이동 에이전트 기반 전자 상거래 모형 시스템의 설계 및 구현

김 평 중[†] · 윤 석 환[††]

요 약

이동 에이전트는 네트워크로 연결된 컴퓨터들을 사용자를 대신하여 작업을 수행하며 이동위치를 스스로 결정할 수 있는 자율성(autonomy)과 이동성(mobility)을 가진 소프트웨어 프로그램이다. 이동 에이전트는 네트워크 연결을 계속 유지한 상태로 작업하지 않고 실행 코드와 상태가 시스템간을 이동하며 임무를 수행할 수 있기 때문에 네트워크 연결이 불안정한 무선 망 환경이나 부하가 많이 걸리는 환경에서 활용될 수 있다. 본 논문에서는 이동 에이전트 패러다임을 사용함으로써 이동 컴퓨팅 환경에서 상품 검색 및 매매를 효율적으로 수행할 수 있는 전자상거래 시스템을 구축할 수 있음을 보여주고 있다. 사용자를 대신하여 복수 개의 이동 에이전트들이 시장으로 직접 이동하여 서비스나 상품을 자율적으로 협상하거나 매매하고, 매매결과는 에이전트 실행 코드와 함께 출발지 시스템으로 이동하여 보고한다. 따라서, 우리는 이동 마켓 시스템을 설계하고 시범 구현함으로써 이동 에이전트 패러다임이 이동 컴퓨팅 환경에서 전자상거래 시스템 구축에 적합함을 보인다.

Design and Implementation of Electronic Commerce Prototype System based on Mobile Agent

Phyoung-Jung Kim[†] · Seok-Hwan Yoon[††]

ABSTRACT

Mobile agent is a software program that provides autonomy and mobility for the users in the networked computers by deciding its own movement place. Since mobile agent does not keep up the network connection and migrate its executable code under its own control, it can be often used in the unstable network conditions such as the wireless network and the heavy traffic network. In this paper, we show that the electronic commerce system can be developed efficiently to retrieve and trade the goods in the mobile computing environment by using the mobile agent paradigm. Multiple mobile agents migrate into the market directly, then negotiate and trade autonomously relevant goods and services. The execution result and executable code of the mobile agent are returned to the home place and reported to the users. We show that the mobile agent paradigm is suitable to build the mobile electronic commerce system by designing and prototyping the mobile market system.

키워드 : 이동 에이전트(Mobile agent), 전자상거래(Electronic commerce), 이동마켓 시스템(Mobile market system), 이동 컴퓨팅(Mobile computing)

1. 서 론

전자상거래는 컴퓨터 네트워크를 통하여 정보, 제품, 서비스 등을 팔고 사는 행위로 정의할 수 있고, 정보통신기술을 활용한 모든 종류의 거래 행위들을 포함한다[1]. 전자상거래는 상거래 주체간의 관계에 따라 크게 두 가지 유형으로 나눌 수 있으며, 첫 번째는 기업과 소비자간의 상거래 관계이고, 두 번째는 기업과 기업간의 상거래이다. 기업간 상거래는 전통적인 VAN(Value Added Network) 기반의 EDI (Electronic Data Interchange)를 이용하여 대기업을 중심으로 부분적으로 실현되었으며, 최근 인터넷 기술과 보안 기술이 발전하면서 생산자와 생산 활동을 위해 조달하여야 하는 부품회사들과의 거래는 물론이고, 생산자와 물건을 공급하는 채널에 있는 중간 유통업자, 도매업자 및 소매업자와의 거래를 모두 포함한다. 기업과 소비자간의 거래는 전자 점포나 전자 결재를 통해서 개인 소비자와 거래를 실시하는 것으로 WWW 사용이 일반인에게 확산되면서 급속히 발전하였으며, 인터넷 쇼핑몰(Internet shopping mall), 전자 상점(electronic retailing), 가상 점포(virtual storefront), 온라인 점포(online shop) 등의 다양한 이름으로 불리고 있다.

웹 기술을 이용하여 인터넷 세상에서 사용자가 원하는 상품을 효율적으로 검색하여 구매 또는 판매하는 것은 일반화된 현실로 받아들이고 있다. 상품 검색을 용이하게 할

† 정 회 원 : 충북과학대학 컴퓨터정보과학과 교수
†† 종신회원 : 정보통신연구진흥원 기술가치평가팀 팀장
논문접수 : 2001년 10월 4일, 심사완료 : 2001년 12월 7일

수 있도록 상품 비교 사이트가 출현하는가 하면, 개인들의 상품 매매를 위한 경매나 역 경매 사이트들의 수도 급속하게 늘어가고 있다[2-4].

이동 에이전트는 네트워크로 연결된 컴퓨터들을 사용자를 대신하여 작업을 수행하며 이동위치를 스스로 결정할 수 있는 자율성(autonomy)과 이동성(mobility)을 가진 소프트웨어 프로그램이다. 이동 에이전트는 네트워크 연결을 계속 유지한 상태로 작업하지 않고 실행 코드와 상태가 시스템간을 이동하며 임무를 수행할 수 있기 때문에 네트워크 연결이 불안정한 무선 망 환경이나 부하가 많이 걸리는 환경에서 활용될 수 있다[4, 5].

본 논문에서는 이동 에이전트 패러다임[5-8]을 사용함으로써 이동 컴퓨팅 환경에서 상품 검색 및 매매를 효율적으로 수행할 수 있는 전자상거래 시스템을 구축할 수 있음을 보여주고 있다. 사용자를 대신하여 복수 개의 이동 에이전트들이 시장으로 직접 이동하여 서비스나 상품을 자율적으로 협상하거나 매매하고, 매매결과는 에이전트 실행 코드와 함께 출발지 시스템으로 이동하여 보고한다. 따라서, 우리는 IBM의 aglets[7]를 이용하여 이동 마켓 시스템을 설계하고 시범 구현함으로써 이동 에이전트 패러다임이 이동 컴퓨팅 환경에서 전자상거래 시스템 구축에 적합함을 보인다.

본 논문의 구성은 다음과 같다. 2장에 자바 기반 이동 에이전트 시스템과 전자상거래에 적용을 비교 분석하고 3장에서는 이동 에이전트 기반 전자 상거래 모형 시스템을 설계하고 4장에서 이를 기반으로 시범 구현 결과에 대하여 기술한다. 마지막으로 시범 구현한 전자상거래 모형 시스템을 고찰하고 향후 연구 과제에 대하여 기술한다.

2. 이동 에이전트 시스템과 전자 상거래 적용 분석

이동 에이전트 시스템은 이동 에이전트를 생성, 이동, 수행, 전송, 해석, 및 폐기 등 에이전트의 생명 주기를 관리할 수 있는 플랫폼이다. 이 시스템은 이동 에이전트를 프로그래밍하기 위한 언어, 에이전트를 설치하기 위한 라이브러리, 이동 에이전트를 수행시키기 위한 인터프리터, 이동 에이전트를 전송하기 위한 프로토콜, 및 이동 에이전트를 보호하기 위한 보안등으로 구성된다. 이동 에이전트를 구현하는데 사용하는 프로그래밍 언어의 요구 사항은 호스트에서의 수행 환경을 제공하는 것뿐이다.

최근 들어 전반적인 시스템 경향은 자바 언어에 근거한 기반 구조를 개발하는 추세이다. 대표적인 예를 들어보면, 자바를 포함한 다수의 언어를 지원하는 Ara[9], D'Agent [5], Tacoma[10]가 있으며, 자바 언어만을 기반으로 하는 IBM Tokyo Research Lab.(TRL)의 Aglets[7], Mitsubishi의 Concordia[11], ObjectSpace의 Voyager[12], General Magic의 Odyssey[8] 등이 있다. 이들의 특징별 비교는 <표 1>과 같다.

이동 에이전트 시스템들은 이동 기법에 따라 점프(jump) 모델과 진입점(known entry-point) 모델 등을 사용하고 있다. 첫째, 점프 모델은 시스템이 점프라는 프리미티브 연산을 제공한다. 이 연산은 자동적으로 에이전트의 코드, 객체 상태, 및 제어 상태를 포착하여 다른 시스템으로 보내고, 도착 시스템에서는 상태를 복원하여 점프한 지점에서부터 실행을 계속한다. 이러한 시스템은 D'Agent[5], Ara[9], Tacoma[10] 등이 있다. 둘째, 진입점 모델은 에이전트의 코드와 객체 상태만을 포착하여 다른 시스템으로 보내고, 도착 시스템에서는 진입점(특정 메소드)에서 실행을 계속한다. 이러한 시스템으로 Aglets[7], Concordia[11], Voyager[12], Odyssey[8] 등이 있다. 자바 기반 이동 에이전트 시스템은 자바가상머신에 아무 변경 없이 실행시키는 진입점 모델을 사용하고 있다. 점프 모델이 에이전트 프로그래머에게 훨씬 편리하지만, 제어 상태를 포착하는 루틴이 기존 인터프리터에 추가되어야 한다는 단점이 있다.

이동 에이전트 시스템은 악의를 가진 에이전트로부터 각 호스트를 방어해야 하기 때문에 보안이 중요하다. 이동 에이전트 소유주와 보낼 호스트는 디지털 서명을 하여 전송하고, 서명을 검증한 후 접수 또는 거부한다. 서명과 이동 약력에 따라 자원 접근을 할당하고 안전한 수행 환경에서 이동 에이전트를 실행시켜야 한다. 그러나, Ara[9], Aglets [7], Odyssey[8] 등 대부분의 시스템들은 호스트의 그룹이나 에이전트를 위한 보호막을 제공하지 못하고 있다.

<표 1> 자바 기반 시스템의 특징별 비교

특징 \ 시스템	Aglets[7]	Voyager[12]	Odyssey[8]
Remote creation of agents	No	Yes	No
Remote Java message	None	State-of-the-art	None
Messaging to mobile agents	None	Transparent	None
Messaging mode	Sync, Future	Oneway, Future, sync	None
Life spans	Explicit	5 different modes	Explicit
Mobile naming service	None	Integrated	None
Move to program	Yes	Yes	Yes
Move to object	No	Yes	No
Itineraries	Yes, special API	Yes, no special API	Yes, special API
Connectivity	Restricted	Full	Restricted
Security	Security manager	Security manager	None
Partial connection	No	No	No

<표 1>에서 Voyager는 원격의 이동 에이전트 클래스를 생성할 수 있는데 비하여 Aglets나 Odyssey는 원격 생성이 불가능하다. Voyager에서 일단 클래스가 생성되면 네트웍상의 어디에서나 쉽게 객체를 생성 할 수 있고, 그 객체에

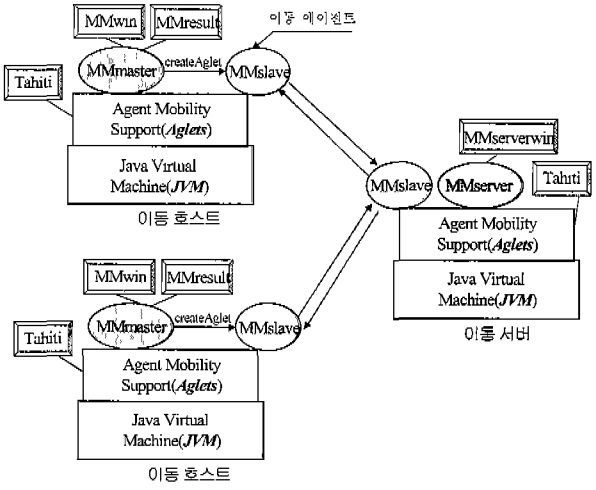
자바 메시지를 보낼 수 있다. Voyager는 synchronous, one-way, 및 future message 모드를 제공한다. 객체가 이동하면 새로운 위치로 이동된 객체에게 메시지를 전송하기 위하여 이동하기 전의 위치에 secretary를 남겨두고 이동한다. 이에 비해 Aglets는 특정 URL에 존재하는 고정 에이전트에게 스트링 명령어를 보낼 수 있지만 그 명령어를 수행하는 책임은 에이전트에게 달려있다[13].

이동 에이전트 시스템의 전자 상거래 적용은 이동 에이전트로부터 자연스럽게 대두되었고 많은 시스템이 연구용과 상용으로 개발되고 있다.

3. 이동 에이전트 기반 전자상거래 모형 시스템의 설계

3.1 전체적인 구성

전자상거래 모형 시스템은 이동 마켓 시스템(mobile market system)이다. 이동 마켓 시스템은 대등 관계(peer-to-peer) 모델이기 때문에 마스터(MMmaster)와 서버(MMserver)로 나누어 설계한다.



(그림 1) 전자상거래 모형시스템(이동마켓 시스템)의 전체적인 구성

(그림 1)에서 이동 호스트와 이동 서버에는 공통적으로 Windows NT 4.0 운영 체제, 자바 가상 머신(JVM)과 Aglets 서버가 설치되어 있다. 이동 호스트에는 MMmaster (Mobile Market Master)와 MMslave(Mobile Market Slave), 이동 서버에는 MMserver (Mobile Market Server)와 MMslave가 각각 설치된다.

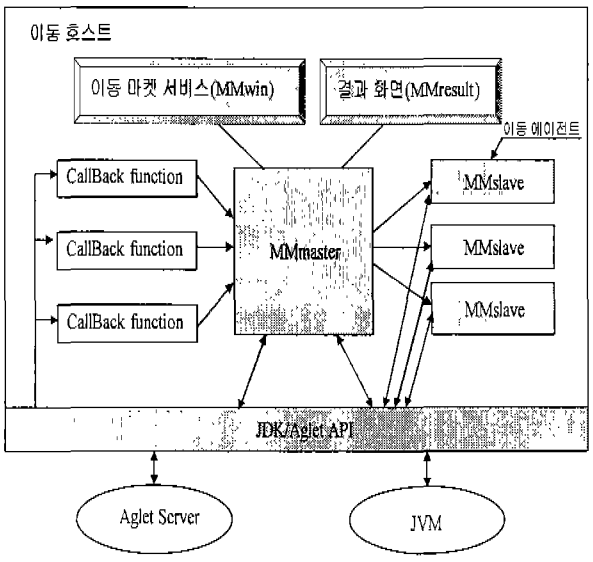
Aglets 서버는 ATP(Agent Transfer Protocol) 데몬, Aglets 컨텍스트, 및 Aglets viewer(Tahiti)로 구성된다[7]. ATP 데몬은 현재 동작중인 Aglets과 통신하거나 전송하기 위하여 ATP 메시지를 처리한다. Aglets 콘텍스트는 서버에 존재하는 Aglets을 수행시키며, Aglets의 생명 주기를 제어한다. Aglets viewer는 이동 에이전트 자바 클래스를 생성하여 동작시키는 사용자 인터페이스와 이동 에이전트의 생명 주기를 제어는 사용자 인터페이스를 갖는다.

MMmaster는 이동 호스트의 고정(stationary) 에이전트로서 사용자와 인터페이스 하면서, MMslave의 인스턴스(instance)인 이동 에이전트를 생성시켜 이동 서버인 MMserver로 보내고, 처리 결과를 보고 받아 사용자에게 그 결과를 보여준다. MMslave는 MMmaster로부터 생성되는 이동 에이전트로서 방문하고자하는 서버 목록에 따라 사용자의 구매 또는 판매 물품에 대한 요구 사항을 싣고 이동 서버인 MMserver로 직접 이동하여 이동 서버에서의 처리 결과를 가지고 이동 호스트로 다시 돌아와 MMmaster에게 처리 결과를 보고한다. 한편, MMserver는 이동 서버에 존재하는 고정 에이전트로서, MMmaster가 보낸 MMslave가 도착하면 이를 대기 큐에 넣고 요구 사항에 따라 처리하고 그 결과를 MMslave에 넘겨준다.

3.2 MMmaster

MMmaster는 사용자 인터페이스 MMwin으로부터 사용자의 요구 사항을 입력받고 Buy 버튼이나 Sell 버튼이 눌려지면 MMslave 인스턴스를 생성시킨다. MMslave로부터 수행 결과를 받으면 MMresult 화면을 띄워 임무 수행 결과를 사용자에게 보고한다. (그림 2)는 MMmaster 클래스, MMwin 클래스, MMresult 클래스, Aglets 서버가 제공하는 API, JDK1.1.8이 제공하는 API, 콜백 (callback)함수, MMslave간 수행 관계를 보여준다.

사용자가 xmas.demo.MMmaster를 Tahiti 화면에서 실행하면, MMmaster는 MMS 화면인 MMwin을 보여준다. MMwin 화면에서 사용자 이름, 소속, 전화번호, e-mail 등의 사용자 정보와 물품 종류, 제목, 수량, 단가 등의 물품



(그림 2) MMmaster 수행 중 구성 요소들 간의 관계

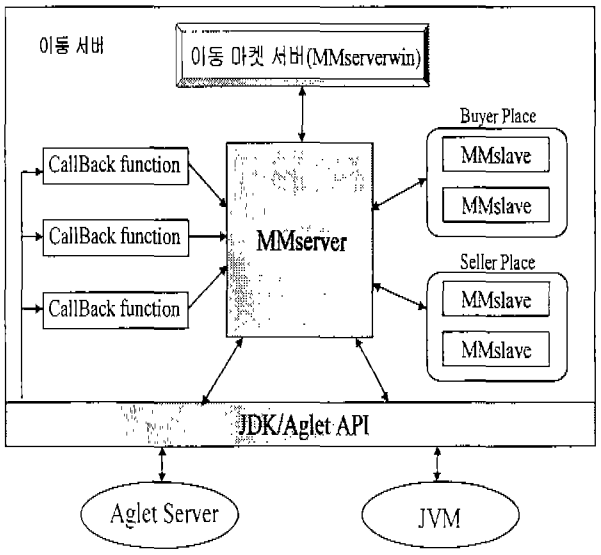
정보를 입력하면, MMmaster는 MMrfp 인스턴스를 생성시키고 Buy 버튼이나 Sell 버튼을 누르면 MMslave를 생성시킨다. MMmaster는 생성된 MMrfp 인스턴스를 이동 에이전트인 MMslave에게 전달하고 방문 목록에 등록된 곳(MMserver)으로 dispatch한다. MMslave는 주어진 임무를 완수하고 홈 에이전트인 MMmaster에게 되돌아와 수행 결과를 보고한다. 이를 위해 MMslave는 MMmaster Id와 MMmaster URL를 갖고 있다. MMmaster는 생성된 MMslave의 정보(MMslave Id, MMmaster Id)를 관리하고, 수행 후 돌아와 앉아 있을 자리를 유지한다. MMslave가 자리에 앉아 있고 자기 차례가 되면 MMmaster에게 보고 메시지를 띄워 보고한다. MMmaster는 MMresult 화면을 통하여 수행 결과를 보여주고 MMslave를 폐기한다.

3.3 MMslave

MMslave 클래스는 이동 에이전트를 구현하는 것으로서 Aglets 서버에서 제공되는 API들을 기본적으로 이용한다. 주요 기능은 목적지에 따라 MMserver로 이동하여 부여된 임무를 수행하고, MMserver에 도착하여 등록하고, MMserver에게 가져온 MMrfp 정보를 전달하여 수행시키고, 그리고 수행 결과를 받아 MMmaster로 되돌아와 결과를 보고하는 기능 등을 갖는다.

3.4 MMserver

이동 서버에 존재하는 MMserver는 (그림 3)과 같이 MMserver 클래스를 중심으로 구성된다. MMserver에서는 여러 개의 MMslave들이 이동 방문하여 사용자의 구매 또는 판매 정보를 주고받는다. 이동 호스트로부터 MMserver를 방문한 이동 에이전트인 MMslave들은 MMserver에 등록한 후 상태 목록에 표시한다. MMserver는 MMslave의 요구



(그림 3) MMserver 수행 중 구성 요소들 간의 관계

정보에 따라 구매 MMslave이면 구매자 대기소, 판매 MMslave이면 판매자 대기소에 넣고 가져온 정보를 해석한다. 각 MMslave가 가져온 MMslave ID와 사용자 정보, 물품 정보들을 Item List에 추가하여 표시한다. MMserver는 MMslave가 가져온 물품 정보에 따라 MatchMaking을 처리한다. MMserver는 물품 정보가 일치되면 바로 그 결과를 MMslave에게 되돌려준다. MatchMaking은 각 MMslave가 도착할 때마다 수행되고 Item List에 표시한다. MMslave가 만족한 결과를 얻었을 경우, MMserver는 MMslave들을 홈 에이전트인 MMmaster에게 되돌아가도록 한다.

4. 이동 에이전트 기반 전자상거래 모형 시스템의 시범 구현

Aglets 기반의 이동 에이전트 기술을 이용하여 전자 상거래 모형 시스템 MMS(Mobile Market System)를 구현하였다. 개발 환경은 Windows NT 4.0, 프로그래밍 언어로 자바 (JDK1.1.8), 이동 에이전트 시스템으로 IBM Aglets Workbench 1.0.3 버전을 사용하여 설계하였다. Aglets 시스템은 현재 JDK1.1 버전만을 지원하고 있으므로 JDK1.1.8을 사용하였다.

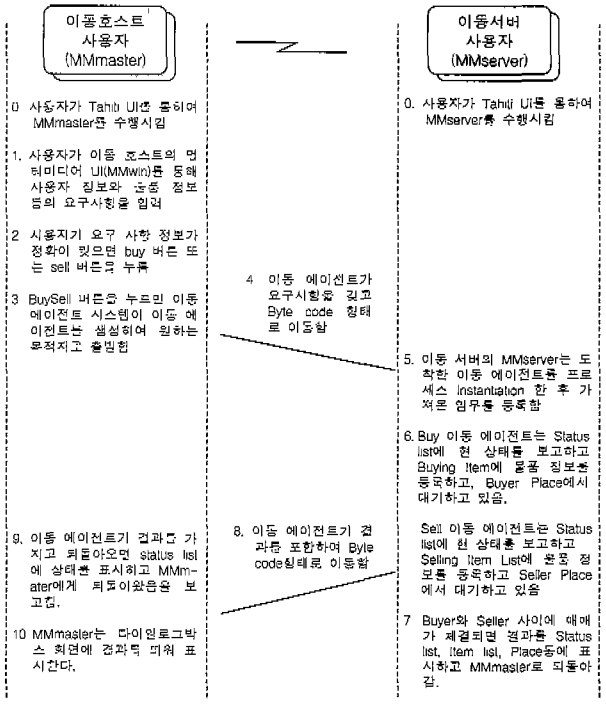
MMS는 이동 에이전트가 사용자를 대신하여 사용자의 구매 또는 판매 정보를 가지고 시장에 돌아다니면서 적절한 구매자 또는 판매자를 찾아 관련 정보 및 상품을 협상하거나 교환하는 서비스이다. MMS에서는 물건을 구 매하거나 판매하고자 하는 사람을 MMmaster로, 구매자 또는 판매자들의 물품 매매를 대행하는 사람을 MMslave로, 물품의 매매가 이루어지는 장터를 MMserver로 시뮬레이션하였다. MMserver에는 MMmaster들의 물품 매매 대리인인 MMslave들이 모여 매매를 체결하는 장소가 된다. MMS의 수행 절차는 다음과 같다.

① 이동 호스트의 사용자, 즉 MMmaster가 시간과 장소에 구애되지 않고 원하는 정보 및 상품을 매매할 수 있도록 사용자 인터페이스를 통해 요구사항을 MMmaster에게 입력한다. MMmaster는 사용자가 구매 또는 판매를 선택하면 사용자 요구사항을 포함하는 이동 에이전트 MMslave를 생성시키고, 이를 MMserver에게 보낸다. 이 때 복수 개의 MMslave를 생성하여 복수 개의 MMserver로 병렬 전송할 수 있다.

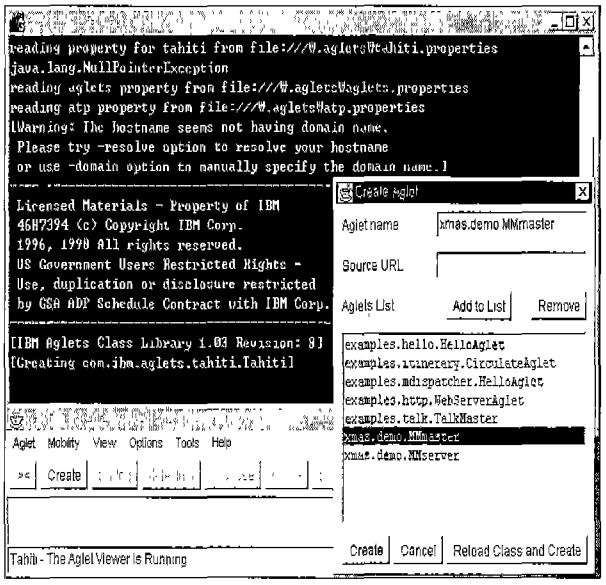
② MMserver에 도착된 MMslave들은 자신의 임무를 수행하기 위하여 시장에서 물건을 판매하려고 대기하고 있거나 물건을 구매하기 위하여 다른 MMserver들을 찾아 다니며 matchmaking을 수행한다. 만약, 구매 이동 에이전트가 원하는 물건을 팔면 구매 내역 결과를 저장하고 홈 에이전트인 MMmaster에게 되돌아가고, 동시에 물건을 팔았던 판매 이동 에이전트도 판매 내역 결과를 저

장하여 MMmaster로 되돌아간다.

③ MMmaster에 도착된 MMslave들은 구매 또는 판매 내역 결과를 사용자에게 보고한다.



(그림 4) MMS의 동작 흐름



(그림 5) MMmaster 실행

사용자는 (그림 5)에서와 같이 Tahiti를 통하여 이동 호스트의 MMmaster와 이동 서버의 MMserver를 각각 기동시킨다. MMmaster는 (그림 6)의 MMwin 화면을 띄우고 MM server는 (그림 7)의 화면을 띄운다.

MMmaster 클래스는 약 1200 라인으로 구성되고 이동 호스트에서 동작하는 모든 기능을 제어하는 핵심 클래스이다. 사용자가 MMS를 invocation하면 MMmaster 클래스의 onCreation() callback 함수가 불리고, 이 함수 내에서 MMwin 클래스를 호출함으로써 이동 마켓 서비스 화면이 나타난다. 아래의 코드는 onCreation() callback 함수를 보여준다.

```
public void onCreation (Object obj)
{
      ac = getAgletContext() ;       // get the currently executing
                                        agletContext
      uiwin = new MMwin(this) ;  // show the MMS UI window
      uiwin.show() ;
}
```

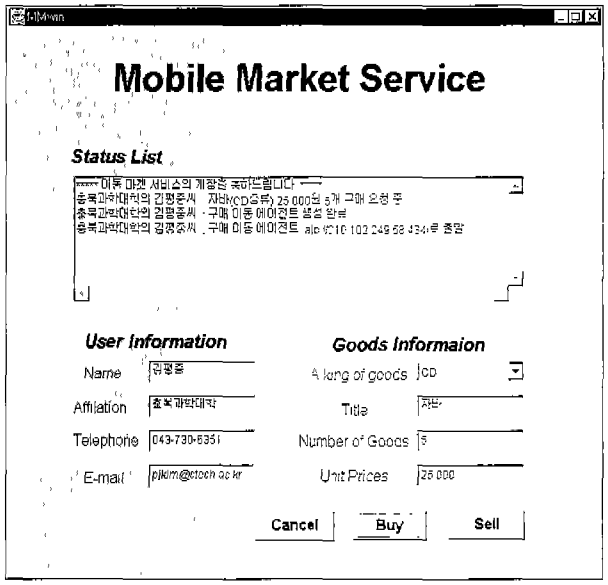
MMwin 클래스에서 Buy 버튼이나 Sell 버튼을 누르면 이동 마켓 서비스 화면의 사용자 정보(사용자의 이름, 소속, 전화 번호)와 물품 정보(물품 종류, 물품 이름, 수량, 단가)를 MMrfp 인스턴스에 저장하여 MMmaster 클래스에게 전달한다. MMmaster 클래스는 buyPressed()나 sellPressed() 함수를 호출한다.

buyPressed()나 sellPressed() 함수는 이동 에이전트인 MMslave 클래스의 인스턴스를 생성하고, 요구받은 정보를 RFPTOSLAVE 메시지를 만들어 이동 에이전트에게 넘겨준 다음, 주어진 itinerary로 dispatch한다.

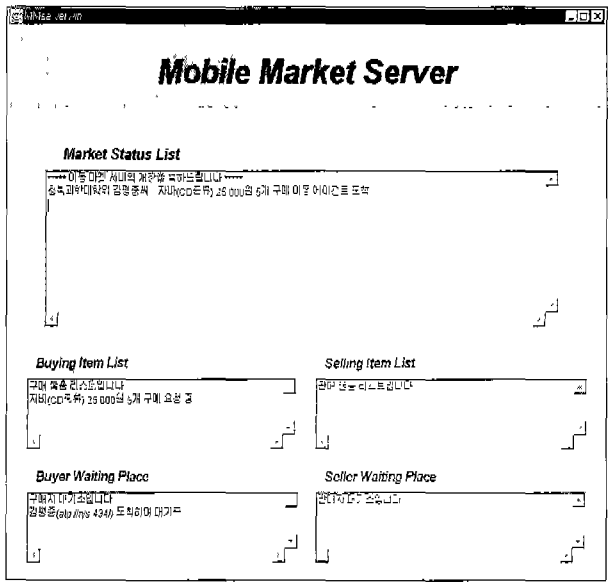
```
public void buyPressed(MMrfp rfp)
{
      rfp.masterId = getAgletID() ;    // get MMmaster ID
      // Creates an instance of the specified aglet class
      sProxy = ac.createAglet(null, SlaveClassName, rfp.masterId) ;
      rfp.slaveId = sProxy.getAgletID() ; //get Aglet Id of MMslave
      // get the URL of the daemon serving this context
      rfp.masterUrl = ac.getHostingURL() ;
      rfp.slaveUrl = getSlaveUrl() ;   // get Destinations of Slave
      rfp.buying = true ;              // set the buying aglet
      Message msg = (Message)rfp.constRfp() ; // construct
                                          RFPTOSLAVE message
      sProxy.sendMessage(msg) ; // Sends a message to MMslave
      putMasterTable(rfp) ;          // put into Vector table
      sHashTable.put(rfp.slaveId, rfp.slaveUrl) ; // put into hash
                                                    table
      sProxy.dispatch(rfp.slaveUrl) ; // Dispatches the aglet to the
                                        MMserver
}
```

이동 서버의 MMserver는 도착한 Buy 또는 Sell 이동 에이전트를 프로세스로 생성하여 초기화시키고 주어진 임무를 수행하도록 한다. 이동 서버에 도착한 이동 에이전트는 (그림 7)의 화면과 같이 Market Status List에 도착을 알리는 메시지를 보여 주고 구매 또는 판매 의뢰에 따라 Buying Item List 또는 Selling Item List에 물품 정보를 등록하고 새로운 쓰레드를 생성하여 요구된 작업을 수행한다.

MMserver 클래스는 약 1500 라인으로 구성되고 이동 서버에서 동작하는 모든 기능을 제어하는 핵심 클래스이다.



(그림 6) MMwin 화면으로부터 정보 입력



(그림 7) 이동 서버에 도착한 Buy 이동 에이전트

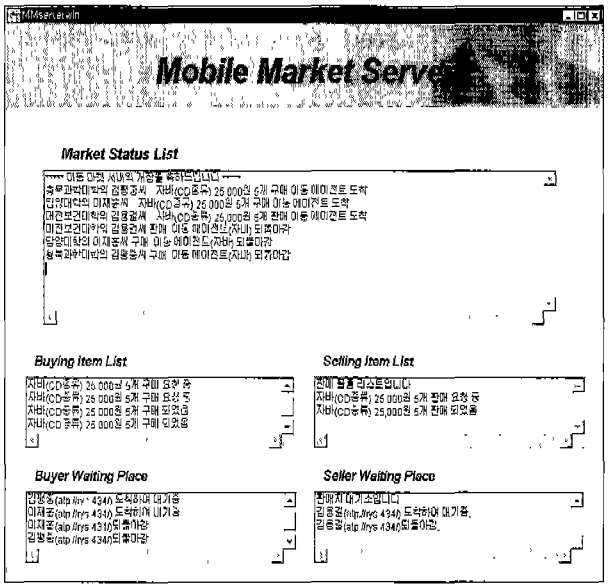
사용자가 이동 마켓 서버 응용 프로그램을 invocation하면 MMserver 클래스의 onCreation() callback 함수가 불리고, 이 함수 내에서 MMserverwin 클래스를 호출함으로써 이동 마켓 서버 화면이 나타난다. 그리고 run() callback 함수가 불린다. 외부로부터 메시지를 접수받기 위하여 handleMessage() callback 함수를 사용한다. MMserver 클래스는 Taihiti 서버로부터 instantiation 될 때 onCreation() 함수, run() 함수가 차례로 호출되고, 그 이외의 어떤 행위도 발생되지 않는다.

```
public boolean handleMessage(Message msg)
{
    if (msg.sameKind("REGITOBA"))
    {
        regiSlave(msg) ;  // The message is sent to MMserver
                          for registration
        return true ;
}
public void regiSlave(Message msg)
{
    MMrfp r = new MMrfp() ;      // user information
    r.uname = new String((String)msg.getArg("NAM")) ;
    r.uaffil = new String((String)msg.getArg("AFF")) ;
        ......
    // goods information
    r.kind = new String((String)msg.getArg("KND")) ;
    r.title = new String((String)msg.getArg("TIT")) ;
    r.price = new String((String)msg.getArg("PRI")) ;
        ......
    // create new thread to process the MMrfp
    RfpThread rt = new RfpThread(r) ;
    // The following code would then create a thread and start
       it running :
    Thread thr = new Thread(rt) ;
    thr.start() ;
}
```

이동 서버에서 구매자와 판매자 사이에 매매가 체결되면 그 결과를 (그림 8)과 같이 Status List, Item List, Waiting Place에 적절한 메시지를 표시한다. 매매가 체결된 이동 에이전트들은 실행 결과를 갖고 자신의 홈 에이전트인 이동 호스트의 MMmaster로 되돌아간다. 이동 서버에서의 매매 체결은 구매자와 판매자의 상품 정보(상품명과 가격)를 비교함으로써 이루어진다.

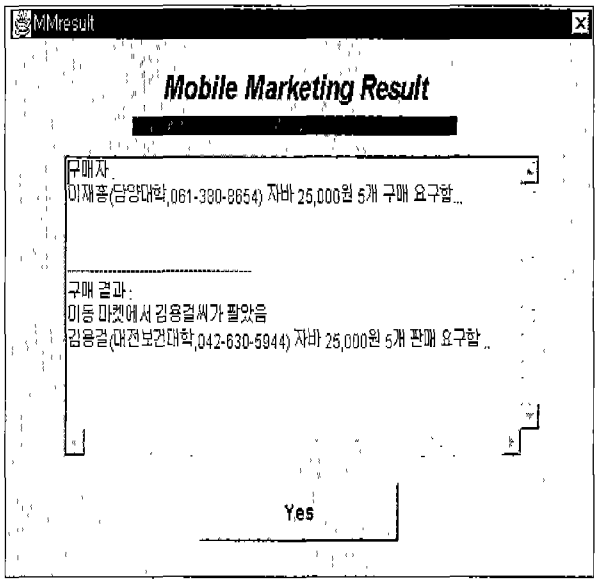
```
class RfpThread implements Runnable
{
    MMrfp r ;
    RfpThread(MMrfp pr) {
        r = pr ;
    } ;
    public void run() {
        doRfp() ;
    }
    // Create one thread per one registration
    synchronized void doRfp ()
    {
        Vector ans = null ; // Result for title-Matchmaking
        // display RFP-Registration into MMserverwin
        regiDisplay(r) ;
        MMrfpAnalysis(r) ;
        if (r.buying )
        {
            ans = matchSTitle(r) ;
            if (ans.size() != 0)
            {
                // construct BUYRESULTTOSLAVE message
                Message msg = constBuyResult(r, ans) ;
                AgletProxy sp =
                    getAgletContext().getAgletProxy(r.slaveId) ;
                removeBTable(r) ;  // remove buyer-entry from
                                      regiBTable
                Object obj = sp.sendMessage(msg) ; // send the
                                      result to slave
```

```
            goBack(sp, r) ;
        }
    }
```



(그림 8) 이동 서버에서 매매 체결

Buy 또는 Sell 이동 에이전트가 처리 결과를 가지고 이동 호스트로 되돌아오면 MMmaster는 Status List에 상태를 표시하고 (그림 9)의 MMresult 화면에 매매 결과를 보여 준다.



(그림 9) 이동 호스트에서의 이동 에이전트 수행 결과 화면

5. 결론 및 향후연구

본 논문에서는 사용자를 대신하여 복수 개의 이동 에이전트들이 시장으로 직접 이동하여 서비스나 상품을 자율적으로 협상하거나 매매하는 이동 에이전트 기반 전자상거래 시스템을 개발하였다. 여기에서 협상은 에이전트간 동일 물품에 대하여 값을 비교하는 매칭 기법을 활용하였다. 사용자가 노트북과 같은 이동 호스트를 이용하여 사용자 정보와 물품 정보를 입력하고 이동 에이전트로 하여금 물품 매매를 대행하게 하고 매매가 체결되면 매매 결과를 보고 받는 이동 마켓 시스템의 시범 구현을 기술하였다. 이것은 소비자 구매 행동 모델[1,3]에서 2단계의 "상품 탐색(무엇을 살 것인가)"과 3단계의 "판매자 탐색(누구로부터 살 것인가)"에 해당되는 과정을 이동 에이전트 기술을 이용하여 전자상거래 서비스의 일부 기능들을 구현하였다. 특히, 사용자가 상품을 검색할 경우 복수개의 이동 에이전트를 생성하여 복수개의 서버로 병렬 전송하여 수집함으로써 병렬 정보 검색 가능성을 보여주고 있다.

향후 연구로는 전자 상거래 서비스 중에서 판매자 탐색에서 가격과 수량만을 이용하였으나, 배달 조건·유지 보수·지불 방법 등을 고려한 4단계의 협상 과정을 추가 구현하여야 한다. 또한, 시범 구현된 시스템은 이동 에이전트 기술을 전자 상거래에 적용할 수 있음을 보인 초보적인 서비스이며, 사용자의 구매 패턴(구매 빈도가 높은 물품 종류, 협상 조건 등)을 이용한 이동 서버의 선택, 이동 에이전트의 이동 서버 탐색 방안에 대한 연구, 협상 과정의 구현, 웹 상의 쇼핑 사이트들로부터 상품 정보의 자동 추출, 필터링 기법 등의 연구가 진행될 예정이다.

참 고 문 헌

[1] M. Bloch, Y. Pigneur, & A. Seegev, "On the Road of Electronic Commerce-a Business Value Framework, Gaining Competitive Advantage and Some Research Issues," *http : //www.hec.unil.ch/mblock/docs/roadtoec.html*, Mar. 1996.

[2] A. Chavez and P. Maes, "Kasbah : An Agent Marketplace for Buying and Selling Goods," *Proc. of 1st International Conference on the Practical Application of Intelligent Agents & Multi Agent Technology*, 1998.

[3] R. B. O. Doorenbos and D. Weld, "A Scalable Comparison-Shopping Agent for the World Wide Web," *Proc. of the First International Conference on Autonomous Agents(Agents 97)*, pp.39-48, 1997.

[4] Michael P. Wellman and Peter R. Wurman, "Real Time Issues for Internet Auctions," *Proc. of the First IEEE Workshop on Dependable and Real_time E-Commerce Systems(DARE 98)*, 1998.

[5] R. Gray, D. Kotz, S. Nog, D. Rus and G. Cybenco, "Mobile Agents for Mobile Computing," *Proc. of the 2nd Aizu Int'l Symposium on Parallel Algorithms/Architectures Synthesis (pAs 97)*, Japan, pp.17, Mar. 1997.

[6] Jae-Hong Lee and Pyeong-Jung Kim, "Mobile Agent Sys-

tem Architecture for the Partial Connection Problem Management," *Proc. of International Conference on Advanced Communication Technology(ICACT'2000)*, pp.165-169. 2000.

[7] D. B. Lange and M. Oshima, *Programming and Deploying Mobile Agents with Aglets*, Addison-Wesley, ISBN : 0-201-32582-9, 1998.

[8] J. White, "Mobile Agents White Paper," General Magic White paper, *http : //www.genmagic.com/agents/Whitepaper/whitepaper.html*, pp.28, 1996.

[9] H. Peine and T. Stolpmann, "The Architecture of the Ara Platform for Mobile Agents," *Proceedings of the First Int.l Workshop on Mobile Agents(MA97)*, Berlin, Germany, Apr. 1997.

[10] D. Johansen, F. B. Schneider and R. Renesse, "Operating System Support for Mobile Agents," *Proc. 5th IEEE Workshop on Hot Topics in Operating Systems*, 1998.

[11] D. Wong, and et. al, "Condordia : An Infrastructure for Collaborating mobile Agents," *Proc. of 1st Int'l Workshop on Mobile Agent(MA'97)*, 1997.

[12] ObjectSpace, "Voyager : ORB3.0 Devcloper Guide," ObjectSpace Inc. *http : //www.objectspace.com/Voyager/*, 1999.

[13] ObjectSpace, "A Comparison : ObjectSpace Voyager, General Magic Odyssey, IBM Aglets," *http : //www.objectspace.com/Voyager/*, 1997.

김 평 중

email : pjkim@ctech.ac.kr

1985년 충남대학교 계산통계학과(학사)

1995년 KAIST 전산학과(공학석사)

2000년 충남대학교 컴퓨터과학과(이학박사)

1995년 정보처리기술사 취득

1987년~1988년 포항종합제철㈜ 전산시스템부

1988년~1998년 ETRI 멀티미디어연구부(선임연구원)

1998년~2000년 한국정보처리학회 학회지 편집위원

1998년~현재 충북과학대학 컴퓨터정보과학과 조교수

관심분야 : 이동에이전트, 전자상거래, 컴퓨터네트워크, 분산 멀티미디어

윤 석 환

e-mail : yoonsh@iita.re.kr

1982년 아주대학교 산업공학과 졸업(공학사)

1984년 건국대학교 산업공학과(공학석사)

1992년 품질관리 기술사 자격획득

1996년 아주대학교 산업공학과(박사)

1986년~1997년 한국전자통신연구원 책임연구원

1998년~현재 정보통신연구진흥원 기술가치평가팀장 (책임연구원)

1998년~현재 한국정보처리학회 이사 겸 학회지 편집위원장

관심분야 : 분산처리, 소프트웨어 공학, 품질보증, 개발체계, ERP