멀티 에이전트를 위한 통신 프레임웍의 설계 및 구현

성현[0], 곽재연, 김정선
한 양 대 학 교 컴 퓨 터 공 학 과
{hseong, jykwak, jskim}@cse.hanyang.ac.kr

Design and Implementation of Communication Framework for Multi-Agents

Hyun Seong[0], Jaeyeon Kwack, Jungsun Kim
School of Electrical Engineering and Computer Science, Hanyang University

요 약

최근에 들어와 그 응용분야가 확대되어 가고 있는 에이전트 소프트웨어는 점점 그 크기와 복잡성이 커져 감에 따라 소프트웨어 개발의 효율성을 필요로 하는 추세이다. 이에 따라 네트워크에 분산되고 이질적인 시스템에 존재하는 에이전트들은 자신의 정보를 효율적으로 교환할 수 있도록 하기 위해 통신 프레임웍을 필요로 하게 되었다. 본 논문에서는 네트워크에 존재하고 동일한 환경 또는 서로 다른 환경에 존재하는 에이전트들 간의 의사 소통을 위한 통신 프레임웍의 설계 및 구현을 제시한다.

기본적으로 에이전트 소프트웨어 아키텍처는 KRIL(KQML Router Interface Labrary), Router, Facilitator로 구성되어 진다. 같은 호스트에 존재하는 에이전트들은 같은 주소를 가지기 때문에 각기 다른 Router를 가질 필요 없이 하나의 Router를 공유한다. Router는 로컬 호스트에 존재하는 에이전트들의 리스트를 유지 관리하며, 로컬 에이전트가 보낸 메시지의 수신자가 자신의 리스트에 없을 경우 Facilitator를 통해 다른 호스트에 존재하는 에이전트에 메시지를 전달한다. 서로 다른 환경에 존재하는 에이전트들 간의 통신과 스레드 관리, 병행처리와 동기화 등을 위해 KRIL은 ACE(The Adaptive Communication Environment) 라이브러리를 사용하였으며, Router와 Facilitator는 Java로 구현하였다.

1. 서 론

현재 소프트웨어들은 점차 복잡하고 대형화되어 가고 있어 이를 지원하기 위한 소프트웨어 환경 또한 이질적이고 분산된 환경을 지원하도록 요구되고 있다. 이와 같이 시스템이 복잡해지고 대형화되어감에 따라 소프트웨어 환경 또한 적합한 해결책을 찾고 있다. 그 하나의 해결책으로 에이전트 시스템[1]이 나타나게 되었다.

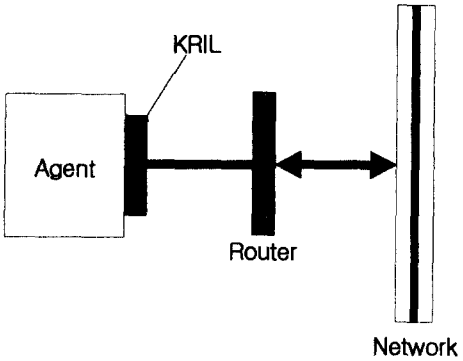
네트워크에 분산된 각각의 에이전트들은 상이한 시스템에 존재할 수 있기 때문에 그들간의 정보교환수단이 미비하게 되어 현재 자신이 가진 정보의 일관성과 공유화를 위한 수단으로는 어려움이 있다. 이러한 문제에 대한 효율적인 문제해결로써의 에이전트 시스템은 시스템간의 상이한 입출력 규약에 따른 정보 교환의 문제점을 매우 유연하고 강력한 기능을 갖는 에이전트 통신 언어(Agent Communication Language; ACL)를 사용하여 완화시킬 수 있으며, 적절한 시기에 분산된 시스템의 각 컴포넌트에게 정보를 전달할 수 있다. 즉, 네트워크로 연결된 환경 하에서 각각의 프로그램들이 상호 통신을 통하여 사용자가 원하는 처리를 해줄 수 있게 구현한 것이 에이전트 시스템이다.

에이전트는 ACL로 다른 에이전트와 통신하는 소프트웨어 컴포넌트로써 상황에 따라 적당한 ACL을 주고받으며, 다른 에이전트와 정보 및 서비스를 교환하고 협동적으로 일을 수행한다. ACL은 에이전트가 수행할 수 있는 일의 범위를 결정짓는 중요한 요소이다. 현재 널리 쓰이는 ACL로는 에이전트 표준화 단체인 FIPA(Foundation for Intelligent Physical Agent)의 권고안을 수용한 KQML(Knowledge Query And Manipulation Language)[2, 3, 4]이 있다.

본 논문에서는 상이한 네트워크 상에 존재하는 에이전트들이 자신이 가지고 있는 정보나 서비스를 다른 에이전트들에게 효율적으로 전달 및 상호 교환할 수 있고 에이전트간의 서로의 존재를 확인하고 상호 이해할 수 있는 메커니즘이 지원되는 통신 프레임웍을 설계 및 구현한다. 그리고, 에이전트간에 의사 소통할 수 있는 언어로써 KQML을 사용하며, 각 에이전트간의 통신환경으로 ACE(The Adaptive Communication Environment)[5]를 사용한다.

2. 에이전트 기본 아키텍처

그림 1은 하나의 에이전트가 다른 에이전트와 통신하기 위해 필요한 기본적인 컴포넌트들을 소개한다[6].



<그림 1> KQML 소프트웨어 구조

2-1. KRIL (KQML Router Interface Library)

KRIL은 KQML Router와 에이전트 사이의 API를 제공하며, 에이전트의 질의를 KQML 메시지로 변환하거나 수신된 KQML 메시지를 에이전트가 이해할 수 있는 형태로 바꾸어주는 역할을 한다.

2-2. KQML Router

KQML Router는 다른 에이전트로 송신되거나 수신되는 KQML 메시지를 관리하고 제어한다. 보내려는 KQML 메시지는 point-to-point 및 multicasting을 통해 전달될 수 있다.

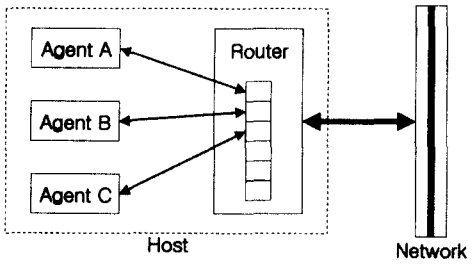
2-3. Facilitator

Facilitator는 KQML Router와 협력하여 송신 에이전트가 원하는 수신 에이전트를 동적으로 투명성 있게 선택할 수 있도록 하여준다. 즉, Router는 에이전트가 보낸 메시지의 수신자를 자신의 가지고 있는 에이전트 리스트에서 못 찾을 경우 Facilitator를 통해 수신자의 주소를 알아낼 수 있다.

3. Shared Router Policy

그림 1에서 본 것처럼 하나의 에이전트는 KRIL과 Router를 각각 하나씩 필요로 한다. 하지만, 본 논문에서는 하나의 호스트에 하나의 Router만을 둠으로써 하나의 Router가 여러 로컬 에이전트를 관리한다.

그림 2는 여러 에이전트들이 하나의 Router를 공유하는 모습을 보여준다. Router는 로컬 에이전트들의 정보를 유지하기 위해 해시 테이블을 갖는다. Router는 에이전트로부터 받은 메시지의 헤더에 적혀있는 수신자의 아이디를 확인한 후에 자신의 리스트와 비교한다. 리스트에 수신자가 있을 경우 Facilitator의 도움 없이 직접 메시지를 보내며, 없을 경우엔 Facilitator를 통해 수신자의 주소와 포트번호를 얻는다.



<그림 2> KQML 소프트웨어 구조

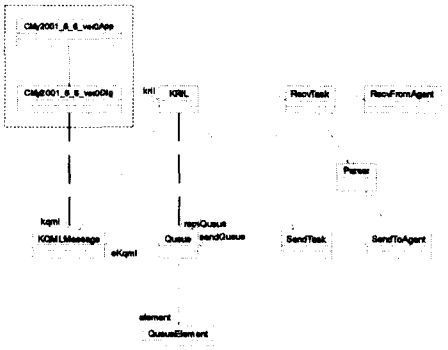
4. KRIL과 Router의 설계

그림 3는 KRIL의 클래스 다이어그램이다. KRIL의 KRIL 클래스는 다음과 같은 메시지 전송 인터페이스를 제공한다.

```
void isend(string rec_agent), KQMLMessage*);
KQMLMessage* send(string rec_agent, KQMLMessage*);
```

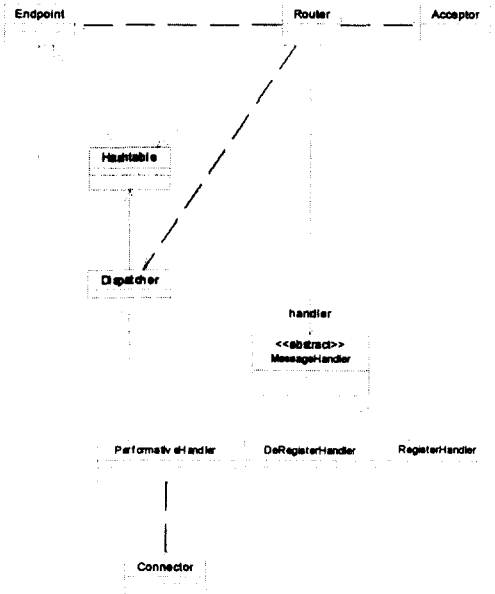
첫 번째 함수는 non-blocking 함수로써 에이전트로부터의 Send의 요구가 로컬 버퍼에 전송되어 버퍼링됨으로써 제어가 즉시 호출 프로세스에게 되돌려지는 형태의 메시지 전송 인터페이스이다.

두 번째 함수는 blocking 함수로써 Send 명령이 실행했을 때, 메시지의 수신이 완료될 때까지 호출한 프로세스가 blocking되는 메시지 전송 인터페이스이다.



<그림 3> KRIL의 클래스 다이어그램

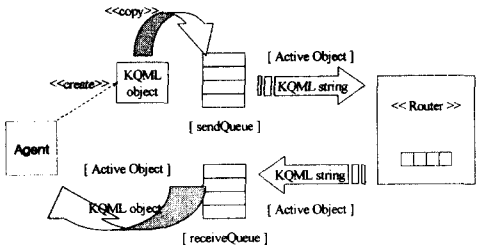
그림 4는 Router의 클래스 다이어그램이다. Router는 에이전트로부터 수신되는 메시지의 헤더로부터 메시지의 포맷을 알아낼 수 있으며, 각각의 메시지 포맷을 처리하는 핸들러를 Dispatcher 클래스를 통해 등록하므로써 수신되는 메시지 포맷에 따라 자동적으로 알맞은 핸들러가 불려진다.



<그림 4>Router의 클래스 다이어그램

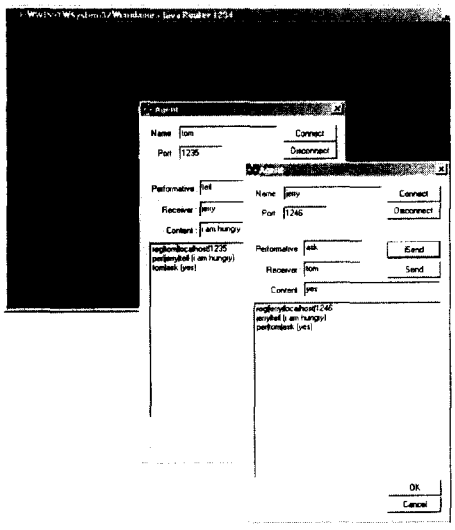
5. KRIL과 Router와의 통신

에이전트는 KRIL을 이용하여 Router에 KQML 메시지를 보낸다. 메시지를 보내는 방식으로는 non-blocking message passing과 blocking message passing방식을 이용한다. 그림 5는 non-blocking message passing 방식을 보여준다. 에이전트는 보내려는 메시지를 KQML 오브젝트로 생성한 후, KRIL 클래스의 isend() 인터페이스를 통해 sendQueue에 삽입한 후, 즉시 리턴한다. KRIL이 생성한 액티브 오브젝트인 sendTask 오브젝트는 sendQueue에 KQML 메시지 오브젝트가 추가되면 이를 확인하여 가져온다. sendTask는 sendQueue로부터 가져온 KQML 메시지 오브젝트를 스트링으로 변환 후, 소켓을 통해 Router에 전송한다. 에이전트가 Router로부터 메시지를 수신할 경우에도 마찬가지로 액티브 오브젝트 receiveTask가 스트링 형태로 수신한 메시지를 KQML 메시지 오브젝트로 변환한 후, receiveQueue에 삽입한다. 또 하나의 액티브 오브젝트인 readTask는 receiveQueue에 KQML 메시지 오브젝트를 확인하여 에이전트가 가져갈 수 있게 알려준다.



<그림 5> non-blocking message passing

6. 구현 결과



<그림 6> 실행 예제

멀티 에이전트를 위한 통신 프레임웍을 이루는 KRIL은 Windows 2000 환경에서 Visual C++ 6.0을 이용하여 구현하였다. 하지만, ACE 라이브러리를 사용하였기 때문에 약간의 코드의 수정만으로도 UNIX에서도 실행 가능하다. Router 및 Facilitator는 jdk 1.3을 이용하여 구현하였다. 그림 6은 에이전트가 KRIL과 Router를 사용하여 통신하는 장면이다. 에이전트는 MFC 라이브러리를 사용하여 GUI를 구성하였다.

7. 결론 및 향후 과제

본 논문은 네트워크에 존재하는 에이전트들 사이에서 원활한 의사 소통을 위한 통신 프레임웍을 설계 및 구현하였다. 다른 환경에 존재하는 에이전트들과도 손쉽게 통신하기 위해 ACE 라이브러리를 사용하였으며, 로컬 호스트에 존재하는 에이전트들간의 효과적인 통신을 위해 Router를 공유하였다.

앞으로의 연구방향은 Facilitator를 Jini 아키텍처를 이용하여 개발하여 보다 자동적으로 유지 관리될 수 있는 프레임웍을 개발하는 것이다. 또한 이질적인 시스템간에 존재하는 데이터의 불일치를 위해 CDR(Common Data Representation)을 사용하여 구현해야한다.

7. 참고 문헌

[1] N. R Jennings, K. P. Sycara, and M. Wooldridge A Roadmap of Agent Research and Development In Journal of Autonomous Agents and Multi-Agent Systems. 1(1), pages 7-36. July 1998.

[2] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In Jeffery M. Bradshaw, editor, Software Agents. MIT Press, 1995.

[3] Y. Labrou and T. Finin. A Proposal for a new KQML Specification. TR CS-97-03, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County(UMBC), Feb. 3, 1997

[4] Y. Labrou and T. Finin. A semantics approach for KQML - a general purpose communication language for software agents. In Third International Conference on Information and Knowledge Management.

[5] The Adaptive Communication Environment (ACE) http://www.cs.wustl.edu/~schmidt/ACE.html

[6] T. Finin, R. Fritzson, D. Mckay and R. McEntire, KQML as an Agent Communication Language. To appear in The Proceedings of the Third International Conference on Information and Knowledge Management(CIKM'94), ACM Press, November 1994.