기능성 다중 중개매체를 이용한 모빌 에이전트 시스템

윤영준°, 송종길, 조영임, 김유신
부산대학교, 평택대학교
{yjyoon00, jksong, yooshin}@pusan.cc.ac.kr
{yicho}@ptuniv.ac.kr

Mobile Agent System using Classified Multi Middleman

Young-Jun Yun° Jong-Kil Song Yoo-Shin Kim
Dept. Electronic Engineering, Pusan National University
Young-Im Cho
Dept. Computer Science, Pyongtaek University

요 약

우리들은 현재 엄청난 양과 질의 정보세상에서 생활하고 있다. 때문에 이들 정보들을 보다 효과적으로 활용하고자 하는 것은 당연한 욕심이다. 그 중 대표적인 예로 RPC(Remote Procedure Call)가 있다. 그리고 mobile code를 이용해 새로운 패러다임을 선보이는 Mobile Agent가 있다. 현재 여러 mobile agent 시스템이 Java RMI(Remote Method Call)를 활용하고 있다. 이전의 RPC는 하나의 완전한 객체가 아닌 일반 data만을 네트웍을 통해서 전송할 수 있었으나, Java RMI를 적극 활용하는 mobile agent는 자신이 하나의 객체로 구현되어 네트웍을 통해 목적지 서버로 혹은 다른 호스트로 이동한 후 원격지에서 직접적으로 자기 내부의 메소드를 실행할 수 있다[1,2,3,4]. 이는 사용자의 간섭을 배제한 agent의 자율적이고 독립된 행동을 지원하기 때문에 분산처리분야에서 새로운 패러다임을 제시하고 있다. 그리고 agent가 적합한 서비스를 제공하는 서버를 효율적이고 정확하게 찾는 것이 무엇보다 중요하다. middle agent가 바로 이런 기능을 가진 모듈이다. 본 논문에서는 효율적인 모빌 에이전트 시스템을 구축하기 위해 서버와 클라이언트와의 상호 연결을 담당하는 새로운 middle agent로서 MiddleMan을 제안하였으며 이것을 이용한 다중 사용방법에 대해서 연구하였다[5].

1. 서 론

요즘의 우리들은 무궁한 양의 정보세상에서 살고 있다. 때문에 이들 정보들을 보다 효과적으로 활용하고자 하는 것은 당연한 욕심이다. 아직까지는 사람이 직접 서버로 접근하여 필요한 정보를 찾고 서비스를 예약하는 것이 주를 이루고 있었으나 새로운 패러다임을 예고하는 mobile agent가 앞으로의 생활습관을 바꾸어줄 것이다.

Mobile agent는 하나의 작은 프로그램이다[1,6]. 이 개념은 1996년 General Magic사의 Telescript를 통해서 세상에 처음으로 소개되었다[6]. 그 뒤 IBM의 Aglet, Baltimore의 Agent Tcl 등의 후속모델들은 Telescript에 기반을 두고 있다[7,8]. 여기서 말하는 mobile agent는 사용자의 요구사항을 가지고 네트웍을 통해 목적지로 이동하는 것만을 의미하지 않고, 원격지 내에서 사용자의 직접적인 간섭을 배제한 채 자율적으로 자신의 업무를 수행함을 의미한다[2].

Mobile Agent는 그 개념을 바탕으로 여러 가지 장점을 가진다[1,6]. 첫째는 RPC에 비해서 네트웍 트래픽의 양이 현저히 줄어든다는 점이다. 한 번 원격지로 전송되면 더 이상 사용자와의 통신이 필요 없기 때문이다. 둘째, agent가 원격지 내부에서 직접적으로 행동하므로 업무처리 시간이 상당히 단축된다. 셋째는 Java라는 객체지향언어를 사용하여 구현하기 때문에 상속성을 이용하여 agent의 생성이 용이하다는 것이다. 그리고 넷째는 mobile agent의 특성상 네트웍 결함 등의 경우에 추후 재접속을 통한 출발지로의 복귀가 가능하므로 매우 안정적이라는 점이다.

Mobile agent 시스템의 구현에 있어서 최우선 과제는 agent에게 mobility를 부여하는 것이다[6]. 이것은 앞서 설명했듯이 단순히 agent가 네트웍을 통해서 원격 서버로 이동하는 것이 아니라, 원격지에서 독립된 객체로서 자율적인 자신의 업무수행을 가능하도록 하는 것이다. 기존의 여러 연구에서는 mobile agent의 개념만을 나타낼 뿐 명확한 모델을 제시하지 않고 있다. 때문에 본 논문에서는 우리의 mobile agent system을 그림1을 통해서 분명하게 제안하고 있다.

또한 agent가 최적의 서버로 신속 정확히 찾아가게끔 도와줄 환경조성이 필요하다. 이런 역할을 하는 모듈이 middle agent인데[5], 이에 대한 연구가 국내외적으로 부족하다. 따라서 본 논문에서는 middle agent의 특성화된 모델인 MiddleMan을 다중 형태로 제안하여 신속하고 정확한 중계서비스 보장해주도록 설계하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 mobile agent를 구현하기 위한 기본 요소 및 구조 및 다중 MiddleMan에 관하여 제안한다. 3장에서 이를 이용한 간단한 시나리오를 설명한 뒤 4장에서는 향후 연구과제에 관하여 설명하고자 한다.

2. Mobile Agent

2.1 기본 구조

General Magic사의 Telescript는 mobile agent를 최초로 소개하면서 places, agents, travel, meetings, connections, authorities, permits 등의 7가지 기본 개념을 제시하였다[6]. 그 후 등장했던 있는 Aglet, Agent Tcl 등도 위의 개념에 비롯되었다[7,8].

지금까지 mobile agent의 응용 사례는 여럿 있으나, 그 모델의 명확한 제안은 없었다. 때문에 본 논문에서는 기본 구조 및 개념을 아래 그림1 과 같이 정리하여 제안하고자 한다.

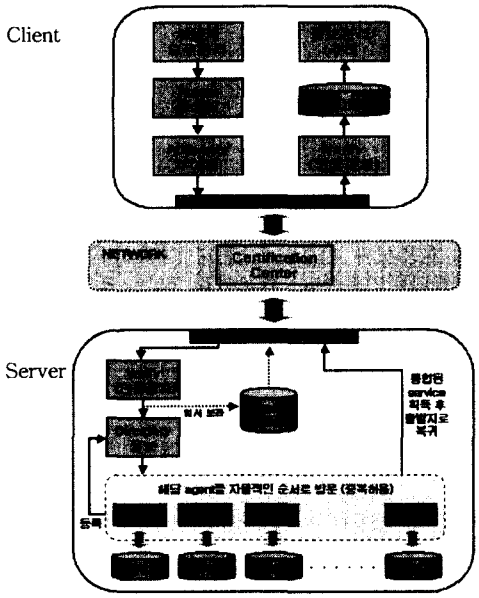

그림 1. Mobile Agent System Architecture

위의 그림 1 은 현재 우리가 구현 중에 있는 mobile agent 시스템의 기본 구조이다. 먼저 클라이언트 쪽에서는 인터페이스를 통해 사용자의 요구사항을 입력받고 분석하여 이를 처리하기에 적합한 에이전트를 생성한다(*agent*). 그 후 에이전트에게 authority를 부여하여 네트웍을 통해 목적지의 서버로 전송시킨다(*authority, travel*). 업무를 수행하고 돌아온 에이전트는 다시 인증절차를 거친 뒤 DB에게 정보들을 저장한다. 클라이언트 내의 관리 에이전트는 이 DB내의 정보들을 활용하여 사용자에게 추천한다.

서버 측면에서의 관점은 다음과 같다. 네트웍을 통하여 서버의 내부로 들어온 에이전트는 인증절차를 거친 후 서버내의 Directory에서 자신이 원하는 서비스를 제공받기 위해서 어떤 DB에 접근해야 하는지 그 목적지와 접근 순서를 판단하게 된다(*connections, authorities*). 이 때, 서버에서는 내부의 오류에 대비하여 클라이언트 에이전트를 내부의 안전한 장소에 보관해둔다. 그리고, 서버의 각 DB를 담당하는 에이전트는 미리 자신의 DB에 대한 정보를 Directory에 등록해두어야 한다(*places*). 클라이언트 에이전트는 DB를 담당하는 여러 에이전트들과의 통신과정을 통해서 원하는 서비스를 제공받는다(*permits, meetings*). 이 때 각 DB를 방문하는 순서는 정해진 것이 아니라 상황에 따라서 유동적이며, 반복적이다. 모든 정보를 획득한 후 에이전트는 출발지로 복귀하거나 또 다른 서버로 이동하게 된다.

네트웍 내부에 있는 인증센터는 에이전트의 인증을 담당하고 있다. 이것은 바이러스와 같은 악성 에이전트의 전송을 사전에 차단하는 역할을 한다.

2.2 Middle agent(MiddleMan)

Mobile agent에서는 클라이언트의 에이전트는 네트웍의 수많은 서버들 중에서 어느 것에 서비스를 요청해야 하는지 어떻게 알 수 있는가가 매우 중요하다. Middle agent가 바로 클라이언트와 서버간의 효율적 연결 기능을 가진 모델이다[5]. 여러 연구에서는 같은 형태의 middle agent를 복수로 두어서 사용할 수 있음을 말하고 있다. 그러나, 본 논문에서는 보다 신속하고 정확한 연결을 위해 MiddleMan이라 부르는 새로운 방식의 middle agent를 다중 형태로 제안하고자 한다.

2.2.1 MiddleMan의 정의

기본적인 MiddleMan은 서버로부터 서비스와 관련된 일련의 정보들을 제공받아서 저장한 후, 해당 서비스를 요청하는 클라이언트의 에이전트가 내부로 들어오게 되면 서버와 클라이언트를 연결해주는 역할을 한다.

MiddleMan의 몸체는 서버로부터의 정보들의 집합체와 서비스를 요청하는 클라이언트로부터의 정보들의 집합체로 이루어진다. 서버에서 제공해야 하는 정보는 서비스의 카테고리와 세부 명칭, 서버의 위치 등의 구체적인 사항들이다. 물론 서버는 클라이언트의 에이전트를 인자로 받아서 수행할 수 있는 자신의 원격 객체를 MiddleMan 내부에 따로 등록해야 한다. 클라이언트는 필요로 하는 서비스의 구체적 사항과 자신의 agent를 등록한다. 이상 설명한 내용이 아래의 표1 에 보다 구체적으로 명시되어 있다.

표 1. MiddleMan 내부 정보의 종류 및 형태

정보종류	제공자	형태
카테고리/이름	서버, 클라이언트	• reservation: hotel/train/..... • purchase: house/book/.....
인자	서버, 클라이언트	• Max/Min time,cost,size, ...
지리적 위치	서버	• Korea: Seoul/Busan/ .....

MiddleMan은 위의 정보들을 자신의 내부에 저장하고 필요한 사항들을 잘 검토하고 비교하여 상호 매칭이 잘 되는 서버와 클라이언트를 최대한 빠른 시간 안에 맺어주는 것이 진정한 목적이다. 대량의 agent가 동시에 서비스를 요청할 경우 신속한 서비스를 보장해 주기 위해 다음절에서 Multi-MiddleMan을 새로이 제안하고 있다.

2.2.2 Multi - MiddleMan

서비스를 요청하는 클라이언트가 무수히 많을 경우에 MiddleMan 한 개로 그 많은 요구를 다 처리하려면 무리가 따른다. 특히 서비스 수행 시간이 지연될 것이다. 이것을 해결하기 위해 본 논문에서는 MiddleMan을 여러 계층으로 분산시켜 활용하도록 하는 Multi- MiddleMan을 제안하며, 이 관계는 그림 2에서 개념적으로 나타내고 있다.

예를 들면, Main_MM에는 각 서비스의 카테고리 정보와 세부명칭, 위치정보를 저장한다. Pos_MM 내부에는 각 서버의 위치별 정보를 내장하고 있다. Main_MM은 클라이언트 에이전트를 매개변수로 위치적으로 적합한 Pos_MM에게 넘겨준다. 최종적으로 Pos_MM은 에이전트를 적합한 서버와 직접 연결하여 준다. 정보의 분류가 간단할 경우에는 굳이 이와 같은 방법을 사용할 필요가 없다. 그러나 보다 고급화되고 복잡한 서비스를 위해서 서버에서 제공하는 정보의 종류와 수가 많을 경우 매우 유용하다.

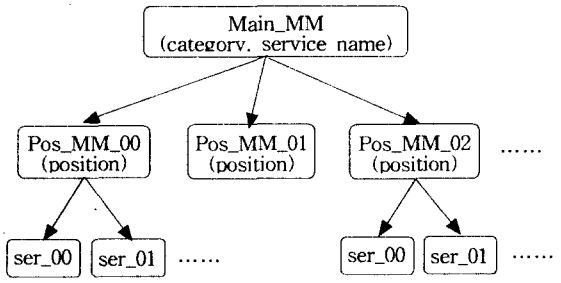

그림 2. Multi-MiddleMan (Multi-MM)의 개념도

3. 가상 시나리오

다음에 소개될 가상 시나리오는 지금까지 설명하였던 mobile agent 시스템의 효용성을 보다 구체적이고 명확하게 증명하고 있다.

음식점 예약 서비스의 경우를 예로 들어보자. 사용자는 아들의 돌잔치를 하기 위해 시내에서 가족들과 같이 식사를 하기를 원하고 있다. 그가 원하는 음식점은 교통이 편리하고 돌잔치 행사를 지원하며 일정한 크기의 방과 한식과 고기 및 주류를 제공하는 곳이다. 그리고 가격은 중간정도를 원하다. 그가 생성하는 agent가 제공하는 정보는 카테고리 및 정보명칭이 {reservation}{hotel | restaurant |.....}이며, 그 외 인자는 {[(Min)cost-medium}],[(Min)size-medium]} 등이 될 것이다. 지리적 위치는 {[Korea][Seoul].....}로 주어질 수 있다.

서버역할을 하는 음식점도 제공할 서비스와 룸의 크기, 서비스별 가격, 지리적 위치, 그 외 사용자에게 유용한 정보 등을 미리 MiddleMan에게 제공해야 한다.

사용자의 에이전트는 만족하는 서비스를 찾을 때까지 한 음식점 내의 여러 DB들을 검색하고, 일정한 수준의 결과를 획득하고 예약을 했을 경우 그 정보를 가지고 출발지로 복귀하며, 그렇지 못했을 경우 또 다른 서버를 찾아서 여행할 것이다. 다른 서버의 정보들은 맨 처음 MiddleMan에서 필요한 서버를 찾을 때 적합한 여러 서버의 위치를 자신의 내부에 가지고 있는 것이 보다 효율적일 것이다. 이전 방문지에서 이 에이전트를 다른 곳으로 전송하는 역할을 한다.

4. 결론

본 논문에서는 Mobile agent 시스템의 기능에 대해 살펴보았으며, middle agent의 필요성 및 본 논문에서 제안하는 Multi-MiddleMan에 관해 간단한 시나리오를 통해서 설명하였다.

모빌 에이전트는 정보분야에서 새로운 패러다임을 제시하고 있다. 한 번 목적지로 향하면 더 이상 사용자의 관리를 받지 않고 독자적으로 업무를 수행한다. 또한 사용자는 네트웍 접속을 끊고 있다가 재접속 했을 때 agent의 수행결과를 전송 받을 수도 있다.

그러나 아직 모빌 에이전트가 활성화되어있지 못하기 때문에 그 효용성과 기능을 충분히 살리기 어렵다. 많은 서버들과 클라이언트들의 협조아래 모빌 에이전트는 자신의 진정한 능력을 발휘할 것이다.

앞으로는 Multi-MiddleMan의 운영방법에 관한 보다 효율적인 알고리즘 개발하여 이를 우리의 모빌 에이전트 시스템에 적용할 예정이다.

5. 참고문헌

[1] Colin G. Harrison, David M. Chess, and Aaron Kershenbaum. Mobile Agents. Are they a good idea? Technical report, IBM Research Division, T.J.Watson Research Center. March 1995.

[2] Alfonso Fuggetta, Gian Pietro Picco, Giovanni Vigna. Understanding Code Mobility. IEEE Transactions on Software Engineering. vol. 24, no. 5, May 1998.

[3] Ann Wollrath, Jim Waldo, Roger Riggs, Java-Centric Distibuted Computing, IEEE Micro, 1997.

[4] Jim Waldo, Remote procedure calls and Java Remote Method Invocation, IEEE, July 1998.

[5] Keith Decker, Katia Sycara, Mike Williamson, Middle-Agents for the Internet, Proc.of the 15th International Joint Conference on Artificial Intelligence(IJCAI-97), Nagoya, Japan, 1997.

[6] Jim White. Mobile Agents White Paper. Copyright 1996 by General Magic.
http://www.yl.is.s.u-tokyo.ac.jp/~masatomo/mobile/White/whitepaper.html

[7] D.B. Lange, Java Aglets Application Programming Interface, IBM Corp. white paper, Feb. 1997.

[8] R.S.Gray, Agent Tcl:A Transportable Agent System. Proc. CIKM Workshop on Intelligent Information Agents, Baltimore,Md., Dec. 1995.