□ 기술해설 □

분산 환경 기반 개방형 에이전트 구조

한국전자통신연구원 민병의*·박상규*·장명욱**
이광로*·황승구***

● 목 차 ●

1. 서 론

분산 환경하에서 수행되는 에이전트는 초고속 통신망과 일반 유무선 망을 기반으로 하는 분산 컴퓨팅 환경에서 사용자의 요구 사항을 능동적으로 처리할 수 있는 지능형 소프트웨어를 지칭한다[1]. 기본적으로 네트워크 환경에서 에이전트는 다른 시스템에서 대리자간의 정보교환 및 협동을 통하여 정보를 처리하는 기능을 갖추고 있어야 한다.

이러한 분산 환경을 기반으로 하는 에이전트 응용의 중요성은 그림 1에 정리되어 있는 바와 같이 최근의 컴퓨팅 환경의 변화와 함께 그 중요성이 급속도로 확산되고 있다.

첫째 '90년대 중반을 중심으로 인터넷의 폭발적 확산에 따라 대부분의 컴퓨터는 네트워크로 연결되어 서로 정보를 주고 받는 환경으로 변화하였으며, 이에 따라 효율적인 컴퓨팅 자원의 활용과 관리가 최우선의 해결 과제로 등장하였고 이의 해결책으로 지능형 에이전트가 대두되고 있다.

둘째로 인터넷의 확산에 따라 각종 정보 통신 서비스 수용자는 많은 정보의 홍수 속에서 보다 편리하고 쉽게 원하는 정보를 검색할 수 있는 환경에 대한 요구 사항이 증가하고 있다.

셋째, 정보 통신 서비스가 다양화됨에 따라 서비스의 상호 연계 및 보다 쉽게 서비스를 제작할 수 있는 기술이 요구되고 있다.

마지막으로 컴퓨터의 소형화, 휴대 및 이동 단말이 출현하고 있으며 2000년대에는 소형 및 이동 가능한 단말이 보편화 될 것으로 예상된다. 이에 따라 다양하고 편리한 서비스를 제공받기 위해 소형 단말에 간단한 사용자 인터페이스만 장착하고 대부분의 컴퓨팅 파워는 유무선 네트워크에 연결된 서버에 의존하는 경향이 확산될 것이다.

이상과 같은 Network Computing, Mobile Computing, Computing On Demand 등의 컴퓨터 분야의 추세에 따르는 새로운 소프트웨어의 파라다임이 요구되고 있으며 이를 해결해 줄 방안의 하나로 한국전자통신연구원에서 개발된 분산 환경 기반 개방형 에이전트 구조를 소개하고자 한다.

2. 개방형 에이전트 구조의 개요 및 특성

개방형 에이전트 구조는 하나의 응용 에이전트의 간단한 접속 및 수행(Plug-and Play : PnP)을 통하여 네트워크 상에 존재하는 다른 응용 에이전트들과 서비스와 정보를 공유하며 상호 협동 작업을 할 수 있도록 제안된 컴퓨팅

*비 회 원
**정 회 원
***종신회원

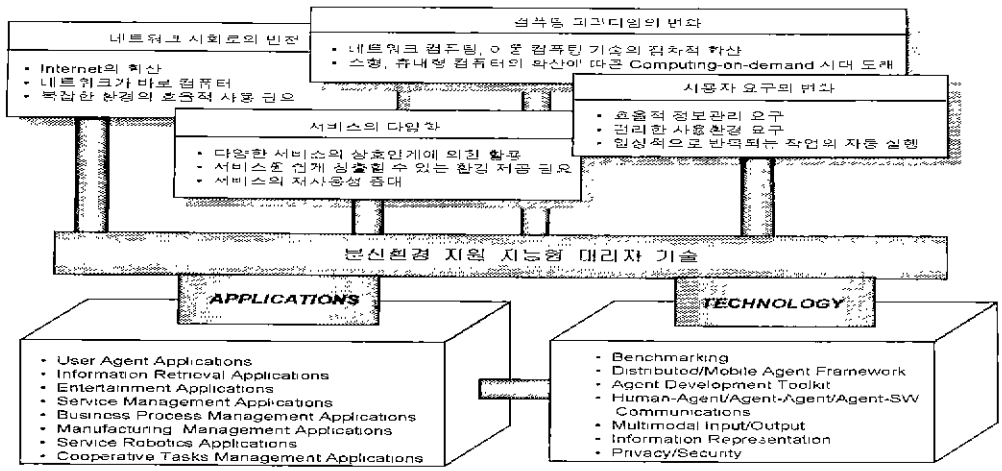


그림 1 컴퓨팅 환경의 변화

아키텍쳐이다.

멀티 에이전트의 개발이 어려운 이유는 개개의 에이전트가 갖는 이형질성(Heterogeneity)과 분산 개념이다. 일반적으로 소프트웨어의 이형질성은 하드웨어, 운영체제, 프로그래밍 언어, 데이타 구조, 시스템 구조 등 매우 다양한 원인에 기인한다. 여러 에이전트들이 이형질성을 가지게 되는 것도 같은 이유에 기인한다. 이러한 이형질성을 극복하는 가장 좋은 방법은 하나의 표준화된 기법이나 언어를 정의하고 동일한 환경하에서 모든 에이전트를 새로 개발하는 것이다. 그러나, 지금까지 개발된 유용한 소프트웨어를 재사용하면서 이형질성을 극복해야 하는 것이 주어진 문제이다.

기존의 프로그램을 에이전트로 재사용하여 멀티 에이전트 기반 응용을 구축하는데 필요한 것은 각자의 작업 수행만을 고려하여 개발된 기존의 프로그램을 하나의 에이전트 그룹의 구성원으로 통합시킬 때 다른 구성원과의 상호작용이 가능하도록 해 주는 것이다. 그런데 이러한 상호작용의 정확성 및 일치성을 보장하기 위해서는 정보공유(Information Sharing), 조정(Coordination), 모순해결(Conflict Control) 등의 기능이 요구된다. 이러한 기능에 의해 이형의 에이전트가 전체적으로 멀티에이전트 시스템 내에서는 동일한 성질의 에이전트처럼 보이게 된다.

이형질성을 극복하기 위해 ARCHON 시스템[2]에서는 도메인이 한정된 기존의 시스템 위에 에이전트 계층을 두어 메타지식을 표현하고, 다른 에이전트와의 상호작용을 전적으로 관리하게 하였다. 메타지식은 에이전트의 성질이나 여러 에이전트간의 상호작용을 위한 제어에 필요에 지식으로서, 에이전트가 수행할 수 있는 기능, 에이전트의 작업 수행 상태, 문제해결에 필요한 지식 등을 포함한다. 그러나 ARCHON 시스템의 단점은 각 에이전트 계층에 자신의 기능 이외에 다른 에이전트의 기능에 관한 정보를 기록하도록 되어 있고 ICL에 대한 기능이 결여되어 있어 진정한 PnP의 기능을 구현하기가 어렵다는 단점이 있다.

분산환경에서 동작하는 지식 기반 시스템이나 에이전트 시스템 간의 지식 공유를 위한 도구로써 ICL을 이용한 대표적인 연구로 KQML[3]이 있다. KQML은 기본적으로 서로 다른 지식 표현 기법이나 추론 기법을 가진 시스템 간의 정보나 능력 공유를 위해 표준화된 네트워크 프로그래밍 언어와 메세지 프로토콜을 제시한다.

또한 고수준 통신을 위한 세가지 계층의 언어를 정의하고, 통신 조정자라는 특수 에이전트를 두고 에이전트간의 상호 작용을 관리한다. 그러나 KQML의 구문(Syntax)과 어의(Semantis)에 대해서는 아직 많은 논란이 있다[4]. 위의 두 가지의 접근 방법은 에이전트의 이형질성을 극복하기 위해서는 매우 중요한

기술이지만 서로의 보완이 없는 어느 하나만으로는 부족하다.

즉, 에이전트 계층을 이용하는 방법은 메타지식을 공통적으로 가짐으로써 작업의 분산적 문제 해결에 장점이 있지만 에이전트의 지식 공유 측면에서는 한계가 있다. 한편 ICL을 이용하는 방법은 여러 에이전트의 지식 공유를 가능하게 하는 방법론에 촛점이 맞추어져 있고 메타지식의 표현 및 공유가 문제해결에 미치는 영향을 간과한 단점이 있다.

이러한 측면에서 한국전자통신연구원에서는 위의 두가지 접근방법을 통합 보완한 분산환경 기반 멀티 에이전트 기반구조인 EMAF(Extensible Multi-Agent Framework)을 제안하였다[5].

3. EMAF

제안된 EMAF의 특성은 이미 기술된 메타지식 및 ICL이외에 서로 다른 응용 에이전트가 동일한 행동 양식으로 교류하기 위한 제어기를 제공한다. 제어기는 에이전트가 수행되는데 요구되는 최소의 기능, 즉 ICL로 표현된 입력 메시지를 읽고, 요구되는 작업을 수행하며, 출력될 메시지를 작성하는 작업을 수행한다.

EMAF가 다양한 분야의 멀티 에이전트 시스템의 개발에 활용될 수 있도록 하기 위해서 특정 도메인에 국한된 부분은 응용 에이전트가 개별적으로 담당하도록 하였다. EMAF는 다만 에이전트 공통의 기능을 추출하여 구성된다. EMAF는 메타지식의 표현, ICL 및 통신 기법, 그리고 에이전트의 제어 방식을 공개함으로써 이방식에 준하는 어떤 에이전트의 접속 및 수행을 가능케 해준다. EMAF의 또 다른 특성은 하나의 멀티 에이전트 시스템의 지원뿐만 아니라 다수의 멀티 에이전트 시스템을 연결하여 하나의 새로운 응용 에이전트 시스템을 구축을 지원 함으로서, 기존 자원의 재활용성 및 시스템의 확장성을 부여하고 있다.

복잡한 문제의 해결을 위하여 EMAF는 블랙보드 기반 모델[6]을 이용한다. 조정자에 해당하는 블랙보드는 전문가에 해당하는 각 에이전트에게 문제를 분산하여 분배하고, 각 에이전트는 자신에게 부여된 문제의 답을 블랙보드에 제시 함으로서 다른 에이전트와 상호 협력하게 되어 있다. 이러한 블랙보드 모델은 EMAF가 제시하는 ICL과 함께 사용되어 이형의 에이전트가 교류하는 메카니즘을 제공한다.

최소 단위의 멀티 에이전트 시스템은 하나의 조정 에이전트와 둘 이상의 응용 에이전트로 구성된다. 조정 에이전트는 응용 에이전트 간의 교류를 위한 장소 및 조정 기능을 제공한다. 반면에 응용 에이전트는 특정 도메인의 작업을 전문적으로 처리하는 역할을 수행한다. 조정 에이전트를 통해서만 응용 에이전트는 다른 에이전트와 정보를 교환할 수 있다. 조정 에이전트는 문제를 해결하는 동안 어느 에이전트에게 어떤 작업을 수행시킬 것인가를 판단하고 해당 에이전트를 선택하여 작업을 넘겨주는 조정 기능을 수행한다. 한편 응용 에이전트는 자신의 기능을 조정 에이전트에게 미리 등록함으로써, 추후의 작업 분배를 가능하게 한다. 조정 에이전트나 응용 에이전트는 각각 독립된 프로세스로 존재한다. 개개의 프로세스는 통신을 위한 프로토콜에 따라 메시지를 교환한다. 다음은 EMAF의 주요 구성요소인 메타지식, ICL 및 제어기의 내부 구조를 설명한다.

3.1 메타지식

메타지식은 조정 에이전트가 사용하게 되는데, 어떤 에이전트에게 작업을 맡길 것인가를 결정하는데 참조된다. 이러한 메타지식은 각 응용 에이전트가 조정 에이전트에게 연결되거나 등록될 때 공지된다. 기존의 프로그래밍 기법 측면에서 보면 일종의 API와 같다고 볼 수 있는데, 어떤 에이전트가 어떤 기능을 가지고 있는 지를 알 수 있다. 응용 에이전트를 개발해서 제공하는 측은, 자신의 프로그램을 어떠한 플랫폼에서, 어떠한 프로그래밍 언어와 컴파일러로 작성하였던지 관계없이, 그 기능을 메타지식의 표현법에 따른 각 에이전트의 능력으로 대응시키는 작업을 함으로써, 자신의 프로그램을 에이전트화 하는 첫 단계를 수행하게 된다.

프로그램간의 재사용성이나 교류를 중요시하는 객체 지향형 프로그램 작성에 있어서도 프

로그래머가 제일 먼저 만나게 되는 문제는 프로그래밍 언어의 선택이다.

다른 컴파일러로 작성한 객체들은 상호 호환성을 갖기가 어렵다. 그러나 메타지식은 컴파일러와 무관하게 해석되어 질 수 있으므로 이러한 한계점을 극복할 수 있다.

이와 같이 자신의 기능을 메타지식으로 표현하여 조정자에게 알려 놓음으로써, 그 에이전트는 다른 이형의 에이전트와도 정보와 서비스를 교류할 수 있게 된다. 메타지식의 한 예를 보면 다음과 같다.

(hitel, [get-weather(Date, Region, Result), get-movie(City, Movies)])

이것은 HiTEL이라는 이름의 에이전트가 날씨를 알려주는 기능과 상영 중인 영화의 제목들을 알려주는 기능을 제공할 수 있다는 것을 표현한 것이다.

3.2 에이전트간 교류 언어(ICL)

분산된 네트워크 환경 하에서의 응용 에이전트는 다양하게 존재한다. 이 에이전트들은 서로 다른 플랫폼 위에서, 서로 다른 프로그래밍 언어를 이용하여, 서로 다른 사람들에 의해서, 서로 다른 시간대에, 서로 다른 문제 해결 방법을 적용하여 구현된다. 따라서 이들 에이전트들간의 인터페이스는 일정하지 않다. 이를 극복하고 에이전트들간의 협동 작업을 수행하기 위해서는 에이전트간의 교류 언어가 필요하다. 언어를 이용한 에이전트들 간의 교류가 갖는 장점으로는 언어라는 것은 플랫폼이나 컴파일러, 프로그래밍 언어와 무관하게 이해되고 다루어 질 수 있다는 것이다. ICL은 상호 교류를 위해 약속된 프로토콜과 메시지 형태이다. ICL의 프로토콜은 메시지를 어떻게 해석하고 어떠한 일련의 행위를 할 것인가를 암시해준다.

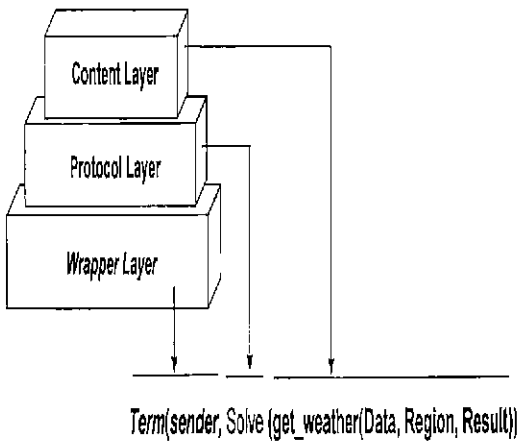


그림 2 ICL의 구조

그림 2에 나타난 바와 같이 ICL은 content 층, protocol 층, wrapper 층으로 구성된 3개의 층으로 이루어진다. Content 층은 에이전트간에 교류 되는 메시지 내용을 나타낸다. Protocol 층은 에이전트들 간의 약속된 규칙을 의미한다. 예를 들면, solve라는 프로토콜이 담긴 메시지를 받은 에이전트는 solved 라는 프로토콜이 담긴 메시지를 되돌려 주어야 한다는 규칙이 내포되어 있다. Wrapper 층은 에이전트가 그 내용을 보고 그 메시지가 ICL로 해석되어 질 수 있는 메시지인지를 판단하며, 어느 에이전트가 보낸 것인가를 알아낼 수 있도록 한다.

term(user-interface, post-query(get-weather('95/5/3', 'Seoul', R)))

이라는 예를 들어 살펴본다. 이는 "user-interface라는 에이전트가 서울의 95년 5월 3일 날씨를 알려달라고 조정 에이전트에게 부탁한다."로 해석된다. 이 메시지는 term( )으로 포장되어 있으므로 ICL 메시지로 인정된다. 이 메시지를 받은 조정 에이전트는 post-query라는 프로토콜에 따라, (1) 작업을 맡길 에이전트를 메타지식을 이용하여 찾아내고, (2) 언제 답이 도착할 지 모르므로 이와 관련된 트리거를 설치한 후, (3) solve라는 프로토콜에 메시지를 실어서 해당 응용 에이전트들에게 보내는 일련의 작업을 수행한다. 이 경우에 조정 에이전트가 응용 에이전트에게 보내는 ICL 메시지는 다음의 형태를 가진다.

term('Broker', solve(2, get-weather('95/5/3', 'Seoul', R)))

여기서, solve의 첫번째 인수인 2는 이 요구

사항의 확인자이다. 이 확인자는 요구된 작업에 대한 고유한 표시로서, 들어오는 메시지의 내용과 트리거를 일치시킬 때 참조된다. 이것이 필요한 이유는 같은 작업 요구가 동시에 여러 에이전트에 의해서 요청될 수 있기 때문이다. 요구된 작업의 수행이 완료되면 HiTEL 에이전트는 다음과 같은 ICL 메시지를 조정 에이전트에게 돌려준다.

term('hitel', solved(2, hitel, get-weather('94/5/3', 'Seoul', R). [get-weather('94/5/3', 'Seoul', fine)])

여기서 solved의 두번째 인수는 이 작업을 수행한 에이전트의 이름을 나타내며, 세번째 인수는 원래의 작업 요청 내용을 나타낸다. 마지막 인수는 수행 결과를 나타내는데, 작업요청에 포함된 변수의 값이 정해지어 나타난다.

Protocol 층은 생략될 수도 있다. 예를 들면 단순한 정보의 교류나 명령어 수행을 요구할 경우, 또는 새로 프로토콜을 정의하여 사용할 필요가 있을 때는 이 층을 생략한다. 만일 읽혀진 메시지의 내용이 어느 프로토콜에도 해당되지 않으면, 도메인 지식 속에서 메시지를 만족시킬 기능을 찾은 후 수행한다. 이것이 다른 에이전트의 기능을 사용할 수 있도록 해준다.

ICL을 위해 post-query, solve, solved로 이루어진 3개의 기본 프로토콜이 제공된다. Post-query는 조정 에이전트에게 문제를 해결해 달라고 요청할 때 사용된다. 이 프로토콜에 따라 조정 에이전트는 문제를 잘 해결할 것 같은 응용 에이전트들을 찾고, 그 에이전트에게 문제를 해결하라는 지시를 solve라는 프로토콜에 실어서 보낸다. 그리고 나서 트리거를 설치하여 추후에 solved라는 프로토콜에 실려온 답을 문제를 제기한 에이전트에게 전달할 수 있도록 하는 작업을 하게 된다.

Solve라는 프로토콜을 받은 에이전트는 그 메시지의 내용을 수행하고 나서 그 결과를 solved라는 프로토콜에 실려 조정 에이전트로 보내준다. 이 메시지는 조정 에이전트의 트리거와 일치되어 다시 문제를 제기한 에이전트로 전달된다. 조정 에이전트의 조정을 위한 지식 속에는 주요한 기능으로서 add-trigger, post-trigger, register-solvable-goals, read-bb, write-bb 등이 포함된다. Add-trigger와 post-trigger는 다른 에이전트에게 트리거를 설치하여 그 에이전트로 하여금 원하는 작업을 조건이 만족될 때마다 수행하도록 할 때 사용된다.

Register-solvable-goals는 응용 에이전트가 자신의 능력을 조정 에이전트에게 공지하는 것을 가능하게 하여 주며, read-bb,와 write-bb는 각각 조정 에이전트가 관리하는 블랙보드에서 자료를 읽거나 블랙보드에 자료를 쓰기 위해 사용되는 것이다.

3.3 제어기

멀티에이전트 시스템에서 다루어지는 대상들은 (1) 각 에이전트의 메타지식, (2) 에이전트들 간의 교류언어, (3) 명령어 또는 작업 수행 요구, (4) 비 동기적인 에이전트의 수행을 위한 트리거로 정리된다. 이러한 대상에 가해지는 기능들은 다음과 같다.

- 에이전트들간의 연결을 성립시키는 기능
- 각 에이전트들의 능력을 선언하는 기능
- 에이전트들간의 ICL 메시지를 생성, 해석하는 기능
- 메시지를 보내고 받는 기능
- 외부에서 온 메시지의 내용에 따라 수행하는 기능

이러한 기능들은 에이전트로서 기본적으로 갖추어야 하는 기능들로서 EMAF의 제어기 안에 구현된다. 제어기는 사용자로부터 에이전트의 고유한 이름 및 자신이 연결하고 싶은 에이전트의 이름을 받은 후 시스템에서 제공하는 통신 메카니즘을 이용하여 연결을 해준다. 조정 에이전트의 입장에서 보면, 연결되는 즉시 도착하는 데이타가 그 에이전트의 이름으로 간주되므로, 제어기는 연결의 성립 직후 에이전트의 이름을 보낸다. 그리고 나서 register-solvable-goal을 이용하여 에이전트의 능력을 보고한다.

제어기는 입력되는 메시지에 대하여 'term('으로 시작되었는가 검사하여 ICL 메시지인가 확인한다. ICL 메시지로 확인이 되면, 그 첫번째 인수(즉 다음 콤마를 만날 때까지의 자료)를 기억하였다가, 나중에 답을 돌려줄 때 사용

한다. 그리고 나서 두번째 인수를 protocol 층 및 content 층으로 분류한다. 프로토콜이 solve 이면 그 내용을 도메인 지식을 이용하여 수행하고, 그 결과를 solved라는 프로토콜을 이용하여 되돌려 준다. 물론 이때 wrapper 층으로 마무리를 하여 내보내 준다.

제어기는 post-query에 대한 처리는 안한다. Post-query 프로토콜은 조정 에이전트의 도메인 지식에 구현되어 있다.

그림 3에 나타난 바와 같이 제어기와 도메인 지식이 합쳐져 하나의 에이전트를 구성한다. 제어기는 에이전트 기반 구조에 의해 제공되는 한편 도메인 지식은 응용 에이전트 개발자가 제공하게 된다. 응용 에이전트 개발자는 제어기를 자신의 프로그램과 연결시킴으로써 에이전트를 만들어 나갈 수 있다.

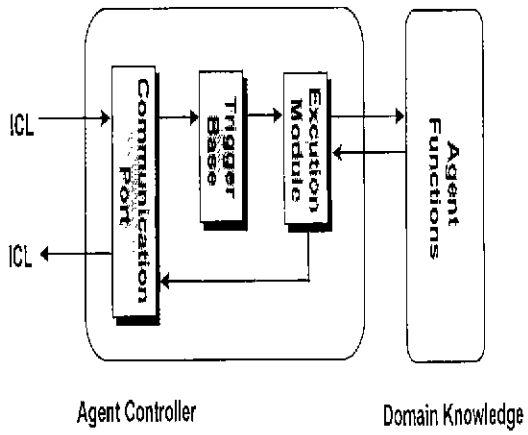


그림 3 한 에이전트의 구조 및 데이타 흐름

제어기가 제일 먼저 하는 일은 조정 에이전트와 연결하고 도메인 지식에 있는 에이전트의 능력을 조정자에게 보내는 것이다. 이것이 완료되면 다음과 같은 과정을 주기적으로 반복한다.

(1) ICL 메시지를 읽고 내용을 추출한다(통신포트).
(2) 메시지를 트리거 베이스의 내용과 비교하여 일치된 것이 있으면, 트리거로부터 처리할 메시지를 추출한다(트리거베이스).
(3) 메시지에 따른 수행을 한다(수행부). 만일 바로 수행할 수 없으면, 도메인 지식을 이용한다(도메인 지식).
(4) 수행된 결과를 포함하여 돌려줄 메시지를 생성(ICL의 생성)하고, 이를 돌려준다(통신포트).

4. EMAF의 확장 : DMAF

인터넷의 확산, 소형/휴대 단말의 확산, 다양한 서비스의 상호연계 등 컴퓨팅 파라다임의 변화에 대응하기 위하여 현재 한국전자통신연구원에서는 Networked Virtual Computing의 개념을 도입한 새로운 에이전트 프레임워크를 개발중에 있다.

즉, 기존의 EMAF를 기반으로 인터넷 및 이동 에이전트[7] 개념을 통합한 새로운 구조인 DMAF(Distributed Mobile Agent Framework)를 개발 중에 있다. 이동 에이전트는 개념상으로는 현재의 JAVA와 유사하다. 즉, 자바 응용에 자율적인 이동성을 부여함으로써 자바가 지니고 있는 시스템에 무관한 호환성 및 자율적 이동에 의한 Computing On Demand를 실현할 수 있다는 특성을 갖고 있다. 이동 에이전트는 자신이 원하는 기능을 수행시킬 수 있는 네트워크 상에 존재하는 플랫폼으로 능동적으로 이동하여 작업을 수행 후 결과를 회신

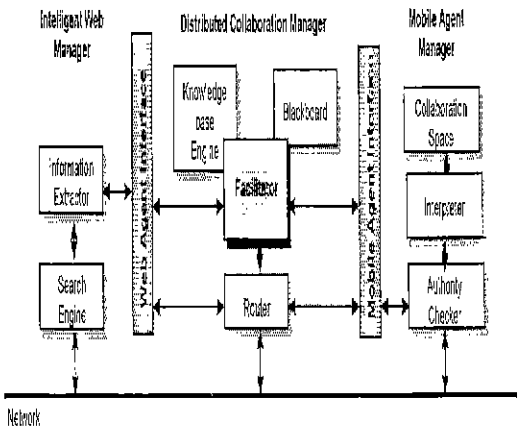


그림 4 DMAF(Distrbuted Mobile Agen Framework)

하여 주는 새로운 개념의 컴퓨팅 파라다임이다. 그러나 가장 문제가 되는 것은 다른 시스템으로 이동 및 수행 시에 발생되는 보안(Security)이 가장 큰 문제로 대두된다. 이러한 문제 때문에 현재 설계 중인 시스템은 그림4에 나타나 있는 것과 같이 중앙의 기존의 EMAF를 확장시킨 DCM(Distributed Collaboration Manager)을 중심으로 Intelligent Web Manager는 주로 인터넷을 기반으로, Mobile Agent Manager는 주로 자체적으로 보안 문제를 해결할 수 있는 Intranet을 기반으로 하고 있다.

DCM은 블랙보드를 이용하여 구현된 EMAF에 이동 에이전트의 능동적 이동성 및 웹 에이전트의 지능적 정보검색(Intelligent Information Navigation)을 위해 통신 및 지식 처리 기능을 강화하여 설계를 진행 중이다.

참고문헌

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

[2] T. Wittig, N.R. Jennings and E.H. Mamdani, "ARCHON-A Framework for Intelligent Cooperation," IEE-BCS Journal of Intelligent System Engineering-Special Issue on Real-time Intelligent Systems in ESPRIT, Vol.3. No.3. pp. 168-179, 1994.

[3] Hans Chalupsky, Timm Finin, Rich Fritzson, Don Mckay, Stu Shapiro and Gio Wiederhold, "An Overview of KQML : A Knowledge Query and Manipulation Language," P. 36, 1992.

[4] Timm Finin, Richard Fritzson, Don Mckay and Robin McEntire, "KQML as an Agent Communication Language," CIKM '94, November 1994.

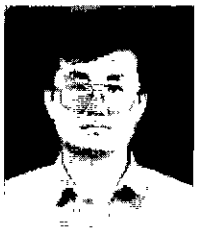
[5] 백순철 외, "이형 분산 환경에서 에이전트들간의 이형성을 극복하기 위한 멀티에이전트 기반구조," 정보과학회 논문지, 제2권, 제1호, pp. 24-37, 1996. 3.

[6] S.C. Baeg, et. al, "Cooperation in Multi-agent Systems," International Workshop on Intelligent Computer Communication, pp. 1-12, Romania, 1995.6.

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

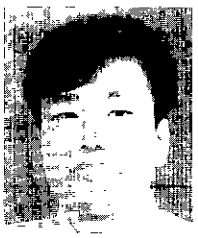
민 병 의

1982 한양대학교 졸업(학사)
1984 한국과학기술원 전기 및 전자공학과 졸업(석사)
1984~1987 대림산업기술연구소
1987~현재 한국전자통신연구원 인공지능연구실 실장
1992 한국과학기술원 전기 및 전자공학과 졸업(박사)
관심분야 : 멀티미디어시스템, 에이전트



박 상 규

1982 서울대학교 컴퓨터공학과 졸업(학사)
1984 한국과학기술원 전산학과 졸업(석사)
1984~87 대림산업 전산실 근무
1987~현재 한국전자통신연구원 인공지능연구실 선임연구원
1989~현재 한국과학기술원 전산학과 박사과정 수료
관심분야 : 에이전트 시스템, 정보검색, HCI



장 명 욱

1990 고려대학교 전산과학과 졸업(학사)
1992 한국과학기술원 전산학과 졸업(석사)
1992~현재 한국전자통신연구원 연구원
관심분야 : 에이전트시스템, 패턴인식, HCI

이 광 로

1986 일본 Fukuoka 공업대학교 전자기계과 졸업(학사)
1988 일본 Ritsumeikan대학교 전기공학과 졸업(석사)
1988～현재 한국전자통신연구원 인공지능 연구실 선임연구원
1994～95 미국 SRI International(International Fellow)
관심분야 : 에이전트 시스템, HCI, 패턴인식

황 승 구

1979 서울대학교 전기공학과 졸업
1982～현재 한국전자통신연구원 책임연구원, 멀티미디어연구부장
1986 플로리다대 전기공학과 박사
관심분야 : Multimedia Computing, Intelligent Computing, Visual Computing

●'97 총회 및 춘계학술발표회●

- 일 자 : 1997년 3월 29일
- 장 소 : 이화여자대학교
- 주 최 : 인공지능연구회
- 문 의 처 : 이화여자대학교 전자계산학과 박승수 교수
  T. 02-360-2316

●HPC ASIA '97 학술대회●

- 일 자 : 1997년 4월 28일～5월 2일
- 장 소 : 서울 힐튼호텔
- 주 최 : 병렬처리시스템연구회
- 문 의 처 : 대회사무국
  T. 02-501-7065