제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

효율적 에이전트 이주를 위한 전역 디렉토리 시스템의 설계 및 구현

조영희*, 강윤희*
*천안대학교 정보통신학부
e-mail:{arden78,yhkang}@cheonan.ac.kr

Design and Implementation of Global Directory System for Efficient Agent Migration

Young-Hee Cho*, Yun-Hee Kang*
*Division of Information & Communiation, Cheonan University

요 약

본 논문에서는 에이전트의 효율적 이주를 제공하기 위한 전역 디렉토리 스킴을 제안한다. 제안된 스킴은 동적으로 잦은 변경을 갖는 분산 시스템에서 에이전트의 서비스 수행을 위한 에이전트 및 환경에 대한 트래킹을 통해 효율적인 이동 에이전트의 이주를 지원하는 것을 목적으로 한다. 본 연구에서는 이를 위해 대표적인 이동에이전트 시스템인 IBM's Aglets Software Development Kit을 확장하여 에이전트 수행 환경에 대한 모니터링 모듈을 구현하고 이를 기반으로 기존의 마스터-슬레이브 형태의 이동 에이전트 시스템을 확장하여 구현하였다.

1. 서론

최근 인터넷의 기하급수적 성장과 웹을 통한 다양한 정보 및 서비스의 증가에 따라 네트워크를 통한 이형의 자원 접근에 대한 요구가 증가하고 있다. 이를 위해 이동 에이전트는 분산 및 이형 시스템을 위한 새로운 프로그래밍 패러다임으로 등장하고 있다.

이동 에이전트는 하나의 시스템으로부터 수행의 제어가 네트워크로 연결된 다른 시스템으로 이동할 수 있으며 이주 후 수행을 재개한다. 에이전트의 이동성은 많은 양의 데이터를 주고받는 대신에 수행을 위한 환경으로 코드를 이주한 후 비동기적으로 수행의 최종 처리 결과를 전달받아 처리한다. 또한 네트워크 부하의 감소 및 분산시스템의 강건성 보장 등 많은 잠재적 장점을 제공한다[1].

본 논문에서는 이동 에이전트를 기반으로 확장 가능한 분산 시스템(scalable distributed system) 환경에서 이동 계산을 위한 확장된 전역 디렉토리 스킴을 제안한다. 확장된 전역 디렉토리 스킴은 동적으로 잦은 변경을 갖는 분산 시스템에서 에이전트의 서비스 수행을 위한 에이전트 및 환경에 대한 트래킹을 통해 효율적인 이동 에이전트의 이주를 지원하는 것을 목적으로 한다.

본 연구에서는 이를 위해 대표적인 이동 에이전트 시스템인 IBM's Aglets Software Development Kit[2]을 확장하여 에이전트 수행 환경에 대한 모니터링 모듈을 구현하고 이를 기반으로 기존의 마스터-슬레이브 형태의 이동 에이전트 시스템을 확장하여 구현하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 이동 에*이전트의 관련 연구를 기술하며 제 3장에서는 제안된 형*상 관리 스킴을 위한 시스템 아키텍츄어를 보이고 제 4장에서는 이동 에이전트의 위치 기반 스킴의 구현을 보이며 결론 및 향후연구를 설명한다.

2. 관련 연구

본 절에서는 대표적인 이동 에이전트 시스템Concordia[3], Odyssey[4], Voyager[5] Telescript[6],에 대해 간단히 살펴보고 MASIF의 디렉토리 관련 API를 기술한다.

2.1 이동 에이전트 시스템

Concordia는 MEITCA에서 1997년에 개발을 시작하였고 자바 모바일 에이전트의 개발과 실행을 위한 프레임워크이다. 구성요소로는 에이전트 실행과 전송을 담당하는 서버와 원격 서버 관리를 위한 관리자가 있다. Concordia는 다른 노드들로 이동하면서 정보 에이전트와 상호작용하며 응용 프로그램 내의 에이전트 그룹을 하나 이상 형성한다. 그룹에 속한 서로 다른 전문 에이전트들이 태스크를 동시에 수행 후 협동을 통해 얻어진 산출물들을 기반으로 하여 결과를 서로 연관시킨다[3].

Odyssey는 General Magic사의 상용 제품으로써, 자사의 에이전트 개발언어인 Telescript의 기술을 자바에 접목하여 개발한 이동 에이전트 시스템이다. Odyssey는 Odyssey클래스 라이브러리를 제공하고, 사용자는 이를 이용하여 자신의 이동 에이전트를 개발할 수 있도록 지원하고 있다[4].

Voyager는 agent-enhanced 분산 컴퓨팅을 위한 자바로된 플랫폼이다. Voyager는 클래스가 생성되면 네트웍상의 어디에서나 쉽게 객체를 생성할 수 있고, 자바 메시지를 보낼 수 있다. 객체가 이동하면 새로운 위치로 이동된 객체에게 메시지를 전송하고 전의 위치에 secretary를 남겨둔다[5].

Telescript는 Telescript엔진에 의해 실행되는 인터프리터 언어이며, Telescript Development Environment라는 개발 환경과 함께 제공된다. Telescript 언어는 동적이며 영속적이고 이식가능하며 안전하다는 특징을 갖는다[6].

2.2 MASIF 디렉토리 관련 API

이동 에이전트의 표준화 그룹인 OMG의 MASIF(Mobile Agent System Interoperability Facility.)는 모바일 에이전트시스템을 위한 상호 운영이 가능한 인터페이스를 제공하는 정의와 인터페이스들을 정의한다.
또한 MASIFFinder 인터페이스는 에이전트의 에이전트, 장소, 에이전트 시스템을 등록, 해제, 그리고 위치시키기(locate) 위한 운영들을 정의하며 데이터베이스의 인터페이스로써 작동한다[7].

MASIFFinder은 클라이언트가 MASIFFinder에게 객체를 찾아달라고 요청하기 전에, 클라이언트는 MASIFFinder에 대한 객체 참조를 얻어야만 한다. 객체 참조를 얻기 위해, 클라이언트는 코바 네이밍 서비스 또는 에이전트 시스템에 의해 제공되는 내부적 인터페이스를 사용할 수 있다.

MASIF에서 정의한 MASIFFinder을 위한 API를 살펴보면 다음과 같다.

o lookup_agent() : 파라미터에 명시한 에이전트의 위치를 반환한다. 이 메서드는 이름 또는 특정 에이전트 프로 파일로 에이전트를 검색할 수 있다.

o lookup_place() : MASIFFinder에 등록된 플레이스의 위치를 구할 수 있다.

o. lookup_agent_system() : MASIFFinder에 등록된 에이전트 시스템의 위치를 찾을 수 있다.

o register_agent() : MASIFFinder에 등록된 에이전트의 리스트에 에이전트를 추가시키는 메서드이다. MASIFFinder 내의 이미 존재하는 에이전트 이름으로 호출된다면 이 오퍼레이션은 관련 정보를 가장 최근에 호출된 정보와 대치시킨다.

o unregister_place() : MASIFFinder에 등록된 에이전트의 리스트에서 플레이스를 등록 해제시키는 메서드이다.

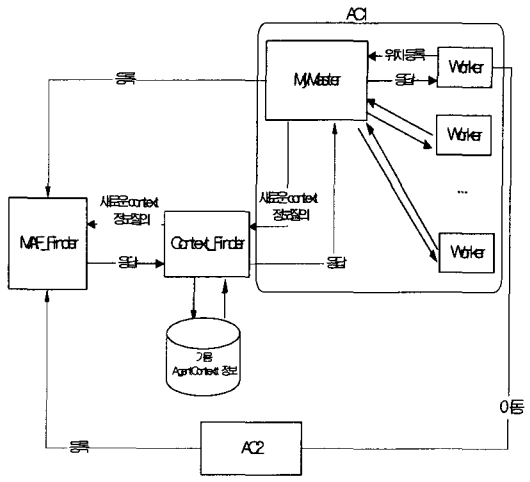
o unregister_agent_system() : MASIFFinder에 등록된 에이전트의 리스트에서 에이전트 시스템을 등록 해제시키는 메서드이다.

3. 제안된 이동 에이전트 디렉토리 시스템

수행중인 에이전트의 실행을 위해 에이전트의 이동을 위한 장소를 결정하는 것은 에이전트의 작업 수행의 성능에 중요한 요소이다. 본 절에서는 에이전트의 적응형 이주 계획을 지원하기 위한 전역 디렉토리시스템 아키텍쳐어 및 적응형 에이전트 여행경로 결정을 기술한다.

3.1 확장된 에이전트-장소 모델

제안된 디렉토리 시스템은 에이전트의 수행 모델인 에이전트-장소 모델을 기반으로 확장한다. [그림 1]은 전역 디렉토리 시스템의 아키텍쳐어를 보인 것이고 [표 1]은 주요 구성요소의 기능을 기술한 것이다.



[그림 1] 전역 디렉토리 아키텍쳐어

[그림 1]에서 에이전트는 에이전트 수행 환경인 AgentContext에 위치하며 현재의 AgentContext에서 다른 AgentContext로 이동하며 수행한다. MasterAgent MyMaster는 AgentContext인 AC1가 생성된 후 AC1에 배치되며 MAF_Finder에 등록되고 실제로 이동하여 작업을 수행할 Worker를 생성한다.

MyMster에 의해 생성된 Worker는 AC1에게 이동하여 작업할 Context의 정보를 질의하게 된다. Worker의 질의를 받은 MasterAgent는 Context_Finder에게 새로운 Context정보를 질의하고 응답을 받게 된다. 이 때 Context_Finder가 유지하고 있는 정보가 특정 시간 동안에 유효하다면 AC1에게 응답하고 그렇지 못하다면 MAF_Finder에게 질의하여 새로운 정보를 가져오게 된다. 또한 Context_Finder가 유지하고 있는 정보가 특정시간이 지났다하더라도 질의가 없다면 Context_Finder는 MAF_Finder에게 질의하지 않는다.

만일 AC1이 아닌 AC2가 등록하여 새로운 AgentContext에 대한 정보가 있다면 MAF_Finder는 Context_Finder의 질의에 응답하고 Context_Finder는 MasterAgent에게 응답하여 결국 Worker가 AgentContext인 AC2로 이동하게 된다.

[표 1] 제안된 아키텍츄어의 주요 구성요소 기능

구성요소	기능
ContextFinder	context들의 정확한 위치 정보를 가지며, Supervisor에게 context 정보를 알려준다. 또한 이동 후 worker의 존재 여부를 점검한다.
Supervisor	ContextFinder로부터 수집한 context 정보를 기반으로 다음 이동 위치를 결정하여 Worker에게 전달하고 작업을 지시한다.
Worker	실제로 context를 이동하면서 작업을 수행한다.

3.2 적응형 에이전트 여행 경로 결정

에이전트는 주어진 타스크 수행을 위해 에이전트 실행 환경을 이동하여 수행하며, 수행의 경로의 결정이 초기에 결정된 후 변경이 불가능한 경우 정적인 여행 경로를 갖는다. 그리고 여행 경로의 변경이 수행과정에서 변경 가능한 경우 동적인 여행 경로를 갖는다.

본 연구에서 제안한 적응형 에이전트 여행 경로 결정은 동적인 여행 경로 결정 기법으로 에이전트의 여행 경로를 결정하기 위해 Worker 에이전트는 ContextFinder 및 Supervisor와의 상호작용을 수행한다.

본 아키텍츄어에서 타스크의 수행은 이동 에이전트의 여행을 통해 이루어지며, 환경 변화에 따른 반응(reaction)과 자신의 지식에 따른 능동수행(proaction)에 의한 여행 경로(itinerary)의 변경이 수행될 수 있으며, 또한 환경 변화에 따른 동적인 반응만을 고려한다.

[표 1]에서의 Supervisor는 Worker에 대한 launch를 수행하고 수행된 결과를 수집하는 작업을 수행한다. 또한 현재 수행 중인 에이전트의 위치에 대한 추적을 수행하며 ContextFinder로부터 수집된 네트워크 및 노드의 상태를 기반으로 다음 이동 위치를 결정하여 Supervisor을 통해 Worker에게 전달한다.

ContextFinder는 현재 노드에서 다른 노드로 에이전트의 안전한 이동을 보장하며, 이동 후 에이전트에 대한 위치 정보를 Supervisor에게 알리고 worker의 존재 여부를 점검한다.

Worker는 자신의 주어진 작업을 노드로 이동 후 수행하며 주기적으로 자신의 존재 여부를 Supervior에게 알린다. 또한 현 노드에서 수행을 완료한 후 다음 노드로의 여행을 진행하게 된다. 이 과정에서 Worker는 Supervisor에 의해 작업이 시작되어 이주를 수행하는 모든 에이전트를 Supervisor로 전달되어야 한다.

여행 계획(Itinerary plan)의 결정은 서비스의 가용성, 접근 비용을 고려하여 Supervisor에 의해 결정된다.

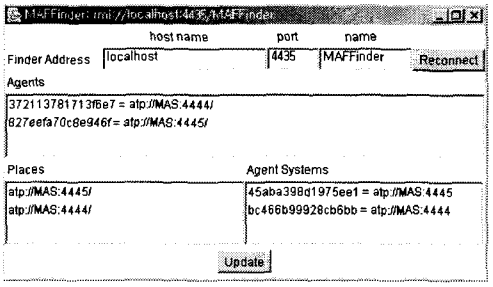
제안 시스템의 장점은 새로이 등록된 context 정보를 그때그때 제공받아 사용할 수 있어 MasterAgent가 따로 데이터를 저장하고 있을 필요가 없고 WorkerAgent의 위치를 쉽게 찾아볼 수 있다.

3.3 ContextViewer

ContextViewer는 Worker 에이전트의 이동을 결정하기 위해 MasterAgent가 필요로 하는 context 정보를 제공하기 위한 GUI 프로그램으로 자바 AWT를 사용하여 개발하였다. [그림 2]는 ContextViewer를 보인 것이다.

ContextViewer에서는 MAFFinder의 localhost의 port를 FinderAddress에서 보여주며, 이동 에이전트인 aglets의 위치와 AgentContext명인 Place와 AgentSystem의 식별자에 대한 정보를 제공한다.

MasterAgent는 ContextViewer를 통해 이동할 context들의 정보를 얻어올 수 있다.



[그림 2] ContextViewer

4. 전역 디렉토리 시스템 구현

[그림 3]는 전역 디렉토리 시스템을 기반으로 한 확장된 마스터-슬래이브 패턴을 확장한 디렉토리 운영을 시퀀스 다이어그램으로 보인 것이다.

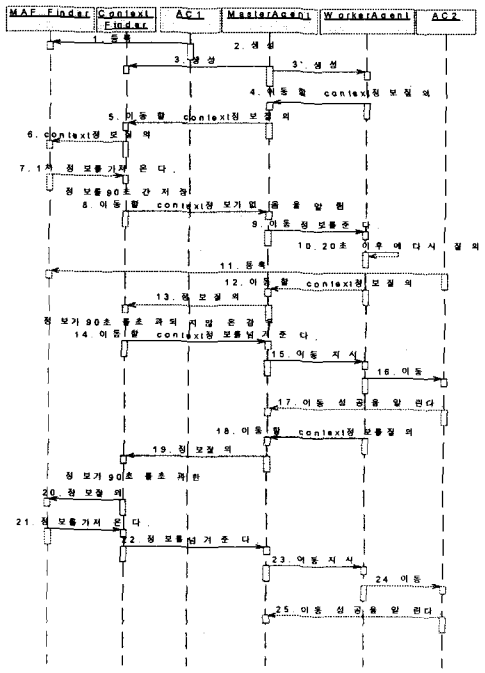
AC1이 MAF_Finder에 등록 후 MyMaster가 생성되고 MyMaster는 Worker와 Context_Finder을 생성한다.

MAF_Finder은 context들의 위치를 데이터베이스로 갖는다.

Woker의 이동정보 요구에 따라 MyMaster는 Context_Finder에게 이동할 context 정보를 질의한다.

Context_Finder는 처음 context 정보를 요구받았을 때 MAF_Finder로부터 context 정보를 가져온 후 90초간 그 정보를 유지한다. Context_Finder는 90초 동안 모든 질의에 대해 MAF_Finder에게 정보를 요구하지 않고 응답한다. 만일 다음 질의가 90초를 초과한 다음에 있다면 MAF_Finder에게서 새로운 정보를 가져다가 응답한다. Context_Finder는 질의가 있을 때만 MAF_Finder에게 새로운 정보를 질의하게 된다.

Worker는 실제로 이동하여 작업을 수행하는 Agent이며 MyMaster에게 이동 가능한 context 정보를 질의하여 이동하며, 이동이 끝나면 이동이 성공적임을 MyMaster에게 알린다.



[그림 3]전역 디렉토리 운영을 위한 시퀀스 다이어그램

5. 결론

본 논문에서는 이동 에이전트의 효율적인 이주를 지원하기 위한 전역 디렉토리 시스템을 설계한 후 구현하였다. 이를 위해 Aglets SDK를 사용하여 전역 디렉토리의 구성요소인 MAF_Finder와 통신을 수행하여 최신의 에이전트 및 에이전트 컨텍스트 정보를 유지하는 Context Finder를 개발하였으며, 기존의 정적인 여행경로를 갖는 마스터-슬래이브의 에이전트 프로그램을 확장하여 적응형 여행경로를 지원할 수 있도록 구성하였다.

향후 네트워크의 상태 정보를 적응형 여행 경로의 설정에 적용하기 위해 NWS(Network Weather Service)를 적용하여 Context_Finder의 경로 설정 과정에서 네트워크의 대역폭과 지연을 반영하여 확장할 예정이다.

참고문헌

[1] D.B. Lange and M.Oshima, "Seven Good Reasons for Mobile Agents," Communications of the ACM, vol. 42, no.3, pp.88-89, 1999.

[2] D.B. Lange and M.Oshima, "Programming Mobile Agents in Java-with the Java Aglet API," 1997.

[3] D. Wong et al., "Concordia: An Infrastructure for *collaborating Mobile Agents," First Int'l Workshop on* Mobile Agents(MA). Springer Lecture Notes in Computer Science 1219, Berlin, Apr. 1997

[4] General Magic, Odyssey - Mobile Java Agents Located at: http://www.genmagic.com.agents/

[5] ObjectSpace, in "Voyager Technical Overview," 1998

[6] J. White. Telescript Technology: Mobile Agents, 1996.

[7] Florin Muscutariu and Marie-Pierre Gervais. Modeling an OMG-MASIF Compliant Mobile Agent Platform with the RM-ODP Engineering Language. In Proceedings of the 2nd International Workshop on Mobile Agents for Telecommunication Applications, volume 1931 of Lecture Notes in Computer Science, Paris, France, September 2000. Springer Verlag.