이동 에이전트의 개념과 응용

석황희* · 김인철*

1. 서 론

최근 들어 컴퓨터 및 통신 기술이 급속히 발전함으로써 이들이 자연스럽게 상호 결합하여 새로운 컴퓨팅 서비스를 생성하고 있다. 예로, 휴대용 네트워크 컴퓨터나 PDA(personal digital assistant) 등의 이동 호스트(mobile host)는 원하는 장소로 자유롭게 이동하면서 사용자의 요구에 따라 작업을 수행하고, 필요한 경우 인터넷에 접속하여 작업을 수행할 수 있다. 이처럼 이동 컴퓨팅(mobile computing) 환경에서 사용자의 요구에 따라 여행 정보나 주식 정보 등을 인터넷에서 검색하여 보여주거나 그 밖의 일상적인 활동이나 업무를 사용자를 대신하여 처리해주는 자율적인 프로세스(autonomous process)를 일반적으로 에이전트(agent)라 하고 여러 유형의 소프트웨어 에이전트들이 존재할 수 있으나, 에이전트가 처음 수행을 시작한 특정 호스트 컴퓨터에 머물러 있지 않고 스스로 네트워크 상의 여러 컴퓨터 시스템들 사이를 옮겨 다니면서 사용자를 대신하여 전자상거래(electronic commerce) 행위나 정보 검색(information retrieval) 행위, 네트워크 관리(network management) 등을 수행할 수 있는 에이전트들을 특히 이동 에이전트(mobile agent)라고 한다.

이러한 이동 에이전트는 일시적인 접속 세션(session)동안에 에이전트를 통해 정보를 주고받을 수 있으므로 지속적으로 연결상태를 유지할 필요가 없기 때문에 네트워크의 낮은 대역폭과 배터리의 제한을 받는 이동 클라이언트에 매우 유리하다. 또한 이동에이전트에 기반을 두고 트랜잭션 처리(transaction processing)를 하는 경우, 에이전트만 상태정보를 가지고 있으면 클라이언트쪽과 서버쪽 어디에도 처리 상태에 대한 정보를 보관할 필요가 없다. 이 외에도 이동에이전트는 질의(Query)에 맞는 충분한 정보를 얻을 수 있는 의미적 정보검색(Semantic Information Retrieval)과 클라이언트와 서버간의 유기적인 연결을 지원하는 의미적 라우팅(Semantic Routing)을 용이하게 해준다. 하지만, 이러한 장점들이 있는 반면 이동 에이전트에는 낮은 전송효율, 바이러스처럼 나쁜 영향을 미칠 수 있다는데 대한 보안문제, 수많은 서버들 중에서 자신이 원하는 서비스를 제공하는 서버를 어떻게 효율적으로 찾는가와 이동 에이전트 서버가 있는 시스템으로만 이동이 가능하다는 점, 그리고 표준화된 이동 에이전트의 실행 환경의 부재 등 해결해야 할 문제가 많이 남아있다.

본 논문에서는 이동 에이전트에 대한 기초 개념과 개발환경, 기술 표준화, 그리고 이동 에이전트를 이용한 응용시스템들에 대해 살펴본다. 다음 장에서는 에이전트에 대한 전반적인 개념을, 제3장에서는 이동에이전트를 개발할 수 있는 환경에 대해 소개한다. 제 4장에서는 이동에이전트를

*경기대학교 전자계산학과

활용한 응용 시스템들을 소개하고 제 5장에서는 이동 에이전트의 기술 표준화를 살펴본다. 마지막으로 제 6장에서는 결론을 기술한다.

2. 이동 에이전트의 개념

일반적으로 소프트웨어 에이전트(software agent)는 자율적(autonomous)이고 주도적(pro-active)이고 반응적(reactive)인 특성을 가지고 있으며 학습(learn)하고 협동(cooperative)하고 이동(mobile)하는 능력을 바탕으로 사용자를 대신해서 행동하는 개체라고 할 수 있다. 특히 이동 에이전트(mobile agent)는 네트워크상의 서로 다른 위치간에 이동할 수 있는 소프트웨어 에이전트이며, 다음과 같은 여러 가지 형태의 모델로 특성화될 수 있다.

- 생명 주기 모델(life-cycle model) - 에이전트에 대한 생성(creation), 파괴(destruction), 시작(start), 정지(stop) 등의 서비스를 기술한다.
- 계산 모델(computational model) - 자료 조작과 쓰래드 제어 primitive를 포함하는, 에이전트의 계산 기능(computational capability)에 관한 것이다.
- 보안 모델(security model) - 에이전트가 네트워크 자원에 접근하는 것 뿐 아니라 네트워크로부터 에이전트의 내부에 접근하는 방법을 기술한다.
- 통신 모델(communication model) - 에이전트간 그리고 에이전트와 다른 개체와의 통신을 정의한다.
- 이동 모델(navigation model) - 서로 다른 위치에 있는 두 개체간에 에이전트를 운반하는 것에 대한 모든 문제를 처리한다.

이동 에이전트는 이동 코드(mobile code)나 원격 객체(remote object)를 이용하여 구현될 수 있는데, 이동 코드를 이용한 대표적인 접근 예로는 Agent Tcl, Telescript 등이 있고, 원격 객체를 이용한 예로는 Aglets Workbench 등이 있다. 이동 에이전트를 사용하기 위해서는 이동성을 제공하는 에이전트 프레임워크(agent framework)가 있어야 하며, 이러한 이동 에이전트 프레임워크는 이동성뿐만 아니라 위에서 기술한 그 밖의 일반적인 에이전트 모델을 뒷받침할 수 있는 모든 설비(facility)들을 가지고 있어야 한다.

이동 에이전트는 네트워크를 통해 이동하여 원거리에 있는 호스트에서 실행될 수 있는 프로그램이나 사람마다 강조하는 측면에 따라 여러 가지 이름으로 사용되기도 한다. 예를 들면,

- 이동 계산(mobile computation) - 프로그램의 실행이 한 네트워크 노드(node)에서 시작하여 다른 네트워크 노드(node)에서 이어질 수 있음을 강조한다.
- 이동 객체 (mobile object) - 이동 computation을 자체적으로 가지고 있는 자율적인 객체로, 이동시에 상태와 코드를 수행할 수 있음을 강조한다.
- 확장 시스템 (extensible system) - 원거리에 있는 시스템의 기능성을 확장시키는데 사용될 수 있음을 강조한다.

이동 에이전트를 사용하는 이유는 전통적인 클라이언트/서버 모델이 갖는 장점 이외에 다음과 같은 장점을 추가적으로 가지고 있기 때문이다.

- 효율성(efficiency) - 데이터(data)가 계산(computation) 쪽으로 이동한다라고 하기보다는 계산(computation)이 데이터(data) 쪽으로 이동하기 때문에 네트워크 자원을 덜 소모한다.
- 내고장성(fault tolerance) - 필요한 데이터를

얻기 위해 네트워크 상의 다른 호스트나 접근 경로를 선택할 수 있다.

- 이동 컴퓨팅 환경 (mobile computing environment) - 노트북, PDA 등 이동 단말기를 이용한 이동 컴퓨팅 환경에서 호스트 서버에 대한 지속적인 접속이 필요 없다.
- 쉬운 유지 및 분산(easy maintenance and distribution) - 이동 에이전트를 기초로 구성된 소프트웨어 시스템은 한 곳에서 수정, 보완한 다음 쉽게 분산 설치하여 실행할 수 있다.
- 편리한 패러다임(convenient paradigm) - 계산(computation)이외의 통신 채널과 같은 하부의 세부적인 사항들을 숨길 수 있다.
- 유연성 및 맞춤성(flexibility and customization) - 클라이언트와 서버 모두 프로그래밍에 의해 서로의 기능을 확장할 수 있다.

3. 이동 에이전트 개발 환경

3.1 Telescript

Telescript[12]는 다른 RP-언어처럼, Telescript 엔진에 의해 실행되는 인터프리터(interpreter) 언어이다. Telescript 언어는 *Telescript Development Environment*(TDE)라는 개발 환경과 함께 제공된다. 일반적으로 한 Telescript 에이전트는 그림 1과 같은 하나의 전자시장(electronic marketplace) 내에서 활동한다. Telescript 엔진이 동작하는 사이트(site)를 장소(place)라 부르며, 하나의 전자시장은 이러한 장소들의 집합으로 구성된다. 장소들은 그림 1과 같이 내포(nest)될 수 있다. Telescript 에이전트는 go 명령으로 장소를 이동하고 서비스는 meet 명령으로 사용된다. 이것은 그 두 에이전트 즉, 클라이언트 에이전트와 서버 에이전트가 통신하기 위해 서로 만난다는

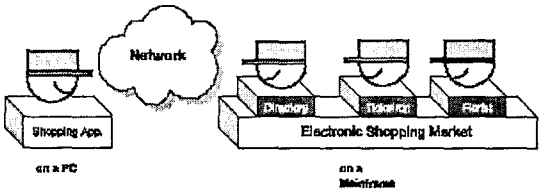


그림 1. Telescript 전자시장

것을 의미한다.

안전을 위해 각 에이전트는 특정 장소로 이동하기 전에 permit를 제시해야한다. permit는 접속하고자하는 지역(region)과 서비스의 자원에 에이전트가 접근(access)하는 것을 제어한다. Telename은 에이전트를 구분하는데 사용된다. 따라서 permit 개념으로 접근은 제한될 수 있고 각각의 행위(action)는 에이전트의 식별자(identifier)를 사용하여 기록(log)될 수 있다. 이것은 지불 서비스와 같은 계산 메쏘드를 가능하게 한다 : 만일 에이전트가 서비스를 사용했다면 그것은 소위 Teleclicks라는 전자화폐 단위로 이 노력의 대가를 지불할 수 있다. Teleclicks와 화폐간의 환율을 정의할 수 있다.

Telescript는 다른 객체지향의 고수준 언어처럼 뿌리가 Object 클래스(class)인 계층화된 클래스 집합을 사용한다. 상위 클래스(superclass)로부터 파생되는 클래스들은 인터페이스(interface)와 구현 부분을 물려받을 수 있다. 사용자가 C++와 매우 유사한 고수준의 Telescript를 다루는 반면 그 엔진은 저수준의 인터프리트된 형태를 다룬다. 컴파일러는 두 가지 다른 형태의 Telescript 언어 사이를 번역한다.

Telescript 언어의 특성은 다음과 같다 :

- 동적이다(Dynamic) : 이동하는 에이전트에 의해 사용되는 클래스(class)가 목적지(destination)에 알려져 있지 않아도, 그 클래스(class)

들이 에이전트의 일부이고 함께 가기 때문에 에이전트는 기능을 계속한다.

- 영속적이다(Persistent) : 프로그램 계산기를 포함하는 에이전트의 모든 데이터는 비휘발성(non-volatile)으로 저장되기 때문에 데이터는 전력이 차단된 경우에도 손실되지 않는다.
- 이식가능하고 안전하다(Portable and Safe) : 명령은 Telescript 엔진에 의해 인터프리트된다. 따라서, 컴퓨터의 자원을 직접 조작할 수 있는 에이전트는 없다. 이것은 기계를 바이러스로부터 보호한다.
- 통신중심이다(Communication-centric): PostScript처럼 언어는 통신목적으로 전문화되는데, 복잡한 이미지를 표현하기 위해 전문화된다. 이것은 프로그래머가 통신을 하기 위한 고수준 명령을 사용할 수 있게 해준다(예 - go, meet).

응용시스템 개발자는 다음과 같은 3가지의 API를 통해 Telescript 엔진에 접근한다 :

- Storage API : 엔진의 장소와 에이전트를 보호하기 위해 엔진이 사용하는 플랫폼의 비휘발성 기억장소에 대한 접근을 제공하기 위한 API이다.
- Transport API : 다른 엔진과 에이전트를 주고받기 위한 고수준의 API이다.
- External application API : 다른 언어로 쓰여진 애플리케이션 부분과 상호 작용하기 위한 API 이다.

두 Telescript 엔진은 Telescript 프로토콜을 통해 통신을 할 수 있다. 프로토콜 모음은 광범위한 네트워크 형태 즉, TCP/IP를 기반으로 하는 것들, X.25 기반 프로토콜, 혹은 전자 우편 등을 지원한다. Telescript 프로토콜은 두 단계로 동작한다 : 하위 단계는 에이전트 전송을, 상위 단계는 암호화(encoding)와 해독(decoding)을 수행한다.

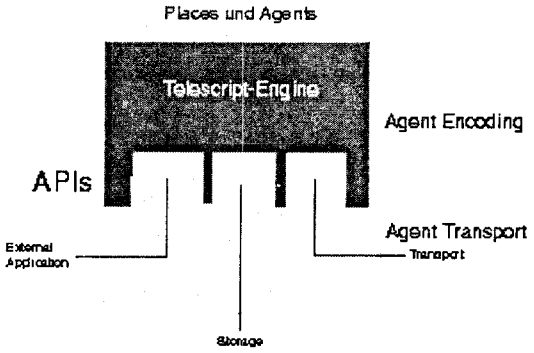


그림 2. Telescript API와 프로토콜

3.2 Aglets Workbench

IBM의 Aglets Workbench[2]는 자바(Java) 프로그래밍 언어로 쓰여지는 플랫폼 독립적인 이동 에이전트의 개발 환경이다. 그리고 aglet은 한 호스트에서 다른 호스트로 이동할 수 있는 하나의 자바 객체이다. Aglets Workbench에서는 응용 개발자를 위해 J-AAPI(Java Aglet Application Programming Interface)를 제공하고 있다.

다음은 aglet 객체 모델을 구성하는 중요한 기초 개념들을 알아본다. 그림 3은 이들간의 관계를 그림으로 보여주고 있다.

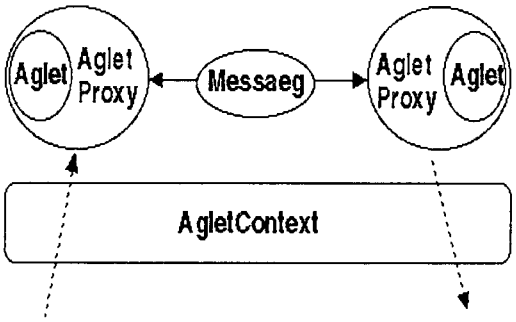


그림 3. Aglet 객체 모델

Aglet :

컴퓨터 네트워크 상에서 허용되는 호스트를 방문하는 이동 자바 객체이다. 호스트에 도착한 후에 자신의 실행 쓰래드(thread)상에서 실행하기 때문에 자율적이고, 들어오는 메시지에 반응하는 능력 때문에 반응적이다.

Aglet Context :

aglet의 작업공간으로, 호스트 시스템이 악의 있는 aglet에 대항하여 안전하게 되는 동안 단일 실행 환경에서, 동작하고 있는 aglet을 유지하고 관리하기 위한 수단을 제공하는 부동 객체이다. 컴퓨터 네트워크상의 한 노드는 다중 context를 갖는 호스트일 수 있다.

Aglet Proxy :

aglet의 대리인으로, public method로 직접 접근해오는 것으로부터 aglet을 보호하는 aglet의 방패역할을 한다. 또한 aglet에 위치 투명성(transparency)을 제공한다. 즉, aglet의 실제 위치를 숨길 수 있다.

Message :

aglet들간에 상호 교환되는 객체로, aglet간에 비동기적인 메시지 전달뿐만 아니라 동기적인 메시지 전달도 고려한다. 메시지 전달은 느슨하게 연결된 유형으로 협동하고 정보를 교환하기 위해 aglets에 의해 사용될 수 있다.

Message Manager :

들어오는 메시지의 동시성 제어를 담당한다.

Itinerary :

aglet의 이동 계획으로, 중요한 이동 유형과 경로지정을 위해 편리한 추상적 개념(abstraction)을 제공한다.

Identifier :

각각의 aglet에 지정되는데, aglet의 수명 전반에 걸쳐 유일하고 불변이다.

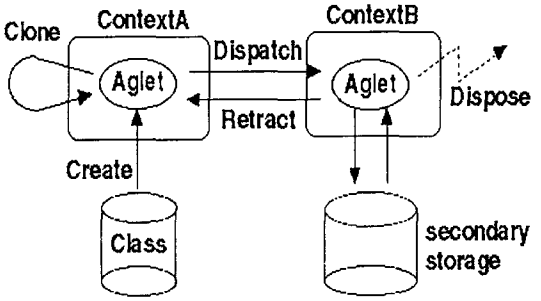


그림 4. Aglet의 생명주기

aglet 객체 모델이 지원하는 행위는 그림 4와 같이 생성(creation), 복제(cloning), 이동(dispatching), 복귀(retraction), 비 활성화(deactivation), 활성화(activation), 소멸(disposal), 메시지 전송(messaging) 등을 포함한다. aglet의 생성(creation)은 context상에서 일어난다. 새로운 aglet은 context에 삽입되는 식별자(identifier)를 할당받고 초기화된다. aglet은 초기화가 성공하자마자 실행을 시작한다. aglet의 복제(cloning)는 같은 context상에 원 aglet에 거의 동일한 복사본을 만든다. 단지 차이는 할당받은 식별자이고 실행은 새로운 aglet에서 다시 시작한다. 단, 실행 쓰래드는 복제되지 않는다. 한 context에서 다른 context로 aglet을 이동(dispatch) 시키는 것은 현재의 context로부터 그것을 제거하고 그것이 실행을 다시 시작할 목적지 context로 삽입한다. aglet의 복귀(retraction)는 현재의 context로부터 aglet을 제거하고 복귀가 요청된 context로 삽입하게 된다. aglet의 비 활성화(deactivation)는 현재의 context로부터 aglet을 일시적으로 제거하고 다른 기억장소에 그것을 저장하는 것을 말하며

aglet의 활성화(activation)는 현재의 context로 그것을 복구하는 것을 말한다. aglet의 소멸(disposal)은 aglet의 실행을 멈추고 그것을 현재의 context로부터 제거하는 것이다. aglet들간의 메시지 전송(messaging)은 동기 및 비동기적으로 메시지를 보내고 받고 이들을 처리하는 것을 말한다.

3.3 Agent Tcl

Agent Tcl[14]은 UNIX 시스템에 보편화되어 있는 대표적인 스크립트 언어인 Tcl(Tool Command Language)을 바탕으로 이동 에이전트를 기술할 수 있도록 확장한 것이다. 따라서 Agent Tcl은 이동성을 제공하는 대표적인 비상업용 내포형 스크립트 언어(embeddable script language)이다. Agent Tcl로 씌여진 에이전트는 jump 명령을 이용하여 기계들간을 이동한다. 에이전트는 jump 후에 즉시 목적지 기계에서 실행을 재개한다.

그림 5에서 보듯이 Agent Tcl의 구조는 agent script를 실행하는 확장된 Tcl 인터프리터(interpreter) 부분과 에이전트가 보내질 수 있는 호스트마다 수행되는 서버(server) 부분으로 구성된다.

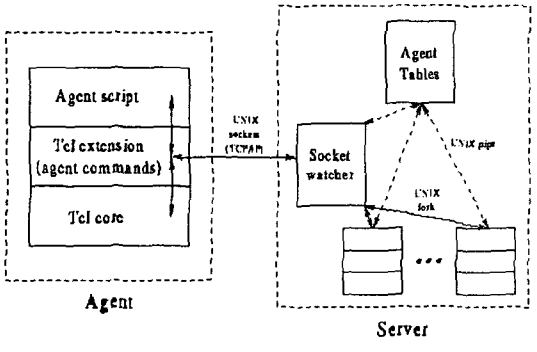


그림 5. Agent Tcl 구조

Agent Tcl은 에이전트의 이동 외에 다음의 기능들을 지원한다 :

- 메시지 전달 (Message passing) - 에이전트는 서로에게 메시지를 보낼 수 있다.
- 직접 접속 (Direct connections) - 에이전트는 대용량의 자료 전송을 위해 서로와의 직접적인 접속을 설정할 수 있다.
- Tk 4.0 - 에이전트는 현재의 머신에 그래픽 인터페이스(graphical interface)를 보이기 위해 Tk 4.0을 이용한다.
- C/C++ - 부동 에이전트는 C/C++로 쓰여질 수 있다.
- 에이전트 생성과 복제 - 에이전트는 자식(child) 에이전트를 생성할 수 있고 자신을 복제할 수 있다.
- 일반적인 타임아웃과 재시도 메카니즘 - 에이전트는 필요한 만큼 여러 번 명령을 재 시도할 수 있고 Tcl 코드를 마음대로 타임아웃 시킬 수 있다.
- 기본적인 보안 - 인정되지 않은 기계로부터 온 어떠한 에이전트, 메시지 혹은 접속도 무시된다. 시스템 관리자는 인정된 기계들을 명시한다.

4. 이동 에이전트 응용

4.1 네트워크 관리(Network Management)

많은 수의 분산된 이질적인 장치와 많은 관리 데이터를 다루는 네트워크 관리 분야는 대표적인 이동 에이전트 응용 분야이다. 특히 다음과 같은 세부 분야에 대한 이동 에이전트 연구가 주로 이루지고 있다.

- 네트워크 모델링(Network modeling)
- 결함 관리(Fault management)

- 회계 관리(Accounting management)
- 구성 관리(Configuration management)
- 수행 관리(Performance management)
- 보안 관리(Security management)

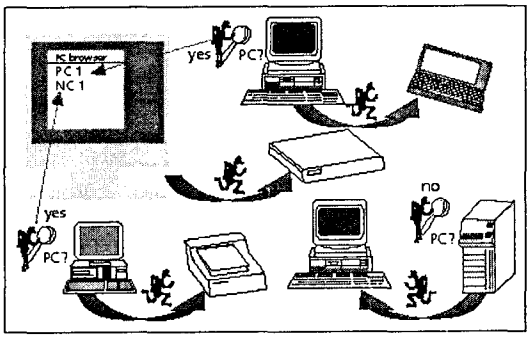


그림 6. 이동 에이전트를 이용한 네트워크 모델링

Perpetuum 프로젝트[4]에서는 그림 6과 같이 deglet이라 불리는 이동 에이전트를 이용해 네트워크 상의 새로운 장치들을 발견하고 이 정보를 바탕으로 네트워크 모델을 생성하는 응용 시스템을 개발하였다. 네트워크 장치의 자동 발견은 네트워크 관리 시스템의 기본 기능중 하나이다. 장치 발견의 복잡성이 증가함에 따라, 전통적인 클라이언트/서버 접근방법을 사용해서 이행하기가 더 어려워진다. deglet이라고 하는 이동 에이전트는 생성자에게 방문한 노드의 식별자를 보내는 단일 업무로 생성될 수 있다. 그러면 deglet은 네트워크로 주입되고 구현된 이동 패턴에 의해 이동한다. ping 알고리즘에서와 같이 해결할 비슷한 문제가 몇 가지 있지만, 그 방법은 네트워크에 대한 세부적인 사전 지식을 가질 필요가 없기 때문에 융통성이 있다. 예를 들어, 업무의 종료는 홉(hop)이나 특정 노드의 평균 방문 수를 셈으로써, deglet의 내부에서 휴리스틱(heuristic)하게 결정될 수 있다.

역시 Perpetuum 프로젝트에서는 그림 7과 같이 네트워크 결함을 진단하는데도 네트워크 모델링과 같은 원리를 적용하였다. 결함의 감지는 분업화된 네트워크 모델을 세우는 과정이다. 예로, 어떤 임계치를 넘는 이용을 하는 노드를 선택적으로 발견하는 일을 수행하는 간단한 deglet은 초과 이용한 노드의 모델을 세운다. 장치 발견시 어떤 제약의 침해를 나타내는 것들을 찾아낼 수 있다면 그 제약을 시험하는 deglet 에이전트는 결함감지 기능을 수행하는 것이다. 때로는 deglet를 이용해 보다 향상된 시험을 수행하거나 장치의 복구절차를 실행하게 할 수 있다.

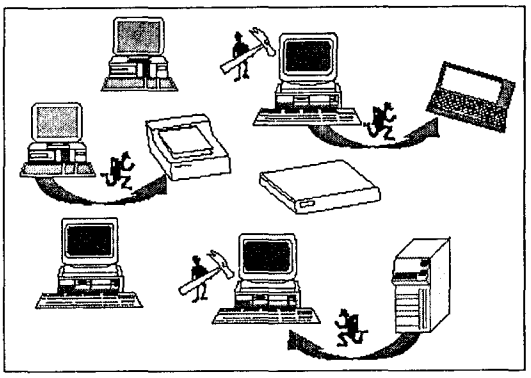


그림 7. 유지보수 에이전트

이동 에이전트는 호스트 자원과 많은 다른 서비스에 대해 안전하고 간접적인 접근을 제공하는 인터페이스를 통해 호스팅 노드와 상호 작용할 필요가 있다. Perpetuum 프로젝트에서 이 인터페이스를 virtual managed component(VMC)라고 한다. VMC는 네트워크 장치의 벤더에 의해 설계되고 구현된다. 장치가 이동 에이전트를 받아들일 수 없다면 proxy상에 존재할 수도 있다. 벤더가 장치의 VMC에 포함할 수 있는 설비 중 하나는 다운로드 가능한 자료 제시 애플릿인데, 그것은

이질적인 네트워크 환경 관리를 더 쉽게 해준다. 장치가 네트워크에 접속될 때, 그 장치에서 사용할 수 있는 기능에 대한 지식이 네트워크상의 다른 호스트나 장치들에 널리 전달된다. 그리고 네트워크 관리자의 워크스테이션에서 웹 브라우저 내부나 독립 애플리케이션으로 실행되는 애플릿은 장치의 벤더가 가장 적당하다고 생각하는 방식으로 자료 제시 애플릿과 상호 작용하여 각종 자료를 전달하고 표현해준다(그림 8). 예로, 그것은 간단한 문자 목록이 될 수도 있고 하드웨어 서비스의 복잡한 그래픽 표현이 될 수도 있다.

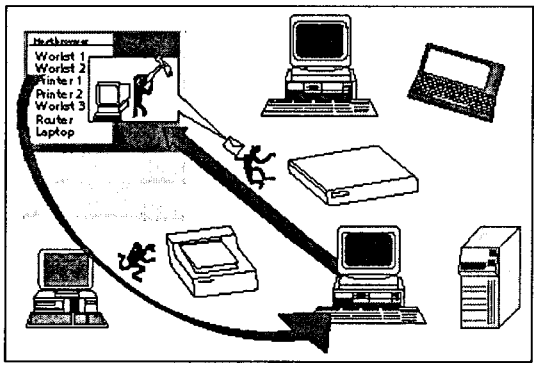


그림 8. 브라우징 에이전트

한편, 원격 통신 네트워크 상에 서비스를 지급하는 것은 복잡한 과정으로, 보통 몇 개의 party를 포함한다. 이동 에이전트는 그 과정을 능률적으로 하도록 도울 수 있다. 이 사실은 서비스 지급 표준, 예를 들어, Telecommunication Information Networking Architecture Consortium인 TINA-C의 활발한 활동을 촉진하고 있다. 또한 active net과 switchlet에 대한 연구는 낮은 단계에 있기는 하지만, 코드 이동성에 기반을 두고, 포괄적이고 융통성 있는 서비스 지급 플랫폼을 제공하려는 면에서 이동 에이전트의 또다른 응용으로 볼 수 있다.

4.2 전자상거래(Electronic Commerce)

MAgNET(Mobile Agents for Networked Electronic Trading) 시스템[6]은 이동 에이전트를 이용한 대표적인 전자상거래 시스템이다. 이 시스템은 생산자와 공급자로 구성되는 하나의 시장 모델을 구현한 것이다. 상품들은 서로 다른 공급자 싸이트들에서 판매되고 구매자는 이동 쇼핑 에이전트들을 보내 공급자들의 싸이트를 방문한 뒤 최적의 협상 결과(best deal)를 가지고 구매자에게로 되돌아 오도록 한다. 뿐만 아니라 MAgNET 시스템에서는 공급자들도 세일즈 에이전트를 구매자들의 사이트로 보내 가격 협상을 벌인 뒤 공급자에게 최대의 이윤이 돌아올 수 있도록 세일즈 활동을 펼치게 한다. 이러한 MAgNET 시스템은 IBM에서 개발된 Java기반의 이동 에이전트 개발 환경인 Aglet Workbench를 이용해 구현되었다.

5.3 가상기업(Virtual Enterprise)

영국의 MSI 연구소에서는 진공 청소기 부품 제조 업체를 대상으로 이동 에이전트를 이용한 가상기업 모델에 관한 연구[8]를 수행하고 있다. 특히 이 연구에서는 가상기업에서 이루어지는 주된 업무 프로세스 중에 그림9와 같은 판매/주문 프로세스(sales/order process)를 자동화하기 위한 이동 에이전트 모델을 제시하고 있다. 판매/주문 프로세스는 일반적으로 네 가지 중요한 구성요소가 있다 : 재고(stock), 판매(sales), 주문(order), BOM(Bill of Material)와 작업순서. 소비자로부터 회사에 주문이 이루어지면 이때 관련 제품 ID를 가지는 하나의 판매 주문이 생성된다. 그러면 완제품이나 혹은 부품의 할당을 요청하는 질의(query)가 재고 데이터베이스(Stock Database)에 주어진다. 재고가 있는 경우, 배달 일시와 함께

판매주문에 대한 할당이 성공적으로 이루어진다. 하지만 주문에 대해 즉시 공급이 이루어질 수 없으면 제품 ID로부터 BOM을 생성한다. 그리고 이 BOM내 해당 제품의 모든 부품에 대한 세부사항도 기록된다. 그리고 이번에는 해당 제품의 부품을 제조하기 위한 원자재의 할당을 요청하는 후속 질의가 만들어져 재고 데이터베이스에 전달이 된다. 원자재 할당이 성공적으로 이루어지면 새로이 작업주문(Works Order)이 만들어져 생산 스케쥴러(production scheduler)에게 제시된다. 이와 같은 프로세스를 위해 이 시스템은 그림 9와 같이 Sales 에이전트, Order 에이전트, 그리고 Stock-Control 에이전트 등 서로 다른 3가지 형태의 에이전트들로 구성된다.

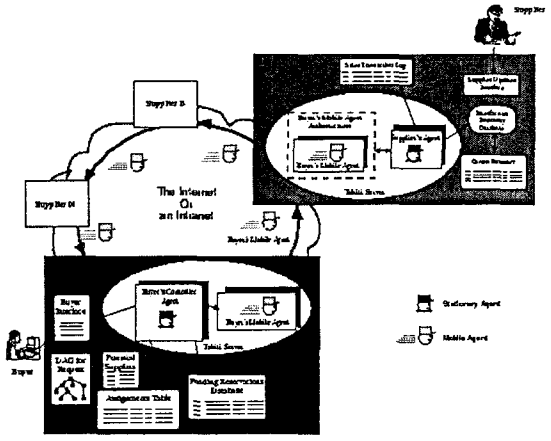


그림 9. 판매/주문 처리를 위한 에이전트 모델

5.4 분산 시뮬레이션(Distributed Simulation)

개방된 컴퓨터 네트워크 상의 대규모 분산 시뮬레이션 분야에도 이동 에이전트를 효과적으로 적용할 수 있다. 이러한 기술을 일반적으로 Mobile Cooperative Technology(MCT)라 부르는데, 그 대표적인 시스템이 WAVE[19] 이다. WAVE는 특수한 재귀 호출 코드에 대한 이동성을 제공할 뿐만 아니라 임의의 지식망(knowledge network)을 동적으로 생성하고 병렬적으로 분산 처리하여 가시화(visualization) 할 수 있는 기능을 제공한다. WAVE에서 이동 에이전트는 완전한 프로그램들에서 기본적인 단위 동작들에 이르기까지 다양하다. 한 이동 에이전트가 모델링하는 탱크나 비행기와 같은 개체들은 자유롭게 컴퓨터 사이를 넘나들 수 있다. 따라서 WAVE는 전쟁과 같은 동적인 상황을 네트워크 상에 분산된 여러 컴퓨터들에 걸쳐 이동 에이전트와 가상현실, 그리고 3D 가시화 기법을 써서 동적으로 시뮬레이션해 볼 수 있게 한다.

4. 이동 에이전트의 표준화

이동 에이전트에 관한 연구는 90년대 초에 Telescript와 이어 Odyssey를 발표한 General Magic사와 Aglet Workbench를 발표한 IBM, Agent Tcl를 연구하는 Dartmouth 대학 등 몇몇 연구기관에 의해 시작되었으나 현재는 이 분야에 대해 연구를 진행하고 있는 기관의 수가 급속하게 늘어나고 있는 실정이다. 이에 따라 서로 다른 개념과 구조에 입각한 상용 시제품들이 서둘러 출시가 되고 있는 상황에서 에이전트 기술에 관한 표준화 문제가 제기되고 표준화 기구들이 등장하여 여러 가지 표준안들을 발표하고 있다. 그러나 대부분이 아직은 초기 시안에 불과하여 다루고 있는 범위가 제한적이거나 구체성이 결여된 면이 많다.

대표적인 에이전트 표준화 기구들로는 FIPA (Foundation for Intelligent Physical Agents)와 Agent Society, 그리고 OMG 등이 있다. 특히 CORBA 등 객체지향 기술의 표준화에 기여해온

OMG에서 처음으로 MAF(Mobile Agent Facilities)라 불리는 이동 에이전트 표준안을 발표하였다. 이 표준안은 플랫폼 독립적인 시안을 담고 있다. MAF에서는 이동 에이전트의 영역(region)들과 이들간의 게이트웨이(gateway)의 필요성에 대해 기술하고 있고 이들을 바탕으로 실제 네트워크 위에 구성되는 에이전트 응용이라는 가상의 층(virtual layer)을 제공하도록 권고하고 있다. 그림 10은 MAF의 구조를 나타내고 있다. 그림에서 한 에이전트 지역(region)은 비슷한 권한과 이동 패턴을 가지고, 서로에게 접근(access)할 수 있는 에이전트 시스템들의 집합으로 정의된다. 이동 에이전트를 위한 기본 기능(facility)에는 에이전트의 저장(storage)과 복구(retrieval), 원격 에이전트의 생성(creation)과 전송(transfer) 그리고 에이전트 메쏘드의 호출(invocation) 등이 포함된다. 이 표준안은 CORBA와 함께 IIOP를 전송 프로토콜로 채용하고 있으며, 명명법(naming)과 같이 이미 CORBA에서 정의된 많은 서비스를 이동 에이전트 활동을 지원하는데도 사용할 수 있음을 암시하고 있다.

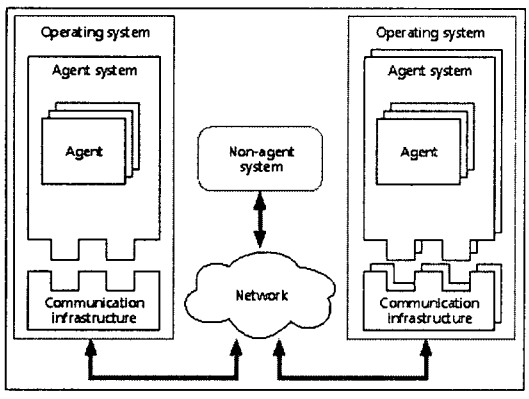


그림 10. MAF 구조

이 표준 명세에서는 두 개의 MAF 객체와 그들의 인터페이스들인 MAFAgentSystemInterface와 MAFFinderInterface를 정의하고 있다. MAFFinder는 에이전트들에 대해 naming 서비스를 제공하는데 지역(region)당 기껏해야 하나의 MAFFinder가 제공된다. MAFFinder는 하나의 CORBA 객체로서 등록되며, 한 에이전트인 MAFClient가 다른 에이전트의 위치를 알아내고 서로 통신할 수 있도록 도와주는 기능을 수행한다. MAFAgentSystemInterface는 수신(receive), 생성(create), 정지(suspend) 그리고 종료(terminate)와 같은 에이전트에 대한 표준 관리 운영 동작(standard management operation)을 제공한다. 이 표준안에서는 에이전트 명명법(agent naming), 권한(authority), 그리고 유형(type)에 대한 세부정의를 IDL(interface definition language) 명세언어로 기술하고 있다. 이와 같은 MAF 명세가 유용한 표준안의 시작지점을 제시했지만, 다루고 있는 범위가 제한적인 문제점을 안고 있다. 즉 통보 서비스(notification service)와 같은 기타 유용한 서비스에 대해서는 정의되지 않았고, 보안(security)에 대해서는 간략하게만 언급되었으며 MAFFinder는 상대적으로 너무 빈약한 인터페이스를 제공하고 있다.

6. 결 론

본 논문에서는 이동 에이전트에 대한 기초 개념과 개발환경, 기술 표준화, 그리고 이동 에이전트를 이용한 응용시스템들에 대해 살펴보았다. 아직 국내외적으로 이동 에이전트 연구의 초기 단계에 머물러 있지만 네트워크에 기초한 다양한 응용분야와 미래의 새로운 계산 모델을 제시한다는 면에서 매우 흥미로운 분야이다. 그 뿐 아니라 이동 에이전트 하부 기술은 이동 코드, 분산 객체, 네트워크 서비스 프로토콜, 암호화, 추론 메카니

즘 등 매우 포괄적인 연관분야를 가지고 있어 인접 관련분야의 연구에 대해 기여할 수 있으리라 본다.

참 고 문 헌

[1] http://agent.cs.dartmouth.edu/general/agenttcl.html

[2] http://www.trl.ibm.co.jp/aglets/JAAPI-whitepaper.html

[3] http://www.cis.ohio-state.edu/~dliang/CIS788.U11

[4] http://www.comsoc.org/pubs/surveys/4q98issue/bies.htm

[5] http://www.omg.org/

[6] http://alpha.ece.ucsb.edu/~pdg/research/

[7] http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole/projects.html#Overview

[8] http://luckyspc.lboro.ac.uk/Docs/Papers/Basys98.pdf

[9] O'Hare, G. and ennings, N., Eds., *Foundations of Distributed Atificial Intelligence*, John Wiley and Sons,1996.

[10] Lesser, V. R., Multiagent Systems: An Emerging Subdiscipline of AI, ACM Computing Surveys, vol. 27, no. 3, ACM Press, Sept. 1995, pp.340-342.

[11] Mobile Agent System Interoperability Facilities, OMG TC Document cf/xx-x-xx, 1998.

[12] White, J. E., Telescript Technology, The Foundation of the Electronic Marketplace, General Magic White Paper, 1994.

[13] S. Appleby and S. Steward, "Mobile software agents for control in telecommunication networks", BT Technol. J., Vol 12, No 2, April 1994.

[14] R. S. Gray, "Agent Tcl: a transportable agent system", In T. Finn and J. Mayfield, Eds, Proc. CIKM'95 Workshop on Intelligent Information agents, Baltimore, Maryland, Dec. 1995.

[15] C. J. Harrison, D. M. Chess A. Kershenbaum, "Mobile agents: are they a good idea?", IBM Research Report, T.J.Watson Research Center, Yorktown Heights, NY 10598, 1995.

[16] D. Johansen, R. van Renesse, F. B. Scheidner, "Operating system support for mobile agents", In Y. Labrou, T. Finin, Eds, Proc. CIKM'94 Workshop on Intelligent Information Agents, Gaithersburg, Maryland, Dec. 1994.

[17] G. Di Marzo, M. Muhugusa, C. Tschudin, J. Harms, "The messenger paradigm and its implications on distributed systems", in Proc. ICC'95 Workshop on Intelligent Computer Communication, 1995.

[18] M. Samela, R. Savola, P. Pulli, "Mobile synthetic environments based on high-speed digital cellular networks", Proc. 12th Workshop on Standards for the Interoperability of Distributed Simulations, IST, Orlando, FL, March 13-17, 1995.

[19] P. S. Sapaty, "WAVE-1: A new ideology of parallel processing on graphs and networks", Future Generations Computer Systems, vol. 4, North-Holland, 1988.

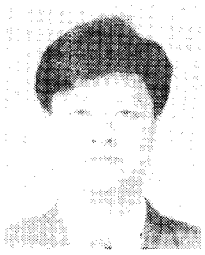
[20] J. E. White, "Telescript technology. The foundation for the electronic marketplace", White paper. General Magic, Inc., 1994.

[21] P. S. Sapaty, M. Corbin, P. M. Borst, and A. Went, "WAVE: a new technology for intelligent control in communication networks", in Proc. Intl. Conf. "The Application of RF, Microwave and Millimetre Wave Technologies" (M'94), Wembley, Oct. 25-27, Nexus, 1994.

[22] S. Seidensticker (Ed.), "The DIS vision. A map to the future of distributed simulation", DIS Steering Committee, Inst. Simulation and Training, UCF, Orlando, 1994.

석 황 희

• 1998년 경기대학교 이과대학 전자계산학과 학사
• 1998년~현재 경기대학교 대학원 전자계산학과 석사과정
• 관심분야 : 지능형 에이전트, 분산인공지능, 계획시스템, 멀티미디어



김 인 철

• 1985년 서울대학교 자연과학대학 수학과 학사
• 1987년 서울대학교 대학원 전산과학과 석사
• 1995년 서울대학교 대학원 전산과학과 박사
• 1989년~1995년 경남대학교 이과대학 전산통계학과 조교수
• 1996년~현재 경기대학교 이과대학 전자계산학과 조교수
• 1999년~현재 (주)프라임테크 이사
• 관심분야 : 지능형 에이전트, 분산인공지능, 계획시스템, 멀티미디어