웹을 기반으로 한 자바 이동에이전트 프로그래밍 모델

(Programming Model for Web-based Mobile Agent)

송 성 훈 † 원 유 헌 ††
(Sung Hoon Song) (Yoo Hun Won)

요 약 현재까지 개발되어 있는 많은 이동에이전트 시스템들이 웹과의 연동을 고려하고 있으며, 웹서버들 또한 향후에 이동에이전트를 지원할 것을 고려하고 있다. 그러나 현재로서는 근본적으로 클라이언트/서버 구조를 가지고 있는 웹프로그래밍과 프로그램 코드의 자치적인 이동이라는 특징을 가진 이동에이전트 프로그래밍을 통합하여 정보시스템을 개발하는 방법이 명확히 정립되어 있지 않다. 본 논문에서는 첫째, 웹상의 이동에이전트 프로그래밍 모델을 제시하므로서 웹프로그래밍과 이동에이전트 프로그래밍을 통합할 수 있는 방법을 명확히 하였다. 둘째, 웹프로그래밍과 이동에이전트 프로그래밍 모두에 많이 쓰이고 있는 자바언어의 API를 개발하므로서, 웹상의 이동에이전트 프로그래밍에 사용할 수 있도록 하였다. 세째, W3C에서 자바로 개발한 Jigsaw 웹서버에 이동에이전트의 실행환경을 제공하기 위한 모듈을 추가하고 테스트하므로서 제안하는 모델의 유용성을 보였다.

키워드 : 이동에이전트, 웹프로그래밍, 자바

Abstract The developers of mobile agent systems are considering integrating the system into the web and the developers of web servers are also considering supporting mobile agents in the future. But they are not clearly suggesting the relationship between web programming which has basically client/server architecture and mobile agent programming which is based on autonomous code mobility. In this paper, firstly, we clarify the method for integrating mobile agent programming into web programming by suggesting the model for mobile agent programming on the web. Secondly, by developing APIs for Java which is widely used for both web programming and mobile agent programming, we made it possible for programmers to use them for programming mobile agent on the web. Thirdly, we show the usefulness of the proposed model by adding and testing modules for execution environment of mobile agents on W3C's Java based web server, Jigsaw.

Key words : Mobile Agent, Web Programming, Java

1. 서 론

이동에이전트는 현재 인공지능 및 소프트웨어엔지니어링 분야에서 많은 연구가 진행되고 있으며, 클라이언트/서버 모델에 대비되는 또 다른 프로그래밍 패러다임으로 자리매김하고 있다. 이동에이전트의 유용성은 이미 많은 논의를 거쳐 확립되어 있다. 이동에이전트의 개념과는 다르지만 코드의 이동성으로 얻을 수 있는 장점은 자바 애플릿에서 실행코드의 자동 이동으로 인한 유지보수의 편리성 등과 같은 예에서 찾아 볼 수 있다.

최근의 컴퓨팅 환경은 웹을 중심으로 빠르게 바뀌어 가고 있다. 그 바탕에는 네트웍을 근본으로 하는 분산 컴퓨팅 환경이 존재하며, 사용자에게는 보다 친숙하게 접근할 수 있는 웹브라우저를 GUI의 수단으로 하는 정보시스템이 증가하고 있다. 웹브라우저는 사용자인터페이스를 제공함과 동시에 클라이언트 플랫폼에 독립적인 자바 및 기타 스크립트 언어의 실행환경을 제공하고 있다. 향후의 정보시스템은 특정 기종의 컴퓨터 혹은 특정 운영체제를 사용하는 플랫폼에서 운영되는 것이 아니라 웹서버와 웹브라우저 상에서 운영되는 방향으로 바뀔 것이다.

이러한 경향을 비추어 볼 때, 이동에이전트의 프로그래

† 비 회 원 : 혜천대학 컴퓨터통신계열 교수
mitchell@hcc.ac.kr
†† 종신회원 : 홍익대학교 컴퓨터공학과 교수
won@cs.hongik.ac.kr
논문접수 : 2001년 12월 6일
심사완료 : 2002년 1월 2일

밍도 웹환경에서 이루어 질 수 있도록 시스템을 구현한다면 유용하게 쓰일 수 있을 것이다. IBM Aglets의 경우에도 이동에이전트의 관리를 웹상에서 운용하기 위한 Fiji의 개발계획에 대하여 언급하고 있고[1], W3C의 Jigsaw 웹서버에서도 웹서버를 자바 언어로 개발한 이유중 하나로서 향후 이동에이전트의 지원을 들고 있다 [2]. 많은 이동에이전트시스템들이 웹환경에서의 프로그래밍에 대하여 언급하고 있고, 웹서버들도 이동에이전트의 지원에 대한 언급을 하고 있지만 실제로 웹환경과 이동에이전트를 이용한 프로그래밍은 서로 독립된 분야로 시스템이 구축되어 있다. 본 논문에서는 웹환경에서 이동에이전트의 프로그래밍이 자연스럽게 이루어질 수 있도록 프로그래밍 모델을 제시하고, 자바 API를 설계하고, 자바 웹서버상에 시스템을 구현하여 그 유용성을 보이고자 한다.

본 논문의 2장에서 이동에이전트와 클라이언트/서버 방식의 웹을 비교 설명을 하였고, 3장에서 웹 기반 이동에이전트의 활용 시나리오를 제시하고 이를 근거로 이동에이전트의 프로그래밍이 웹환경에서 자연스럽게 이루어 질 수 있도록 프로그래밍 모델을 제안하였다. 4장에서는 이동에이전트 프로그래밍을 위한 자바 API 설계에 대해 기술하고 예제 코드를 보였다. 자바 언어는 기본적으로 코드의 이동성을 지원하고 있으며 웹환경과 이동에이전트 시스템 모두에서 많이 사용되므로, 웹환경에서 이동에이전트를 프로그래밍하는 언어로서 가장 실용적이라고 생각된다. 5장에서는 시스템의 구성에 대해 기술하고 W3C의 자바 웹서버인 Jigsaw에 구현 모듈을 추가하는 방법을 설명하였다. 자바 웹서버인 Jigsaw는 자바 프로그램의 실행환경을 잘 지원하므로 자바로 작성된 이동에이전트의 실행환경을 구현하는데 적절하였다. 6장에서는 제안된 모델의 성능평가를 통하여 본 모델의 타당성을 보였다.

2. 관련연구

2.1 코드의 이동성과 실행환경 비교

코드의 이동성과 관련된 언어는 상당수가 개발되어 있다[3]. 그러나 자바 애플릿, 자바 스크립트, Caml 애플릿[4]과 같은 언어들은 웹브라우저가 웹서버에 해당 코드를 요구하는 시점에 웹브라우저로 다운로드되어 실행되는 형식을 지원한다. 하지만 이동에이전트는 스스로의 판단에 의해 이동할 컴퓨터와 이동 시점을 결정한다(표 1).

이동에이전트 스스로가 코드의 이동시점을 결정하고 스스로 다른 시스템으로 이동하도록 지원하는 이동에이전트 언어로는 Obliq[5], Telescript[6] 등이 있다. 이동에이전트를 위한 별도의 언어를 설계하는 경우에는 언어의 설계시점에서 코드의 이동성을 고려하므로서 기존의 언어를 이용하여 이동에이전트를 프로그래밍하는데서 발생할 수 있는 문제들을 미리 명확히 정의할 수 있다는 것이다. 예를 들어, 자바 언어를 이동에이전트에 사용한 경우 클래스변수와 같이 다른 클래스 객체들과 공유하는 변수를 가진 클래스의 객체가 다른 시스템으로 이동하였을 경우 이러한 변수의 바인딩에 대한 정확한 의미 해석을 하기 어렵게 된다[7].

표 1 코드의 이동성과 실행환경

	코드의 이동시점	코드의 실행환경
자바 애플릿	웹브라우저가 요구시	웹브라우저의 자바가상기계
자바 서블릿	코드이동을 지원하지 않음	웹서버상의 자바 가상기계
이동에이전트	스스로 판단	이동에이전트 서버

별도의 언어를 설계하지 않고 기존의 언어를 사용한 시스템으로 IBM Aglets[1], ObjectSpace의 Voyager[8], Mitsubishi의 Concordia[9], NOMADS[10], IKV의 Grasshopper[11]와 같은 시스템이 있다. 이들은 대부분 자바언어를 기반으로 하고, 자바 API를 이용하여 이동에이전트를 프로그래밍하도록 하며, 이동에이전트를 위한 독립된 실행환경을 제공하고 있다. 기존 언어를 사용하여 이동에이전트를 구현하는데에 문제점이 있기는 하지만, 굳이 새로운 언어를 설계하기 보다는 자바와 같은 산업계에서 성공한 언어를 기반으로 하여 이동에이전트의 특징을 추가한다면 실용적인 측면에서 많은 장점이 있다.

2.2 클라이언트/서버 방식과 이동에이전트의 비교

클라이언트/서버 방식과 이동에이전트 방식의 비교는 데이타와 코드를 어떻게 분배하고 이동하느냐로 비교해 볼 수 있다. 클라이언트/서버 방식의 경우에는 코드가 클라이언트와 서버 응용프로그램으로 분리되며 이 양자간에 데이타가 이동하는 방식으로 응용프로그램이 동작한다. 이때, 클라이언트 및 서버 응용프로그램은 사전에 설치되어 있어야 한다. 이동에이전트의 경우에는 코드가 데이타와 함께 이동하는데, 이동에이전트의 코드는 이동에이전트의 실행을 지원하는 호환 가능한 실행환경 사이에서 이동하게 된다(그림 1). 클라이언트와 서버 응용

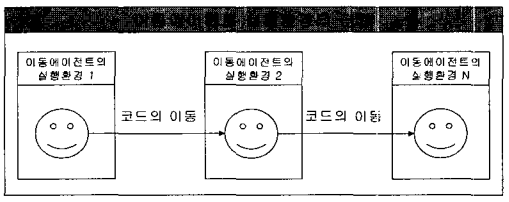

그림 1 이동에이전트 방식

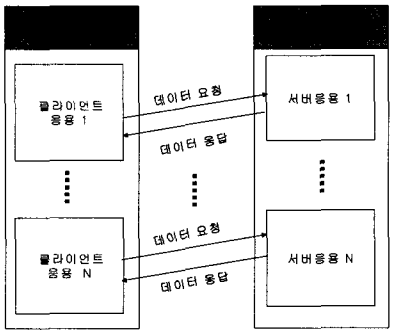

그림 2 클라이언트/서버 방식

프로그램은 서로 독립적인 실행환경에서 실행되며, 양자간에 약정된 프로토콜을 준수하므로서 서로 협조하게 된다(그림 2).

이 두 가지 방식의 효율성을 비교할 때, 데이타와 코드가 이동하는 네트웍상에서의 시간지연이 주요한 관건이 된다. 서버와 클라이언트간에 데이타 송수신에 많은 트래픽을 유발하는 응용의 경우에는 클라이언트를 이동에이전트로 하여 서버로 이동시키는 편이 효율적일 것이다. 클라이언트/서버 방식과 이동에이전트 방식의 응용프로그램 설계는 명확하게 산출할 수 있는 효율성에 기준하여 선택하기는 어렵다. 실제로 이동에이전트를 사용하여 효율적인 설계가 가능한 응용은 클라이언트/서버 방식을 사용하여도 같은 수준의 효율성을 보일 수 있도록 설계할 수 있다[12][13][14]. 이동에이전트 혹은 클라이언트/서버 방식의 선택은 코드 분배의 문제와 응용프로그램의 구조적 특징이 어느 쪽에 더 적합한가를 기준으로 해야 한다고 본다.

2.3 이동에이전트시스템과 웹

기존의 잘 정의된 이동에이전트 시스템들은 코드의 이동성을 중심으로 한 API를 제공하고 이동에이전트의 실행환경을 별도의 프로그램으로 제공하거나, 프로그래머가 실행환경을 구성할 수 있도록 추가의 API를 제공하고 있다. 이동에이전트의 실행환경은 이동에이전트의 실행을 초기화하고, 실행중인 이동에이전트를 관리하며, 사용자 인터페이스를 제공한다. 기존의 이동에이전트 시스템들은 웹을 고려하지 않고도 많은 분야에서 활용할 수 있다. 그러나, 정보시스템의 웹으로의 이동이라는 경향을 고려할 때, 이동에이전트를 웹환경에서 활용할 수 있다면 응용프로그래머에게 많은 도움이 되리라고 생각한다. 이동에이전트를 웹환경에서 이용한다면, 기존의 정보시스템이 웹으로 이동했을 때와 같은 장점을 얻을 수 있다. 즉, 사용자는 장소에 관계없이 어디서나 쉽게 가용한 웹브라우저를 활용하여 이동에이전트로 작성된 응용을 사용할 수 있을 것이다.

이동에이전트가 웹환경에서 사용되기 위해서는 사용자가 웹브라우저를 통하여 이동에이전트를 실행하고, 결과를 확인할 수 있어야 한다. 웹브라우저는 웹서버와의 통신을 위한 도구이므로 이동에이전트가 웹브라우저를 사용자인터페이스로 하려면 웹서버를 통해야 한다. 이를 위해서는 CGI 프로그래밍과 같이 이동에이전트 시스템과 웹서버 사이의 인터페이스를 제공하는 방법을 사용하거나 자바 서블릿과 같이 웹서버에 실행환경이 통합되어 있어야 할 것이다. 본 논문에서는 자바 서블릿과 같이 이동에이전트의 실행환경을 웹서버에 통합하므로서 보다 효율적인 프로그래밍이 가능하도록 했다.

응용프로그래머는 CGI나 자바 서블릿과 같이 주어진 프로그래밍 모델과 API 및 시스템을 사용한다. 웹상에서 이동에이전트를 이용하는 프로그래밍을 하기 위해서도 적절한 프로그래밍 모델, API 및 시스템이 구축되어 있어야 할 것이다. 이러한 시스템의 구축은 응용프로그래머의 몫이 아니라 시스템프로그래머가 제공해야 할 것이다.

2.4 웹을 고려한 기존의 이동에이전트 시스템

기존의 이동에이전트 시스템은 웹과는 독립적으로 설계되어 있으나, 웹과의 연계 방법을 제공하고 있는 시스템들이 있다. Telescript의 웹 도구인 Tabriz의 경우 사용자가 웹페이지상에서 이동에이전트를 호출할 수 있도록 하고 있다. 웹서버와 Telescript의 실행환경은 webtots라는 CGI 프로세스에 의해 연결되는데, webtots는 사용자가 웹페이지의 폼문에서 지정한 모든 매개변수를 Telescript의 자료구조로 변환한후, Telescript의 실행환경에 연결하여 이동에이전트의 실행을 초기화한다[15]. 웹페이지에서 서버상의 이동에이전트를 실행시키는 것은 본 논문에서 제안하고자 하는 모델과 같은 방법이지만, Tabriz의 경우 웹서버와 Telescript의 실행환경은 완전히 독립적이며 서로간의 연계는 CGI를 사용하여 이루어진다.

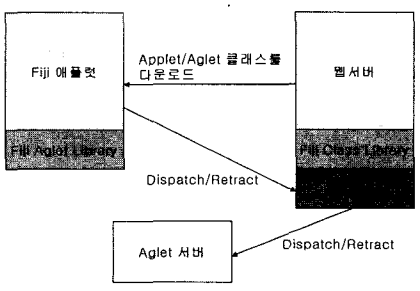

그림 3 Fiji의 동작구조

IBM Aglets의 경우 Fiji라는 도구를 사용하여 Fiji 애플릿을 생성할 수 있도록 하고 있다. Fiji 애플릿은 웹브라우저에서 실행되는 이동에이전트이다. Fiji를 사용하는 경우에도 Aglet 서버는 웹서버와 독립적으로 실행되는 구조이다. 따라서 Fiji 애플릿이 이동할 때에는 독립적으로 실행되는 Aglet 서버로 이동해야 한다. 그러나, Fiji 애플릿은 애플릿의 제한 때문에 자신을 다운로드한 웹서버외에는 네트웍 연결을 할 수 없다. 따라서 Fiji는 웹서버상에 router 모듈을 두어 Fiji 애플릿이 원하는 Aglet 서버로 이동할 수 있도록 라우팅을 해주는 구조를 가지고 있다(그림 3). Fiji는 애플릿을 이동에이전트로 하는 특징을 가지고 있으나, 본 논문의 모델에서는 서버상에서 실행되는 이동에이전트의 실용성을 감안하여 이동에이전트의 실행환경을 웹서버상으로 하고자 한다.

앞에서 살펴본 기존의 시스템들은 모두 웹서버와 이동에이전트의 연계를 위하여 중간 모듈을 사용하고 있다. 본 논문에서는 웹서버와 이동에이전트 시스템을 통합한 모델을 제안하고자 한다.

3. 웹 기반 이동에이전트의 프로그래밍 모델

3.1 활용 시나리오

웹 기반 이동에이전트의 프로그래밍을 모델화하기 위해 먼저 웹상에서 이동에이전트를 활용하는 시나리오를 설정하였다. 이 시나리오는 웹서버에 이동에이전트의 실행환경이 통합되어 있다는 가정하에 작성되었으며, 본 논문에서 작성한 테스트 프로그램을 기준으로 설명하였다.

테스트를 위해 작성된 이동에이전트는 사용자가 지정한 웹사이트에 지정된 날짜 이후에 수정된 웹페이지가 있는가를 알려주는 일을 한다. 이러한 응용은 사용자가 자주 방문하는 웹사이트에 변경된 내용이 있는가를 알아내기 위해 사용된다. 다음에 이동에이전트의 실행과정을 단계별로 기술하였다(그림 4).

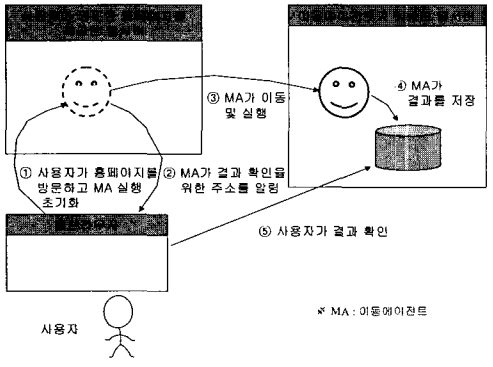

그림 4 웹기반 이동에이전트의 활용 시나리오

① 사용자가 웹브라우저로 이동에이전트를 초기화하는 HTML태그가 포함된 웹페이지를 방문한다(그림 5). 사용자가 방문한 웹페이지를 제공하는 웹서버는 실행할 이동에이전트를 저장하고 있다. 그림 5의 경우 사용자가 목적 URL을 입력하고 확인단추를 누르면, 사용자가 방문한 웹페이지를 제공한 웹서버상에 이동에이전트가 적재된다.

그림 5 이동에이전트를 호출하는 웹페이지

② 웹서버상에 적재된 이동에이전트는 사용자가 웹브라우저를 사용하여 결과를 확인할 수 있도록 하기 위해 결과가 저장될 URL을 사용자에게 미리 표시해 준다(그림 6).

그림 6 이동에이전트가 생성한 안내 웹페이지

③ 이동에이전트는 사용자가 지정한 목적 URL의 웹서버로 이동하여 사용자가 지정한 날짜 이후에 수정된 웹페이지를 검색한다.

④ 이동에이전트는 실행 결과를 이동한 웹서버의 디스크에 웹페이지로 저장한다.

⑤ 사용자는 미리 안내 받은 URL을 사용하여 웹브라우저로 이동에이전트가 저장한 웹페이지의 내용을 확인한다(그림 7).

그림 7 사용자가 웹브라우저로 결과를 확인

이러한 응용은 흔히 웹사이트에서 클라이언트/서버 형태로 개발하여 서비스를 제공하는 경우가 많다. 그러나 웹사이트가 이러한 서비스를 제공하지 않는다면, 사용자가 별도의 응용프로그램을 사용하여 그 웹페이지에 포함된 모든 지역적 링크에 대해 순환적으로 HTTP

GET 요청을 하므로서 링크된 모든 웹페이지에 대한 수정일자를 조사해야할 것이고, 당연히 엄청난 트래픽을 발생하게 될 것이다.

기술한 시나리오에서 웹브라우저는 사용자인터페이스로 사용되며, 웹서버는 이동에이전트의 실행환경이다. 이러한 가정하에 "3.2 프로그래밍 모델"에서 웹 기반 이동에이전트 프로그래밍을 모델화하였다.

3.2 프로그래밍 모델

실행가능한 프로그램 코드를 EU(Executing Unit)로, 프로그램 코드의 실행환경을 CE(Computational Environment)로 정의할 때[16], 이동에이전트는 EU가 컴퓨터 네트웍상에 산재한 다수의 CE 사이를 독자적으로 이동하며 실행되는 경우로 정의할 수 있다. "2.3 이동에이전트시스템과 웹"과 "3.1 활용 시나리오"에서 기술한 가정에 의해 본 모델에서는 이동에이전트의 CE가 웹서버에 포함되어 있다(그림 8). "3.1 활용 시나리오"에서 기술한 대로, 이동에이전트는 사용자가 웹브라우저로 웹페이지를 방문하므로서 실행이 초기화된다. 이때, 사용자가 방문한 웹페이지는 "3.1 활용 시나리오"의 그림 5에서 보인 것처럼 이동에이전트의 실행을 초기화할 수 있도록 구성되어 있다. 이 웹페이지와 실행할 이동에이전트를 저장하고 있는 웹서버를 HCE(Home Computational Environment)로 정의하였다. 이동에이전트는 HCE에서 지역적으로 적재되어 다른 CE로 이동한다. 웹서버는 HCE, CE의 역할을 동시에 한다. 즉, 자신의 시스템에 저장된 이동에이전트를 적재하여 다른 CE로 보내는 경우에는 HCE의 역할을하며, 다른 HCE 혹은 CE에서 수신된 이동에이전트의 CE가 되기도 한다. 이동에이전트는 네트웍상에 산재한 웹서버 사이를 이동하며 실행된다. 이러한 관점에서, 이동에이전트의 실행환경을 네트웍상에 산재한 CE의 집합으로 추상화한 NCE(Networked Computing Environment)로 확장할 수 있다 (그림 8). 본문 중에서 이동에이전트는 자치적으로 이동하는 EU를 의미하며, 인터넷환경 또는 네트웍은 NCE에 해당된다. HCE와 CE는 웹서버로 표기하되 구분이 필요한 경우 명시하였다.

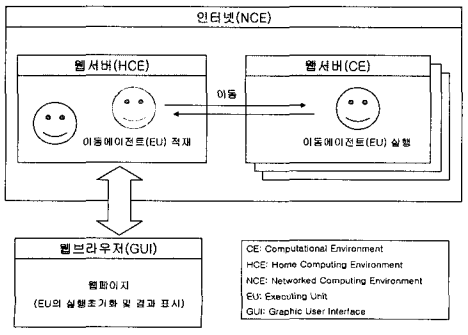

그림 8 웹 기반 이동에이전트의 모델

4. API 설계

4.1 API 설계시의 고려사항

4.1.1 웹브라우저를 도구로 한 사용자인터페이스

웹브라우저는 웹서버와 HTTP 프로토콜을 사용하여 통신하고 송수신되는 데이타는 웹페이지를 기본으로 하고 있다. 따라서 웹페이지상에서 이동에이전트를 실행시키고, 결과를 표시할 수 있어야 한다.

이동에이전트도 자바 서블릿이나 CGI와 같이 <form> 태그를 사용하여 실행시키도록 하였다. 다음은 "3.1 활용 시나리오"의 그림 5를 표시하는 HTML 코드이다. 이 HTML 코드가 이동에이전트를 실행시키고 매개변수를 전달하는 방법은 자바 서블릿이나 CGI와 동일하다.

```
<form name="form" method="get" action="/agent/
  MyAgent">
<h4>검사할 웹페이지의 주소와 기준날짜를 입력하세요.
  <hr color=red>
목적 URL: <input type="text" name="target_url"
  size=30><br>
기준 날짜: <input type="text" name="date" size=20>
  <hr color=red>
<center><input type="submit" value="확인">
  <INPUT type="reset" value="재설정"></center>
</h4>
</form>
```

이동에이전트의 실행 결과를 웹브라우저로 확인하는 방법은 두가지를 고려하였다. 첫 번째로는 자바 서블릿과 같이 프린터 객체를 제공하여 웹브라우저에게 HTTP응답을 할 수 있도록 하였다. 두 번째로는 웹서버(CE 혹은 HCE)상에 웹페이지로 결과를 저장하고, 사용자가 나중에 결과를 확인하도록 하는 방법이다. 이 방법은 기존의 자바 파일 입출력 API로 프로그램할 수 있다. 이 두가지 방법을 조합하여 이동에이전트의 응용프로그래머 나름대로 사용자에게 결과를 제공하는 방법을 고안할 수 있을 것이다. 그림 9은 결과전송방법 두가지 예를 든 것이다. 그림 9의 "결과전송방법1"은 실행을 마치고 HCE로 돌아온 이동에이전트가 사용자의 웹브라우저에 직접 결과를 전송하는 것인데, 이때, 사용자는 이동에이전트를 실행시킨 후 이동에이전트가 결과를 전송해 줄 때까지 기다려야 한다.

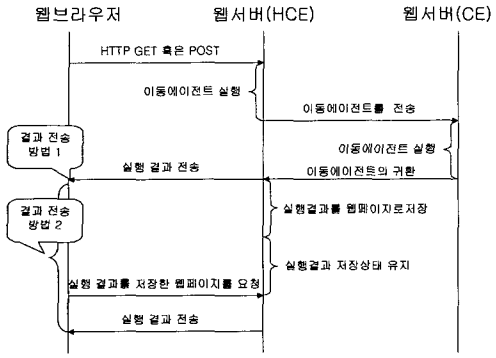

그림 9 사용자가 실행결과를 확인하는 과정

그러나, 사용자가 웹브라우저를 이용하여 이동에이전트를 실행하는 시점에서 이동에이전트가 NCE를 순회하여 결과를 얻어 내는데는 많은 시간이 경과할 수 있고, 또한 모빌 사용자의 경우 이동에이전트가 실행되는 동안 인터넷 연결을 계속 유지하지 않아도 되도록 해야 하므로, 이동에이전트가 실행결과를 HCE에 웹페이지로 저장하고 나중에 사용자가 저장된 웹페이지를 확인하도록 하는 방법이 "결과전송방법2"이다.

4.1.2 자바를 기반으로 한 이동에이전트 구현

이동에이전트는 프로그래밍언어에 많은 영향을 주기 때문에 이동에이전트를 위한 별도의 언어를 설계하는 것이 필요할 수도 있다. 그러나 웹환경에서 확립되어 있는 자바와 같은 기존언어를 이용하여 이동에이전트를 구현한다면, 자바언어로 제공되는 수많은 API 패키지들을 고려할 때, 보다 실용적이라고 생각된다. 자바 언어는 또한 이동에이전트 언어의 구현에 필요한 많은 특징들을 이미 가지고 있다.

자바의 타입시스템은 실행시간형검사를 지원한다. 자바언어에서 실행시간형검사를 지원하는 예는 여러가지이지만[17], 네트웍상에서 동적으로 다운로드한 코드의 형검사를 실행시간에 할 수 있다는 것은 이동에이전트의 구현에 있어서 중요한 특징이다[16]. 자바언어는 정적인 변수의 영역규칙을 사용하고 있으며, 원격지에서 다운로드 받은 클래스의 바인딩은 프로그래머가 명시적으로 별도의 클래스로더를 프로그램하도록 하고 있다. 대부분의 이동에이전트 언어들도 원격지의 자원에 대한 자동적인 바인딩을 지원하지 않는다. 자바를 이용하여 이동에이전트를 작성할 때, 이동하는 코드의 단위는 자연스럽게 클래스 객체가 된다. 코드의 이동을 위해서는 클래스객체를 바이트배열로 변환하는 마셜링(marshall- ing)과 언마셜링(unmarshalling)과정이 필요하다. 자바는 객체직렬화를 위한 방법을 지원하는데 이 방법을 사용할 경우 이동할 객체가 java.io.Serializable 인터페이스를 구현하여야 한다. 만약 클래스 객체 안에 직렬화되지 않는 객체가 포함되어 있다면 예외가 발생한다. 파일디스크립터와 같이 이동할 경우 의미가 없는 객체는 *transient* 형[17]으로 지정해 주어야 한다. 만약 파일디스크립터와 같은 변수를 이동에이전트가 목적 CE로 이동한 후에도 공유하려면 별도의 한정자(modifier)를 자바에 추가하고 자바가상기계에도 수정이 필요할 것이다. 본 논문에서는 이러한 접근대신 이러한 변수들에 대한 바인딩을 해제하고 목적 CE에서 새로 바인딩하는 복사전략[16]을 사용하도록 하였다. 이와 유사한 경우로 클래스변수가 있는데, 클래스변수는 객체에 포함되지 않으므로 직렬화가 이루어 질 수 없다. 따라서 목적 CE에 이동에이전트가 도착한 후에 새로운 값으로 바인딩하도록 한다. 프로세스간의 통신에 있어 세마포, 모니터, 메시지전송, RPC등의 기법을 사용할 수 있는데[18], RPC, 소켓등 자바의 풍부한 메시지 전송 API를 사용한다면, 원격지 변수에 대한 바인딩을 반드시 지원할 필요는 없다고 본다.

4.2 이동에이전트 구현을 위한 API

프로그래머가 자바언어를 사용하여 웹 기반 이동에이전트를 프로그램할 수 있도록 하는 최소한의 자바 API를 정의하였다. 다음에 본 논문의 주요 API를 기술한다.

4.2.1 이동에이전트의 생성 및 이동

public abstract class Agent extends java.lang.Object implements java.io.Serializable

```
void dispatch(java.net.URL destination)
void dispose()
void init(HttpAgentRequest, HttpAgentResponse)
void start()
void stop()
void destroy()
```

Agent 클래스는 이동에이전트 객체를 정의하기 위한 클래스이다. 객체의 이동을 위해 dispatch()라는 메소드를 정의했다. 이 메소드는 목적 CE의 URL을 매개변수로 한다. 또한, 이동에이전트가 실행을 완료한 후 스스로 소멸될 수 있도록 dispose() 메소드를 정의하였다. 이동에이전트의 생성과 실행은 자바 애플릿을 참고하여 사용자가 init(), start(), stop(), destroy() 메소드를 오버라이드하도록 하였다. init() 메소드는 웹브라우저가 전송하는 매개변수를 수신하고, 웹브라우저에 출력을 전송하기 위한 객체인 HttpAgentRequest와 HttpAgent Response 객체를 매개변수로 전달 받는다. 매개변수의 전달은 HTTP GET과 POST에 의해 이루어 있다. 이

메소드는 사용자가 오버라이드할 수 있는데, 이동에이전트가 HCE에 적재될 때에만 실행된다.

public abstract class AgentServer extends java.lang.Object

public static java.net.URL getServerURL(void)

이동에이전트가 스스로 다음 이동 목적 CE를 판단하고 현재의 CE에서 해야할 작업을 판단하기 위해서는 현재 CE의 URL을 알아낼 방법이 필요하다. CE는 AgentServer 클래스로 구현하는데, getServerURL() 정적 메소드는 CE의 URL을 반환한다.

4.2.2 사용자인터페이스

public interface HttpAgentRequest

public abstract String getParameter(String name)

이동에이전트와 웹브라우저 사이의 통신에 사용되는 API는 자바 서블릿의 형태를 참고하였다. 이동에이전트가 웹브라우저에서 매개변수를 전달받기 위해 HttpAgentRequest 인터페이스를 정의하였다. getParameter()는 지정된 이름의 매개변수 값을 반환한다.

public interface HttpAgentResponse

public abstract PrintWriter getWriter() throws IOException

public abstract void setContentType(String type)

실행결과를 웹브라우저로 전송하기위한 HttpAgentResponse 인터페이스를 정의하였다. getWriter()는 웹브라우저에 웹페이지를 전송하기 위한 프린터 객체를 반환한다. setContentType()은 HTTP응답의 Content-Type을 설정하기 위한 메소드이다.

4.2.3 API 사용 예제

다음 예제는 "3.1 활용 시나리오"에서 기술한 내용의 이동에이전트 코드를 보인 것이다.

```
public class MyAgent extends Agent {
    URL url;
    Date date;
    init(HttpAgentRequest ar, HttpAgentResponse as) {
        url = new URL(ar.getParameter("target_url"));
        date = DateFormat.getDateInstance().parse(ar.
            getParameter("date"));
        PrintWriter pout = as.getWriter();
        pout.setContentType("text/plain");
        pout.println("<H4>이동에이전트가 파견됩니다.
            <BR>결과는 <U>"+url+"agent/result.html
            </U>에서 확인하세요.</H4>");
        dispatch(url);
    }
    start() {
        // 목적 시스템에서 필요한 작업을 수행하고 결과를
        // result.html에 저장한다.
    }
}
```

init() 메소드는 HCE에서 이동에이전트의 실행이 초기화될 때에 한번만 실행된다. init() 메소드에서는 웹브라우저에서 전송한 매개변수를 HttpAgentRequest 객체로 전달 받은후 이를 각각 URL과 Date 객체에 기억한다. 그 다음에 HttpAgentResponse 객체를 사용하여 사용자의 웹브라우저에 "3.1 활용 시나리오"의 그림 6에 해당하는 안내 웹페이지를 표시하여 사용자에게 결과를 확인할 웹페이지의 주소를 알려준다. 이러한 작업이 끝난후 dispatch() 메소드로 사용자가 지정한 URL의 CE로 이동한다. start() 메소드는 이동에이전트가 이동한 CE에서 호출한다. 따라서 start()는 이동에이전트가 새로운 CE에 도착했을 때, 프로그램의 시작점이 된다.

5. 시스템 구현

시스템의 구현을 위해서 W3C의 Jigsaw, IBM의 Aglet, Sun의 JDK 및 JSDK의 오픈 소스를 참고로 하였다. 특히, Jigsaw는 W3C에서 자바로 개발한 웹서버로서 HTTP/1.1을 지원하며, 연구 목적으로 사용할 수 있도록 제공되고 있다. 이 웹서버의 중요한 특징으로 임의의 프로토콜 서버를 쉽게 통합할 수 있도록 잘 모듈화되어 있으며, 향후 웹상에서의 이동에이전트 지원을 고려하고 있다는 것이다.

5.1 시스템 구성

시스템의 설계에 있어 ① 웹브라우저와 웹서버(HCE) 간의 통신, ② CE들간의 통신을 고려하여 전체설계를 하였다(그림 10).

Jigsaw 웹서버에서 제공하는 HTTP 데몬(httpd)은 웹브라우저의 HTTP 요청에 대응하여 HCE 모듈을 호출한다. 그림 10에서 HTTP 데몬(httpd)과 HCE 모듈이 함께 표시되어 있다. HCE 모듈은 이동에이전트를 지역적으로 적재하여 자바 가상기계가 실행할 수 있도

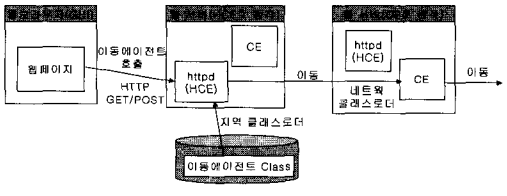

그림 10 시스템 구성

록 넘겨주고, 실행을 감독하며, 이동에이전트의 이동시에 직렬화 및 전송 작업을 지원한다. HCE 모듈을 Jigsaw에 통합하는 방법은 5.2.1에서 설명한다.

다른 웹서버(CE 혹은 HCE)로부터 이동해 오는 이동에이전트를 수신하고 실행하기 위해 웹서버의 내부에는 HTTP 데몬(httpd)과는 별도로 CE 모듈이 실행된다. CE 모듈은 이동에이전트의 수신을 위해 네트웍연결을 기다리는 단순한 네트웍클래스로더로 구현하였다. Jigsaw와의 통합 방법은 5.2.2에서 설명한다.

5.2 Jigsaw와 구현 모듈의 통합

5.2.1 HCE 모듈의 통합

Jigsaw는 철저한 모듈화를 통하여 프로그래머가 손쉽게 웹서버를 확장할 수 있도록 지원하고 있다[2]. Jigsaw는 자원, 프레임, 필터라는 개념을 사용한다. 자원은 웹서버상의 파일, 디렉토리 등을 자바 객체로 정의한 것이다. 프레임은 HTTP를 포함한 지정된 프로토콜을 사용하여 자원을 처리하기 위한 자바 객체로서, 각 자원은 프레임과 결합되어 있다. 필터는 각 자원에 관련하여 입출력의 인증과 같은 일을 하는 자바 객체로서, 프레임에 결합할 수 있다. 자원, 프레임, 필터는 모두 Jigsaw가 제공하는 클래스를 상속받아 프로그래머가 작성할 수 있다. 그림 11은 이동에이전트 /agent/HelloAgent.class에 대한 HTTP 요청을 처리하는 과정이다.

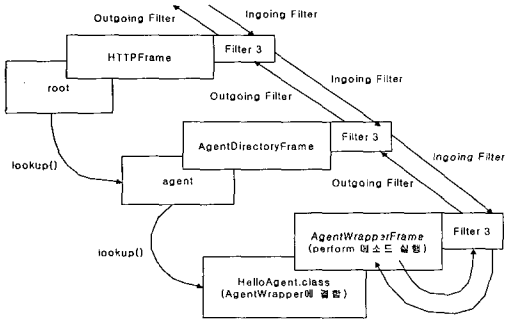

그림 11 웹서버(HCE)상의 이동에이전트의 실행

Jigsaw의 모든 자원은 레지스트리에 등록 되어 있는데, 이동에이전트의 클래스 파일도 여기에 포함된다. Jigsaw의 HTTP 데몬이 이동에이전트에 대한 HTTP 요청을 받으면 등록된 레지스트리의 root로부터 Hello Agent.class까지의 각 자원, 프레임, 필터의 lookup() 메소드가 호출되어 목적하는 자원을 찾는다. 다음으로 root로부터 각 입력 필터가 호출되고 최종적으로 목적 자원에 결합된 프레임(그림 11의 Agent Wrapper Frame)에 속한 perform() 메소드가 호출된다. perform() 메소드는 목적 자원의 처리에 필요한 작업을 수행하고 출력을 각 단계별 출력 필터에 전달한다. Jigsaw의 HTTP 데몬은 이렇게 전달받은 출력을 웹브라우저로 전송한다.

또한 이동에이전트의 클래스 파일은 AgentWrapper Frame이라는 자바 객체에 결합되어 있다. 결국 Agent WrapperFrame은 HCE 모듈로서, 이동에이전트의 적재 및 실행에 관한 일들을 한다. 이동에이전트가 저장된 디렉토리는 AgentDirectoryFrame이라는 프레임에 결합되어 있는데, 웹브라우저에서 이 디렉토리(/agent)에 접근하면 디렉토리의 내용을 표시하도록 되어 있으며 이동에이전트의 실행에는 직접적인 관련이 없다. 자원, 프레임, 필터의 자바 객체들은 Jigsaw에서 제공하는 관리자용 사용자 인터페이스를 사용하여 레지스트리에 등록할 수 있다.

5.2.2 CE 모듈의 통합

W3C에서 제공하는 Jigsaw의 내부에는 기본적으로 HTTP 데몬(httpd)과 웹서버 관리용 서버(JigAdmin)가 실행되도록 되어 있다. 그러나 Jigsaw는 HTTP 이외의 프로토콜을 지원하는 서버들을 기본적인 두 개의 서버와 함께 실행할 수 있도록 지원하고 있다. 이렇게 추가된 서버들은 Jigsaw 내부에서 실행되는 다른 서버들과 웹상의 자원을 공유할 수 있다. 새로운 서버는 자바로 개발되어야 하며, Jigsaw의 관리자가 새로운 서버의 이름과 클래스파일의 위치를 Jigsaw의 시스템 속성 파일에 등록하면 Jigsaw의 구동시 다른 서버들과 함께 Jigsaw의 내부에서 실행된다. Jigsaw는 이렇게 동시에 실행되는 여러 서버를 관리하기 위해 내부적으로 ServerHandler Manager[2]를 가지고 있다. 구현된 CE 모듈은 HTTP를 사용하지 않으므로 이러한 방법으로 웹서버의 구동시에 같이 실행되도록 하였다.

6. 성능평가

6.1 평가 방법의 고려

이동에이전트의 성능은 인터넷 환경, 이동에이전트 실행환경의 구조, 이동에이전트의 응용분야, 자바언어의 성능에 영향을 미치는 가비지콜렉션 등 많은 요소들의 영향을 받는다. 따라서 비교하는 이동에이전트 시스템들을 동일한 조건하에 실험하기 위한 환경설정을 하기가 쉽지 않다[19]. 이동에이전트 시스템과 클라이언트/서버 모델과의 비교에서도 이러한 어려움이 있으며, 특히 이 경우 응용 시나리오와 환경을 구성하는 방법에 따라 두 모델의 성능이 많은 차이가 나게 되는데, 클라이언트와

서버 간에 많은 통신이 필요한 응용의 경우 이동에이전트가 효율적인 것으로 나타나고 있다[20].

위에서 기술한 이동에이전트 시스템간의 성능비교나, 혹은 클라이언트/서버 모델과의 성능비교는 본 논문에서 제안하는 모델의 정당성을 평가하는데는 충분하지 않다. 본 논문에서 제안하는 모델의 정당성은 기존의 이동에이전트가 웹프로그래밍에 사용되는 방법과 비교하면 될 것으로 생각된다. "2.4 웹을 고려한 기존의 이동에이전트 시스템"에서 Telescript의 Tabriz와 IBM Aglets의 Fiji에 대하여 기술하였는데, 본 논문에서 제안하는 모델은 이동에이전트가 서버측에서 실행이 초기화된다는 점에서 Tabriz의 경우와 더 유사하다고 생각된다. Fiji의 경우 웹서버가 라우터로 사용되는 경우로 다른 관점에서 평가할 수 있을 것이다[21]. Tabriz는 CGI를 사용하여 웹서버와 이동에이전트를 연결한다. 따라서, 본 논문에서 제안하는 모델은 기존의 이동에이전트 시스템을 CGI를 사용하여 웹서버와 연동했을 경우와 비교하므로서 평가할 수 있다고 본다.

6.2 성능 평가 및 분석

본 논문의 시스템과 가능한한 동일한 조건으로 비교하기 위해 Jigsaw와 Aglet을 CGI로 연동한 결과를 비교하였다. 실험은 Sun의 Ultra-2 기종 1대와 PC 1대가 인트러넷으로 연결된 상황에서 본 논문의 예제를 사용하여 수행하였다. 이동에이전트 시스템들의 성능 비교에 있어서 전체적인 실행시간을 비교하거나 혹은 이동에이전트의 생성 시간, 이동 시간, 메시지전송 시간 등의 개별 항목을 비교할 수 있다[19]. 본 시스템에서의 평가 항목은 이동에이전트의 생성 시간을 선택했다. IBM Aglets와 같이 완성도가 높은 시스템과 프로토타입 시스템을 비교했을 때 이동에이전트의 생성 시간, 메시지전송 시간등에서 프로토타입 시스템이 앞서게 된다[20]. 그러나 이것은 완성된 시스템의 복잡성 때문으로, 본 시스템과의 비교에서 의미를 가지지 못한다. CGI를 사용한 방법과 본 논문의 시스템의 모델이 차이가 나는 부분은 독립된 이동에이전트 시스템의 이동에이전트를 CGI를 사용하여 호출하는 시간과 웹서버에 통합된 이동에이전트 시스템이 이동에이전트를 생성하는 시간으로 비교할 수 있다고 본다. 다음에 실험결과를 보였다(표 4).

표 4 이동에이전트의 로딩 시간

	1차 호출(ms)
Aglet 서버 구동시간	3620
Aglet 로딩 시간	220
Aglet 클라이언트 구동시간	2960
본 시스템의 이동에이전트 로딩 시간	60

예측한대로 Aglet의 로딩 시간 보다 본 시스템의 로딩 시간이 훨씬 빠르게 측정되었다. 전술한대로 이것은 Aglet의 복잡성 때문이라고 이해할 수 있다. 그러나 웹서버에 이동에이전트 시스템이 통합되어 있지 않은 Aglet의 경우 CGI를 통하여 Aglet의 실행을 초기화해야 한다. Aglet 프로그래머는 Aglet 서버와 함께 Aglet이 실행되도록 프로그램할 수 있다. 이렇게 작성된 Aglet을 CGI를 통하여 호출할 경우 Aglet 서버 구동시간과 Aglet의 로딩 시간을 합한 만큼의 시간이 필요하다. Aglet 서버를 미리 실행시켜 놓고, Aglet의 로딩을 요청하는 좀더 가벼운 Aglet 클라이언트를 작성한 경우에도 결과는 크게 향상되지 않았다. 그 이유는 Aglet 클래스 라이브러리의 로딩 시간 때문인 것으로 판단된다. 실험 결과로 나타난 Aglet 서버의 구동시간이나 Aglet 클라이언트의 구동시간은 웹서버에 이동에이전트 시스템이 통합되어 있다면 불필요한 시간이며, 이 시간이 상대적으로 길다는 것은 본 논문에서 제안하는 모델의 정당성을 뒷받침하는 결과라고 볼 수 있다.

7. 결 론

설계된 이동에이전트 시스템은 웹환경에서 이동에이전트를 프로그래밍할 수 있도록 특화하였다는 점에서 기존의 이동에이전트 시스템과 구별된다. 웹브라우저를 그래픽사용자인터페이스를 위한 도구로 사용하고, 네트웍(NCE)상에 분산되어 있는 컴퓨터들에 존재하는 웹서버들을 이동에이전트의 CE로 추상화하였다. 또한 웹프로그래머들에게 웹페이지상에서 이동에이전트의 실행을 지시하고 결과를 확인하도록 하는 방안을 제시하였다. 이러한 방법은 기존의 CGI나 자바 서블릿의 프로그래밍에 익숙한 웹프로그래머들이 쉽게 이해하고 접근할 수 있게 해 주며, 웹 응용프로그램상에서 이동에이전트의 이용을 용이하게 해 준다.

시스템의 구현을 위하여 W3C의 Jigsaw 웹서버를 확장하였다. Jigsaw는 자바로 구현되어 있어 객체지향언어인 자바의 코드 재활용성을 잘 지원하고 있을 뿐 아니라 전체적인 시스템 설계가 확장성을 지원하도록 되어 있어, 자바를 기반으로 하는 이동에이전트 시스템을 웹서버에 통합하기 위한 플랫폼으로서 적합하였다. 본 논문의 시스템을 구현하는데 있어 이러한 Jigsaw의 특성을 최대한 활용하고, 이동에이전트 시스템과 웹서버의 통합에 있어 Jigsaw의 모듈 확장 방법을 따랐다. 따라서 구현된 시스템을 사용하는 웹서버의 관리자는 자바 서블릿이나 웹페이지와 같은 Jigsaw의 다른 자원들을 관리하는 것과 동

일한 방법으로 이동에이전트의 자원을 관리할 수 있다.

본 논문의 시스템에서는 자바언어를 사용한 이동에이전트의 구현을 위하여 자바의 언어적 특성을 충분히 고려하여 프로그래머가 이동에이전트의 프로그래밍시에 혼란이 없도록 하였으며, 이를 바탕으로 이동에이전트의 프로그래밍을 위한 API를 정의하고 구현하였다. 또한 구현된 시스템 상에서 웹과 관련한 간단하지만 실용적인 테스트 프로그램을 작성하여 실행하므로서 제안된 프로그래밍 모델의 실효성을 보였다.

구현된 이동에이전트 시스템은 웹환경에서의 프로그래밍 특성에 주력한 관계로 기존 이동에이전트 시스템과 같이 시스템의 완성도를 높이기 위한 API를 지원하는 문제는 향후의 연구과제로 하고 있다. 추가되어야 할 문제로는 이동에이전트간의 메시지전달방법, 실행중인 이동에이전트의 관리방법등이 있다. 웹 기반 이동에이전트 프로그래밍 모델을 발전시키기 위해서는 이러한 연구들이 계속되어야 할 것으로 본다.

참 고 문 헌

[1] http://www.trl.ibm.co.jp/aglets/index.html

[2] http://www.w3.org/Jigsaw

[3] http://www.w3.org/MobileCode/

[4] F. Rouaix. "A Web navigator with applets in Caml," In Proceedings of the 5th International World Wide Web Conference, Computer Networks and Telecommunications Networking, volume 28, pages 1365--1371. Elsevier, May 1996.

[5] Luca Cardelli, "Obliq: A Language with Distributed Scope," Technical Report 122, Digital Equipment Corporation, Systems Research Center, 1994

[6] Jim White, "Mobile Agents: A white paper," General Magic, Inc. 1996

[7] Mitsuru Oshima, Guenter Karjorth, and Guichi Ono, "Aglets Specification 1.1 Draft," 1998

[8] http://www.objectspace.com/products/voyager/

[9] R. Koblick. "Concordia," Communications of the ACM, 42(3):96--97, March 1999

[10] Niranjan Suri et al.,, "An Overview of the NOMADS Mobile Agent System" In Proceedings of ECOOP'2000, Nice, France, 2000.

[11] Thomas Gschwind, "Comparing Object Oriented Mobile Agent Systems" In Proceedings of ECOOP'2000, Nice, France, 2000.

[12] Colin G. Harrison, David M. Chess, Aaron Kershenbaum, "Mobile Agents: Are they a good idea?," Technical Report, IBM T.J. Watson Research Center, 1995

[13] Ghezzi, C and Vigna, G., "Mobile Code Paradigms and Technologies: A Case Study"; In Proceedings of the 1 st International Workshop on Mobile Agents; April 1997

[14] A. Fuggetta, G. Picco, G. Vigna, "Understanding Code Mobility," IEEE Transactions on Software Engineering, 24(5):352-361, May 1998.

[15] P. Domel, A. Lingnau, and O. Drobnik, "Mobile Agent Interaction in Heterogeneous Environments," LNCS, vol. 1219, pp. 136--148, Apr. 1997.

[16] G. Cugola, C. Ghezzi, G.P. Picco and G. Vigna, 1996, "Analyzing Mobile Code Languages," 2nd Int. Workshop on Mobile Object System; also in LNCS 1222, pp. 93-109

[17] James Gosling, Bill Joy, Guy Steele, "The Java Language Specification," http://java.sun.com/docs/books/jls/index.html, p.302

[18] Narain Gehani, Andrew D. McGettrick, "Concurrent Programming," p.62, Addison-Wesley, 1988

[19] M. Dikaiakos and G. Samaras. "Quantitative Performance Analysis of Mobile-Agent Systems: A Hierarchical Approach," Technical Report TR-00-2, Department of Computer Science, University of Cyprus, June 2000.

[20] L. Ismail and D. Hagimont. "A performance evaluation of the mobile agent paradigm," In Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications(OOPSLA'99), pages 306-313, 1999.

[21] G. Samaras, M. D. Dikaiakos, C. Spyrou, and A. Liverdos. "Mobile Agent Platforms for Web Databases: A Qualitative and Quantitative Assessment," Submitted for publication, May 1999.

송 성 훈

1981년 ~ 1985년 홍익대학교 전자계산학과. 1985년 ~ 1987년 홍익대학교 대학원 석사과정 전자계산학과. 1987년 ~ 1993년 삼성데이타시스템 선임 연구원. 1993년 ~ 현재 혜천대학 컴퓨터통신계열 교수. 1994년 ~ 현재 홍익대학교 대학원 박사과정 전자계산학과 수료. 관심분야는 프로그래밍 언어, 객체지향 언어, 인터넷, 정보보호

원 유 헌

1972년 성균관대학교 수학과(B.S). 1975년 한국과학기술원(KAIST) 전자계산학과(M.S). 1975년 한국과학기술 연구소 연구원. 1985년 고려대학교 대학원 컴퓨터과학과(Ph.D). 1986년 미국 RPI대학 교환교수. 1976년 ~ 홍익대학교 컴퓨터공학과 교수. 관심분야는 프로그래밍 언어, 객체지향 언어, 실시간 언어, 실시간 시스템, 멀티미디어 시스템, 정보보호