제 28 회 한국정보처리학회 추계학술발표대회 논문집 제 14 권 제 2 호 (2007. 11)

# 에이전트 네트워크 자가 구성방법

박병선*, 윤희용*
*성균관대학교 컴퓨터공학과
e-mail : * parkho79@skku.edu, *youn@ece.skku.ac.kr

# Self-organization for agent network

Byung-Seon Park*, Hee-Yong Youn*
**Dept. of Computer Engineering, Sung-Kyun-Kwan University

**요 약**

에이전트란 사용자의 관점에서 시스템이 사용자를 대신해서 사용자가 원하는 작업을 자동적으로 수행해 주는 소프트웨어로서 오래 전부터 연구되어 1990년대 초부터 관련제품도 대거 등장하게 되었다. 하지만, 나날이 다양해지고 복잡해지는 사용자의 요구를 해결하려면 단독 에이전트의 능력으로는 한계가 있었고 그 해결 방법으로 지역적으로 분산된 다른 에이전트의 도움을 받아 처리하는 분산 협동 처리의 개념이 등장하였다. 여러 에이전트의 분산 협동 처리를 위해서는 에이전트간 통신이 필수적이다. 그 목적은 정보나 작업 처리의 공유와 교환에 있으며 다른 에이전트에게 처리를 요구하기도 하기 때문이다. 이러한 요구사항을 만족시키기 위해서는 지능적인 작업을 수행하기 위한 에이전트, 에이전트 플랫폼이 효율적 구성이 이루어져야 한다. 이러한 에이전트와 에이전트 플랫폼으로 구성된 에이전트 네트워크를 효율적 구성하기 위하여 본 논문에서는 이러한 과정에서 발생할 수 있는 기존 방식의 문제점들을 JXTA의 예를 통해 지적하고, 개선방안으로 에이전트 네트워크 자가 구성방법을 소개하려 한다.

## 1. 서론

에이전트라는 용어의 사전적인 의미는 대행자, 대리인 정도의 의미를 가지지만 컴퓨터 분야에서는 작업을 대행해주는 프로그램으로 해석될 수 있다. 사용자를 대신해서 사용자가 원하는 작업을 자동적으로 수행하는 소프트웨어로서 인공지능 분야에서 오래 전부터 연구되어온 개념이다. 1980년대 말부터 에이전트라는 분야는 인공지능과 분리되어 독자적인 연구 주제로 대두되었고 관련제품도 1990년대 초부터 대거 등장하게 되었다.

하지만 나날이 다양해지고 복잡해지는 사용자의 요구를 해결하기 위해서는 단독 에이전트의 능력으로는 한계가 있었고 그 해결 방법으로 지역적으로 분산되어 있는 다른 에이전트의 도움을 받아 처리하는 분산 협동 처리의 개념이 등장하였다. 여러 에이전트의 분산 협동 처리를 위해서는 에이전트간 통신이 필수적이며, 에이전트간 통신을 위해서는 이를 구성하는 네트워크 즉 에이전트 네트워크가 필요하며 이를 효율적으로 구성하기 위한 방안들이 연구 중이다.

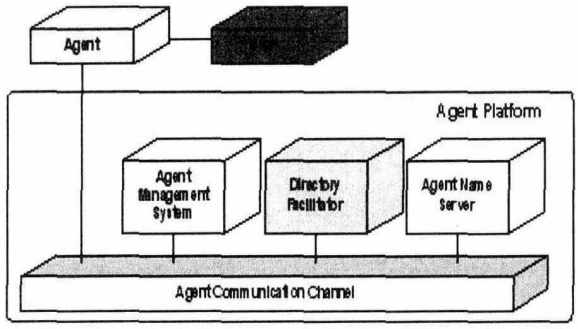
기존의 통신방법으로는 P2P나 RPC와 같은 방식이 있는데 P2P의 경우 PC간의 통신에 맞추어져 있기 때문에 에이전트 네트워크에 적용하기 어려우며, RPC의 경우 구성은 쉽다는 장점이 있지만 네트워크가 끊겼을 경우 문제가 발생하여 이 역시 에이전트 네트워크에는 적합하지 않다. 또한 에이전트 네트워크를 구성하는 방법으로 JXTA[1]를 활용할 경우 에이전트 플랫폼이 증가할수록 탐색시간이 길어지며, 이로 인하여 에이전트 네트워크의 전체 성능저하가 발생한다. [2]

이에 대한 개선안으로 본 논문에서는 ROOT-AP와 TERMINAL-AP라는 개념을 제시하여 에이전트, 해시테이블정보 등을 관리하도록 하였고, TERMINAL-AP는 새로운 AP가 생성될 때 마다 변경되어 ROOT-AP에 생기는 부하를 나누었다. 또한 패스트리라우팅[3] 알고리즘을 이용하여 애드버타이저먼트, 디스커버리를 가능하게 하였고 ROOT-AP의 백업역할, 라우팅 효율화 등의 부가적인 성능향상 효과도 가져왔다. 이와 같은 방법으로 네트워크를 효율적으로 구성하기 위한 자가 구성방법에 대하여 제안한다.[4]

본 논문의 구성은 다음과 같다. 2장에서는 에이전트와 에이전트 플랫폼에 관하여 알아보고, 3장에서는 에이전트 네트워크를 구성하데 활용될 수 있는 방식인 JXTA에 대하여 알아본다. 이어서 4장에서는 에이전트 네트워크 자가 구성방법을 소개하고, 마지막으로 5장에서 결론 및 향후 연구 과제에 관해서 제시하도록 한다.

## 2. 에이전트 플랫폼

에이전트란 사용자의 관점에서 시스템이 사용자를 대신해서 사용자가 원하는 작업을 자동적으로 수행해 주는 소프트웨어라 할 수 있다. 에이전트의 기본적인 특징으로는 자율성, 지능성, 이동성이 있고 이외에도 환경변화에 대해 반응할 수 있는 반응성, 잘못된 정보를 주고받지 않도록 하는 정직성, 에이전트의 활동을 합리적인 방법으로 목적을 달성 할 수 있도록 하는 합리성 등이 있다. 그리고 하나의 에이전트로 해결하지 못하는 작업을 다양한 에이전트간의 협업을 통하여 처리할 수 있도록 하는 멀티 에이전트 구조가 있다. 이런 특징들을 효율적으로 지원하기 위해 에이전트 플랫폼을 사용하며 에이전트 플랫폼의 구조는 그림 1과 같다.

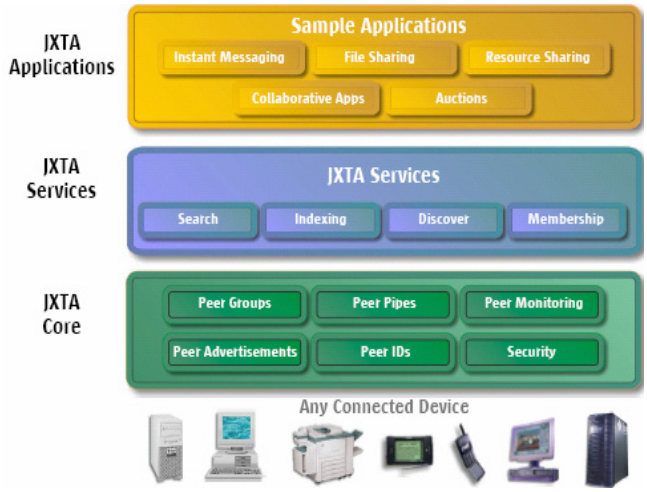



(그림 1) 에이전트 플랫폼의 구조

에이전트 플랫폼은 표준 기관인 FIPA[5]에서 제안하고 있는 구조이며 표준 에이전트 언어인 ACL(Agent Communication Language)을 사용하여 ACC(Agent Communication Channel)을 통하여 에이전트와 에이전트 플랫폼간의 통신을 지원한다. DF(Directory Facilitator)는 에이전트가 제공하는 서비스에 대한 정보를 제공하고 AMS(Agent Management Service)는 에이전트 등록과 제거, 정지와 회복 등 전반적인 에이전트 주기를 관리한다.

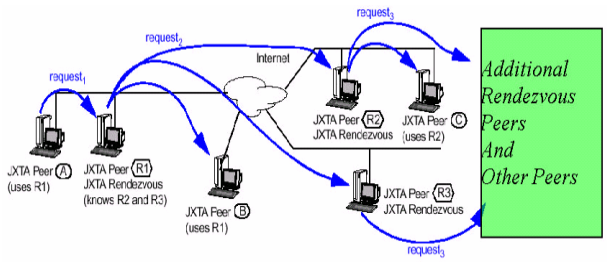
## 3. JXTA

JXTA는 P2P 네트워크 플랫폼이며, "Juxtapose"의 줄임 말로서 네트워크 프로그래밍과 분산 컴퓨팅 환경을 P2P 기반으로 제공하는 공동 연구 프로젝트이다. 썬마이크로시스템즈는 초기 많은 P2P 애플리케이션들이 갖는 공통적인 아키텍처를 분석해서 개념적인 일반적 계층 구조를 발견했다. 그리고 이러한 계층 구조를 바탕으로 그림 2와 같은 소프트웨어 계층 구조도를 작성했다. Core layer에서는 기본적인 동작에 관한 정보를 담고 있으며 peer를 생성하기 위해 unique한 peerID를 제공한며 다른 peer나 peer group을 위한 pipe를 생성하는 기능을 담당한다. Service layer에서는 CMS(Content Management System)나 JXTA peer들 간의 파일공유와 같은 일반적인 기능을 담당한다. Service layer에서는 CMS(Content Management System)나 JXTA peer들 간의 파일공유와 같은 일반적인 기능을 담당한다. Application layer에서는 사용자가 제어 할 수 있는 실제적이고 다양한 어플리케이션을 제공하는 기능을 담당한다.[6,7]



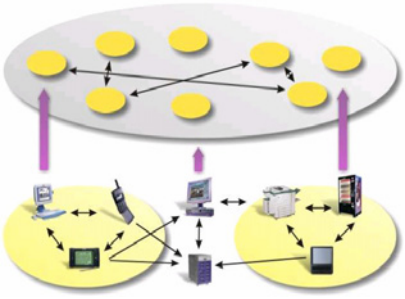

(그림 2) JXTA 소프트웨어 구조

JXTA의 동작과정은 그림 3과 같은 과정을 거친다. 각 peer를 생성하기 위하여 core layer에서 peer ID를 생성한다. peer는 연결 및 활성화되어 있는 동안 advertisements message를 통하여 자신의 존재여부를 rendezvous peer에게 알리게 된다. 확인된 peer는 rendezvous peer에 저장된다.




(그림 3) JXTA 동작과정

이런 과정을 거쳐 virtual network 이 생성되며 생성된 네트워크는 그림4와 같은 형태가 된다.[8,9]



(그림 4) JXTA virtual network

## 4. 제안하는 에이전트 네트워크의 자가 구성

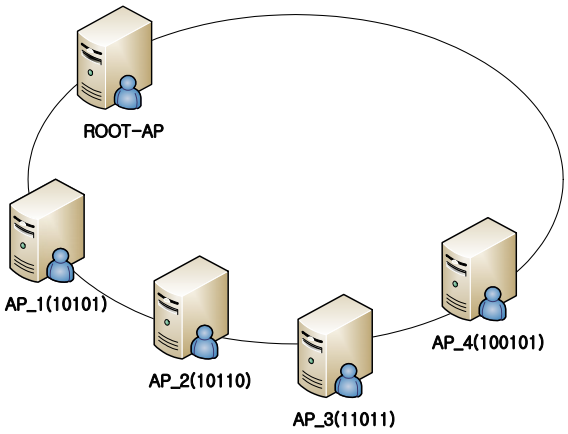
제안하는 에이전트 네트워크 자가 구성방법은 에이전트 플랫폼의 시작부터 최종 네트워크가 구성되기까지 전체의 과정을 단계적으로 살펴본다. 단, 여기서 사용되는 용어는 편의상 JXTA와의 구분을 위해 지은 이름이다.

그림 5와 같이 처음 생성되는 AP(Agent Platform)는 ROOT-AP가 되며 다음부터 생성되는 AP들은 ROOT-AP에게 라우팅 정보, hash value을 포함한 자신의 정보를 등록한다. Hash value(본문에서는 6bits기준으로 한다.)는 패스트리 라우팅 방식에 쓰이며 ID, 포트(AP@IP) 등을 ROOT AP에게 알려준다.


(그림 5) AP 등록과정

이 과정에서 새로 생성된 AP는 기본적으로 ROOT-AP를 찾게 되며, 특정시간(user-input) 동안 찾지 못하게 되면 본인을 ROOT-AP라 판단한다. 이때 만약, 새로 생성된 AP가 ROOT-AP 검색에 실패하여 본인을 ROOT-AP라 간주하고 얼마 후 ROOT-AP를 찾게 되면 본인의 정보를 ROOT-AP에게 전달한다. 그림 5의 경우를 예를 들어보자. 새로 생성된 AP_1은 초기 ROOT-AP와의 통신이 원활하지 못하여 본인이 ROOT-AP라고 가정하고 일정 시간 후 통신이 원활하여 원래의 ROOT-AP를 찾게 되면 본인의 정보를 원래의 ROOT-AP에게 전달하고 일반 AP로 돌아간다.

ROOT-AP를 기준으로 다음부터 생성되는 AP들은 초기 본인이 ROOT-AP가 될 것인지, 일반 AP가 될 것인지를 판단하여 에이전트 네트워크를 구성하게 된다.

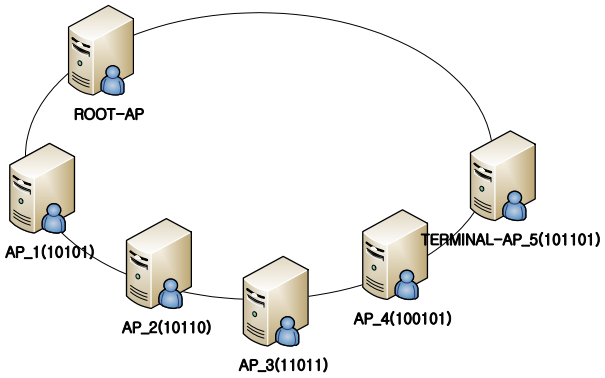



(그림 6) 최초 구성된 에이전트 네트워크

이런 절차로 구성된 에이전트 네트워크가 그림 6이며 AP_1 ~ AP_4는 컨텍스트 별로 라우팅 테이블을 형성한다. 에이전트들은 AP를 통하여 특정 에이전트와 통신이 가능하다.

FIPA spec에 의하여 애드버타이즈먼트, 디스코버리를 할 경우 패스트리 라이팅을 적용한다. 이때, 각 에이전트 플랫폼에서 DF(본 논문에서는 각 에이전트 플랫폼에 1개의 DF만 존재한다고 가정한다.)라는 에이전트가 존재하며 DF를 통해서 어떤 에이전트에서 어떤 서비스를 제공하지는 알 수 있다. 에이전트나 에이전트 네트워크 탐색 시 에이전트는 에이전트 플랫폼의 DF에게 먼저 쿼리를 보낸다. 쿼리를 받은 DF는 다른 DF에게 전달할지를 결정하며 최대의 횟수는 FIPA에서 정한 규정을 따른다.

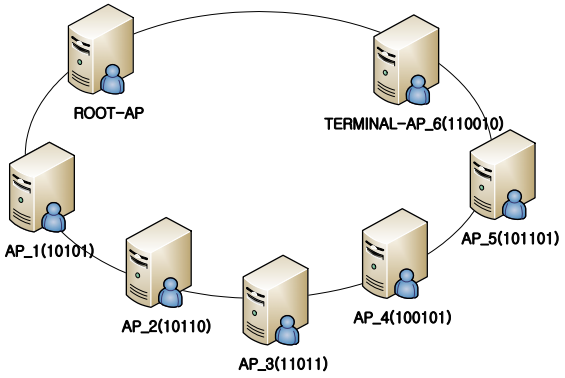
하지만, 그림 6과 같이 생성된 에이전트 네트워크의 경우 그림 5에서 처음 생성된 AP_1은 나중에 생성된 AP_2, AP_3, AP_4의 정보를 알 수 없는 문제가 발생한다. 따라서, 마지막에 등록한 AP를 TERMINAL-AP라 칭하고 이 TERMINAL-AP는 ROOT-AP에게 라우팅 정보를 포함한 자신의 정보를 등록하며 이미 등록된 AP들의 정보를 받게 된다. 이 정보를 토대로 나머지 AP에게 자신의 존재를 알리게 된다.




(그림 7) TERMINAL-AP가 생성된 에이전트 네트워크

그림 7의 예를 들어보자. 마지막에 생성된 AP는 ROOT-AP를 찾게 되고 ROOT-AP를 찾게 되면, ROOT-AP에게 자신의 정보를 등록한다. ROOT-AP에서는 새로 추가된 AP가 TERMINAL-AP임을 알게 되고 기존에 등록되어 있던 AP_1 ~ AP_4까지의 정보를 전송하여 준다. ROOT-AP로부터 자신이 TERMINAL-AP임일 알게 된 AP는 자신의 존재를 모르고 있는 AP_1 ~ AP_4들을 알게 되고 자신의 존재를 알린다. 이때 그림8과 같이 다른 하나의 AP(TERMINAL-AP)가 추가로 등록 되면 앞에서의 과정에서 본 것처럼 ROOT-AP는 TERRMINAL-AP를 정하고 이를 기존의 TERMINAL-AP(AP_5)에게 전달한다. 기존의TERMINAL-AP(AP_5)는 새로운 TERMINAL-AP가 생성된 것을 알게 되고 자신의 정보를 포함한 기존 AP(AP_1 ~ AP_4)의 정보를 알려주고 자신은 일반 AP(AP_5)가 되며 기존의 정보를 삭제한다.

마지막으로 생성된 TERMINAL-AP는 앞에서의 과정을 반복하게 된다.
지금까지의 과정에서 본 것처럼 최초의 AP는 ROOT-AP가 되고 최후의 AP는 TERMINAL-AP가 되며 ROOT-AP와 TERMINAL-AP는 특정시간(user-input)을 기준으로 서로 동기를 맞추어 AP들의 정보를 갱신한다. 이런 갱신을 통하여 ROOT-AP가 통신이 끊어질 경우 TERMINAL-AP는 ROOT-AP가 통신이 원활할 때까지 임시 ROOT-AP 역할을 한다.




(그림 8) 최종 생성된 에이전트 네트워크

## 5. 결론 및 향후 연구과제

지금까지 에이전트 플랫폼과 에이전트 네트워크를 구성하는 대표적인 방법인 JXTA를 알아보았고 JXTA의 문제제기와 이의 개선안으로 에이전트 네트워크 자가구성을 제시하였다. 본 논문에서 제안한 에이전트 네트워크 자가 구성방법은 에이전트 네트워크를 어떻게 효율적으로 구성할 수 있는지에 대한 개선방향이다.

기존 JXTA의 문제점으로 제시한 에이전트 플랫폼이 증가할수록 탐색시간이 길어져 발생하는 성능저하와 특정 에이전트에게 트래픽이 집중되는 경우 대처방안을 ROOT-AP와 TERMINAL-AP를 만들어 개선하였다. 이외에 패스트리 라우팅 알고리즘을 이용하여 애드버타이저먼트, 디스커버리를 가능하게 하였다. 또한, ROOT-AP의 백업역할, 라우팅 효율화 등의 부가적인 성능향상 효과도 가져왔다.

향후 과제로서 아직까지 사용자 입력이 필요한 ROOT-AP를 판단하는 시간, ROOT-AP와 TERMINAL-AP의 동기화 시간 등 아직까지 사람의 판단에 의존하는 부분이 존재한다. 주기나 양에 관하여 유동적으로 대처할 수 있는 방안에 대한 연구를 할 예정이다.

## 참고문헌

[1] Bernard Traversat, Mohamed Abdelaziz, Mike Duigou, Jean-Christophe Hugly, Eric Pouyoul, Bill Yeager,, "Project JXTA Virtual Network
[2] Qusay H. Mahmoud, "Middleware for Communications", 2004
[3] A. Rowstron and P.Druschel “Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November 2001.
[4] Gong Li, "JXTA : A Network Programming Environment", IEEE Internet Computing, 2001
[5] FIPA, Foundation for Intelligent Phisical Agents, Fipa domains and policies specification, www.fipa.org
[6] JXTA KOREA, www.jxtakorea.net, 2002
[7] Sun Microsystems, "Project JXTA v2.0:Java Programmer''s Guide", www.jxta.org, 2003.
[8] Jin Soo Han, Tae Hyung Kim, "Connecting heterogeneous P2P systems using JXTA“, KIISE, pp.790~792, 2004
[9] Sun Microsystems, "JXTA v2.0 Protocols specification", www.jaxt.org, 2004