이동 에이전트를 이용한 가상 상점 환경의 설계와 구현†

이명섭, 김칠수, 박창현

Design and Implementation of Virtual Market System Using Mobile Agent

Myung-Sub Lee, Chil-Su Kim, Chang-Hyeon Park

요약

Recently, according to the rapid development of Internet and IT, many applications and technologies related to Internet and IT have been studied and developed. The agent technology, which was an essential topic in early AI, is being researched with the combination of the recent network technologies. Mobile agent especially takes high interest of many researchers because of its mobility with which it can move around the internet and complete its goal. This paper presents a virtual market system using mobile agent, in which the transaction agents can do their transaction about some product with each other instead of real buyer and seller. The presented virtual market system consists of market agent, customer agent and database agent to provide the virtual market places for the effective completion of transactions.

1. 서 론

최근의 정보통신 기술의 발달은 인터넷 및 WWW(World-Wide Web)의 사용자의 수를 급속도로 증가시키고, 또한 이에 따라서 인터넷 및 WWW 에 관련된 다양한 기술 및 소프트웨어들이 개발되고 있다. 이러한 추세에 부응하여 과거 인공지능의 한 분야로 연구되었던 에이전트 기술이 네트워크 기술과 결합하여 다시 그 연구가 활발해지고 있다.

에이전트에 대한 정의는 여러 분야에서 다양하게 언급되고 있으나, 대체적으로 다른 대상을 위해 행동하는 객체로 정의하고 있다[15]. 컴퓨터 분야에서는 계산적인 객체(Computational entity)로서의 에이전트를 언급하면서, 다른 객체를 위해서 행동하는 객체,

* 영남대학교 컴퓨터공학과

적절한 반응적(reactive) 행동과 순응적(proactive) 행동을 나타내며, 이동성(mobility) 및 협력성(cooperation)을 보이면서 행동하는 소프트웨어 시스템을 에이전트 시스템으로 언급하고 있다[15]. 에이전트 시스템의 종류는 대개 지능형 에이전트와 이동형 에이전트로 대별되는데, 지능형 에이전트는 특정의 작업을 위해서 지식을 이용하는 정적인(static) 대상을 의미한다[1]. 이는 과거의 인공지능 연구에서 주류를 이루던 것으로, 그 연구의 대부분이 인간의 능력을 모방할 수 있는 지적인 객체를 생성하려는 노력으로서, 이는 에이전트 개념의 한 부분인 특정 객체의 할 일을 대행하는 객체를 형성하려는 노력과 상통한다. 반면에 이동형 에이전트는 지적인 작업 능력을 최소화하면서, 네트워크를 통해 여러 시스템을 방문하면서 다양한 작업을 수행할 수 있는 동적인 시스템을 지칭하고 있다. 최근의 대부분의 에이전트 시스템에 대한 연구는 이와 같은 이동형 에이전트에 집중되고 있다[2,3,4,25].

본 논문에서는 이동 에이전트를 이용하여 가상 시장을 설계하고 구현하였다. 본 논문에서 기술하는 이동 에이전트를 이용한 가상 시장 환경은 시장 에이전트, 고객 에이전트, 데이터베이스 에이전트로 크게 세 부분으로 나눈다. 시장 에이전트는 팔고자 하는 상품의 정보를 데이터베이스 에이전트에게 알려주고, 데이터베이스 에이전트는 이 정보를 저장하고 기존의 시장 정보 중 가장 적당한 시장 정보를 시장 에이전트에 보낸다. 시장 에이전트는 이 정보를 가지고 원하는 시장에 가서 고객과 만나 협상을 통해 상품을 판매하고 처음 파견한 곳에 정보를 보고 후 소멸한다. 고객 에이전트는 자신이 사고자 하는 상품의 정보를 가지고 데이터베이스 에이전트에 가서 시장 정보를 얻은 다음 원하는 상품을 구매하고 처음 파견된 곳에 보고 후 소멸한다

2. 관련연구

본 장에서는 시스템의 구조 설계를 위해, 우선 에이전트에 대한 연구를 기술하고, 에이전트 중심의 전자상거래 시스템에 대한 소개 및 그 사례를 연구 한다.

2.1 에이전트

제안 하고자 하는 시스템은 에이전트 기반의 시스템이므로 에이전트에 대한 기반 연구가 필요하다.

2.1.1 에이전트 시스템의 구조 모델

에이전트 시스템들의 분류 기준은 여러 가지 이다. 예를 들어, 에이전트 시스템은 네트워크를 통한 이동성 유무에 따라 이동형 에이전트와 정적 에이전트로 분류된다. 또한 에이전트 시스템은 다른 에이전트들과 협력능력, 자동성, 학습능력의 유무에 따라 협력 에이전트, 인터페이스 에이전트, 협력 학습형 에이전트 등으로 분류 될 수 있다. 또 에이전트 시스템을 구조적 관점에서 숙고형 에이전트(Deliberative Agent), 반응형 에이전트(Reactive Agent), 복합형 에이전트(Hybrid Agent)로 분류 하기도 한다. 이러한 분류는 에이전트 시스템에 지적 능력을 부여하는 방법과 관련된다. 숙고형 에이전트는 전통적인 심볼형 인공지능 패러다임에 근거하여 구조를 사용하는 에이전트를 의미한다. 숙고형 에이전트 구조에서는 심볼 기반의 지식 표현 언어 및 패턴 매칭과 같은 심볼 처리를 통해 지적 능력을 부여하는 방법이 추구된다. 일반적으로 숙고형 에이전트 시스템은 크게 두 가지

해결해야 할 문제를 가지게 되는데, 첫 번째가 실세계를 어떻게 정확하고 적절하게 심볼로 표현할 것이냐는 문제이고, 두 번째는 이러한 심볼 모델로부터 어떻게 복잡한 실세계의 문제에 대한 답을 추론해 낼 것이냐는 문제이다. 숙고형 에이전트 시스템들의 원조격인 시스템들로는 STRIPS[5], NOAH[7], NONLINE[10]등과 같은 전통적인 계획 수립 시스템들과 MYCIN[6]과 같은 전문가 시스템들이 있다. 에이전트 시스템이란 용어를 얼마나 넓은 의미의 용어로 사용하느냐에 따라 위와 같은 전통적인 계획 수립 시스템들이나 전문가 시스템들은 에이전트 시스템이라고 부를 수도 있고 아닐 수도 있다. 숙고형 에이전트에 관련된 최근의 연구로는 믿음(Belief), 소망(Desire), 의도(Intension)등과 같은 속성에 기초를 둔 에이전트 이론을 실제로 에이전트 시스템으로 구조화한 IRIMA[11]와 GRATE*[8]와 간단한 자연어 능력과 계획 수립 능력 그리고 계획 실행 능력을 가진 시뮬레이션 된 로봇 잠수함인 HOMER[13]등이 있다.

반응형 에이전트란 어떤 유형의 심볼화된 실세계 모델과 복잡한 심볼 추론(Symbolic Reasoning)도 포함하지 않는 반응형 구조(Reactive Architecture)를 사용하는 에이전트를 의미한다. 반응형 에이전트 개념은 숙고형 에이전트에 대한 반성에서 출발되었다. Brook 은 포함 구조에 대한 연구[9]를 통해 숙고형 구조가 아니더라도 지적 능력을 가질 수 있다고 주장하였다. 심볼형 인공 지능에 기초를 둔 숙고형 구조와는 달리, 반응적 에이전트 시스템들은 내부적으로 복잡한 심볼 모델이나 추론 기능이 없이도, 마치 로봇과 같이 직접 실세계에 놓여서 실세계 환경으로부터 발생하는 응급 상황에 대한 반응에 대한 결과로서 지능적인 행위를 할 수 있도록 시도하는 시스템들이다. 반응형 에이전트에 대한 대표적인 연구들에는 문헌[8,9,12] 등이 있다.

혼합형 에이전트란 숙고형 구조와 반응형 구조를 결합한 구조인 혼합형 구조(Hybrid Architecture)를 갖는 에이전트를 의미한다. 혼합형 구조는 심볼 모델을 기반으로 계획 생성을 수행하는 숙고모듈(Deliberation Module)과 복잡한 추론 없이 발생하는 사건들에 빠르게 반응할 수 있는 반응 모듈(Reactive Module)을 계층구조화 하여 두 구조의 장점들을 함께 살리려고 시도한다. 혼합형 에이전트 시스템에 대한 대표적인 연구로는 문헌[21,22,16,19]등이 있다.

반응형 에이전트 구조 모델이나 혼합형 에이전트 구조 모델은 숙고형 에이전트 모델은 인공지능에서 매우 오랜 역사를 가지고 있음에도 불구하고 실용적인 문제들의 해결보다는 이론적이고 실험적인 문제들에 대한 연구를 벗어나지 못했다는 평이 있다. 하지만 반응형 에이전트나 혼합형 에이전트 모델도 실용적인 연구 결과를 충분히 얻지 못했으며 아직 연구가 더 필요한 상황이며, 마찬가지로 숙고형 에이전트 모델에 대한 연구도 계속 필요하다고 할 수 있다. 최근에는 계획수립 에이전트와 같은 숙고형 에이전트를 멀티 에이전트 환경 아래에서 하나의 구성 원소로 사용하기 위한 연구들이 많이 이루어지고 있다[23,24].

2.1.2 에이전트간 통신 언어

에이전트 간의 통신을 위해서는 에이전트가 메시지를 생성하고 접수된 메시지를 해석해야 한다. 이를 목적으로 메시지의 정형적이고 일반적인 표현법인 에이전트 통신언어(Agent Communication Language : ACL)가 필요하다. ACL 은 외부언어와 내부 언어로 구성되어 있다.

ACL 은 기본적으로 응용분야에 독립적으로 만들어진 언어로써 전자상거래 시스템을 구현하기위해서는 내부 언어를 정의해야 한다.

2.2 전자상거래 시스템 사례

2.2.1 Kasbah[17]

Kasbah 는 MIT Media Lab 에서 연구한 에이전트 기반 가상 시장이다. 구매자나 판매자는 사거나 팔 물건에 대한 기술, 희망 가격, 그리고 받아들일 수 있는 최고 또는 최저 가격을 입력하고, 주어진 몇 개의 교섭 전략 중 하나를 선택하고 이를 자신의 에이전트에게 전달한다. 에이전트는 사용자의 입력을 기반으로 교섭할 대상을 찾고, 선택된 전략에 따라 교섭하여 거래를 성사시킨다. Kasbah 는 사용자를 대신해서 교섭 가능하고, 일련의 거래 활동은 자동화가 이루어지는 장점이 있다. 단점으로는 시간에 따른 가격의 변화율을 채택한 세 가지의 단순한 교섭 전략을 제공함으로서 효과적인 거래를 기대하기 어렵고, 모든 에이전트가 가상 거래 공간에서 동작하므로 서버의 부하가 커진다. 또 실제의 거래 적용에 한계가 있고 단지 가격에 의한 교섭만을 제공한다.

2.2.2 MAGNET[24]

MAGNET 은 구매자가 요구하는 상품을 가지고 있는 판매자에게 직접 접근할 수 있는 이점을 지닌 산타바바라 대학에서 제안한 쇼핑 모델이다. 이 시스템은 구매자와 구매자를 위한 이동 에이전트, 판매자들을 포함하고 있다. 구매자는 요구한 상품 목록을 가지고 있는 판매자 목록을 가지고 있는다. 상품을 구매하고자 하는 구매자는 판매자 사이트들이 있는 목적지 정보와 상품을 구매하기 위한 기준을 정의한 이동 에이전트를 생성한다. 그리고 요구한 상품 목록을 가지고 있는 이동 에이전트를 보낸다. 이동 에이전트는 각각의 판매자 사이트들을 방문하게 되고, 구매자의 기준에 따라 상품 카탈로그에서 해당 상품을 찾고, 상품을 찾으면 구매자 사이트로 돌아오게 된다. 구매자는 찾아온 상품에 대해 최상이라고 판단되면 거래를 하고, 그렇지 않으면 동작을 멈추도록 이동 에이전트를 처리한다. 그러나 해당 상품에 대한 대상이 단지 가격에만 의존하는 단점이 있다[4].

3. 가상 시장의 환경 모델

본 논문의 가상 시장 환경은 Aglets 기반의 이동 에이전트로 동작하며 시장 에이전트, 고객 에이전트, 데이터베이스 에이전트 등 크게 세 부분으로 구성된다. [그림 1]에서 이동 에이전트를 이용한 가상 상점의 내부 동작 방식을 보인다.

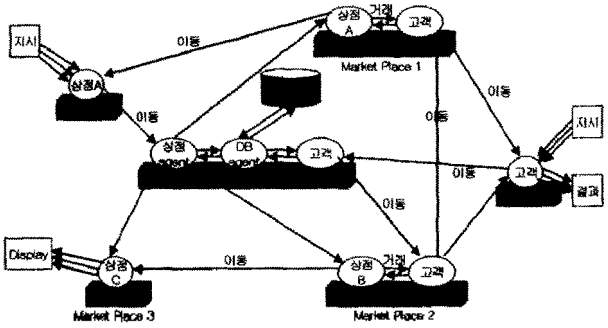

Fig.1. Design architecture of a market place

먼저, 상점(Market) 에이전트는 상품의 가격, 수량, 종류와 자신의 URL 정보를 가지고 데이터베이스 에이전트로 이동한다. 이동한 상점 에이전트는 데이터베이스 에이전트와 대화를 통해 자신의 정보를 데이터베이스 서버에 저장한다. 그런 다음 자신의 상품을 판매할 시장의 정보를 얻어 시장으로 이동하여 고객을 기다리고, 고객 에이전트를 만나 서로 협상을 통해 상품을 매매한다. 상품을 판매하고 나면 자신을 복제하여 처음 파견된 곳과 데이터베이스 에이전트에 그 정보를 보고하고, 물건을 다 판매하고 나면 데이터베이스 에이전트에는 복제된 에이전트를, 자신이 처음 파견된 곳에는 직접 돌아가서 에이전트 프로세스를 소멸한다. 고객 에이전트는 고객이 사고자 하는 상품의 정보 즉, 가격, 종류, 수량 등의 정보를 가지는 슬레이브(Slave) 에이전트를 복제하여 데이터베이스 에이전트에 파견한다. 그러면, 이 데이터베이스 에이전트와 협상을 통해 자신이 원하는 상품을 판매하는 정

보를 얻어 시장으로 이동한다. 고객 에이전트와 상점 에이전트간의 협상을 통하여 조건이 충족되면 구매가 이루어지지만, 만약 조건이 충족되지 않는다면 다른 시장으로 이동한다. 고객 에이전트는 원하는 상품을 구매했으면 처음 파견된 곳의 마스터(Master) 에이전트에 그 결과를 보고하고 소멸한다. 고객 에이전트에는 상품을 비교하여 사고 팔 수 있는 협상 알고리즘을 사용하였으며, 데이터베이스 에이전트에는 상점 에이전트에서 상품의 정보와 시장의 정보를 얻어 이를 관리할 수 있도록 구현하였다.

3.1 상점 에이전트(Market Agent)

상점 에이전트는 판매할 물건을 사용자에게 GUI(Graphic User Interface)를 통해 상품과 시장 URL 을 입력하고 시장으로 이동한다. 시장에 도착하면 자신을 복제(Clone)해서 그 복제된 에이전트를 데이터베이스 에이전트가 있는 곳으로 이동시키고 자신은 고객을 기다린다. 복제된 에이전트는 데이터베이스 에이전트에게 상점 에이전트의 ID, 상품, 가격, 개수, 시장 URL 을 알려주고 처음 파견되었던 HOME 으로 이동하여 데이터베이스 에이전트에 접속한 결과를 알리고 종료한다. 고객 에이전트가 상품을 구입하게 되면 상점 에이전트는 자신을 복제하여 상품 구입 결과를 다시 데이터베이스 에이전트에게 보내고 고객을 기다린다. 복제된 에이전트는 데이터베이스 에이전트에게 판매된 상품의 개수를 알려주고 HOME 으로 이동하여 팔린 내용을 출력하고 종료한다. 상품이 모두 판매되고 나면 자신이 직접 데이터베이스 에이전트로 이동하여 상품이 모두 판매되었다는 것을 알리고 HOME 으로 이동하여 판매된 결과를 출력하고 종료한다. [그림 2]에서 상점 에이전트의 순서도를 보인다.

순서도에서 상점 에이전트의 UI(User Interface)를 제외한 핵심적인 코드를 보면 [그림 3]와 같다. runState 변수는 상점 에이전트의 진행 상태를 저장하고 판단하기 위한 가장 핵심이 되는 변수로 상점 에이전트는 runState 변수의 값에 따라 단계를 진행해 나가게 된다.

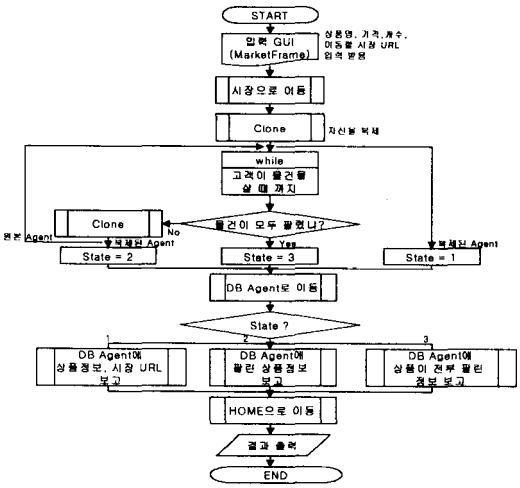

Fig. 2 Flowchart of market agent

```
public void run()
    // agent 원본이 수행되는 부분
    if (runState == 1)              // Created
        setText("Hello Master.");
        waitMessage();

      else if (runState == 2) // 목적지에 도착
        setText("Market Agent");
        sendResult();
      // DB Agent 에 등록, 결과보고
        printList();  // Print List
        waitMessage(); // 대기(고객을 기다림)

      else if (runState == 4)
      // 상품을 다 팔고 Home 에 도착한
          setText(".. Market Master ..");
          printList();     // 결과 출력
          waitMessage(2000);
          dispose();              // 종료

      else if (runState == 11)
      // 상품을 다 팔고 DB Agent 로 이동
          setText(".. Market master ..");
          connectDBAgent3();
          // Market Master Connect DB
          goHome();   // Home 으로 이동

      else if (runState == 12)
          // 상품을 다 팔고 Home 으로 이동
          setText(".. Market master ..");
          printList();     // 결과 출력
          System.out.println(tmpMsg);
          // DB Agent 접속결과 출력
          waitMessage(2000);
          dispose();              // 종료
```

Fig. 3 Execute code of Market Agent

3.2 고객 Agent

고객 에이전트는 사용자에게 GUI(Graphic User Interface)를 통해서 구입할 상품과 개수와 구입 상한가를 입력받고 슬레이브(Slave) 에이전트를 생성 시켜 상품을 구매해 오도록 하고 자신은 결과를 기다린다. 슬레이브 에이전트는 데이터베이스 에이전트에게 이동하여 원하는 상품을 판매하는 상점 에이전트들의 URL 을 얻은 후 최초 URL 이 있는 시장으로 이동한다. 이때, 시장에서 대기하고 있는 상점 에이전트에게 원하는 상품에 대한 정보를 얻어낸 후 다른 시장으로 이동한다. 상품 구입이 끝나면 HOME 에서 기다리고 있는 마스터(Master) 에이전트에게 결과를 바로 원격으로 접속하여 전달하고 종료한다. 대기하고 있던 Master 에이전트는 결과를 받아서 GUI 를 통해서 출력하고 사용자의 다른 상품 구매를 기다린다. [그림 4]에서 고객 에이전트 순서도를 보인다.

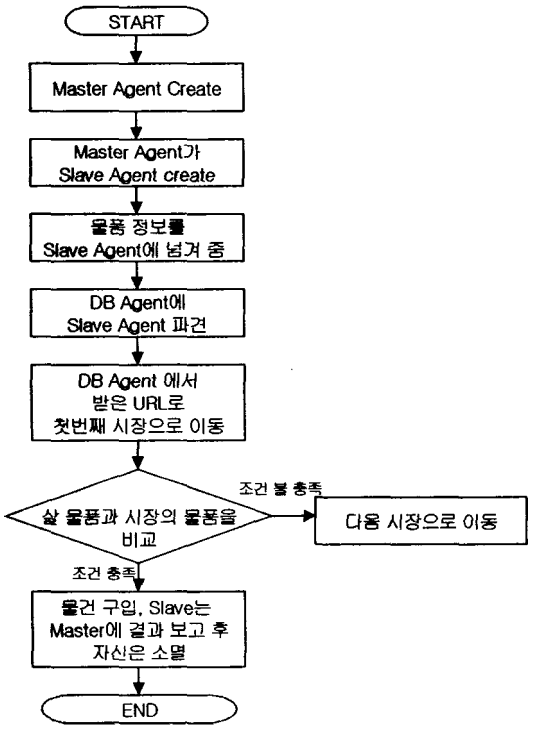

Fig. 4 Flowchart of Customer agent

[그림 5]에서와 같이 고객 마스터(Master) 에이전트는 슬레이브(Slave) 에이전트를 생성한 후, 입력된 조건을 슬레이브 에이전트에 넘겨준다. 슬레이브 에이전트는 모드 메시지(Message)를 수신하고 이 메시지를 사용자가 볼 수 있도록 TextArea 에 보여준다. [그림 5] 에서 고객 마스터 에이전트의 주요 부분을 보인다

```
public void onCreation(Object init)
    {
        send_dialog = new SendDialog(this);
        send_dialog.pack();
        send_dialog.resize(send_dialog.preferredSize());
        send_dialog.setResizable(false);
        send_dialog.show();
    }

 /* Agent 가 생성될 때 처음으로 실행되는 부분으로 구매할
제품정보를 입력받기 위한 다이얼로그박스를 생성 */

 // Slave 를 생성 시켜 시장에 파견
 public void dispatchSlave()
   {
    try
     {
     AgletContext context = getAgletContext();
     AgletProxy    proxy    =    context.createAglet(null,
                              "CliSlave", getProxy());
     //실제로 구매할 Slave Agent 를 생성
     infoprint("Slave Created...");
     proxy.sendMessage(new          Message("itemName",
                                        Name));
     proxy.sendMessage(new       Message("itemQuantity",
                                        Quantity));
     proxy.sendMessage(new      Message("itemMaxPrice",
                                        MaxPrice));
     proxy.sendMessage(new       Message("haveMoney",
                                        Money));
     URL url = new URL("atp://165.229.191.104:9000");
     remoteProxy = proxy.dispatch(url);
     infoprint("Slave dispatched to DB-Agent");
    }catch (InvalidAgletException ex)
    }catch (Exception ex)
    }
}
```

Fig. 5 Code of Customer Master agent

3.3 데이터베이스 에이전트

데이터베이스 에이전트는 고객 에이전트가 물품을 구매하기 위해, 상점 에이전트가 물품을 판매하기 위해 시장으로 이동할 때 협상을 담당한다. 데이터베이스 에이전트는 고객 에

이전트가 원하는 물품을 살 수 있도록 상점 에이전트에 대한 정보를 데이터베이스에 저장하게 된다. [그림 6]에서 상점 에이전트가 데이터베이스 에이전트에 파견되어 데이터베이스 서버에 상품 정보를 등록하는 과정을 보인다. 각각의 시장에 대한 정보를 서버에 보관하여 [그림 7]과 같이 고객 에이전트가 사고자 하는 상품의 정보를 가지고 데이터베이스 에이전트에 방문하면 서버에 저장된 정보를 알려주어 원하는 시장을 찾을 수 있도록 도와준다.

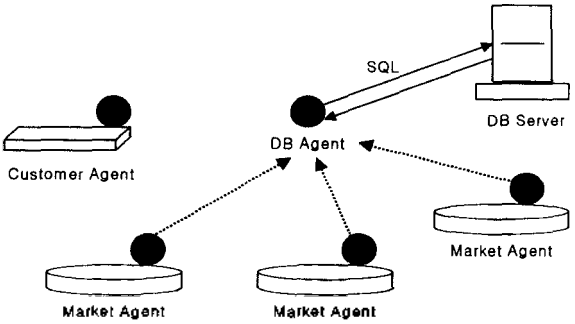

Fig. 6 Entry process of goods information

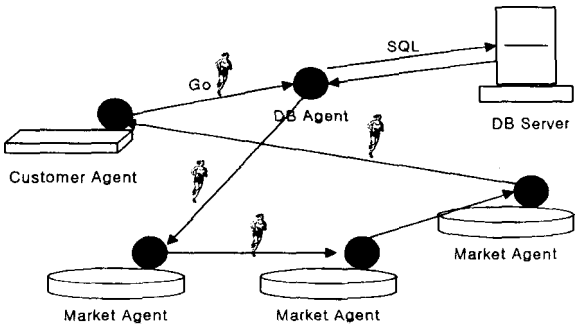

Fig. 7 Acquisition of goods information

[그림 8]에서 데이터베이스 에이전트의 순서도를 보인다.

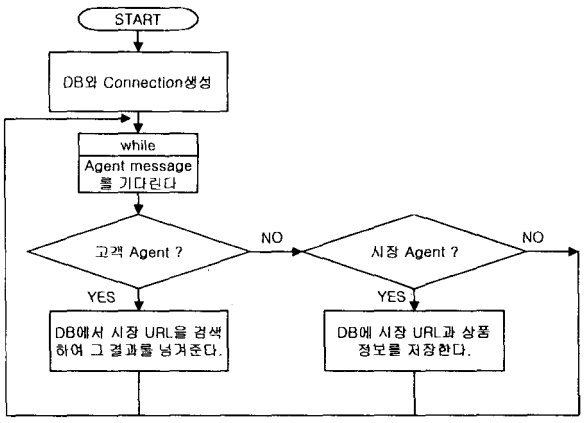

Fig. 8 Flowchart of Database agent

3.4 웹 서버 에이전트

웹 서버 에이전트는 웹 브라우저를 통해서 각 에이전트를 제어하기 위한 에이전트로 Java Class 파일을 디렉토리에서 읽어 브라우저에게 넘겨준다. 브라우저는 Java Class 파일이 포함된 애플릿(Applet)을 실행시킨다. 실행된 애플릿은 [그림 9]와 같은 GUI(Graphic User Interface)를 통해서 현재 상주하고 있는 에이전트들의 정보를 보여주며 새로운 에이전트를 실행시키는 역할을 한다.

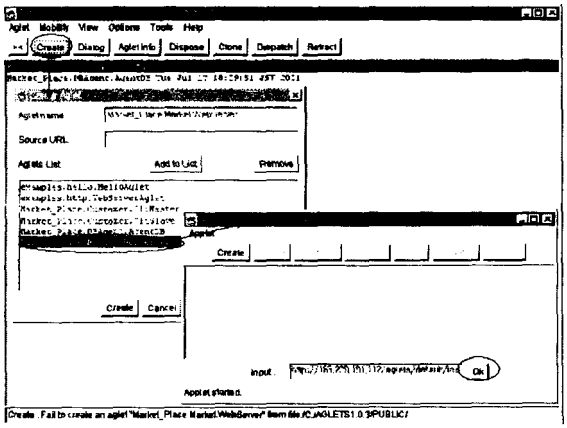

Fig. 9 GUI of Web Server Agent

4. 시스템 구현

본 논문에서는 자바로 구현된 이동 에이전트로 Aglets 개발 툴인 ASDK(Aglets Software Development Kit)를 이용하여 가상 시장을 설계하고 구현하였다. [그림 10]에서 가상 시장의 프로토콜 스택을, [그림 11]에서 시스템의 전체 구성도를 보인다.

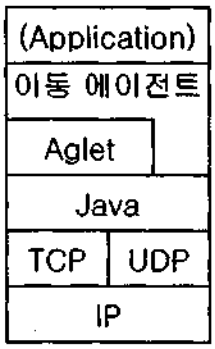

Fig. 10 Protocol Stack of Virtual Market

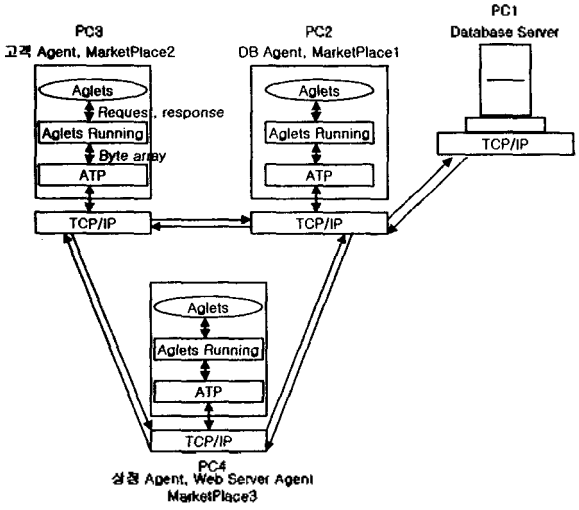

Fig. 11 Architecture of market place

구현한 시스템은 4 대의 시스템으로 구성되었으며, 동작 과정은 다음과 같다. 첫째, PC4 의 웹 서버 에이전트에서 상품의 정보를 가지고 PC2 의 데이터베이스 에이전트로 이동하여 상품 정보와 자신의 정보를 등록하고 3 개의 시장 가운데 하나의 시장으로 이동해서 고객 에이전트를 기다린다. 둘째, 상점 에이전트가 각각의 시장에 파견되어지면 PC3 의 고객 에이전트는 자신이 사고자 하는 상품의 정보를 가지고 먼저 PC2 의 데이터베이스 에이전트로 간다. 셋째, 데이터베이스 에이전트는 PC1 의 데이터베이스 서버에서 그 정보를 가지고 와서 고객 에이전트에게 알려준다. 넷째, 고객 에이전트는 이 정보를 참조하여 원하는 시장을 찾아가서 상점 에이전트와 협상 후 상품을 구매한다. 구매 후에는 파견된 PC3 에 결과를 보고 후 소멸한다. [그림 12]에서 고객 에이전트의 실행화면 및 DB Query 화면을, [그림 13]에서 상점 에이전트의 실행 화면을 보인다.

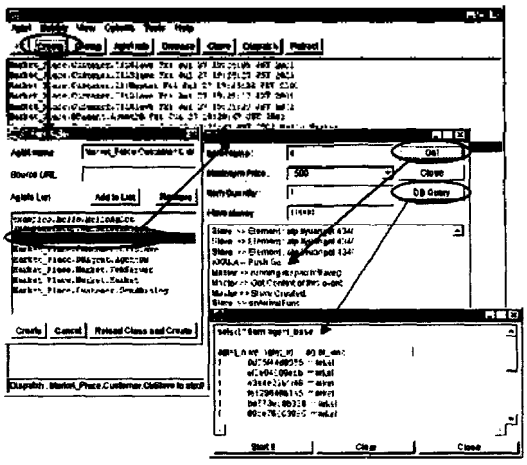

Fig. 12 GUI of Customer Agent and DB Agent

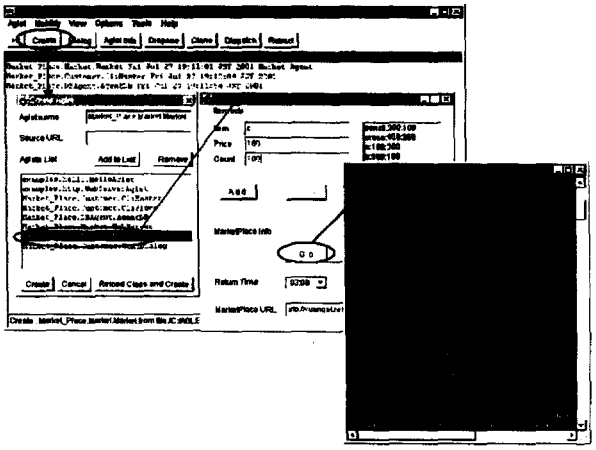

Fig. 13 GUI of Market Agent

5. 결론

최근 인터넷 및 정보통신 기술이 급속하게 발달함에 따라서 네트워크를 기반으로 하는 소프트웨어 및 기술들이 많이 개발되고 있으며, 또한 과거의 인공지능 분야에서 많은 연구가 있었던 에이전트 기술이 최근의 네트워크 기술과 결합한 에이전트 시스템 개발에 많은 노력이 진행 중이다. 특히, 인터넷의 정보를 이용하기 위한 정보 검색 에이전트는 많은 프로그램들이 개발되어 있다. 본 논문에서는 이동 에이전트 시스템을 이용하여 가상상점을 구현하였다. 본 논문의 가상 상점 환경은 각 에이전트가 Aglets 기반의 이동 에이전트 환경 하에서 수행되며, 각 에이전트와 웹 서버 사이의 통신은 HTTP 를 이용하였다. 본 논문에서 구현하지 못한 보안 문제는 현재 연구가 진행 중이며, 보안 문제까지 구현된다면 보다 효율적인 이동 에이전트 시스템이 될 수 있을 것이다.

참고문헌

[1] M. Wooldrige, N. R. Jennings, "Intelligent agents : Theories, Architectures, and Languages", Lecture Notes in AI, Vol. 890, Springer-Verlag, 1995.

[2] J. E. White, "Telescript Technology: Mobile Agents", White Paper, Gerneral Magic, Inc., Sunnyvale, CA., 1996

[3] D. S. Milojicic, M. Condict, F. Reynolds, D. Bolinger, P. Date, "Mobile Objects and Agents", Second USENEX Conference on Object Oriented Technologies and Systems(COOTs), 1996.

[4] M. Straβ er, J. Baumann, F. Hohl, "MOLE: A Jana Based Mobile Agent System", European Conference on Object Oriented Programming, pp. 301-308, 1997.

[5] R. A. Brooks, " A Robuse layered Control System for a Mobile Robot" , IEEE Journal of Robotics and Automation, 2(1), pp.14-23, 1986.

[6] B. G. Buchanan and E. H. Shortliff, Rule-Based Expert Systems: The MyCIN Experiments of the Standford Heuristic Programming Project, Reading, MA: Addison-Wesly, 1984.

[7] E. D. Sacerdoti, " Problem Solving Tactics" , Proceedings of IJCAI-79, Tokyo, Japan, 1979.

[8] N. R. Jennings " Specification and implementation of a belief desire joint-intension architecture for collaborative problem solving" , Journal of Intelligent

and Cooperative Information Systems, 2(3), pp.289-318, 1993.

[9] R. A. Brooks, " A Robust layerd Control System for a Mobile Robot" , IEEE Journal of Robotics and Automation, 2(1), pp.14-23, 1986.

[10] A. Tate, " Generation Project Networks" , Proceedings of IJCAI-77, Boston, USA, 1977.

[11] A. S. Rao and M. P. Georgeff, " Modeling Rational Agents within a BDI-architecture, Proceedings of KR&R-91, pp.473-484, Morgan Kaufmann, 1991.

[12] P. Agre and D. Chapman, " PENGI: An Implementation of a Theory of Activity" , Proceedings of AAAI-87, pp.268-272, Seattle, WA, 1987.

[13] S. Vere and T. Bickmore, " A Basic Agent" , Computational Intelligence, 6:41-60, 1990.

[14] S. Wood, " Planning and Decision Making in Dynamic Domains" , Ellis Horwood, 1993.

[15] P. Chan, R. Lee, "The Java Class Libraries", Second Edition, Vol. 2, Addison-Wesley, 1998.

[16] M. P. Georgeff and A. L. Lansky, " Reactive Reasoning and Planning" , Proceedings of AAAI-87, pp.677-682, Seattle, Wa, 1987.

[17] A. Chvez, P. Maes, " Kasbah: An Agent Marketplace for Buying and Selling Goods" , PAAM96, pp.257-263, 1996.

[18] S. Rosenschein, " Formal Theories of Knowledge in AI and Robotics" , New Generation Computing, pp.345-357, 1985.

[19] J. P. Muller and M. Pischel, " Modeling Interacting Agents in Dynamic Environments" Proceedings of ECAI-94, pp.709-713, 1994.

[20] B. Aoun, " Agent Technology in Electronic Commerce and Information Retrieval on the Internet" , Proc. Of AusWeb96, 1996.

[21] B. Burmeister and K. Sundermeyer, " Cooperative Problem Solving Guided by Intensions and Perception" , Proceedings of MAAMAW-91, pp.77-92, 1991.

[22] I. A. Ferguson, " Towards an Architecture for Adaptive, Rational, Mobile Agents" , Proceedings of MAAMAW-01, pp.249-262, 1992.

[23] O. Etzoini, N. Lesh and R. Segal, " Building softbots for UNIX" , Software Agent - Papers from the 1994 AAAI Spring Symposium, pp.9-16. AAAI

[24] P.Dasgupta, N. Narasimhan, L.E. Moster, and P.M. MelliarSmith. " MAGNET : Mobile Agents for Networked Electronic Trading" , IEEE Transactions on Knowledge and Data Engineering, 1999

[25] W. Li, D. G. Meserschmitt, "Java-To-Go: Itinerative Computing Using Java", Univ. of California, Berkeley, "http://ptolemy.eecs.berkeley.edu/~wli/group/java2go /java-to-go.html".