情報管理研究 vol. 31, no. 4, 2000, pp. 71-87

에이전트의 시대가 오고 있다

Advence of Agent Age

이근상*

Keun-Sang Lee

= 초 록 =

최근 네트워크로 연결된 이기종의 유용성을 높이고 기존의 분산객체 컴퓨팅의 문제해결을 위해 이동 에이전트(Mobile agent)의 연구가 활발히 이루어지고 있다. 이런 연구를 통해 기존의 분산시스템뿐만 아니라 전자상거래, 네트워크관리, 정보검색 등 많은 분야에 이동 에이전트를 적용시키는 연구가 진행되고 있으며, 많은 시스템이 개발되고 있다. 본고에서는 에이전트의 연구에 있어 관련된 사항을 알아보고 에이전트간 통신기능향상을 위해 연구한 내용을 바탕으로 전반적인 내용을 다루고자 하며, 마지막으로 에이전트의 미래와 적용분야에 대해 다루고자 한다.

= 키워드 =

이동 에이전트, 네트워크, 로봇에이전트, 소프트웨어 에이전트, 분산객체컴퓨팅, 통신기법, 공유메시지 저장소

ABSTRACT

Recently, researches of mobile agent systems have done actively to enhance usability of heterogenenous environment linked via network and to solve problems of existing distributed-object computing. Through these research and developments, many studies have done to be applicated to many areas that are existing distributed systems as well as E-commerce, network maintenance, and information retrieval etc.

This paper reviews some related issues to agent studies, comprehensive studies

* 다산씨앤아이 E-BIZ 사업부 책임연구원/실장
(DASAN C&I E-BIZ Dept., Senior R&D Engineer)

to enhance telecommunication functionality among agents, and future and application fields of agent.

KEYWORDS

Mobile Agent, Network, Robot Agent, Software Agent, Distributed-Object Computing, Telecommunication Method, Common Message Pool

1. 에이전트의 소개

대기업 전산실에 근무하는 권씨는 아침 출근을 위해 지하철에 올랐다. 순간 모바일컴퓨팅 기능을 탑재한 PDA(Personal Digital Assistant)에서 비프음이 울렸다. PDA의 액정에 나타난 내용은 [LA행 항공권 구입 성공] 이란 내용이었다. 이틀전에 에이전트에게 구입을 부탁한 내용이었다. 잠시 후 또 다른 비프음과 함께 [거래 성사 39만원 낙찰] 이라는 또 다른 에이전트의 메시지가 나타났다. 지하철에서 내려 회사에 출근한 권씨는 느긋한 마음으로 컴퓨터에 앉아 서류 작업을 위해 워드프로세서 아이콘을 클릭하였다. 암호를 입력하고 수 초정도 시간이 흐르고 화면에는 마이크소프트에서 제공하는 윈도우즈 화면과 백두에서 제공하는 사전과 한봉시스템의 글자 폰트들이 로드되기 시작했다. 로드가 완료된 후 권씨는 서류작업을 끝내고 작성한 문서는 인터넷을 통해 웹하드에 저장되었다. 권씨의 컴퓨터는 저장 스토로지가 없는 NC(Network Computer)이며 같은 회사원 전체가 1년 전부터 사용하고 있다. 또한 원하는 정보를 찾기위해 검색사이트를 이용해 검색된 결과를 하나 하나 클릭해보기 보다는 권씨는 필요한 정보만 주면 알아서 찾아오고 정리하여 우선순위에 따라 보여주기까지 하는 로봇 에이전트를 이용해 정보를 얻고 있다. 이런 시나리오는 앞으로 10년 뒤가 아니라 지금 이 순간에 사용되고 있다. 바로 에이전트가 있기에 가능한 것이다. 에이전트의 활용을 설명하기 위해 아주 간단한 시나리오를 만들어 보았는데, 그럼 소프트웨어 에이전트, 즉 에이전트는 무엇인가?

1.1 에이전트는 무엇인가?

일반적으로 에이전트의 정의는 의뢰인(client)이 특정한 작업을 요청하고, 요청한 작업을 대행해주는 사람을 일컫는다. 즉 누군가를 대신해서 원하는 작업을 대행해 주는 사람을 말한다. 이렇게 어떤 일은 대신해 주는 사람의 기능을 소프트웨어적으로 구현한 것을 소프트웨어 에이전트라

고 한다. 전산학적으로 보통 에이전트(Agent)라고 부르며 본 지에서도 에이전트라고 부르겠다. 그러면 에이전트의 정의는 무엇일까? 간단하게 말하면 특정한 환경하에서 그 환경의 일부로서 환경을 인지하면서, 독자적으로 어떠한 목적을 수행하기 위해 사용자를 대신하여 지속적으로 환경과 연동하며 작업을 수행하는 자율적인 시스템을 말한다. 여기서 환경이란 운영체제나, 네트워크, 기타 프로세스가 동작을 할 수 있는 환경을 말한다. 다음은 에이전트에 대해 좀더 정확히 정의한 내용이다.

자율적인 에이전트(autonomous agent)는 어떤 복잡한 동적 환경에 살며 이 환경을 자율적으로 감지하면서 동작하는 컴퓨터 시스템이며 그렇게함으로써 그들이 고안되게 됐던 목적이나 일들을 수행한다. - MIT 미디어 연구실

또한 에이전트가 일반적으로 운영체제에서 동작하고 있는 다른 프로그램과는 어떻게 다른가 ? 그것은 바로 자율성(autonomy) 및 지능(intelligence)으로 에이전트와 일반 소프트웨어를 구별해 주는 가장 큰 특징이다. 특히 이동 에이전트(Mobile Agent)는 이동의 자율성을 가지고 사용자의 지시나, 혹은 부가적인 요구없이도 스스로 목적을 달성하기 위해 일련의 작업을 수행한다. 즉, 사용자의 요구를 달성하기 위해 목적지향(goal-oriented)과 자율적 활동(self-starting & autonomous mobility) 능력이 포함된다고 할 수 있다. 에이전트의 인공지능은 자율성과 유기적인 결합을 통해 이동 에이전트의 특성이라고 할 수 있다. 이를 위해 에이전트는 지식베이스(knowledge base)를 채택하고 있으며 추가적으로 추론(reasoning) 및 계획(planning) 능력을 포함하고 있다. 또한 사용자로부터 혹은 스스로 목적에 맞게 작업을 수행하면서 학습(learning) 능력을 가지고 후에 더 나은 작업수행을 위하여 지식베이스를 늘려 나가게 된다. 이것은 인간이 경험을 축적해가는 것과 같다고 볼 수 있는데 반드시 자율성과 지능성을 동시에 가져야 하는 것은 아니다. 또한 다른 에이전트와의 통신을 통한 정보교환 등으로 협동 작업을 통해 서로의 문제을 해결하는 사회성(social ability) 등의 특성을 가지는 에이전트도 있다. 즉 위의 특성들을 모두 갖추고 있어야 에이전트라고 말하는 것이 아니라 위의 특성들 중 어느 것이라도 지니고 있으면 에이전트라고 부를 수 있다.

1.2 그렇다면 에이전트는 새로운 개념인가?

에이전트는 예전부터 인공지능을 이용해 사용자의 편의를 도모하고자 연구되어져 왔다. 그러다 인공지능의 연구가 잠시 주춤할 무렵 분산처리와

에이전트의 통신개념이 등장하면서 인공지능과는 학문적으로 분리되어 독자적인 학문으로 분리되어 더욱 발전되고 있다. 최근들어서는 인공지능을 가지고 이동의 자율성까지 포함한 지능형 이동 에이전트의 연구가 활발히 진행되고 있다. 또한 에이전트를 이용한 네트워크망관리부터 컴포넌트 소프트웨어 개념까지 모두 에이전트 개념을 이용하고 있는 추세이다.

2. 에이전트의 종류

에이전트는 어떤 종류가 있나? 에이전트의 분류는 일반적으로 로봇 에이전트와 소프트웨어 에이전트로 나눌 수 있다.

2.1 로봇 에이전트

로봇 에이전트란 인터넷을 통해 웹을 서핑하며 각 HTML 페이지들의 정보를 수집하는 프로그램이다. 즉 웹서버에 접속하여 HTML 화일을 읽어온다. 또는 가져온다는 표현이 더 어울린다. 가져온 HTML을 분석하고 정의된 작업 내용에 따라 필요한 부분을 추출하여 데이터베이스화 하는 작업을 한다. 또한 이렇게 모아진 정보를 이용해 다른 사용자가 손쉽게 정보와 관련된 URL로 접근을 용이하게 해주도록 하는 것이다. 로봇 에이전트가 혼자 능동적으로 HTML페이지를 찾아다니며 필요한 정보를 분류함으로 예를 든 권씨와 같이 귀찮은 일을 하지 않아도 되고 엄청난 시간을 절약할 수 있게 된다. 또한 홈페이지와 링크된 수많은 내용이나 HTML 안에 수많은 그림파일들을 하나씩 저장하는 과정이 필요없어진다. 또는 쇼핑몰 가격을 비교해 주는 사이트의 관리자는 매 시간마다 링크를 하나하나 접속해 보면서 끊겨진 링크를 일일이 찾아서 고칠 필요가 없어진다. 로봇 에이전트는 이와 같이 웹페이지를 일일이 돌아다니면서 할 수 있는 다양한 일들을 자동적으로 해 주는 가장 강력한 프로그램이다.

로봇 에이전트는 다음과 같은 일을 할 수 있다.

(1) 통계분석(Statistical Analysis)

로봇 에이전트의 최초의 사용 목적은 웹서버의 수를 파악하기 위해 처음 사용되었다. 1993년 MIT의 Mattew Gray는 "World-Wide Web Wanderer" 로봇 에이전트를 이용하여 전세계에 몇 개의 웹서버가 있는지 알아보기 시작하였다. Matthew Gray 뿐만 아니라 워싱턴 대학의 Brian Pinkerton도 웹서버의 수를 조사하였다. 이후에 통계 분석용 프로그램을 서치엔진용 로봇 에이전트로 바꾸어 현재 WebCrawler라는 서치엔진을 운영중이기도 하다. 또한 웹서버의 수만 계산하는 로봇 에이전트와

어떤 웹서버를 사용하는지 조사하는 로봇 에이전트도 운용되고 있다. 이렇게 로봇 에이전트를 이용하여 웹서버를 발견하고 서버의 수를 세고 서버가 가지고 있는 문서의 평균 페이지수 등의 통계 조사를 수행할 수 있다.

(2) 유지보수(Maintenance)

웹서버 유지 보수의 어려움은 관리자가 아니면 모를 정도로 엄청난 코스트 및 부하를 주게 된다. 개개인이 가지고 있는 파일과 어디에 데드링크(dead links)가 발생하였는지 웹서버 관리자는 모르게 된다. 데드링크란 원래의 URL이 변경되거나 없어져 버림으로써 기존의 사용자가 알고 있는 URL로써 접근이 불가능한 경우를 말한다. 이때 유용하게 사용할 수 있는것이 로봇 에이전트이다. 주변에서 쉽게 사용해 볼 수 있는 로봇 에이전트가 있는데 네스케이프 경우 브라우저 안에 로봇 에이전트를 내장하고 있어 데드링크에 대한 통지를 받을 수 있다.

2.2 소프트웨어 에이전트

일반적으로 소프트웨어 에이전트는 컴퓨터 환경에서 사용자를 위해 필요한 작업을 수행하는 소프트웨어 프로그램으로 앞서 제시한 내용에 부합되는 능동적인 프로세서를 말한다. 또한 대부분의 4세대 소프트웨어 어플리케이션이 소프트웨어 에이전트로 정의될 수 있다. 좀더 세분화하면 다음과 같이 분류할 수 있다.

(1) 다중 에이전트 시스템

분산 환경에서 상호 협력을 통해 작업을 수행하는 에이전트로 에이전트간의 대화기능을 가지고 에이전트간의 협동을 통해 작업을 수행해 나간다.

(2) 이동 에이전트

보통 네트워크 에이전트 또는 순회 에이전트라고도 하며 프로그램 자체가 네트워크를 이동하면서 수행되는 에이전트이다. 이 에이전트는 이동 에이전트가 도착할 곳에 인증 및 활동에 필요한 환경을 제공해 주는 서버가 필요하다. 이를 에이전트시스템이라고 부른다. 또한 보통 자바 애플릿을 생각할 수 있는데 애플릿은 사용자가 홈페이지를 방문한 결과로 실행되기 때문에 자율성을 가지는 이동 에이전트라고 말하지는 않는다.

(3) 사용자 인터페이스 에이전트

사용자가 컴퓨터를 사용하기 편하도록 지원하는 에이전트로 음성인식, 자연어 처리, 음성합성 등의 사용자 인터페이스 기술과 인공지능 기술이 융합된 종합응용 분야에 이용되고 있다.

(4) 지능형 에이전트

학습능력이나 추론능력, 계획능력 같은 지능적인 특성을 갖는 에이전트로 사용자가 원하는 작업에 대해 기존 처리 방법이나 다른 시스템에 있는 에이전트의 경험과 지식을 바탕으로 작업처리방법을 파악하고 그에 따라 문제를 해결하는 추론 에이전트와 여러 에이전트가 협력해 하나의 작업을 처리하기 전에 에이전트간의 통신과 에이전트의 작업수행을 어떤 방식으로 진행할 것인가에 대해 미리 계획하고 그 계획에 따라 통신 및 작업을 수행해 나가는 계획 에이전트가 있다.

이런 모든 종류의 에이전트는 최근 들어서 에이전트간의 통신을 이용해 협동작업 개념으로 발전하고 있다. 이는 분산처리의 단점을 해결하는 방법으로 가장 활발히 연구되는 분야이기도 하다. 또 이질적 에이전트를 통합관리하는 에이전트 분야도 많은 연구가 진행중이다.

3. 에이전트간 통신

앞서 제시한 여러 에이전트의 분산협동처리를 위해서는 에이전트간 통신이 필수적이다. 그 목적은 정보나 작업처리의 공유와 교환에 있다. 이것은 이동 에이전트 뿐만아니라 일반적인 에이전트간 통신에 있어서 가장 큰 문제점은 각 에이전트가 가지고 있는 이형질성(heterogeneity)이다. 이를 위하여 상호 이해 가능한 프로토콜 개발이 추진되고 있으며 좀 더 현실적인 접근방법으로 동일 에이전트간의 메시지 교환 방식을 통해 이형질성의 통신 문제를 해결하려고 하고 있다.

3.1 이동 에이전트간 통신방법

기존의 이동 에이전트간의 통신에 관한 연구는 Odyssey나 AgentTCL에서 있었으나 클라이언트/서버 형식과 저수준의 메시지 전달 매카니즘을 사용하기 때문에 안정적인 통신망 연결과 통신을 위해 동기화가 이뤄져야 하는 단점을 가지고 있다. 또다른 방법은 미팅(Meeting) 기반 매카니즘이 있는데 이 기법을 사용하는 ARA와 Mole은 이동 에이전트들이 동일시간에 미팅 장소, 즉 동일 에이전트시스템에 존재하고 있어야 하는 가장 큰 제약을 가지고 있다. 따라서 최근에는 이동 에이전트가 에이전트시스템에 동시에 존재할 필요가 없는 블랙보드(blackboard)를 사용하는 에이전트시스템이 연구되고 있다. 블랙보드 매카니즘은 주로 에이전트간의 작업조정을 위한 방법론으로써 연구, 개발되고 있으며 블랙보드에서 공유되는 정보는 작업 규칙이나 작업에 대한 결과를 스트링 형태로 교환하는 정도에 그치고 있다. 즉 프리미티브 타입의 결과값만을 공유하는 단점이 있다.

본고에서는 이런 문제점을 해결하기 위해 블랙보드기반 매카니즘을 이용하여 공유메시지 저장소(Common Message Pool)를 설계, 구현하여 에이전트의 통신문제를 해결하는 방법을 살펴보기로 하겠다.

3.2 기존 에이전트의 통신방법

시스템에 정적으로 존재하는 에이전트간 메시지 전송을 위한 단순한 통신 기능은 이동 에이전트에 있어서는 매우 중요하다. 더구나 그룹 에이전트 협력 작업을 수행하는 모델일 경우에는 협력하는 이동 에이전트간의 의사전달 수단 및 정보공유를 위해 효율적인 통신기능이 제공되어야 한다. 이를 위해 기존의 이동 에이전트들은 크게 다음과 같은 통신기법을 사용하고 있다.

(1) 동기화 기법

이동 에이전트들은 동시에 에이전트시스템 내에 존재하면서 서로간에 메시지 전달 혹은 정보공유를 해야 한다. 이 방법은 효과적으로 통신제어를 할 수 있는 장점은 있으나 다중 에이전트간 통신 프로토콜(Protocol)이 일치해야 하는 단점이 있다. 따라서 이를 해결하려면 상호간에 항상 동기화가 필요하다는 단점이 있다.

· 요청(request)/답변(reply)

가장 일반적으로 사용되는 간단한 방법으로, 동기화 제약이 있는 프로시저(procedure) 호출 방법이 있다. 예를들어 에이전트 A가 필요한 정보를 에이전트 C로부터 제공받아야 한다고 한다면 에이전트 A는 시스템 B에게 요청을 시작하면서 동시에 시스템 B에서 답변이 올 때까지 기다리게 된다. 시스템 B는 A 요청을 서비스하기 위해 에이전트 C에게 요청한 후 A와 마찬가지로 기다리게 되는 것이다. 이러한 요청-답변 방법은 프로그램 구현이나 실행 흐름의 제어가 쉬운 장점이 있다. 그러나 필요조건으로 동기화 제약이 있으며, 이동 에이전트에 범용으로 적용하기엔 매카니즘 자체가 가지고 있는 단점이 크다. 대표적으로 RPC(Remote Procedure Call)와 자바의 RMI(Remote Method Invocation)가 있다.

(2) 비동기 통신기법

비동기 통신의 가장 큰 장점은 정보교환시 통신하려는 에이전트들이 동시에 한 시스템에 존재할 필요가 없다. 즉, 메일박스 개념을 이용해 메시지를 메일박스에 남겨두거나, 메시지 도착을 통보하는 방법으로 상대 에이전트가 어디에 있는지, 언제 메시지를 읽는지에 대해 생각할 필요가 없다. 따라서 이런 기법은 에이전트들의 라우팅 스케줄이나 위치를 예측할 수 없는 이동 에이전트 응용에 적합하다. 비동기 통신기법으로는 다음

과 같은 방법이 있다.

- 콜백(callback)

이 방법은 동기화기법과 비슷하다. 다만, 요청과 답변을 기다리지 않고, 에이전트 A와 시스템 B는 요청과 답변을 서로 대화하여 알고 있으며, 기다리지 않고 수행을 계속한다. 마찬가지로 시스템 B와 에이전트 C도 상호 대화하여 완료되었을 때, 원 호출자를 다시 호출하여 답변을 전달한다. 이때 원 호출자는 수행을 잠시 멈추고, 콜백을 처리한다. 콜백 기법은 비동기적 처리를 지원한다. 그러나, 프로그램 실행흐름을 복잡하고 어렵게 하는 단점이 있다.

- 우편함(mailbox)

에이전트 A가 작업을 수행하기 위해 시스템 B에게 요청과 답변을 자신의 우편함에 놓도록 대화한다. 그리고 에이전트 A는 수행을 계속하면서 주기적으로 자신의 우편함을 조사하거나 일정기간 요청-답변 기법처럼 B를 기다릴 수 있다. 마찬가지로 시스템 B와 에이전트 C는 같은 방법으로 대화한다. 우편함 방법은 요구/답변과 콜백 방법보다 구현하기 더 어렵지만, 비동기적 처리를 제공하면서 동시에 프로그램 실행흐름을 방해하지 않는다는 장점이 있다. 또한 이 두 가지 장점은 이동 에이전트를 분산시스템에 적용하는데 매우 중요한 요소이기도 하다.

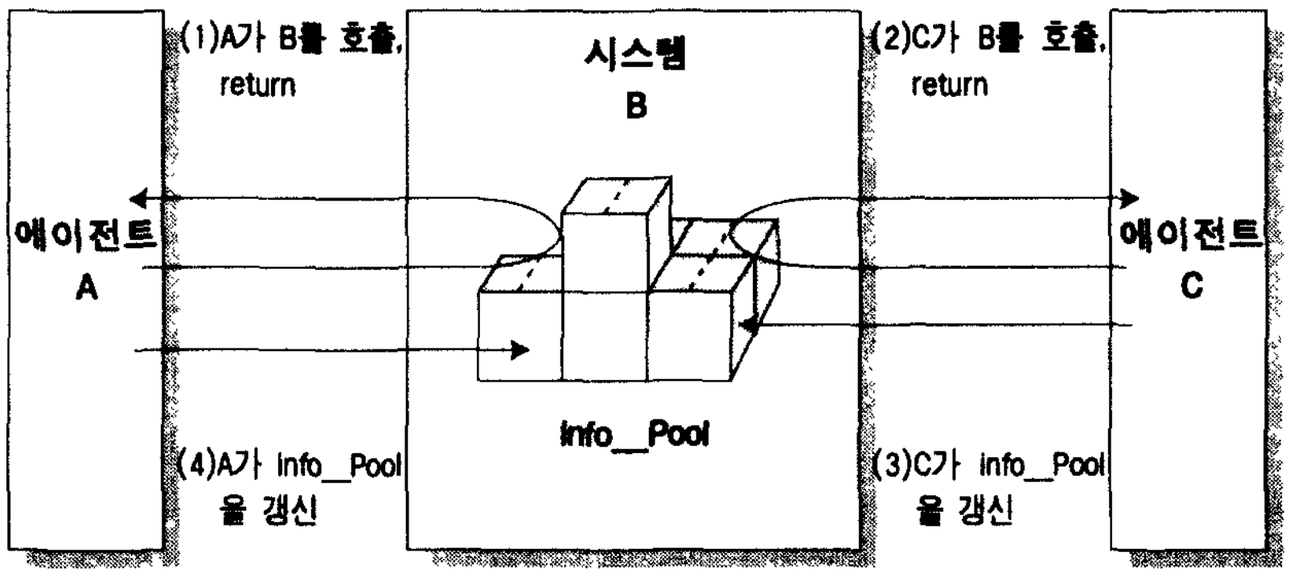
그러나, 우편함 기법이 효율적이라 하더라도 이동 에이전트를 사용하는 본질적인 목적에는 다소 문제가 있다. 즉, 이동 에이전트의 가장 근본적인 목적은 가능한 한 통신망 지연과 대역폭의 효율성을 높이는 것이다. 그러나, 이동 에이전트와 함께 전송되어야 하는 많은 트랜잭션 혹은 상태정보에 덧붙여 다른 에이전트와 통신을 위해서 우편함도 같이 수반해야하므로 대역폭의 효율은 낮아지게 된다.

3. 공유 메시지 저장소

앞서 살펴본 에이전트간의 통신을 위한 여러 매카니즘의 단점을 어느정도 해결하고 특별히 이동 에이전트에 적합한 기법으로 공유메시지 저장소 개념을 살펴보기로 한다.

이동 에이전트간의 통신을 위해 제안된 기법은 다음 <그림 1>에서 나타낸 바와 같다. 이동 에이전트시스템인 MAS(Mobile Agent System)가 메시지 보관, 관리를 위해 공유 메시지 저장소인 CMP(Common Message Pool)를 제공하고 이동 에이전트는 필요한 정보를 CMP를 이용해 교환함으로써 내용에 대한 일관성(consistency)을 보장받는다. 또한 이동중인 에이전트간의 메시지 교환 서비스를 위해 별도의 중계(broker) 시스템을 사용하는 단점을 보완한다. 그리

〈그림 1〉 공유메시지 저장소의 역할



고 MAS는 하나의 호스트에 다수의 이동 에이전트를 위한 가상 실행 환경으로 한 호스트에 다수의 MAS가 존재할 수 있다. 따라서 각각의 MAS에 존재하는 CMP는 상호 독립적인 관계를 형성한다. 즉, MAS가 구동될 때마다 CMP가 생성되기 때문에 한 호스트에서 다수의 에이전트 실행 환경인 MAS가 구동되더라도 CMP는 상호 배타적으로 메시지 관리를 수행할 수 있다.

3.1 CMP의 설계

이동 에이전트시스템인 MAS에서 제공하고 관리하는 공유메시지 저장소를 CMP라 한다. 이를 제공하기 위해 MAS의 기능이 다소 복잡해지는 단점이 있다. 하지만 이동 에이전트 구현시 통신 기능을 CMP에 의존하게 함으로써 에이전트 설계를 단순화시키며 실행 크기를 줄이는 장점이 있다.

CMP는(s__key__Set, r__key__Set, Objects__Set)의 구조를 갖는 튜플(tuple)로 설계되었다. 여기서 s__key__Set는 송신자를 구분하는 키 집합을 의미하며, r__key__Set는 수신자를 구분하는 키 집합들이다. 두 개의 키 집합은 요청/답변을 수행할 때와 보안을 위해 접근 제어를 제공하는 역할을 수행한다. 그리고 Objects__Set는 정보공유를 위한 값으로 객체형을 유지한다. 객체형으로 공유되는 정보는 이동 에이전트가 작업을 위해 단순히 참조하는 정보에 그치지 않고 직접적으로 이용할 수 있는 바이트 형태의 객체형이다. 또한 이동 에이전트가 CMP에 접근하기 위해 필요한 방법으로 〈알고리즘 1〉과 같이 설계된 접근 메소드(Access Method)를 사용한다. 접근 메소드 writes는 메시지 요청 및 답변을 위한 기능을 수행한다. reads 메소드는 요청(request)을 받아들이는 기능을 제공하며, 동시에 CMP 정보

〈알고리즘 1〉 에이전트간의 정보 접근 메소드

```
public boolean writes(String send, String[] recv, Object[] values)
{
   Lock();
   for (int i =0; i<recv.length; i++) {
      CMP cmp = getChannel(recv[i]);
      cmp.localV.put(send, values[i]);
      if (fail the put) boolean flag = false;
      else flag =true;
   }
   UnLock();
   return flag;
}

public String[] reads(String itself_id, Object values)
{
   Lock();
   CMP cmp = getChannel(itself_id);
   get those recv_keys from the cmp;
   UnLock();
   return recv_keys;
}
```

의 일관성 유지를 위해 writes에 의해 생성된 튜플을 read 후 제거하는 기능을 수행한다.

또 이동 에이전트는 takeAND와 takeOR 메소드를 사용하여 메시지 정보를 얻는다. 메소드 takeAND는 반드시 요청한 에이전트들로부터 결과를 모두 가져오는 것을 만족해야 한다. 그러나 takeOR 메소드는 적어도 하나 이상의 에이전트로부터 값을 가져왔을 경우를 만족한다. 마찬가지로 takeAND/OR 메소드들도 CMP의 일관성을 위해 답변 튜플들을 제거하는 기능을 수행한다. 아울러 에이전트간의 CMP 경쟁상태(race condition)를 회피하기 위해 접근 잠금/해제 기능을 CMP에서 제공한다. 제공된 메소드의 알고리즘은 다음과 같다.

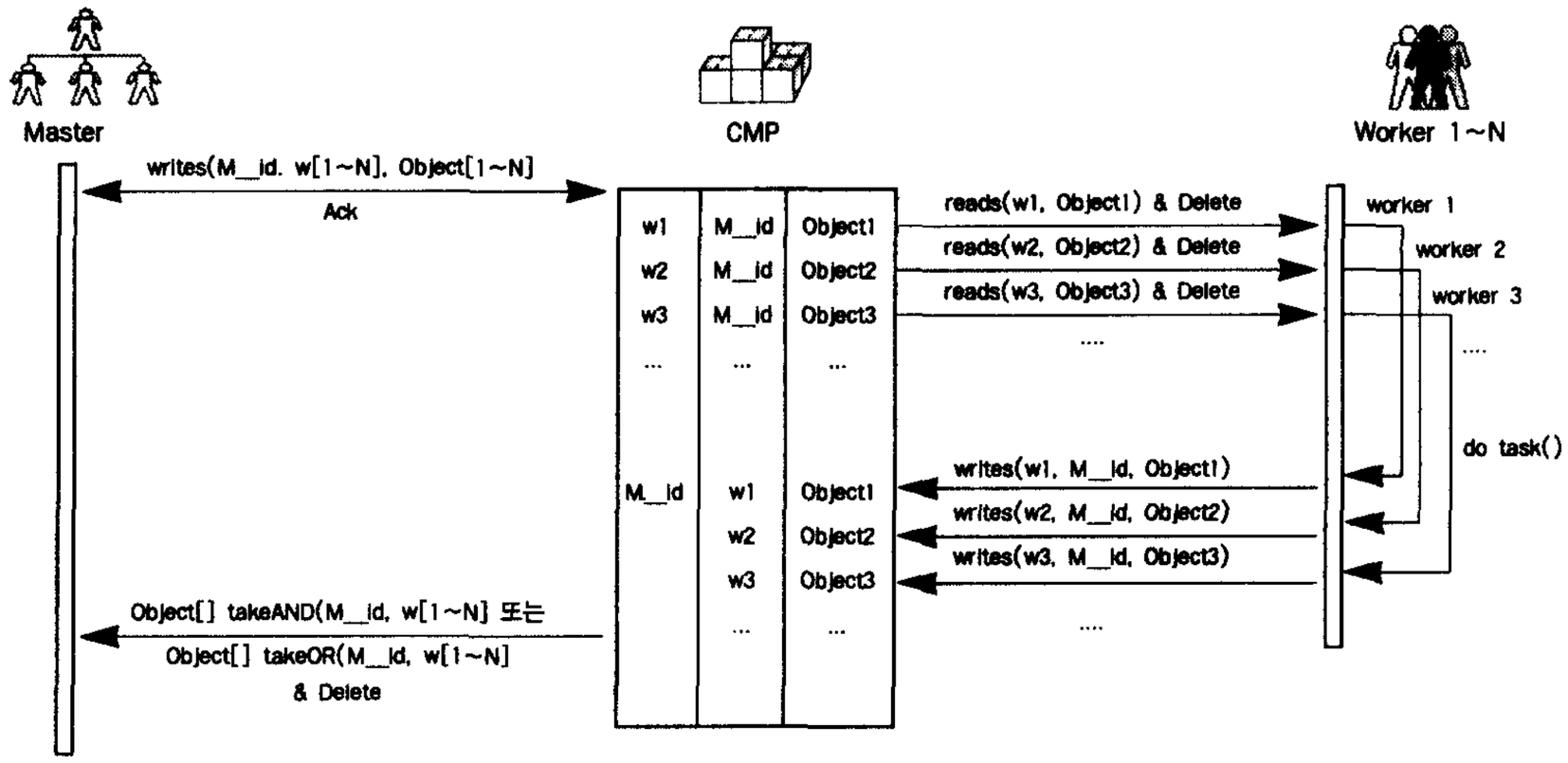
위의 메소드들이 제공하는 통신프로토콜은 반드시 요청과 답변을 위해서 각 이동 에이전트마다 메소드 모두를 적용할 필요는 없다. 예를 들어, 그룹 협력 작업을 하는 경우 주 에이전트(master agent)와 작업 에이전트(worker agent)로 구분되는데, 작업 에이전트는 단지 주 에이전트로부터 부여받은 작업에 대해서만 CMP에 결과를 작성하기만 하면 되고, 반대로 주 에이전트는 각각의 CMP에 저장된 값들을 takeAND 또는 takeOR로 가져오면 된다. 물론 상호간의 요

〈알고리즘 2〉 정보를 가져오기 위한 알고리즘

```
public Object[] takeAND(String send__id, String[] recv__set);
  {
    Lock();
    CMP  cmp = getChannel(send__id);
    while (cmp.localV.size() != recv__set.length) {
       if (Agent is actived) Agent.Deactive(system__times);
       if (system__times is over) {
              send "fail the takeAND";
              Agent.Active();
              stop this method;
       }
       cmp = getChannel(send__id);
    }
    for (int i =0; i<recv__set.length; i++) {
       put cmp.localV[i] of recv__set[i] in Object[i];
       remove cmp.localV[i] from the cmp;
    }
    if (Agent is deactived) Agent.Active();
    UnLock();
    return Object;
}

public Object[] takeOR(String send__id, String[] recv__set);
{
    Lock();
    CMP  cmp = getChannel(send__id);
    while (cmp == null){
       if (Agent is actived) Agent.Deactive(system__times);
       if (system__times is over) {
              send "fail the takeOR";
              Agent.Active();
              stop this method;
       }
       cmp = getChannel(send__id);
    }
    for (int i =0; i<recv__set.length; i++) {
       put cmp.localV[i] of recv__set[i] in Object[i];
       remove cmp.localV[i] from the cmp;
    }
    if (Agent is deactived) Agent.Active();
    UnLock();
    return Object;
  }
}}
```

〈그림 2〉 CMP을 이용한 메시지 멀티캐스트



청과 답변에 대한 인증키들이 보안을 위해 일치해야 한다.

3.2 메시지 멀티캐스트

위의 예에서, 그룹협력작업을 하기 위해 주 에이전트가 다수의 작업 에이전트에게 메시지 요청을 한다고 가정하자. 그룹 협력 모델은 에이전트를 어떠한 형태로 구성하느냐에 따라 주 에이전트가 작업 에이전트들과 직접적으로 통신을 할 수도 있지만, CMP를 이용할 경우 다음 〈그림 2〉와 같이 행해진다.

주 에이전트는 한번의 메소드 호출에 의해 작업 에이전트 모두에게 값을 요청하며(writes), 각 작업 에이전트는 요청을 받아(reads), 결과를 CMP에 갖다 놓는다(writes). 이후, 주 에이전트는 마찬가지로 한번의 메소드 호출(takeAND 또는 takeOR)로 모든 CMP에 저장된 결과값을 객체형으로 가져온다. 이때, 주 에이전트는 〈그림 2〉처럼 동기적으로 혹은 비동기적으로 수행할 수 있다. 이와 같은 멀티캐스트 기능을 제공하는 CMP 접근 메소드는 에이전트간의 통신을 위해 1:1뿐만 아니라 다:다 통신이 가능하도록 지원한다.

4. CMP의 실현

위에서 살펴본 CMP와 통신 메소드를 구현하기 위해 이미 자바(Java) 언어로 개발된 이동 에이전트시스템 MAS를 사용하였다. MAS는 자바 이동 에이전트를 대상으로 입/출 기능이 있는 가상의 장소를 제공하는 환경이며, 한 호스트에 다수의 독립

〈표 1〉 호스트 주소와 운영체제

IP 주소	Port #	운영체제
172.16.53.100	9110 9210	Unix
172.16.51.99	6510 6620	Unix
172.16.53.70	6000	Windows NT

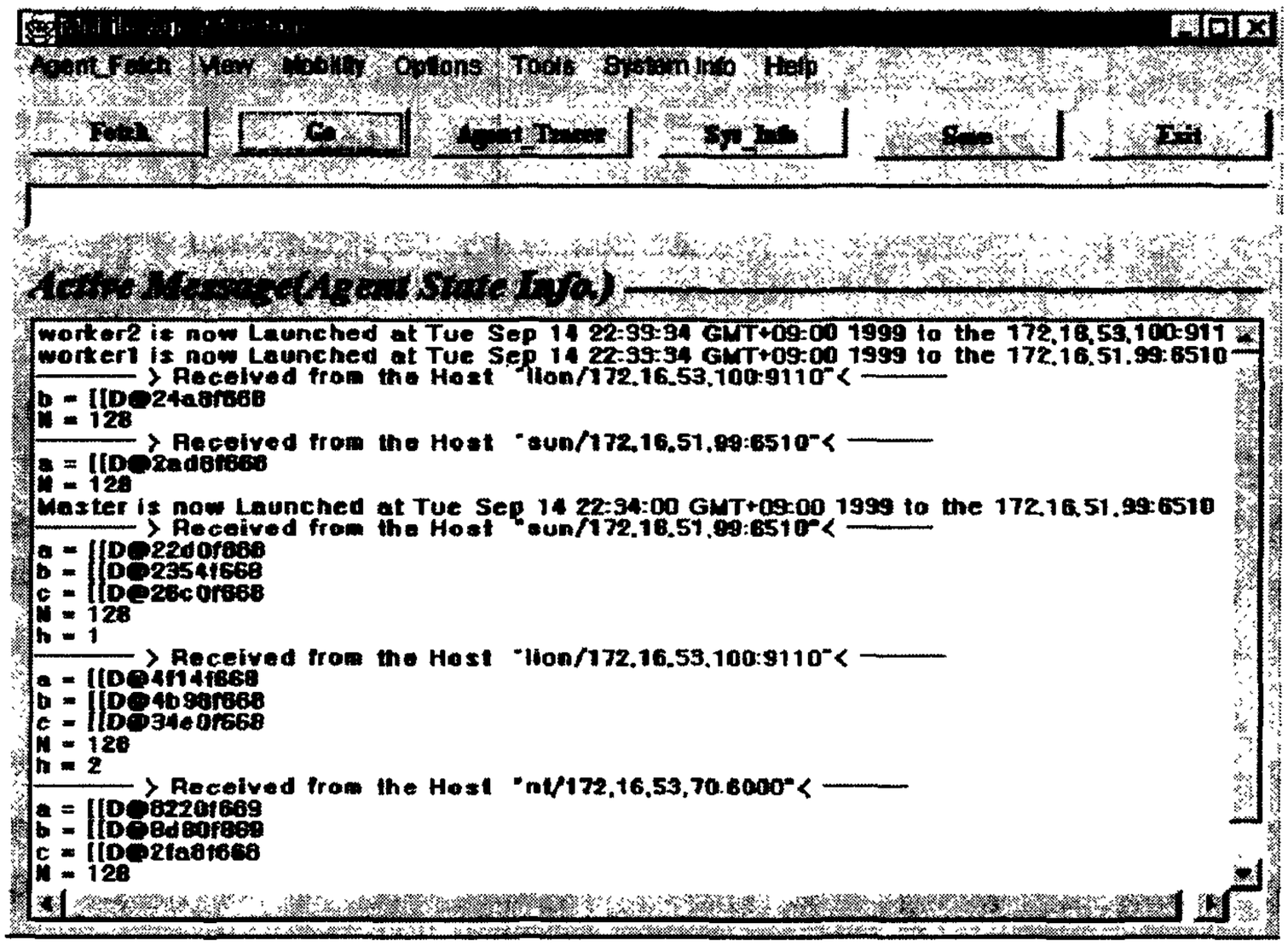
적인 MAS가 존재할 수 있다. 따라서 본 논문에서 제안한 통신 매카니즘을 보이기 위해 실제 통신망에 연동된 MAS의 각 호스트의 주소와 운영체제는 〈표 1〉과 같다.

그룹협력작업을 실험하기 위한 모델로 작업 에이전트 2개와 주 에이전트 1개를 가정한다. 그리고, 두 개의 작업 에이전트는 동시에 서로 다른 MAS들로 이동하여 N*N 행렬을 생성하여 각 MAS의 CMP에 주 에이전트가 가져갈 수 있도록 메시지 a와 b 행렬 객체를 저장한다. 여기서 행렬 원소는 (double) random*1000/100인 수식으로 생성된다. 이후, 주 에이전트는 메시지 a와 b를 받기 위해 순서적으로 두 개의 노드 시스템을 방문하여 가져오고, 세 번째 노드 시스템에서 행렬 곱 c를 생성한 후 사용자에게 전달하는 시나리오를 가정한다.

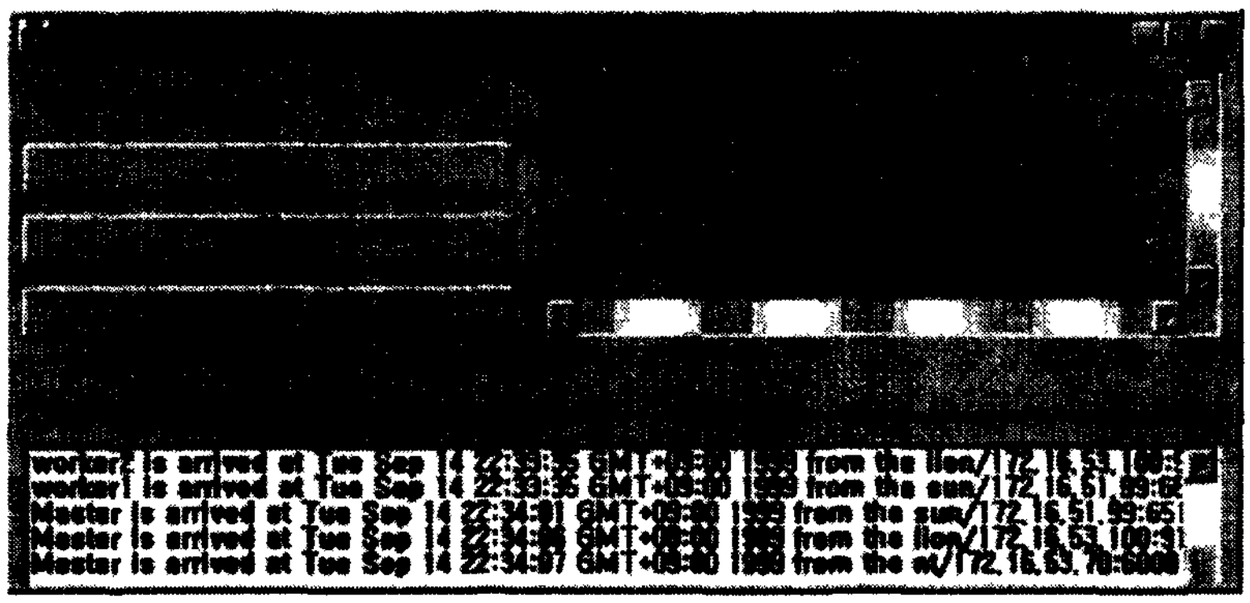
주 에이전트와 작업 에이전트간의 메시지 전달이 수행된 결과로서, 〈그림 3(a)〉와 〈그림 3(b)〉는 실행중인 에이전트들의 상태 정보와 각 실행 결과 및 수행 시간을 보여준다. 〈그림 3〉에서 주 에이전트와 작업 에이전트들 각각의 수행 시간은 행렬 크기 N을 128로 하였을 때, 작업에이전트들의 수행 시간은 적다. 그러나 주 에이전트는 각 노드를 방문하여 메시지를 수집(gathering)하여 마지막 노드로 이동하여 행렬 곱을 수행한 시간이다. 이는 주 에이전트 자체의 크기가 커짐에 따라 통신 대역폭에 영향이 있음을 보여준다. 또한, 〈그림 4〉는 각 호스트에서 에이전트 통신을 위해 적용한 writes와 takeAND 메소드 수행이 되고 있음을 화면 캡쳐한 것이다. 여기서 서버 sun에 작업에이전트 worker 1이 저장한 메시지 객체 "D@cc06e4"를 주 에이전트가 takeAND로 똑같은 메시지를 가져오고 있음을 보여 주고 있다. 마찬가지로 서버 lion에서도 작업에이전트 worker 2가 CMP에 저장한 메시지를 정확히 가져옴을 나타낸다.

〈그림 3〉 각 사례에 따른 에이전트 상태 정보와 실행 결과

(a) 각 에이전트 상태 정보 결과

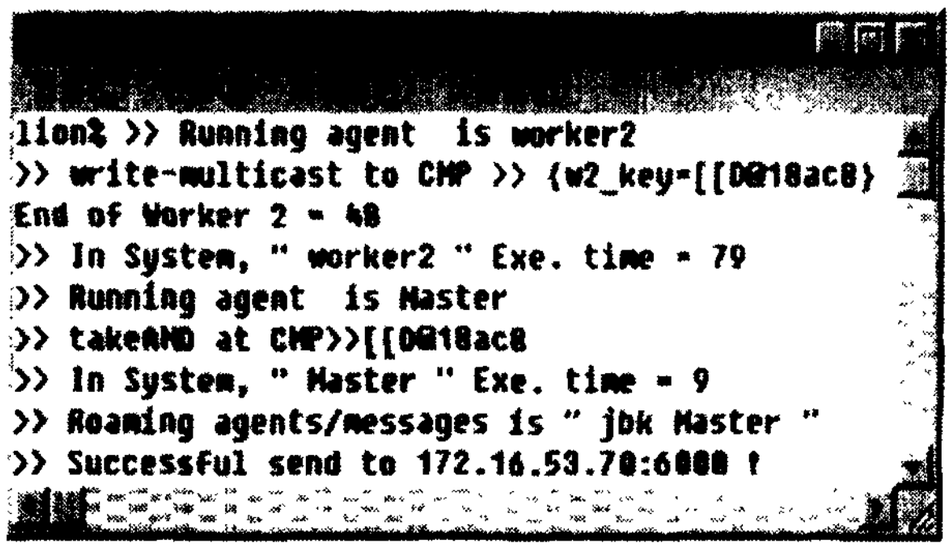


(b) 실행 결과

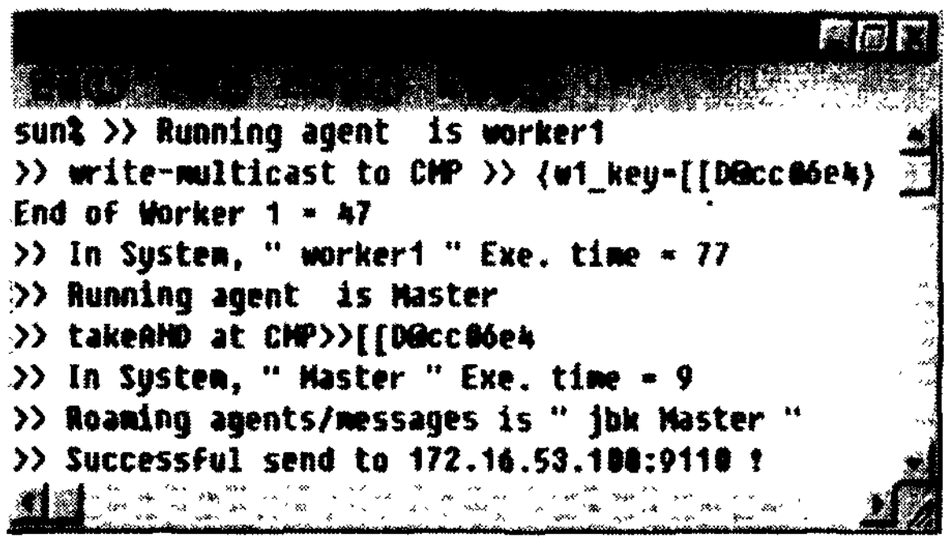


〈그림 4〉 각 서버에서 실행 결과 화면

(a) 서버 SUN에서의 실행



(b) 서버 lion에서의 실행



5. 결론

에이전트에 대한 관심이 최근들어 더욱 증대되고 있다. 이는 인터넷 이용에 따른 사용자의 다양한 정보요구 때문이기도 하다. 더 나아가 전자상거래, 메시징과 같은 이동컴퓨팅 분야의 활용가능성이 점점 커지며 요구가 많아지고 있기 때문이기도 한다.

본고에서 살펴본 에이전트는 자율성(autonomy), 사교성(social ability) 및 이동성(mobility), 지능(AI) 등의 특성을 가지는 프로세스로 정의할 수 있다.

앞으로 에이전트가 연구되어질 방향은 에이전트의 정의와 특성을 명확히 하는 작업이 필요하며 이론에서 정형화된 특성들을 구현하기 위한 에이전트의 구성요소와 제어 및 통신 프로토콜 등을 연구해야 한다. 또한 에이전트를 개발하기 위해 에이전트 특성을 고려한 프로그래밍 언어를 개발하고 연구하는 에이전트 언어도 표준이 정해져야 할 것이다. 나아가 실

제 사용할 수 있는 응용 분야를 개발해야만 한다. 지금의 인터넷 정보검색, 온라인 쇼핑, 메시징, 네트워크 관리만 가지고 에이전트의 미래는 꽉 밝지만은 않다. 좀더 많은 연구와 빠른 표준화만 이뤄진다면 에이전트만이 네트워크를 사용하는 날이 올 것이다.

참고문헌

이근상, 전병국, 최영근. 2000. "그룹 에이전트간의 통신을 위한 공유 메시지 저장소," 『한국정보처리학회논문지』, 7권 8호.

전병국, 이근상, 최영근. 1999. "Java 언어를 이용한 객체이동시스템의 설계 및 구현", 『한국정보처리학회논문지』, 6권 1호.

전병국, 최영근. 1999. "이동 에이전트를 위한 효율적인 이주 정책 설계 및 구현," 『한국정보처리학회논문지』, 6권 7호.

전병국, 최영근. 1999. "인트라넷상에서 자바 객체의 이동시스템 설계 및 구현," 『한국정보과학회논문지(C)』, 5권 2호.

최중민. 1997. "에이전트의 개요와 연구방향," 『정보과학회지』, 제15권, 제3호.

Baumann, J. et al. 1997. "Communication concepts for mobile agent systems," *Pro. 1st Int'l Workshop on Mobile Agents*, Germany.

Cardelli, L. & Gordon, A.D. 1997. "Mobile Ambient", 〈http://www.reserach.digital.com/SRC/personal/Luca__Cardelli/Ambit/Ambit.html〉

Domel, P. et al. 1997. "Mobile Agent Interaction in Heterogenenous Environment," *Proc. Int'l Workshop on Mobile Agents*, LNCS, No.1219.

Durfee, Edmund H. & Rosenschein, Jeffrey S. 1994. "Distributed Problem Solving and Multi-Agent Systems: Comparisons and Examples," *Proc. of Thirteenth International Distributed AI Workshop*, pp. 94-104.

Etzioni, Oren & Weld, Daniel. 1994. "A Softbot-Based Interface to the Internet," *Communications of ACM*, Vol. 37, No. 7, pp. 72-76.

Finin, T., & Fritzson, R. 1994. "KQML as an agent communication language," *Proc. of CIKM 94*

Finnin, T., Richard Fritzson, Don McKay, and Robin McEntire, 1994. "KQML as an Agent Communication Language," *CIKM '94*

Franklin, S. & Graesser, A. 1996. "Is it an agent or just a program," *Proc. of Third International Workshop on Agent Theories, Architectures, and Languages*

Gray, R. 『Agent Tcl: A flexible and secure mobile-agent system』, PhD thesis, Dept. of Computer Science, Dartmouth.

Lange, Danny B. & Oshima, M. 1998. 『Programming and Deploying Java Mobile Agents with Aglets』, Addison Wesley.

Magic, General. "Odyssey", 〈http://camille.is.s.u-tokyo.ac.jp/~masatomo/mobile/White/whitepaper.html〉

Peine, H. & Stolpmann, T. 1997. "The architecture of the Ara platform for mobile agents", *Proc. of 1st Int'l Workshop on Mobile Agents*, Germany.

Roda, C., Jennings, N.R. & Mamdani, E.H. 1991. "The Impact of Heterogeneity on Cooperating Agents," *Proc. AAAI Workshop on Cooperation among Heterogeneous Intelligent Systems*, Anaheim.

White, J. 1996 『Mobile Agents White Paper』, General Magic.

White, James E. 1998. 『Mobile Agents White Paper』, General Magic Co. 〈http://camille.is.s.u-tokyo.ac.jp/~masatomo/mobile/White/whitepaper.html〉 Feb. 1998.