Proceedings of KFIS Autumn Conference 2006 Vol. 16, No. 2.

웹서비스 기반의 개방형 다중 에이전트 시스템 구조

An Open Multiagent System Architecture based on Web Service Infrastructure

황경순, 이승희, 양원섭, 이건명

충북대학교 전기전자컴퓨터공학부, 첨단정보기술연구센터(AITrc)
hks@aicore.cbnu.ac.kr, shlee@aicore.cbnu.ac.kr, soryu@aicore.cbnu.ac.kr, kmlee@cbnu.ac.kr

요 약

분산 환경에서의 자원 및 서비스를 활용하여 응용 시스템을 구축하기 위한 패러다임으로서 다중 에이전트 구조가 많아 연구되어 왔다. 이 논문에서는 다양한 자원 및 서비스를 개방적으로 활용하기 위해, 웹서비스의 개방적인 환경을 활용하여 에이전트 및 웹서비스 객체를 효과적으로 이용할 수 있도록 하는 개방형 다중 에이전트 시스템 구조에 대해서 소개한다.

Key Words : 웹서비스, 다중 에이전트, UDDI, WSDL, SOAP

1. 서 론

다중 에이전트 기반의 시스템 개발 기법은 중요한 방법론으로서 자리잡아가고 있으며, FIPA등의 표준화가 있지만 아직 다른 플랫폼에서 개발된 에이전트간의 통신이나 이용은 원활하지 못한 상태이다. 한편, 다중 에이전트 시스템에서는 각 에이전트들에 대한 기능을 등록하고, 검색할 수 있는 디렉토리 서비스 에이전트 등 미들-에이전트들을 가지고 있지만[1], 이를 효과적으로 다중 에이전트 시스템 외부에 공개하는 메커니즘을 갖추고 있지 않다.

최근 웹서비스는 인터넷상에 분산된 객체를 조합하여 적은 비용으로 원하는 서비스를 제공하는 기술로서 많은 기반 기술과 표준화 및 제품들이 개발되고 있다. 웹서비스의 기반 기술로서 객체에 접근하기 위한 SOAP 기반 메시지 전송 기술, 객체를 등록하여 공개, 탐색, 그리고 통합을 위한 저장소에 관련한 UDDI, 서로 다른 클라이언트가 웹서비스를 통해 상호작용할 수 있도록 WSDL 등이 개발되어왔다[1,3].

서비스를 위한 분산된 객체를 이용하여, 이들의 공동작업 또는 활용을 통해서 서비스를 제공한다는 측면에서, 다중 에이전트 기술과 웹서비스 기술은 공통적이다. 다중 에이전트 시스템의 에이전트들과 웹서비스 객체를 결합하기 위한 여러 시도가 있어왔다[1,3,4,5]. 이 논문에서는 에이전트에 대한 정보를 웹서비스의 UDDI서버에 등록하고, 이를 통해서 적합한 에이전트에 대한 정보를 찾고, 또한 필요에 따라 에이전트가 웹서비스 객체 및 에이전트를 사용하여 작업을 처리할 수 있는 구조에 대해서 소개한다.

본 연구는 첨단정보기술연구센터(AITrc)를 통해서 과학재단 지원으로 수행된 것임.

2. 관련연구

웹서비스는 웹을 통해서 등록, 탐색, 그리고 공개될 수 있는 독립적인 기능의 모듈로서 분산된 서비스 객체를 활용하여 느슨한 결합(loosely coupled) 형태의 응용 서비스를 개발할 수 있도록 한다[2]. 웹서비스는 서비스 제공자(provider)의 정보를 공개하기 위한 저장소에 대한 표준인 UDDI가 있으며, 이를 통해서 서비스 사용자가 응용시스템을 개발할 때 필요한 서비스를 선택할 수 있고, 해당 서비스 객체 호출을 위한 정보를 습득할 수 있다. 서비스 객체에 대한 정보는 WSDL을 사용하여 전송되는 메시지와 메시지 전송에 관련된 연산에 대해서 정의한다. 웹서비스는 광범위하게 사용되는 웹상에서 구현되기 때문에 다중 에이전트 시스템 구현 관점에서도 많은 관심을 가져왔다. 다중 에이전트 시스템 관점에서는 웹서비스 객체를 다중 에이전트 시스템이 작업을 처리 하는 것을 활용하는데 관심을 가져왔다

[3,4,5]. 이를 위해서 에이전트가 직접 UDDI를 통해 필요한 웹서비스 객체를 직접 찾아 호출하거나, 에이전트 개발자가 미리 웹서비스 객체를 선택하여 호출하도록 코딩할 수 있다. 다른 방법으로는 웹서비스 객체에 대응하는 랩퍼(wrapper) 에이전트를 만들고, 이를 다중 에이전트 시스템의 디렉토리 시스템에 등록한다. 특정 웹서비스 객체를 활용한 작업이 필요한 경우, 이에 대응하는 랩퍼 에이전트를 호출하게 되고, 이 에이전트는 해당 웹서비스 객체를 호출하여 결과를 획득한 다음, 이를 반환하는 방법으로 다중 에이전트 프레임워크 안에서 작업을 수행한다. 이러한 랩퍼 에이전트를 자동으로 생성하는 대표적인 예로서 WS2JADE는 웹서비스 객체에 대응하는 JADE 환경의 랩퍼 에이전트를 생성한다[7].

한편, 웹서비스 관점에서 다중 에이전트 시스템을 통합하려면 다중 에이전트 시스템의 디렉토리 정보를 웹서비스의 UDDI 저장소로 변환하여 저장하도록 한다. 에이전트를 통해서 제공되는 서비스를 웹서비스 객체에서 사용하고자 하는 경우에는 중간의 중개자(mediator) 역할을 하는 서비스를 통해서 다중 에이전트에게 서비스를 요청하고 결과를 받아서 전해 주는 방식에 대한 연구가 수행되어 왔다[1].

다중 에이전트 시스템 관점에서 웹 서비스 객체를 통합하는 방법론이나 웹서비스 응용 관점에서 다중 에이전트를 통합하는 방법론은 각기 관점별로 상대되는 기법을 부가적으로 사용하기 위한 접근방법들이다. 따라서 각 접근방법에서 사용하는 표준에 따라 대응하는 변환기법이 개발되어야 한다. 웹 서비스의 경우에는 표준화가 성공적으로 진행되어 실제 현장에서 이를 활용한 다양한 응용 시스템이 개발되고 있다. 그러나 다중 에이전트 시스템의 경우에는 FIPA 표준을 따르는 JADE 등과 같은 프레임워크가 있기는 하지만 아직은 웹서비스의 SOAP, WSDL, UDDI 등의 표준과 같은 위상을 갖는 것은 없다.

3. 웹서비스 기반의 개방형 다중 에이전트 구조

제안한 구조에서는 독립적으로 동작하고 있는 에이전트들과 웹서비스 객체를 에이전트가 활용하여 작업을 진행할 수 있도록 하는데 목적이 있다[3]. 다중 에이전트 시스템에서 다중 에이전트의 공동 작업을 지원하는 주요 구성요소는 에이전트의 역할 정보를 관리하고 질의에 응답하는 디렉토리 서비스, 에이전트 생성 및 소멸을 관리하는 에이전트 관리 시스템 등이 있다. 제안한 다중 에이전트 시스템 구조에서는 개방성을 확보하기 위한 방안으로, 웹서비스 환경에서 에이전트를 운영하는 방법을 제안하였다.

우선 다중 에이전트 시스템에서의 디렉토리 서비스를 웹서비스의 UDDI를 사용하여 제공함으로써, UDDI 표준을 만족하는 형태로 에이전트의 정보를 등록하는 방법으로 웹상에서 동작하는 서비스 등록(Service Register) 에이전트가 자신을 등록하고, 다른 에이전트에서 자신의 존재와 역할을 확인할 수 있도록 한다. 에이전트의 생성과 소멸에 대해서는 별도의 관리 서비스를 구현하지 않고, 각각의 에이전트가 자신의 관리자나 다른 시스템에 의해서 구동되고, 자신의 로직에 따라 종료하거나 강제적으로 종료될 수 있다고 전제한다. 즉, 각 에이전트별로 구동과 종료가 별도의 관리자 또는 시스템에 의해서 관리된다고 가정한다. 에이전트간의 통신은 SOAP 표준을 따라 메시지를 사용하여 수행된다.

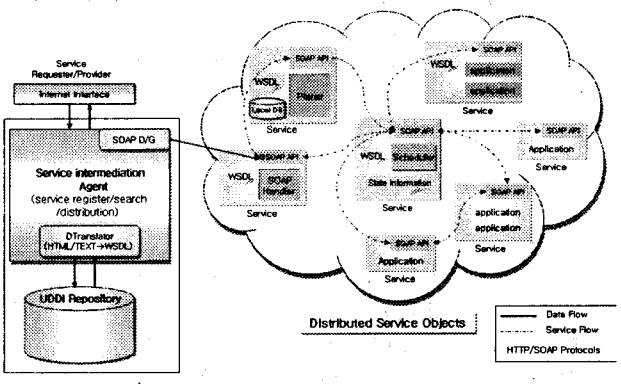

그림 1. 웹서비스 기반의 개방형 다중 에이전트 시스템 구조

[그림 1]은 웹서비스 기반의 개방형 다중 에이전트 시스템에 대한 전체적인 구성도이다. 우선 서비스 요구자(service requester)는 이용하고자하는 서비스를 등록 또는 조회하기 위해 서비스 중개자 에이전트(service intermediator agent)의 도움을 받아 원하는 정보를 획득한다. 서비스 중개자 에이전트는 웹서비스와 에이전트기반의 서비스 그리고 자신에 대한 정보를 UDDI로 운영하게 된다. 즉, UDDI의 정보를 등록, 조회, 배포에 관한 서비스를 하게 된다. 이때 서비스는 UDDI XML (WSDL)문서를 포함한 SOAP 메시지를 생성하여 SOAP 처리 에이전트에게 보내면 시작된다. UDDI의 등록 정보는 식별자, 구분(에이전트/애플리케이션), 이름, 기술, 접속정보, 그리고 정책 등이 있다. SOAP 처리 에이전트(SOAP handler agent)는 서비스 중계자 에이전트에 의해 생성된 XML형식의 SOAP 메시지를 구문분석하고 어떤 서비스 인스턴스를 호출할지를 결정하게 된다. 계획 에이전트(planer agent)는 SOAP 처리 에이전트에서 생성된 서비스 인스턴스 리

스트들의 작업 순서를 계획하고 이를 스케줄러 에이전트(scheduler agent)에 전달하게 된다. 단, 단일 작업에 대해서는 계획 에이전트를 걸치지 않고 스케줄러 에이전트가 수행된다. 스케줄러 에이전트는 적당한 작업 흐름과 일치하는 서비스들을 인스턴스화하고 각각의 서비스 객체들에 상태 정보 및 결과를 저장 관리한다. 이때 에이전트 간에 통신은 SOAP API모듈로 이루어지며, 반드시 분산된 서비스 객체는 네트워크 접속이 용이한 플랫폼 상에 존재해야 한다.

3.1 웹서비스를 이용한 에이전트

제안한 다중 에이전트 구조에서는 에이전트는 독립적으로 운영되는 것뿐만 아니라, 웹서비스에 의해서 구현된 것도 허용한다. 기본적으로 에이전트는 자신의 상태정보를 가지면서 자율적으로 자신의 일을 수행할 수 있는 것을 말한다. 그런데 에이전트를 웹서비스 개체로 구현하는 경우, 웹서비스는 상태가 없는(stateless) 프로그램 모듈이기 때문에 에이전트 구현에 어려움이 있다. 따라서 제안한 방법에서는 이를 위해 에이전트를 구현하는 웹서비스 객체는 데이터베이스 내에 자신의 상태와 관련된 데이터를 저장하도록 하고, 해당 에이전트에 대응하는 웹서비스 객체가 호출될 때는, 이에 대응하는 데이터베이스에 접근하여 상태값을 읽어오고, 해당 서비스 개체가 종료할 때는 현재 상태값을 데이터베이스에 저장하는 방법을 사용한다.

제한된 시스템에서는 SOAP 처리 에이전트가 서비스 중개 에이전트로부터 받은 SOAP 메시지 문서를 파싱(parsing)해서 문서의 내용을 추출한다. 추출한 내용은 구문분석을 통해 문서에 포함된 원소와 속성 값을 파악한다. 따라서 구문분석 후에는 다음 서비스의 인스턴스가 무엇인지와 입력에 대한 파라미터가 어떤 형식인지 결정한다. 이때, 에이전트 간에 메시지를 송·수신하기 위하여 SOAP API 모듈을 사용하며, 만약 서비스 인스턴스가 단일 작업을 수행하는 것이라면 스케줄러 에이전트가 바로 서비스를 인스턴스화하여 수행한 후, 그 결과를 서비스 요구자에게 전달해 주고, 작업을 종료한다. 만약 단일 작업이 아닌 경우, 계획 에이전트에서 사용자에 의해 미리 정의된 워크플로우로(local db)부터 일치하는 작업들을 바인딩한다. 그리고 스케줄러 에이전트에서 워크플로우에 해당되는 서비스들을 인스턴스화하고 그 객체들의 상태정보를 기록하고 관리한다. 일련의 작업이 끝나면 사용자에게 그 결과를 전달한다.

3.2 UDDI에서 에이전트 정보 기술

에이전트 정보를 UDDI에 등록한다고 할 때, 다중 에이전트 시스템에서는 특정 작업을 의뢰할 에이전트를 찾아야 하는데(contract-net이나 auction 등의 프로토콜을 사용하여), 이를 위해서는 에이전트의 역할(role) 정보를 UDDI에 저장해 두어야 한다. 그런데, UDDI에는 이러한 역할 정보를 표현할 정의된 필드가 없기 때문에, 제안한 방법에서는 UDDI의 description 필드에 role에 대한 태그와 함께 이에 대한 역할 정보를 기술하도록 하는 방법을 사용하고 있다. UDDI에 등록된 모듈이 에이전트인 경우에는 해당 에이전트를 접근하기 위한 접근정보가 포함되고, 해당 모듈이 에이전트인지 아닌지의 정보가 기록된다.

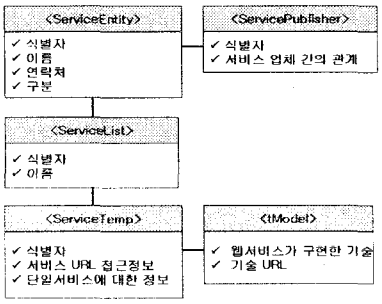

그림2. UDDI 데이터 구조

[그림 2]는 UDDI 데이터 구조를 보여주고 있다. 이 스펙은 SOAP 메시지에 포함된 XML 구조의 구체적인 정보를 담고 있다. 즉, 서비스 개체(service entity)에는 식별자, 이름, 연락처, 그리고 서비스 구분(에이전트/애플리케이션)등의 정보를 포함하고, 서비스 리스트(service list)에는 제공되는 하나 이상의 일련의 서비스들의 목록 정보를, 서비스 탬프(service temp)에는 서비스에 대한 접근점 URL(조회 URL, URL)과 tModel대한 조회 정보를, 그리고 tModel에서는 특정 스펙 또는 웹서비스가 따르는 행위나 에이전트의 역할에 대한 정보를 포함한다.

3.3 에이전트 검색을 위한 방법

웹 영역으로 다중 에이전트 시스템에서 사용할 수 있는 서비스가 개발되기 때문에, 자신의 작업에 최적인 서비스를 검색하는 것은 수작업으로 처리하는 것은 무리가 따른다. 다중 에이전트 시스템에서는 필요한 작업을 대행할 에이전트를 찾는 것은 일반적인 요구사항이다. 그런데 웹서비스 환경에서는 서비스 및 에이전트의 등록에 대한 충분한 검증이 수반되지 않기 때문에, 적합한 웹서비스나 에이전트를 선정하는 것에 문제가 있을 수 있다. 따라서 자동으

로 웹서비스나 에이전트를 선정하는 것을 지원하기 위해서, 제안한 방법에서는 요청 작업에 대한 입력 및 출력 또는 이들의 패턴에 대한 예제를 작업요청 에이전트에 제공하여 이를 이용하여 작업 에이전트를 선정하도록 하는 방법을 채택하고 있다. UDDI를 통해서 작업을 수행할 수 있는 에이전트 또는 웹서비스 객체를 등록된 description 필드에 기술된 정보를 토대로 일차적으로 후보를 선발한다. 다른 입출력 예제를 사용하여 후보 에이전트에 입력을 제공하고 그에 대한 출력을 관찰하는 방법으로서 에이전트 타당성을 평가한다. 한편 정확한 수치적인 결과를 기대하기 어려운 경우에는, 출력의 형태를 수치적이지 않는 방법을 개발하여 적합성을 판단한다.

3.4 웹서비스 객체로 구현된 에이전트의 상호작용 프로토콜

에이전트 간에는 단순 one-way broadcast, notification 뿐만 아니라 request-response, solicit-response 등과 같은 상호작용을 해야 한다[2]. 특히 contract-net 나 auction 프로토콜 등은 더 복잡한 다중 에이전트 프로토콜이다. 일반 에이전트나 웹서비스로 구현된 에이전트나 모두 웹서비스 통신프로토콜에서 사용되는 SOAP으로 코딩되어 전달된다. 이들 에이전트는 진행 중인 프로토콜의 진행상태를 파악하여야 한다. 한편, 웹서비스로 구현된 에이전트인 경우에는 해당 에이전트가 진행하고 있는 상호작용에 대한 정보를 저장하고, 에이전트 자체가 상호작용 상태를 파악하여 이에 대한 적합한 조치를 취한다. 각 상호작용에는 고유 식별자를 부여하여, 에이전트 간에 상호작용을 허용한다.

한편, 에이전트 간에 정보를 공유하기 위하여 상태 정보를 생성하고 저장 관리해야 한다. 따라서 상태정보에 대한 테이블 필드에는 작업이름, 작업순서, 발신자, 수신자, 제어필드 등을 포함한다. 제어필드에는 통신의 연결 및 종료 그리고 데이터전송 모드를 기술한다. 데이터 전송 모드에는 다른 에이전트에게 메시지를 보내는 send message, 다른 에이전트로부터 메시지를 기다리는 receive message, 메시지를 보내고 확인응답을 보내는 request-respond, 메시지를 보내고 응답을 기다는 solicit-respond, 그리고 에러를 알리는 notify 가 있다. 스케줄러 에이전트에 인스턴스화된 모든 서비스 에이전트들 간에 통신이 종료되면 상태 정보 테이블도 삭제된다.

4. 결 론

본 논문에서는 개방형 다중 에이전트 시스템의 구조인 웹서비스의 프레임워크 안에서 다중 에이전트 시스템을 구현하는 방법을 제안하였다. 즉, 웹서비스를 이용하는 에이전트들의 작업 수행 과정, UDDI에서 에이전트 정보 기술 스펙, 에이전트 검색방법, 그리고 웹서비스 객체로 구현된 에이전트 간에 상호작용을 하기 위한 프로토콜에 대하여 설명하였다.

본 논문의 향후과제로는 에이전트들의 기능을 좀 더 지능적으로 향상시키기 위하여 시맨틱 웹서비스와 온톨로지 및 추론 기능을 추가한 연구가 필요하다.

참 고 문 헌

[1] K. Sycara, M. Paolucci, J. SOudry. N. Srinivasan, Dynamic Discovery and Coordination of Agent-based Semantic Web Service, IEEE Internet Computing, pp.66-73, 2004.

[2] Tyler Jewell, David Chappll, "Java web services", O'REILLY, 2002

[3] B. Srivastava, J. Koehler, Web Service Composition Current Solutions and Open Problems, Proc. of the 1st Int'l Conf. on Automated Planning and Scheduling (ICAPS 2003) Workshop on Planning for Web Services. Trento: AAAI Press, 2003. 28-35.

[4] R. Rao, X. Su, A Survey of Automated Web Service Composition Method, WAIM 2004, Dalian, China, July 15-17, 2004. LNCS 3129, Springer-Verlag.

[5] D. McDermott. Estimated-regression planning for interactions with web services. In Proc. Of the 6th Int. Conf. on AI Planning and Scheduling, AAAI Press, 2002.

[6] S. MxIlraith, T. C. Son, H. Zeng. Semantic Web Service, IEEE Intelligent Systems, Vo..16, No.2, 2001.

[7] Thang Xuan Nguyen, Ryszard Kowalczyk. WS2JADE: Integrating Web Service with Jade Agents, Faculty of Information and Communication Technologies Centre for Intelligent Agents and Multi-Agent Systems, 2005