멀티 에이전트 시스템의 에이전트 플랫폼 간의 에이전트 탐색 단축 기법

정준희*, 윤희용*
*성균관대학교 정보통신공학부
e-mail:jhbebop@naver.com, youn@ece.skku.ac.kr

A Method Reducing Agent Search between Agent Platforms in Multi-Agent System

Joon Hee Chung*, Hee Yong Youn*
*School of Information and Communication Engineering, SungKyunKwan University

요 약

유비쿼터스 시대를 맞이하여 기존의 클라이언트-서버 모델을 벗어나 분산된 자원을 활용하는 방법이 연구 중인데 그 중 대표적인 것이 멀티 에이전트 시스템이다. 그러나 멀티 에이전트 시스템은 에이전트를 탐색할 때 깊이 우선 탐색 기법을 사용하는데 이는 에이전트 시스템과 맞지 않는 면이 있어 멀티 에이전트 시스템을 분석하여 탐색을 단축하는 기법을 제안하고자 한다. 본 논문에서는 에이전트 시스템의 에이전트 탐색에 매니저 DF를 두어 확률적으로 탐색확률이 높은 에이전트 플랫폼을 우선탐색함으로써 에이전트 탐색 성능을 향상시킬 수 있다. 또, 매니저 DF에 네트워크 장애 등의 사고를 대비하여 미리 인접한 DF에 자료를 백업해두어 장애 발생 시에도 백업해 둔 자료를 활용하여 빠르게 그 업무를 대체할 수 있도록 하였다.

1. 서론

지금까지 컴퓨터간의 네트워크를 구성하는 가장 일반적인 방법은 클라이언트-서버 모델이었지만 다수의 클라이언트가 소수의 서버에 접속하여 서비스를 제공받았기에 서버가 갖추어야할 처리 능력과 네트워크 대역폭, 저장 공간은 기하급수적으로 증가하였다. 따라서 최근에는 분산되어 있는 다양한 자원을 활용하는 방법이 연구 중인데 그 중 대표적인 것이 멀티 에이전트 시스템이다.

멀티 에이전트 시스템은 자율적이고 지능적인 여럿의 에이전트들이 협업을 통해 일을 한다. 현재 Foundation for Intelligent Physical Agent(FIPA)[1]에서 제안하는 에이전트 플랫폼에서 에이전트는 AMS(Agent Management System)에 의해 관리되고, DF(Directory Facilitator)는 옐로우 페이지로서 협업할 에이전트를 찾아주는 역할을 한다. 에이전트들끼리 협업을 하기 위해서는 먼저 수많은 에이전트들 중에서 원하는 서비스를 제공하는 에이전트를 효과적으로 검색하는 것이 중요한데, 이는 바로 DF의 역할이다. DF는 주로 해당 에이전트 플랫폼 안에 속해있는 에이전트를 탐색하는데, 만약 해당 에이전트 플랫폼 내에 원하는 에이전트가 존재하지 않고, 에이전트 플랫폼 외부까지의 탐색 요청이 있다면 다른 에이전트 플랫폼의 DF와 연합을 이루어 그 검색영역을 넓힌다[2].

하지만 현재 FIPA에서 제안하는 멀티 에이전트 시스템의 DF는 다른 에이전트 플랫폼의 DF를 검색할 때 깊이 우선 탐색 기법을 사용한다[1]. 깊이우선탐색 기법은 노드(에이전트 시스템에서는 DF)를 탐색하는 데는 훌륭한 기법이긴 하지만 에이전트를 검색하는데 있어 찾고자 하는 에이전트가 존재할만한 확률을 고려하지 않고 모든 에이전트 플랫폼을 동일하게 보고 검색을 하기 때문에 그리 효율적이지 않다. 또, 깊이우선탐색을 통한 검색결과는 해당 에이전트까지의 최단경로를 보장하지 못한다는 문제점이 있다.

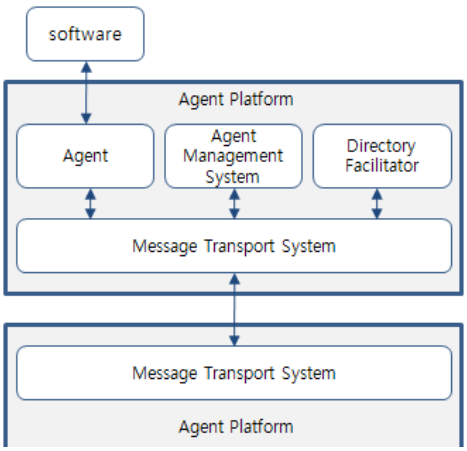
본 논문에서는 위의 문제점들을 해결하기 위해 멀티 에이전트 시스템 간의 에이전트 탐색을 단축하는 기법을 제안한다. 하나의 에이전트 플랫폼에서 다른 에이전트 플랫폼의 에이전트를 검색할 때 모든 DF를 동일하게 보는 깊이 우선 탐색 기법에 의한 순차적인 탐색이 아니라, 매니저 DF를 두어 다른 DF들의 정보를 갖고 있어 DF들에게 우선순위를 두어 탐색될 확률이 높은 DF부터 검색하도록 하여 검색성능을 향상시키고자 한다. 또한, 평상시 매니저 DF의 자료를 다른 DF에 미리 백업해 두어 매니저 DF에 장애가 발생할 시에도 백업해 둔 자료를 활용하여 다른 DF가 빠르게 그 업무를 대체할 수 있도록 하는 장치를 해 두었다.

본 논문의 구성은 다음과 같다. 2 장에서 멀티 에이전트 시스템의 기본구조, 특징과 깊이 우선 탐색 알고리즘에 대해 분석한다. 3 장에서는 DF의 탐색 기법에서 발생하는 문제점과 이에 대한 해결책에 대해 언급한다. 끝으로 4 장은 논문의 결론을 담고 있다.

2. 관련 연구

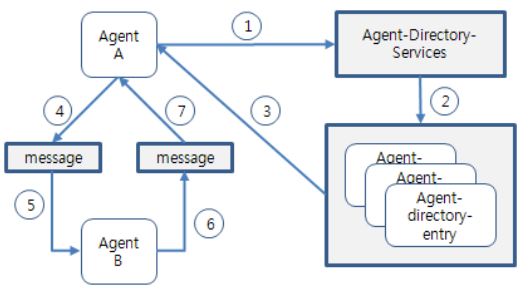
2.1 멀티 에이전트 시스템

멀티 에이전트 시스템은 에이전트 혼자만의 능력이나 지식으로는 해결하기 힘든 문제를 다른 에이전트들과 상호 협력하여 문제를 풀어나가는 시스템으로 각 에이전트의 목표와 문제 해결 상황의 차이에 따라 합의, 교섭, 설득, 경합 등의 프로세스를 통해 문제 해결을 추구하는 시스템이다[3]. FIPA에서 제안하는 에이전트 시스템의 기본 구조는 아래 그림 1과 같다.



(그림 1) Agent System의 기본 구조

그림 1에서 보이는 바와 같이 에이전트는 에이전트 플랫폼 위에 생성되고, 하나의 에이전트 플랫폼 위에는 하나 이상의 에이전트들이 존재한다. 에이전트는 FIPA에서 제안한 ACL(Agent Communication Language)를 이용해서 메시지를 주고받는다. AMS(Agent Management System)는 에이전트로서 에이전트 플랫폼의 제어와 사용을 관리하는 역할을 한다. 하나의 에이전트 플랫폼에는 오직 하나의 AMS만이 존재한다. DF(Directory Facilitator) 또한 하나의 에이전트로서 옐로우 페이지 역할을 한다. DF는 에이전트 플랫폼 내의 에이전트의 정보를 갖고 있고, 다른 에이전트 플랫폼의 DF와 연계를 통해 협업할 에이전트를 찾는데 사용된다[4].



(그림 2) Agent System의 기본 동작

그림 2를 통해 에이전트의 기본 동작을 간단하게 알아볼 수 있다[5]. 먼저 에이전트 A가 Agent directory service에게 찾고자 하는 에이전트 검색을 의뢰하면, 에이전트 플랫폼 내에 등록되어 있는 에이전트 중에서 가장 적합한 에이전트를 Agent Directory Entry에서 찾아, 그 결과인 에이전트 B를 리턴 한다. 에이전트 A는 HTTP 형식이나 SMTP 형식으로 검색된 에이전트 B와 통신을 하게 되며 메시지를 주고받으며 작업을 수행한다.

멀티 에이전트 시스템에서 DF를 통해 에이전트를 검색하는데, 만약 같은 에이전트 플랫폼 내에 원하는 에이전트가 존재하지 않는 경우, 다른 플랫폼의 에이전트를 검색하게 된다. DF는 타 에이전트 플랫폼의 DF를 등록해두고 검색요청이 들어오면 깊이 우선 탐색 기법을 통해 타 에이전트 플랫폼의 DF를 검색한다.

2.2 깊이 우선 탐색 알고리즘

깊이 우선 탐색 알고리즘[6]은 출발 노드로부터 시작하여 노드를 계속적으로 확장하고, 가장 최근에 생성된 노드를 먼저 확장시키는 탐색 기법이다. 만약 후계 노드가 없다든지, 어떤 노드의 모든 후계 노드를 방문했지만 목표 노드를 발견하지 못하였을 경우에는 즉각적으로 바로 직전 노드로 돌아간다. 경우에 따라서는 해가 없는 경로를 계속해서 따라갈 수 있으므로, 필요에 따라 상위 노드로 되돌아가는 백트랙킹(backtracking)을 할 수 있는 방안을 마련해야 한다. 일반적으로 적당한 깊이 제한을 두어 어떤 노드가 그 이상의 깊이를 갖게 되면 이 노드를 더 이상 확장시키지 않고, 깊이 제한을 넘지 않는 것 중에서 가장 깊은(즉, 가장 최근에 생성된) 노드를 골라서 확장한다.

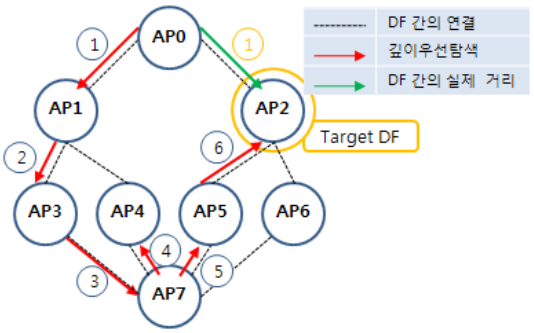
깊이 우선 탐색 기법의 장점으로는 신속하게 원하는 목표를 찾을 수 있으며 구현하기가 용이하며 많은 가지들을 가지고 있는 상태 공간에서 매우 유용하다.

3. 에이전트 탐색 의 문제점과 단축 기법

위에서 말한 바와 같이 현재 멀티 에이전트 시스템의 DF는 타 플랫폼의 DF를 검색할 때 깊이우선탐색 기법을 사용한다. 이 기법을 설명하기 위해 실제로는 피지컬한 네트워크를 논리적인 네트워크로 가정한다. 깊이 우선 탐색 기법에는 몇 가지 단점이 존재하는데 그 중 하나인 “원하는 결과가 없는 경로로 계속해서 탐색할 수 있다”는 점은 에이전트가 검색을 요청할 때부터 원하는 깊이(max-depth)를 지정함으로써 해결될 수 있었다[4].

하지만 깊이 우선 탐색 기법이 노드(에이전트 시스템에서는 DF)를 탐색하는데 효율적인 기법이지만 에이전트를 검색을 하는데 있어 목표로 하고 있는 에이전트가 존재할만한 확률을 고려하지 않고 모든 에이전트 플랫폼을 동일하게 보고 검색을 하기 때문에 탐색 시간이 오래 걸리게 된다. 아래 그림 4와 같은 멀티 에이전트 시스템이 있을 때 깊이 우선 탐색 기법 사용 시 목표로 하는 에이전트 플랫폼의 DF가 바로 연결되어 있음(1번의 탐색)에도

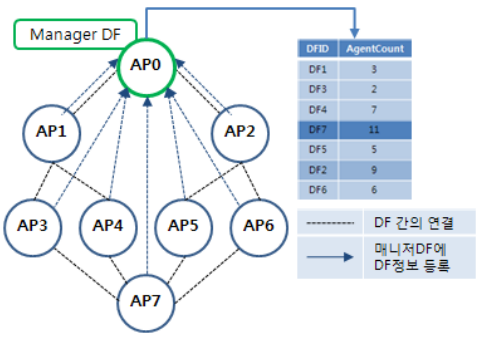
불구하고 6번의 탐색 끝에 찾게 되는 단점이 있다.



(그림 4) 깊이 우선 탐색 기법의 단점

위에서 언급한 바와 같이 멀티 에이전트 시스템의 에이전트 검색은 다른 에이전트 플랫폼의 DF를 검색 시 깊이 우선 탐색 기법을 사용하지만, 이는 훌륭한 탐색 기법임에도 불구하고 에이전트 검색에는 맞지 않는 모습을 보인다. 이에 한 에이전트 플랫폼의 DF가 매니저 역할을 하여 다른 에이전트 플랫폼 DF의 정보를 갖는다. 이 정보를 바탕으로 탐색에 우선순위를 두어 탐색될 확률이 높은 DF부터 검색하도록 하여 탐색 시간을 단축시키는 기법을 제안하고자 한다.

제안된 기법에서는 에이전트 플랫폼을 검색하기 위해 다른 DF를 검색하는 기법을 깊이 우선 탐색이 아닌 매니저 DF를 두어 다른 DF들의 정보를 관리하도록 설계하였다.



(그림 5) 매니저 DF를 적용한 시스템 구조

먼저 처음으로 만들어진 에이전트 플랫폼의 DF가 매니저 DF의 역할을 한다. 그 후에 연결되는 DF는 매니저 DF에 자신의 ID와 소유하고 있는 에이전트의 수를 등록한다. 에이전트들이 DF에게 자신을 등록하는 것과 마찬가지로 DF 또한 자신의 정보(ID, 소속 에이전트의 수)를 매니저 DF에게 등록한다. 매니저 DF와 직접 연결되어 있지 않은 DF도 매니저 DF와 접해있는 DF를 통해 연결을 생성하고 자신을 등록한다. 에이전트의 정보가 변경되는 경우 DF는 실시간으로 매니저 DF에 변경내용을 업데이트한다. 매니저 DF의 필드와 그 속성은 다음과 같다.

<표 1> 매니저 DF의 필드와 속성

필드	속성
DFID	DF의 고유ID(유일하다)
AgentCount	DF가 소유하고 있는 에이전트의 수

매니저 DF는 다른 DF가 소유하고 있는 에이전트의 정보까지 갖지는 않는다. 이는 만약 매니저 DF가 에이전트의 정보를 모두 갖게 된다면 서버의 역할을 하는 것과 다름없어 과중한 네트워크 트래픽, 처리능력, 저장 공간 등을 요구하게 되기 때문이다. 그런 반면 DF가 소유하고 있는 에이전트의 수를 매니저 DF의 필드 값으로 하는 이유는 매니저 DF가 요청을 받아 탐색할 에이전트 플랫폼의 DF를 선택할 때 소유하고 있는 에이전트의 수에 따라 우선순위를 부여하기 위해서다.

에이전트가 협업할 에이전트의 탐색을 DF에 요청하고 DF가 자신의 에이전트 플랫폼 내와 다른 DF를 통해 에이전트를 탐색하는 것은 기존의 방식과 동일하다. 하지만 다른 에이전트 플랫폼의 DF를 탐색할 때는 요청을 매니저 DF에게 전송한다.

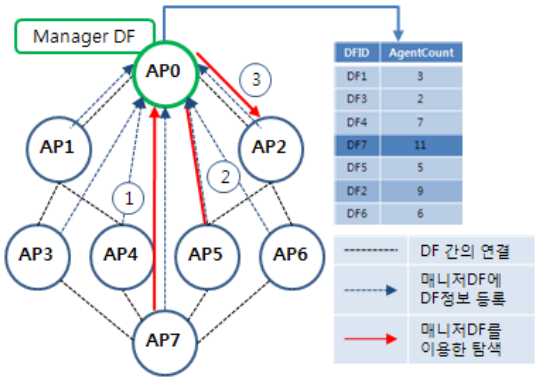
매니저 DF는 자신이 갖고 있는 모든 DF의 정보를 이용해 요청을 적절한 DF에게 전달한다. 이때 요청을 전달할 DF의 선택 기준은

(1) 이미 검색했었던 DF를 제외하고,
(2) 가장 많은 수의 에이전트를 소유하고 있는 DF

이다. 이는 에이전트를 많이 소유한 DF일수록 원하는 에이전트가 있을 확률이 높기 때문이다. 선택된 DF에서 원하는 에이전트를 찾지 못한 경우, 선택 기준에 의거 차순의 DF를 선택하고 이런 과정을 반복하여 에이전트를 찾을 때까지 반복한다.

예를 들어 위의 그림 6과 같이 에이전트 플랫폼 1~7의 DF는 매니저 DF(에이전트 플랫폼 0)에게 자신의 정보(DF의 ID, 소유한 에이전트의 수)를 전달한다. 에이전트 플랫폼 7의 에이전트 A가 에이전트 플랫폼 2의 에이전트 B를 필요로 할 경우 에이전트 A는 자신의 에이전트 플랫폼의 DF에게 협업에 필요한 에이전트를 요청한다. DF는 자신이 갖고 있는 동일 에이전트 플랫폼 내의 에이전트를 검색해보고 일치하는 에이전트가 없기에 매니저 DF에 요청을 전송한다. 매니저 DF가 갖고 있는 DF들의 정보 중 이미 검색해 본 7을 제외한 나머지 중 가장 많은 에이전트를 소유하고 있는 에이전트 플랫폼 3의 DF를 검색한다. 하지만 에이전트 플랫폼 3에도 일치하는 에이전트가 없기에 그 다음 우선순위의 에이전트 플랫폼 2의 DF를 검색한다. 에이전트 플랫폼 2의 DF에서 원하는 에이전트(에이전트 B)를 찾게 된다. DF는 에이전트 B에게 요청을 전달

하고, 에이전트 B는 에이전트 A에게 메시지를 보낸다.

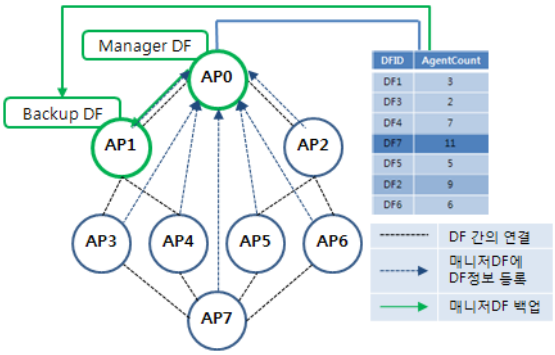


(그림 6) 매니저 DF를 적용한 시스템에서의 에이전트 탐색

위와 같이 매니저 DF를 사용함으로써 "모든 DF의 id와 소유한 에이전트 수"만으로도 에이전트 탐색을 단축하게 된다.

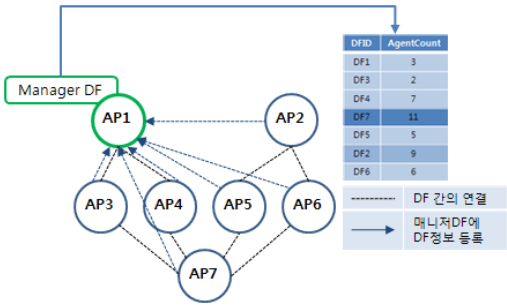
위에서 설명한 매니저 DF를 활용한 시스템에서 매니저 DF는 핵심적인 역할을 하지만 만약 매니저 DF에 네트워크 장애 등의 사고가 발생하는 경우가 생긴다면 해당 시스템에서는 에이전트 탐색에 바로 어려움을 느끼게 될 것이다.

이런 어려움을 해결하기 위해서 아래 그림 7과 같이 매니저 DF에 인접한 DF에 미리 매니저 DF가 갖고 있는 다른 DF들의 정보를 복사해 놓는다.



(그림 7) 매니저 DF의 백업

그래서 만약 매니저 DF에 장애가 발생하더라도 매니저 DF의 정보를 미리 복사해서 갖고 있던 DF가 매니저 DF가 되어 그 역할을 대신한다(그림 8). 이렇게 함으로써 장애가 발생하더라도 해당 네트워크 시스템은 무리 없이 제 역할을 수행할 수 있도록 유지한다.



(그림 8) 사고 발생 시 새로운 매니저 DF

4. 결론

본 논문에서는 FIPA에서 제안하는 멀티 에이전트 시스템에서 원하는 에이전트를 탐색하는 알고리즘(깊이 우선 탐색)과 해당 알고리즘에서 유발되는 문제점을 알아보고, 매니저 DF를 활용하여 멀티 에이전트 시스템을 구축하여 에이전트 탐색을 단축시키는 방법에 대해 다루었다.

기존의 깊이우선탐색 기법을 이용한 에이전트 탐색은 목표로 하고 있는 에이전트가 존재할만한 확률을 고려하지 않고 모든 에이전트 플랫폼을 동일하게 보고 검색을 하기 때문에 탐색 시간이 커지게 된다.

이런 문제점을 해결하기 위해 본 논문에서는 매니저 DF를 두어 에이전트 탐색을 단축시키는 기법을 제안하였다. 다른 에이전트 플랫폼 검색 시 매니저 DF를 활용하여 에이전트 플랫폼이 우선순위를 두어 검색함으로써 목표로 하는 에이전트를 찾을 확률이 높은 DF를 먼저 검색하여 에이전트 탐색을 단축시켰다. 향후 연구 과제로 본 논문이 제안한 매니저 DF의 검색 효율을 더 높일 수 있는 방안에 대한 구체적인 분석이 필요하다.

참고문헌

[1] FIPA (Foundation for Intelligent Physical Agents), web resource available at http://www.fipa.org/

[2] Dan W. Patterson, "Introduction to Artificial Intelligence and Expert Systems", 지성출판사, 1990

[3] OHare, G. and Jennings, N. "Foundations of Distributed Artificial Intelligence", John Wiley & Sons, 1996

[4] M. Wooldridge, "An Introduction to Multiagent Systems", John Wiley & Sons. 2002

[5] Weiss, Gerhard, "Multiagent systems : a modern approach to distributed artificial intelligence", MIT Press, 1999

[6] M. Luck, P. McBurney and C. Preist, "Agent Technol-ogy: Enabling Next Generation Computing", AgentLink community, 2003