2000 년 한국정보처리학회 춘계 학술발표논문집 제 7 권 제 1 호

워크플로우를 위한 이동 에이전트 시스템의 통신 기반구조 설계

이효경*, 송상범, 유정준, 이동익
광주과학기술원 정보통신공학과
e-mail : {pharos, sbsong, jjyoo, dilee}@csrl.kjist.ac.kr

Design of Inter-Agent Communication Infrastructure for Agent-based Workflow System

Hyo-Kyoung Lee*, Sang-Bum Song, Jeong-Joon Yoo, Dong-Ik Lee
Dept. of Information and Communications
Kwang-Ju Institute of Science and Technology (K-JIST)

요 약

이동 에이전트 기반 워크플로우 시스템에서 워크플로우는 프록시 에이전트라는 이동 에이전트에 의해 수행되며 워크플로우를 구성하는 각 작업들은 서브 에이전트라는 여러 이동 에이전트들에게 위임된다. 워크플로우의 수행을 위해서는 프록시 에이전트와 서브 에이전트간 상호협력이 필요하며 이를 위해 에이전트간 통신 기반구조가 요구된다. 본 논문에서는 에이전트간 통신 방법으로서 동기적 통신과 비동기적 통신을 위한 기반구조를 설계한다.

1. 서론

기업의 규모가 점점 커짐에 따라 처리해야 할 업무가 다양, 복잡해지며 업무 내용에 잦은 변화가 생기게 되어 업무처리의 자동화가 불가피하게 되었다. 워크플로우 시스템이란 이러한 기업내의 다양한 업무 흐름을 정의하고 자동화하는 시스템이다. 오늘날 컴퓨터와 네트워크의 빠른 확산으로 인해 워크플로우 시스템의 효율적이고 사용자 위주의 편리성을 제공하는 자동화 시스템이 가능하게 되었다.

이동 에이전트란 네트워크 상에서 호스트 사이를 옮겨 다니며 자율적으로 동작하는 소프트웨어 프로그램을 말한다[1][2]. 이동 에이전트의 가장 큰 장점인 이동성 때문에 에이전트가 작업을 수행하는 동안 사용자와 지속적인 연결이 필요 없게 되므로 클라이언트와 서버간의 빈번한 정보 교환으로 인한 트래픽을 감소시킬 수 있고, 여러 클라이언트가 동시에 한 호스트에 정보를 요구함으로써 발생하는 서버의 부담을 줄일 수 있게 되었다. 그리고 무엇보다 불안한 네트워크 환경에서 이동 에이전트 시스템을 적용함으로써 불안정한 네트워크로 인한 작업처리 지연을 감소시킬 수 있다.

이동 에이전트 개념을 워크플로우 시스템에 적용함으로써 워크플로우 엔진의 부하를 관찰하여 작업부하분산을 구현하여 엔진의 부하를 분산시킬 수 있다. 그리고 동시 수행이 가능한 작업의 경우는 여러 에이전트들에게 동시에 수행하도록 함으로써 작업 처리시간을 줄일 수 있으며, 에이전트 기반의 워크플로우 시스템은 분산되어 있기 때문에 특정 부분의 고장이 시스템 전체의 고장으로 확대되지 않는다[3][4].

이동 에이전트를 적용하여 분산 워크플로우 시스템을 구현함에 있어서 단계별로 워크플로우를 수행할 수 있도록 원격 호스트는 서비스를 제공한다. 이러한 원격 호스트들을 옮겨 다니며 작업을 수행하는 중에 같은 호스트에 있거나 다른 호스트에 흩어져서 작업을 수행하는 에이전트들간에 통신을 할 수 있는 방법이 제공되어야 한다. 기존의 이동 에이전트 시스템은 기본적으로 비동기 통신을 제공한다. 단순한 상태 보고나 간단한 에러 발생을 알리는 경우에는 비동기 통신만으로 충분하나 비동기 통신 한가지만을 분산 워크플로우 시스템에 적용하는 것에는 한계가 있다. 전송도중 메시지 분실 위험뿐만 아니라, 이동 에이전트간에 협력하여 작업을 처리하는 경우, 특정 에이전트에게 정보를 요청하거나 응답을 하는 과정에서 상대 에이전트와 지속적인 메시지 교환을 해야 하는 경우가 생기게 된다. 이때 비동기 통신으로 통신을 한다면 신속한 통신이 이루어질 수 없고, 에이전트가 자신의 작업을 수행하는 과정에서 호스트들을 이동하기 때문에 메시지가 원격 호스트에 도착하기 전에 상대 에이전트가 이동 한다면 이동한 위치를 찾아서 다시 메시지를 전송하는 이중의 작업을 해야 한다[5]. 만약 이런 상황이 계속 발생한다면 메시지를 제때에 전달하지 못하고 에이전트가 이동한 위치를 찾아 다니기만 하는 경우가 발생할 수 있다. 이와 같이 비동기 통신만을 지원하는 에이전트 시스템

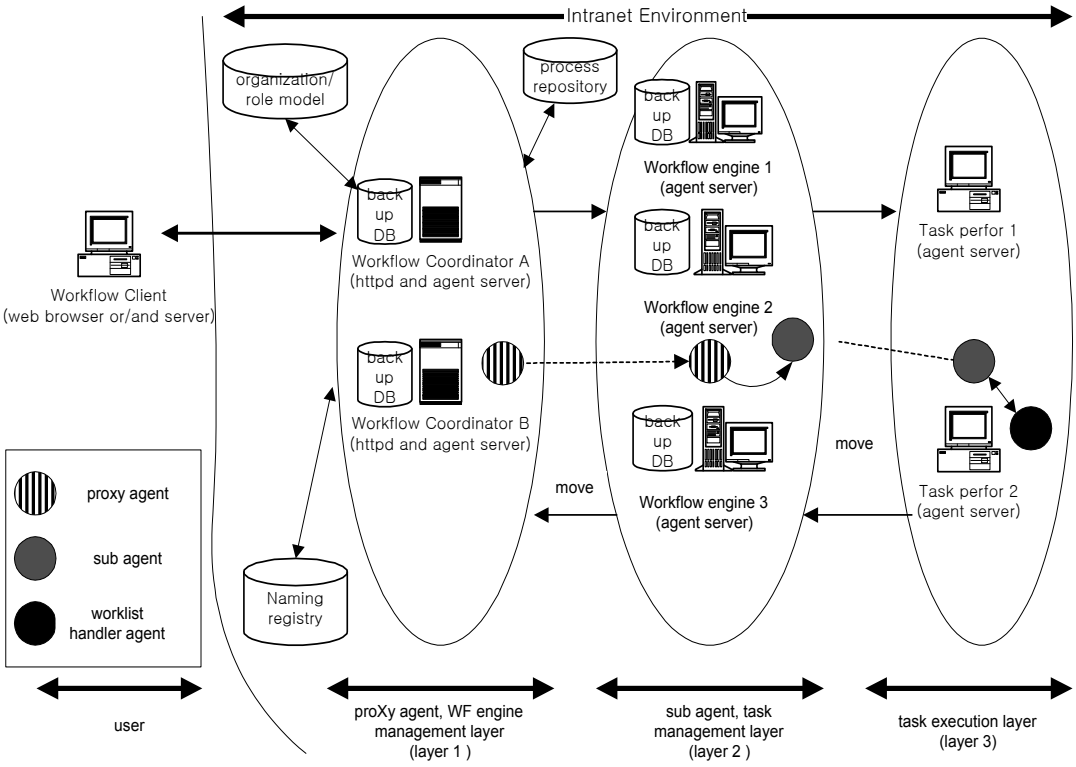

그림1: 에이전트 기반 워크플로우 시스템 구조

은 비효율적이고, 특히 워크플로우를 위한 이동 에이전트 시스템에서 에이전트의 기능을 충실히 제공할 수 없다. 따라서 비동기 통신뿐만 아니라 동기 통신도 제공되어야 한다. 본 논문에서는 이동 에이전트를 기반으로 한 워크플로우 시스템에서 에이전트간 통신 방법으로서 반드시 요구되는 동기 통신과 비동기 통신을 위한 기반구조를 설계한다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련연구로서 이동 에이전트 기반 워크플로우 시스템 구조와 수행구조, 기존 이동 에이전트 시스템의 통신 방법에 대해 언급하고, 3 장에서는 워크플로우 시스템에서 이동 에이전트간 통신이 필요한 경우를 기술함으로써, 비동기 통신외에 동기통신의 필요성을 보인다. 4 장에서는 이러한 동기,비동기 통신을 위한 통신기반 시스템을 설계한다. 5 장에서는 결론을 맺고 앞으로의 연구과제에 대해 설명한다.

2. 관련 연구

2.1 이동 에이전트 기반 워크플로우 시스템의 구조 및 수행구조

이동 에이전트 기반 워크플로우 시스템 구조는 그림 1 과 같이 워크플로우 조정자(Workflow Coordinator), 워크플로우 엔진(Workflow Engine), 워크플로우 클라이언트(Task Performer)의 세 가지 요소들로 구성된다[4]. 각각의 요소들의 기능을 살펴보면 다음과 같다.

• 워크플로우 조정자

- 사용자들에게 웹 서버로서 동작하면서 웹 환경의 사용자 인터페이스를 제공한다. 사용자들의 프로세스 생성이나 프로세스 모니터링에 관한 요청을 받아서 처리한다.
- 웹 서버를 통해 받은 워크플로우 클라이언트들로부터의 프로세스 생성 요청에 따라 요청된 프로세스의 수행을 위해 프록시 에이전트를 생성하여 등록 후, 특정한 워크플로우 엔진으로 보낸다.
- 웹 서버를 통해 받은 워크플로우 클라이언트들로부터의 모니터링 요청에 따라 프로세스 수행 상황을 모니터링하여 결과를 보여 준다.
- 워크플로우 감독자에게 수행중인 프록시 에이전트에 대한 통제를 통해 수행중인 프로세스를 제어하고 감독할 수 있는 기능을 제공한다.

• 워크플로우 엔진

- 다수의 워크플로우 엔진들의 집합으로, 워크플로우 협동자에 의해 조정되는 분산 워크플로우 엔진을 구성한다.
- 하나의 워크플로우 엔진은 프록시 에이전트를 통해 동시에 여러 프로세스 인스턴스들을 수행 관리한다.
- 각각의 프록시 에이전트들은 각자 하나의 프로세스 인스턴스에 대해 그 수행을 관리한다.
- 프록시 에이전트가 서브 에이전트(sub agent)를 생성하고, 이들을 관리 감독할 수 있는 환경을 제공한다.

• 워크플로우 클라이언트

- 작업수행자가 작업을 수행할 수 있는 도구와 인터페이스를 제공한다.
- 작업처리 에이전트가 상주해 있으면서 프록시 에이전트가 보낸 서브 에이전트와 작업 수행자간의 상호작용을 담당할 수 있는 환경을 제공한다.

본 구조하에서 실제 워크플로우 프로세스 수행구조는 다음과 같다. 워크플로우 조정자가 워크플로우 엔진들의 부하를 관찰·관리하게 된다. 여러 개의 워크플로우 조정자가 필요한 이유는, 한 워크플로우 조정자의 고장으로 시스템 전체에 미치는 파급효과를 줄이고, 상호 예비저장을 함으로써 시스템의 신뢰성을 높이기 위해서이다. 사용자로부터 요구

가 들어오면 사용자가 요구하는 작업에 대한 워크플로우 템플릿을 읽어 필요한 정보와 함께 인스턴스인 프록시 에이전트가 생성되며, 인스턴스의 수행은 작업 부하가 적은 워크플로우 엔진에 의해 수행된다. 프록시 에이전트는 워크플로우 엔진의 도움을 받아 워크플로우를 수행하는데, 실제 작업의 수행은 서브 에이전트가 생성되어 작업 수행자에 존재하는 작업처리 에이전트에게 작업을 요청함으로써 수행된다. 작업처리 에이전트는 외부로부터 들어오는 에이전트의 작업 요청을 받아 응용 프로그램을 호출하거나 사람에게 작업을 지시하는 역할을 담당한다. 서브 에이전트의 작업이 끝나면 프록시 에이전트에 의해 또 다른 서브 에이전트가 생성되어 작업이 수행되며, 한 워크플로우가 완료되면 결과는 워크플로우 협동자에 저장되고 사용자에게 통보된다.

2.2 기존 이동 에이전트 시스템의 통신방법

기존의 이동 에이전트 시스템인 TACOMA, ARA, Mole 등에서 제공하는 에이전트간 통신 방법은 다음과 같다.

•Ara

에이전트들간의 통신은 같은 호스트내의 service point 라고 불리는 특정 장소에서만 이루어지고, n:1 의 클라이언트-서버 방식으로 메시지가 전송된다. 에이전트들이 동시에 서버에 요청을 하면 서버는 에이전트들에게 메시지들을 보내어 응답을 한다. 서버는 메시지에 에이전트의 이름을 넣어서 보낸다. 즉 클라이언트 서버방식의 비동기 통신이 이루어진다. 원격 호스트에 흩어져 있을때도 통신을 지원하는데 이 경우도 간단한 비동기 메시지 전송이 이루어진다[6].

•Concordia

이벤트 방식의 비동기 통신과 여러 에이전트들이 협력(collaboration)하여 작업을 수행하는 경우를 위해 동기 통신을 제공한다. 이벤트 방식에는 두 가지가 있는데 선택적 이벤트(Selected Event)와 그룹 이벤트(Group-Oriented Event)방식이 있다. 선택적 이벤트는 각 에이전트가 받기를 원하는 이벤트 타입과 에이전트 객체가 위치한 곳에 대한 레퍼런스를 이벤트 매니저에게 등록하면 이벤트 발생시 이벤트 매니저는 등록 테이블을 참조하여 이벤트의 타입을 비교한 후 해당 에이전트에게 전달해준다. 그룹 지향 이벤트는 그룹에 속한 모든 에이전트들에게 이벤트를 전달한다. 그리고 하나의 작업을 수행하기 위해 여러 에이전트가 협력하여 수행하는 경우 그룹 지향 이벤트 방식도 사용하지만 동기 통신도 사용한다. 동기 통신을 하기 위해서 같은 그룹의 에이전트들은 동일한 호스트내의 특정 장소에 모여야 한다[7].

•TACOMA

에이전트가 서비스 에이전트와 통신을 하기 위해서는 meet 이라는 프리미티브를 사용하여 통신을 시작한다. 서비스 에이전트는 meet 에 의해 활성화되기를 수동적으로 기다리며 통신 방식은 비동기 통신을 지원한다 [8].

•Mole

Java 에서 제공하는 RMI 를 이용하여 전역적(global)으로 메시지를 교환하는 비동기 통신을 지원한다. 통신 상대인 서비스 에이전트의 위치를 알아내기 위해서는 에이전트 이름과 그 에이전트가 제공하는 서비스에 대한 정보를 대응시킨 리스트로 이루어진 주소 테이블을 관리하면서 이 테이블을 이용하여 통신을 하려는 서비스 에이전트 위치를 찾아내는 지역적인 서비스 탐색(local service lookup) 메커니즘을 이용한다[9].

•ffMAIN

HTTP 를 통하여 통신을 한다. 유일(unique)한 URL 은 각각 데이터 아이템을 나타내고 이곳에 데이터들이 저장되며 저장된 데이터들을 읽을 수 있다. HTTP 를 이용하여 통신 서비스를 하기 때문에 비동기 통신을 제공한다[10][11].

지금까지 알아본 바와 같이 Concordia 는 두 가지 통신방식을 모두 지원하지만 그 외의 Ara, TACOMA, Mole, 그리고 ffMAIN 등 대부분의 이동 에이전트 시스템이 비동기 통신만을 제공한다. 비동기 통신에 더 중점을 두는 이유는 에이전트의 이동성 때문이라고 말할 수 있다. 그러나 이러한 이동 에이전트 시스템이 워크플로우 시스템에 사용되기에는 워크플로우를 위한 이동 에이전트의 기능 지원에 한계를 가진다. 본 논문에서는 비동기 통신뿐만 아니라 같은 호스트에서 또는 원격 호스트에 있는 에이전트들간에 동기 통신을 제공한다.

3. 워크플로우 수행을 위한 이동 에이전트간의 통신

2 장에서 제시한 이동 에이전트 기반 워크플로우 시스템에서 프록시 에이전트와 서브 에이전트, 서브 에이전트와 작업수행 에이전트(Worklist Handler Agent), 그리고 서브 에이전트간 통신이 필요한 경우에 대해 알아본다.

3.1 프록시 에이전트와 서브 에이전트간 통신

프록시 에이전트와 서브 에이전트간에 통신이 필요한 경우는 다음과 같다.

- 오류 발생시
- 상태 관리시
- 수행결과 통보시

프록시 에이전트에게 주어진 하나의 작업에 대해 하나의 서브 에이전트만 생성하여 수행하도록 하는 경우가 있고, 단위작업으로 세분하여 여러 에이전트를 생성하여 각 서브 작업을 수행하도록 할 경우도 있다. 서브 에이전트들은 주어진 임무를 수행하기위해 여러 호스트들에서 서비스를 제공하는 플레이스들을 옮겨 다니며 작업을 수행한다. 작업을 수행하는 과정에서 서브 에이전트에게 오류가 발생한 경우 간단한 오류이면 프록시 에이전트에게 오류 메시지만 보내고 작업을 계속한다. 그러나 작업을 계속 수행하지 못할 정도로 심각한 오류가 발생한 경우 서브 에이전트는 프록시 에이전트에게 오류 메시지를 보내고 작업 수행을 중단한다. 사용자가 프로그램의 디버깅을 위해서 프록시 에이전트내에 오류가 발생했을 때 상세한 내용을 보고하도록 하는 루틴을 넣어놓을 수도 있다.

서브 에이전트가 작업을 끝낸 후 작업이 완료 되었음을 프록시 에이전트에게 알리고 홈 호스트로 되돌아가야 할 경우 홈 호스트에 문제가 생겨 서브 에이전트의 복귀가 어려운 경우가 있다. 이때 프록시 에이전트는 서브 에이전트에게 호스트에 발생한 문제가 해결될 때까지 기다리도록 하는 메시지를 보낸다.

프록시 에이전트는 자신이 생성한 서브 에이전트의 상태를 관리한다. 서브 에이전트에게 상태를 주기적으로 보고하도록 하거나 필요한 경우에 중간 상태를 보고하도록 메시지를 보낼 수 있다. 그리고 서브 에이전트가 자신에게 주어진 작업의 수행을 끝낸 경우 프록시 에이전트에게 작업 수행 결과를 통보해야 한다.

3.2 서브 에이전트와 작업 수행 에이전트간의 통신

서브 에이전트와 작업 수행 에이전트간 통신이 필요한 경우는 다음과 같다.

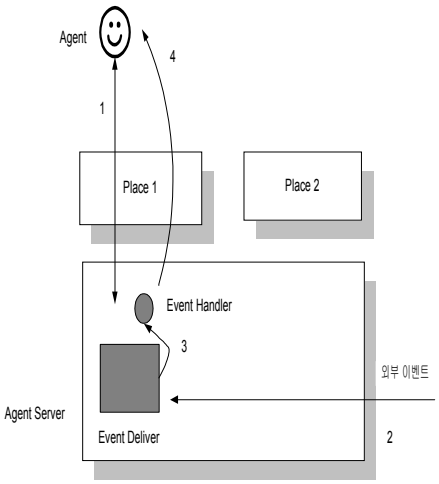

그림2: 에이전트의 이벤트 처리 절차

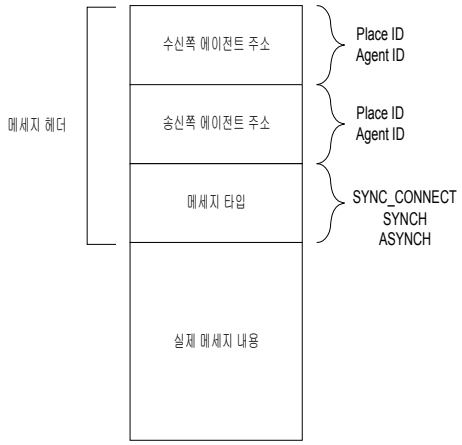

그림3: 메세지 구조

· 작업 수행 요청시
· 수행결과 전달시

서브 에이전트는 자신의 작업을 수행하는 과정에서 실제 작업을 처리하는 작업 수행자에 존재하는 작업처리 에이전트에게 작업 수행을 요청한다. 그리고 응용 프로그램의 수행이 끝나고 난 후 작업처리 에이전트는 서브 에이전트에게 작업이 완료되었다는 메시지를 보낸다.

3.3 서브 에이전트간 통신

서브 에이전트간에 통신이 필요한 경우는 다음과 같다.

· 오류 발생시
· 수행결과 통보시

서로 협력하여 작업을 수행하는 경우 모든 서브 에이전트의 작업이 완료되었을 때 특정 장소에 모여서 각자 수행한 작업의 결과를 모아 전체 작업을 완료해야 한다. 그러나 특정 서브 에이전트에게 오류가 발생하여 더 이상 작업 수행이 불가능할 때 그룹전체 또는 특정 에이전트에게 오류 메시지를 보내야 한다.

특정 에이전트의 작업 수행 결과에 따라 또는 중간 결과에 따라 다음에 자신이 수행해야 할 일이 결정되는 경우 작업을 중단한 상태에서 특정 에이전트의 작업이 완료될 때까지 기다렸다가 수행 결과를 통보 받아야 한다.

위에서 열거한 통신이 필요한 경우 중 동기 통신을 사용하는 것이 더 효율적인 경우는 다음과 같다.
· 서브 에이전트가 작업 수행 중 오류가 발생했을 때 프록시 에이전트에게 오류 발생 원인과 오류 상태에 대한 상세한 기록을 보고하는 경우
· 프록시 에이전트가 서브 에이전트의 작업수행 시작 시점부터 종료 시점까지의 처리 내역에 대한 기록을 보고 받는 경우
· 같은 그룹의 서브 에이전트들이 세부 작업을 완료한 후 특정 장소에 모여 수행한 결과를 가지고 협의하는 과정에서 서로 통신하여 결과를 얻고 이 결과를 프록시 에이전트에게 보고하는 경우
· 서브 에이전트가 단계별로 자신의 작업을 처리하는 과정에서 작업처리 에이전트에게 실제로 작업을 수행하기 위해 응용 프로그램을 실행시킬 필요가 있다[10]. 이때 작업처리 에이전트에게 응용 프로그램 수행에 필요한 조건 리스트와 예상결과를 알려주어야 하는 경우
· 서브 에이전트들이 협력하여 작업을 수행해야 하는 경우 일정 시점에서 특정 서브 에이전트의 작업 수행 결과에 따라 다음에 처리해야 할 일의 순서가 달라지기 때문에 처리 결과에 대한 상세한 보고를 받아야 할 경우

워크플로우 응용에서는 위에서 열거한 경우들 외에도 동기 통신이 필요한 경우가 많이 발생하기 때문에 비동기 통신만을 지원하는 이동 에이전트 시스템을 워크플로우에 적용하는데는 어려움이 많다. 따라서, 워크플로우를 위한 이동 에이전트 시스템의 통신 기반구조는 동기, 비동기 통신을 모두 제공해야 한다.

4. 동기 및 비동기 통신 기반구조의 설계

통신 기반 구조는 그림 2 와 같다. 에이전트는 이벤트 핸들러에게 자신이 받고자 하는 이벤트 타입을 등록하고 이벤트가 발생하면 이벤트 핸들러는 이벤트 테이블을 참조하여 에이전트에게 전달한다. 받은 이벤트에 따라 통신을 해야 할 경우, 에이전트가 직접 하는 것이 아니라 통신을 시작하기 전에 에이전트가 생성한 Comm Object 라는 통신을 담당하는 객체를 통하여 통신이 이루어진다. 이 객체를 통해 전달하고자 하는 내용을 메시지로 만들어서 보내게 되는데, 메시지는 그림 3 과 같이 헤더와 실제 전달하고자 하는 내용으로 이루어져있다. 헤더는 받는 쪽의 주소와 보내는 쪽의 주소, 그리고 동기 통신인지 비동기 통신인지를 나타내는 메시지 타입으로 이루어져있고, 메시지 내용은 실제 전달하고자 하는 내용으로 이루어진다.

4.1 동기 통신

동기 통신은 두 에이전트간에 통신 경로를 설정하여 지속적으로 메시지를 교환하기 위해 소켓을 이용한다. 동기 통신의 수행 구조는 그림 4 와 같다. 통신을 하고자 하는 에이전트는 Comm Object 를 생성한 후(step1), 통신하고자 하는 상대편 에이전트의 위치를 찾기 위해 Comm Object 에게 질의어를 넘겨주어야 한다. 질의어를 만들 때 상대 에이전트를 찾기 위한 적절한 조건들을 넘겨주어야 하는데 조건들에는 에이전트의 고유한 ID, 에이전트의 서비스 타입 즉, 프록시 에이전트 또는 서브 에이전트가 될 수 있다. Comm Object 는 NDS(Naming Directory Server)가 제공하는 lookup 이라는 메소드를 호출하여 에이전트로부터 받은 질의어를 넘

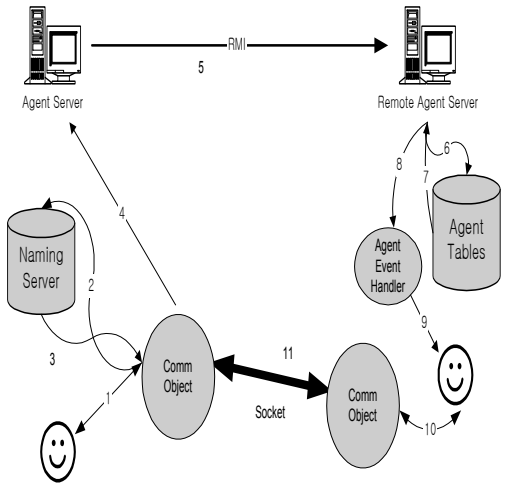

그림4: 동기 통신 구조

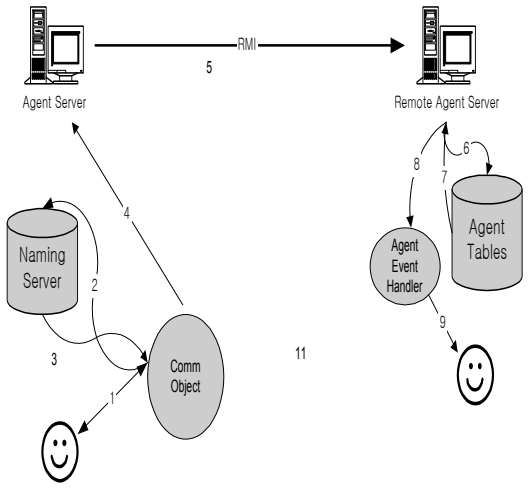

그림5: 비동기 통신 구조

겨준다(step2). NDS 는 이 질의어의 조건을 참조하여 자신이 관리하는 에이전트 테이블에서 적절한 에이전트를 찾아서 Comm Object 에게 결과를 넘겨주는데(step3), 만약 넘겨받은 결과가 null 인 경우는 질의어의 조건에 해당하는 에이전트가 없으므로 통신 상대가 없다는 의미이다. 결과가 null 이 아닌 경우, 만약 넘겨받은 에이전트의 주소가 하나인 경우는 이 에이전트에게만 메시지를 보내면 되지만 질의어의 조건에 해당하는 에이전트가 여러 개인 경우가 생기게 된다. 이때는 에이전트의 주소 리스트중 첫번째 에이전트를 통신 상대로 택한다. Comm Object 는 이 에이전트의 주소로 메시지를 만들어 에이전트 서버에게 전달한다(step4). 메시지의 구조를 살펴보면 헤더는 보내는 쪽 에이전트의 주소, 메시지를 받을 상대편 에이전트 주소, 그리고 "통신을 시작하고자 한다"는 의사표시로써 SYNC_CONNECT 라는 메시지 타입으로 채워진다. 통신을 바로 시작하지 않고 먼저 의사표시를 하는 이유는 상대편 에이전트로 하여금 통신을 할 준비를 하도록 하기 위함이다. 메시지 내용에는 소켓 통신을 위해 Comm Object 가 에이전트 서버에게 요청하여 받은 사용 가능한 포트 번호를 넣어준다. 에이전트 서버는 이 메시지를 받아서 RMI 를 통하여 원격 에이전트 서버에게 전송한다(step5). Comm Object 는 서버 소켓을 열어서 클라이언트 쪽 에이전트가 연결해 오기를 기다렸다가 응답이 오면 양방향 동기 통신을 시작한다. 이때 고려해야 할 사항으로 상대편 에이전트로부터 응답이 올 때까지 무작정 기다릴 수는 없으므로 서버 소켓을 열고 나서 타이머를 설정한다. 일정 시간이 지나도록 클라이언트쪽 에이전트가 응답하지 않으면 통신할 의사가 없거나 연결에 문제가 생겨 통신을 할 수 없다는 뜻이므로 Comm Object 는 통신을 시도한 에이전트에게 false 를 넘겨주고 서버 소켓을 닫고 통신을 중단한다. 통신을 시도한 에이전트는 다시 메시지를 보내어 통신을 시도하거나 자신이 하던 작업을 수행한다.

한편 메시지를 받은 원격 에이전트 서버는 메시지 헤더를 분석하여 메시지를 받을 에이전트가 자신의 서버에 있는지 확인 후(step6, step7) 이벤트 핸들러에게 메시지를 넘겨준다(step8). 이벤트 핸들러는 이벤트 테이블이라는 것을 관리하는데 에이전트들은 이 테이블에 자신들이 받고싶은 메시지 타입을 등록한다. 이벤트 핸들러는 이 테이블을 참조하여 받은 메시지 타입과 에이전트가 등록한 메시지 타입이 일치하면 해당하는 에이전트에게 메시지를 보낸다(step9). 메시지를 받은 에이전트는 동기통신이 가능한 상태라면 Comm Object 를 생성하여(step10) 서버쪽 에이전트가 열어 놓은 포트에 연결하여 동기통신을 시작한다(step11).

4.2 비동기 통신

비동기 통신은 동기 통신과는 달리 지속적인 연결을 위해 통신 채널을 설정하여 메시지 교환을 할 필요가 없고 자바에서 제공하는 RMI 를 이용하여 원격 호스트에 있는 에이전트에게 메시지를 보내기만 한다. 그 외에 통신 절차는 동기 통신과 같다. 수행 절차는 그림 5 와 같다. 메시지를 보내고자 하는 에이전트는 우선 Comm Object 를 생성한 후(step1) 동기 통신에서와 같은 방법으로 질의어를 만들어 Comm Object 를 통하여 NDS 에 넘겨준다. NDS 는 질의어의 조건을 참조하여 에이전트 테이블에서 적절한 에이전트를 찾아서 Comm Object 에게 넘겨준다(step2,step3). 메시지 헤더에는 보내는 쪽의 주소와 받는 쪽의 주소, 그리고 비동기 통신임을 알리는 통신 타입을 넣고, 메시지 내용에는 실제 전달하고자 하는 내용을 넣는다. Comm Object 는 에이전트 서버에서 제공하는 메시지 전달 메소드를 호출하여(step4) RMI 를 통하여 원격 에이전트에게 메시지를 전달하게 된다(step5). 메시지 헤더에 하나의 에이전트 주소만 있다면 특정 에이전트에게만 메시지가 전달되고 에이전트들의 주소 리스트이면 다수의 에이전트들에게, 단일 플레이스 주소이면 그 플레이스 내의 모든 에이전트들에게, 플레이스들의 주소 리스트이면 리스트의 모든 플레이스 내의 모든 에이전트들에게, 에이전트 서버들의 주소 리스트이면 에이전트 서버들내의 모든 에이전트들에게 전달된다. 원격 에이전트 서버는 이 메시지를 받아서 에이전트 테이블에서 에이전트를 찾아서(step6,step7) 이벤트 핸들러에게 메시지를 전달하게되고(step8) 이벤트 핸들러는 이벤트 테이블을 참조하여 에이전트에게 메시지를 전달한다(step9).

5. 결론 및 향후 연구과제

지금까지 이동 에이전트 기반 워크플로우 시스템에서 에이전트간 통신 방법으로서 제안한 동기, 비동기 통신 방법에 대해 기술하였다. 위에서 언급한 워크플로우 시스템에서는 사용자가 요구한 작업을 처리하기 위해 프록시 에이전트를 생성하고 프록시 에이전트는 서브 에이전트들을 생성하여 서브에이전트들에게 각각의 단위작업을 수행하도록 지시한다. 하나의 목적하는 작업을 수행하기위해 서브 에이전트들은 자신이 맡은 세부 작업을 수행하는 과정에서 다른 서브 에이전트와 또는 자신을 생성한 프록시 에이전트, 그리고 작업처리 에이전트와 통신을 위해 동기 통신과 비동기

통신 두 가지를 사용하도록 제안하였다. 앞으로는 워크플로우 시스템의 한 요구 사항인 Reliability 를 통신기반에서 지원하기 위해 오류에 대해 안정적인 신뢰성 있는 에이전트 통신 프로토콜(protocol)에 대한 연구를 수행하고자 한다.

감사의 글

이 연구는 일부 한국 과학재단(KOSEF)의 특정기초연구(98-1012-11-01-3)에 의해 지원되었습니다.

참고 문헌

[1] Neeran M. Karnik, Anand R. Tripathi, “Design Issues in Mobile Agent Programming Systems” IEEE Concurrency, Vol. 6, No. 3, 1998

[2] Hyacinth S. Nwana, “Software Agents : An Overview” Knowledge Engineering Review, Vol11, No3, 00. 205 – 224, 1996.

[3] G. Alonso, D. Agrawal, A. El Abbadi, C. Mohan, “Functionality and Limitation of Current Workflow Management Systems” IEEE Expert Journal, 1996.

[4] 유정준, 서영호, 송상범, 이동익, “에이전트 기반 워크플로우 시스템 구조 및 이동 에이전트 요구사항” 한국정보처리학회 추계 학술발표논문집, 1999.

[5] Murphy, A.L., Picco, G.P. “Reliable Communication for Highly Mobile Agents” Agent Systems and Applications, 1999. Proceedings. First International Symposium on ...and Third International Symposium on Mobile Agents , Page(s): 141 –150, 1999.

[6] H. Peine and T. Stolpmann, “The Architecture of the Ara Platform for Mobile Agents” In Kurt Rothermel, Radu Popescu-Zeletin (Eds.): Proc. of the First International Workshop on Mobile Agents 97, 1997.

[7] D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, B. Peet, “Concordia: An Infrastructure for Collaborating Mobile Agents” First International Workshop on Mobile Agents 97, 1997.

[8] Nils Peter Sudmann , “TACOMA - fundamental abstractions supporting agent computing in a distributed environment” PhD thesis in Computer Science, Department of Computer Science University of Tromsø, Norway, 1996.

[9] J. Baumann, F. Hohl, N. Radouniklis, M. Straßer, K. Rothermel, Univ. Stuttgart, Germany, “Communication Concepts for Mobile Agent Systems” First International Workshop on Mobile Agents 97, 1997.

[10] G. Kappel, S. Rausch-Schott, W. Retschitzegger, “TriGSflow - Applying Active Concepts to WorkFlow Management” GI-Fachgruppen-Treffen Aktive Datenbanken, Datenbank-Rundbrief (GI-FG 2.5.1), Vol. 14, Hamburg, pp. 21-24, 1994.

[11] Anselm Lingnau, Oswald Drobnik, “Agent-user Communications: Request, Results, Interaction” First International Workshop on Mobile Agents 98, pp.209 – 221, 1998.

[12] Georgakopoulos, D., M. Hornick, and A. Sheth, "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure," Distributed and Parallel Databases 3(2): 119-154, 1995.

[13] Jonathan Dale, “A Mobile Agent Architecture for Distributed Information Management” PhD thesis in University of Southhampton, 1997.