 관리를 용이하게 해주고, 인증 받은 서버를 통해 코드의 안전성을 보장해줄 수 있다. 클라이언트는 서비스를 수행할 에이전트 시스템을 직접 지정하거나, 또는 서비스의 수행에 적합한 에이전트를 에이전트 매스터가 선택하도록 할 수도 있다. P2P 응용의 경우 에이전트 시스템과 클라이언트는 하나의 프로그램으로 묶여서 돌아갈 수 있다. 에이전트를 이용한 서비스는 다음과 같은 순서에 의해 동작한다.

① 에이전트 시스템 등록 및 발행

초기에 에이전트 매스터는 서비스를 시작하고 에이전트 시스템이 등록하기를 기다린다. 에이전트 시스템은 자신의 서비스 포트를 열고, 에이전트 매스터에게 등록한다. 등록 시 에이전트 시스템은 서비스 프로파일을 제출하는데, 이 서비스 프로파일에는 통신방식, 네트워크 통신 기능 제공여부, 파일 및 DB 서비스 제공여부, 스레드 수와 최대 할당 메모리 크기 등과 같은 에이전트의 시스템 구성과 능력에 대한 정보를 포함한다.

② 에이전트 시스템 검색

클라이언트는 에이전트 서비스를 이용하기 위하여 에이전트 매스터에 접속하여 가능한 서비스와 가능한 에이전트 시스템을 조사한다. 이러한 검사는 에이전트 시스템의 프로파일을 통해 이루어진다. 에이전트 시스템의 위치와 제공 서비스 수준, 비용 등이 검토된다.

③ 서비스 호출

클라이언트는 에이전트 매스터를 통해 필요한 서비스를 요청한다. 서비스가 실행될 에이전트 시스템을 직접 지정할 수도 있고, 에이전트 매스터에게 일임할 수도 있다. 이후 에이전트의 상태를 보고 받고, 에이전트로부터 보고되는 결과를 분석한다.

④ 에이전트 관리자는 망 내의 에이전트 서비스와 에이전트 매스터를 관리 MIB을 통해 관리한다. 서비스는 관리자에 의해 중단 혹은 재시작 될 수 있고, 매스터의 환경 설정 정보가 관리자에 의해 구성된다.

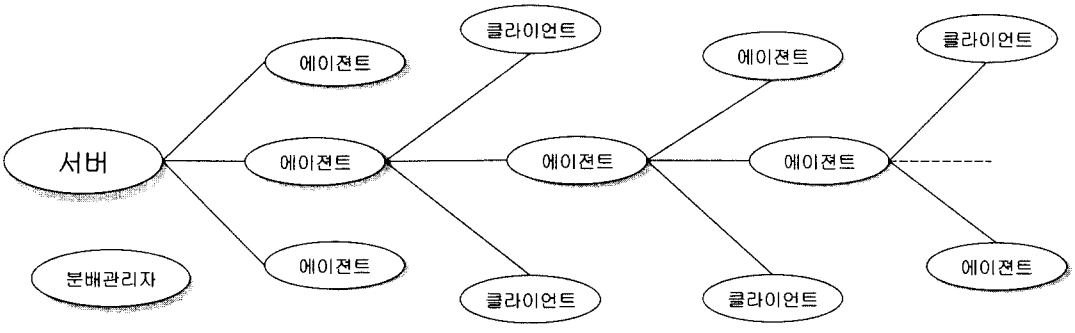
4.2 에이전트 서비스 응용 및 관리 사례

에이전트를 이용한 응용으로 정보를 제공하는 서비스 망을 구현하고 이를 모니터링 하였다. 에이전트를 이용한 컨텐츠 분배 망 서비스를 구현하기 위하여 컨텐츠 분배 서버, 컨텐츠 분배 에이전트, 컨텐츠 분배 관리자, 클라이언트 등의 서비스 컴포넌트를 정의한다. 이러한 구조는 클라이언트/서버 구조에 바탕을 두고 실현되며, 컨텐츠 분배 에이전트와 분배 관리자를 통해 동적인 서비스 망의 구현을 이룬다. 중앙 서버들이 모든 서비스를 담당하는 대신 동적으로 구성되는 지역의 서버가 이를 분배하고 관리하는 구조를 통해 품질 제어와 서비스의 효율을 얻고자 하는 것이다[14, 15]. 컨텐츠 분배 망은 트리 구조를 이루고, 그 정점에 컨텐츠의 원천 서버가 있다. 서버에 다수개의 동적인 에이전트가 연결되어 일차 하위 분배 망을 구성하고, 그 하부에 또 다른 에이전트들이 연결되어 망이 확장된다. 클라이언트는 분배관리자의 설정에 따라 자기의 위치에서 가장 가깝거나 성능에 유리한 에이전트에 연결되어 종단 멤버로 연결되고, 연결된 에이전트를 통해 서버의 컨텐츠 서비스를 제공받는다.

(그림 2)에서 컨텐츠 서비스 분배 망 트리는 분배 관리자가 관리한다. 분배 관리자는 (그림 1)에서의 에이전트 클라이언트의 한 종류이다. 에이전트 매스터를 통해 에이전트 시스템의 프로파일을 검색하여 필요한 에이전트 시스템에 서비스 에이전트를 시작시킨다. 에이전트가 새로 시작하면 분배 관리자에게 메시지를 전달하여 자신을 등록한다. 분배 관리자는 에이전트의 위치에 따라 신설 에이전트를 분배 트리 내의 적절한 위치에 위치시키도록 한다. 이후 신설 에이전트 주변의 컨텐츠 클라이언트가 컨텐츠 서비스를 요구하면 분배관리자는 적절한 에이전트에게 연결시켜 서비스를 제공받도록 해 준다. 컨텐츠 서비스 모델에서의 각 컴포넌트의 기능은 다음과 같다.

① 컨텐츠 서버

컨텐츠 서버는 정보를 제공하는 원천 서버이다. 제공 서비스는 방송, VOD (Video On Demand), 화상회의, 채팅, 주식정보 서비스, 실시간 뉴스 서비스 등 다양하다. 서비스를 제공하는 방식은 스트림을 이용한 방식, UDP(User Datagram Protocol) 데이터그램을 이용한 서비스, RTP (Realtime Transfer Protocol) 실시간 정보 서비스 등 다양한 형태로 존재한다. 또한, 서비스 제공 방식에 있어 클라이언트가 정보를 끌어오는 푸시형 서버, 서버가 정보를 제공해주는 풀형 서버 등 서비스의 형태에 따라 다양한 서버가 존재한다. 서버는 클라이언트에 대한 목록을 가지고 서비스한다. 서비스 분배 목록은 컨텐츠 분배 관리자에 의해 제공받는다.



(그림 2) 컨텐츠 분배 망 트리

② 컨텐츠 분배 에이전트

분배 에이전트는 컨텐츠 서버로부터 컨텐츠를 받아와서 클라이언트나 또 다른 분배 에이전트에게 다시 컨텐츠를 분배한다. 분배 에이전트는 모든 컨텐츠에 대하여 같은 형태가 아니라 서비스되는 컨텐츠의 유형에 따라 다른 형태를 갖는다. 따라서, 에이전트는 서비스를 제공하는 측에서 에이전트를 원하는 위치에 위치시키고, 특정 서비스와 서버의 형태에 따라 고유의 형태로 서비스된다. 에이전트 또한 서버와 같이 클라이언트에 대한 목록을 가지고 서비스한다. 서비스의 품질과 형태에 따라 한 개 혹은 여러 개의 스레드가 분담하여 일을 처리한다. 서비스 분배 목록은 컨텐츠 분배 관리자에 의해 제공받는다. 서비스를 제공하면서 서비스 품질을 모니터링 한다.

③ 컨텐츠 분배 관리자

컨텐츠 분배 관리자는 서버와 클라이언트 사이의 데이터 서비스를 제어하는 컴포넌트이다. 컨텐츠 분배 관리자는 서버와 모든 분배 에이전트, 그리고 서비스를 제공받는 모든 클라이언트에 대한 정보를 갖고, 분배 트리를 조정하여 제공되는 서비스의 형태를 관리한다. 또한 서비스 품질을 모니터링하고 보고되는 서비스 품질을 통해 새로운 에이전트의 위치와 목록을 준비하고, 분배 트리를 재조정한다. 이 시스템은 에이전트 인터페이스에서 클라이언트 기능을 수행하는 컴포넌트에 해당한다.

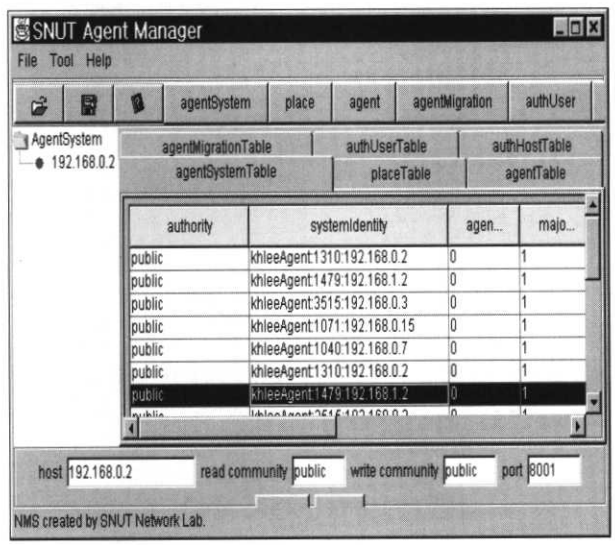
④ 컨텐츠 클라이언트

클라이언트는 에이전트 분배자를 통해 컨텐츠를 제공받고, 이를 이용자에게 적당한 형태로 제공한다. 보통 클라이언트 시스템은 통신 담당 처리 부분과 정보 처리 담당 부분t으로 구성된다.

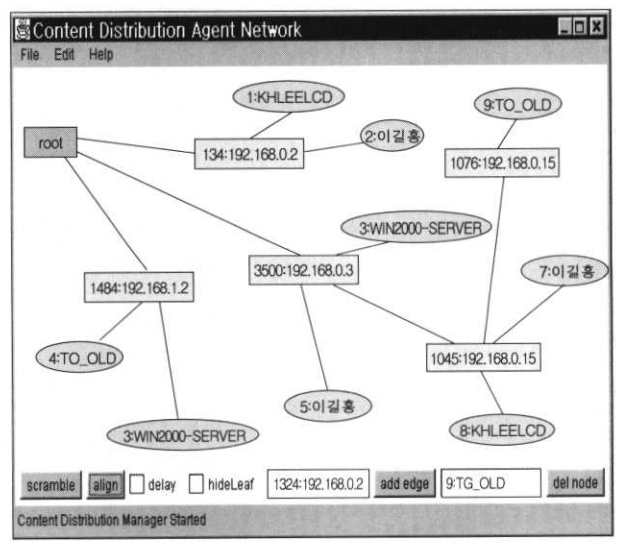
에이전트 매스터는 포트 12012를 열고 에이전트 시스템의 등록을 기다린다. 에이전트 시스템이 시작하면 에이전트 매스터에게 Register 메시지를 통해 등록하고 에이전트 시스템의 서비스를 시작한다. 등록한 이후, 일정 간격으로 에이전트 매스터에게 Status 메시지로 자신의 서비스 정보를 알리는 메시지를 보낸다. 에이전트 매스터가 Service메시지를 통해 에이전트 시스템의 서비스 상태정보를 요구하면, 에이전트 매스터가 Status 메시지로 응답한다.

(그림 3)은 에이전트 서비스를 제공하는 측에서의 에이전트 관리자 프로그램 화면이다. 에이전트 관리자를 통해 현재 활동중인 에이전트 시스템과 서비스중인 에이전트를 모니터링하고 이의 생명주기를 제어할 수 있다. 에이전트 관리자는 에이전트 매스터를 통해 에이전트 시스템과 에이전트들을 모니터링하고 이들의 동작 방식을 관리한다.

(그림 4)는 컨텐츠 분배 망을 에이전트 클라이언트가 에이전트 서비스를 이용하여 구성하고 이를 제어하는 클라이언트 프로그램인 컨텐츠 분배 관리자의 실행화면이다. 컨텐츠 분배 관리자는 컨텐츠 고객에게 양질의 컨텐츠 서비스를 위해 에이전트를 이용한 분배망을 구축하고 제어하는 기능을 갖는다. 컨텐츠 분배 관리자는 에이전트 매스터에게 접속하여 GetList 메시지를 통해 에이전트 시스템의 정보를 요구하면, 에이전트 매스터는 List 메시지로 에이전트 서비스 정보를 리턴한다. 컨텐츠 분배 관리자는 Service 메시지에 이중의 한 에이전트를 선택하거나, 원하는 에이전트에 관한 정보와 서비스할 에이전트 프로그램의 정보를 넣어 에이전트 매스터로 보내 서비스 시작 요청을 보낸다.



(그림 3) 에이전트 제어 및 관리 화면



(그림 4) 컨텐츠 분배 망 제어 화면

에이전트 매스터는 클라이언트의 메시지를 받아 서비스가 실행될 에이전트 시스템으로 Service 메시지를 보낸다. 에이전트 시스템은 Service 메시지 내의 정보를 이용하여 에

이전트 서비스 코드를 얻고, 에이전트를 생성시켜 서비스를 시작한다. 서비스를 실행중인 에이전트는 에이전트 시스템을 통해 Status 메시지로 클라이언트에게 서비스 실행정보를 보내고, 클라이언트는 Service 메시지를 에이전트에게 보내 서비스 실행을 실시간 제어한다. Service 메시지를 받은 에이전트는 명령을 수행하고 Status 메시지로 응답한다. Service와 Status 메시지 내의 구체적인 내용은 컨텐츠 관리자의 설계에 따른다.

컨텐츠 분배 관리자는 컨텐츠를 중계하는 망을 에이전트를 이용하여 구축하고, 컨텐츠 고객의 서비스 요구에 인접한 에이전트의 URL 정보를 갖는 Service 메시지를 리턴하여 고객이 자신에게서 유리한 에이전트에게 접속할 수 있도록 한다.

컨텐츠 분배 망의 구축은 에이전트 클라이언트인 컨텐츠 분배 관리자가 수행하고, 망 내의 모든 에이전트의 행동에 대한 제어는 에이전트 관리자가 수행한다. 에이전트 관리자는 에이전트 매스터를 통해 망 내의 모든 에이전트 시스템과 에이전트에 관한 정보를 수집한다. 에이전트 관리자가 GetRequest 메시지를 통해 에이전트 매스터에게 필요한 정보가 있는 관리 MIB을 요구하면, 에이전트 매스터는 요청받은 관리정보를 GetResponse 메시지로 알린다. 새로운 에이전트 시스템의 시작이나 새로운 에이전트의 생성 등과 같은 사건이 발생하면 에이전트 매스터가 에이전트 관리자에게 Trap 메시지로 이를 알린다.

기존에 에이전트를 관리하는 방안은 [4]에서 제시하는 방안 밖에 아직 나와있지 않은 상태이다. [4]에서 제시하는 방안은 에이전트 시스템이 설치되어 있는 곳에는 이미 SNMP 에이전트가 설치되어 있다는 가정 하에 이루어지는 것이다. 에이전트 시스템마다 SNMP 에이전트를 두는 것은 시스템의 입장에서 부담이 될수 있는 방안이다. 또한, 에이전트가 SNMP 에이전트가 설치되지 않은 에이전트 시스템으로 설치되거나 이동하는 경우에는 이를 제어할 수 있는 장치가 없다. 본 고에서는 다수의 에이전트를 효과적으로 관리하기 위하여 에이전트 매스터를 통해 관리하는 방안을 제시하고 구현을 통해 그 가능성을 실험하였다. 실험 결과 에이전트 매스터를 통해 효과적으로 에이전트의 생성과 서비스의 전개 과정을 관리할 수 있음을 확인하였다.

5. 결론 및 향후과제

다양한 기능을 수행하는 활발한 에이전트 개발의 성공은 인터페이스의 표준화에 달려 있다. 따라서, 에이전트의 효과적인 전개와 관리를 위한 인터페이스 표준을 정의하고, 인터페이스에 필요한 메시지를 정의하였다. 에이전트가 실행되기 위해서는 에이전트 시스템, 에이전트 매스터, 클라이언트, 에이전트 관리자 등의 컴포넌트가 필요하다. 이들 컴포넌트간의 인터페이스를 통한 메시지의 교환을 통해 에이전트 시스템이 생성되고, 이동하면서 필요한 서비스가 실현될 수 있음을 구현을 통해 알아보았다.

에이전트의 효율적인 관리를 위해서 에이전트 매스터를 통해 에이전트 서비스가 이루어질 뿐만 아니라, 에이전트 매스터를 통해 에이전트의 관리도 수행될 수 있음을 보였다. 이러한 구조는 에이전트 시스템을 간략화 시키고, 많은 수의 에이전트를 관리할 수 있는 매우 효율적인 구조이다.

정의된 인터페이스와 메시지는 에이전트를 생성하고 서비스를 이루는 작업에 필요한 최소한의 정의에 해당한다. 이를 토대로 복잡한 서비스의 실행과 에이전트의 세밀한 제어와 관리를 위한 추가적인 인터페이스와 메시지의 정의를 구현할 것이다. 에이전트 매스터간의 정보교환과 이를 이용하는 에이전트의 기능을 실현하는 구조 및 구현 방안을 제시 할 것이다. 또한, 어떠한 응용이 에이전트를 통해 효율적으로 제공될 수 있는지도 계속 검토될 것이다.

참 고 문 헌

[1] Green Shaw, Hurist L., Nangle, B., Cunninggham, P. Somars, F., Evans, R., Software Agents : A review, http://www.cs.tcd.ie/research_groups/aig/iag/toplevel2.html.

[2] Alex L. G. Hayzelden and Rachel A. Bourne, Agent Technology for Communication Infrastructures, Weley, 2001.

[3] Antonio Liotta, George pavlou, and Graham Knight, Exploiting Agent Mobility for Large-Scale Network Monitoring, IEEE Network, Vol.16, No.3, pp.7-15, May/June, 2002.

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

[5] Danny B. Lange and Mitsuru Oshima, "Mobile Agents with Java : The Aglet API," World Wide Web Journal, 1998.

[6] http://www.recursionsw.com/products/voyager/voyager.asp.

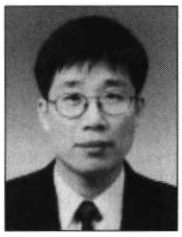
[7] http://www.merl.com/projects/concordia.

[8] R. S. Gray, "Agent Tcl : A Flexible and Secure Mobile-Agent System," Proceeding of the Fourth Annual Tcl/Tk Workshop(TCL '96), 1996.

[9] Danny B. Lange and mitsuru Oshima, Programming and Deploying Java Mobile Agent with Aglets, Addison Wesley, 1998.

[10] GMD FOKUS and IBM, Mobile Agent Facility Specification V1.0, Jan., 2000.

[11] http://www.fipa.org.
[12] XC00023H, FIPA Agent Management Specification, 2001.
[13] 이길홍, "에이전트 관리 방안", 한국컴퓨터산업학회논문지, 제3권 제2호, pp.191-198, 2002.
[14] Hofmann, M., Sabnani, K, Streaming and Broadcasting over the Internet, Proceedings of the IEEE Conference on ATM 2000, pp.251-256, June, 2000.
[15] Damien Stolarz, Peer-to-Peer Streaming Media Dlivery, Proceedings of the First International Conference on Peer-to-Peer Computing, pp.48-52, Aug., 2001.



이 길 흥
e-mail : khlee@snut.ac.kr
1989년 연세대학교 전자공학과(공학사)
1991년 연세대학교 대학원 전자공학과 (공학석사)
1991년~1995년 LG 정보통신 안양연구소 네트워크그룹 주임연구원
1999년 연세대학교 대학원 전기컴퓨터공학과(공학박사)
2000년~현재 서울산업대학교 컴퓨터공학과 조교수
관심분야 : 초고속망, 망관리, 에이전트 응용