Jini 기반 분산 환경에서의 이동 에이전트 시스템

°최 석 환*, 김 효 남**, 원 유 헌*
홍익대학교 컴퓨터공학과*
청강문화산업대학 컴퓨터소프트웨어과**
{shchoi, won}@cs.hongik.ac.kr*, hnkim@mail.chungkang.ac.kr**

Mobile Agent System of Distributed Environment Based on Jini

°Suk-Hown Choi*, Hyo-Nam Kim**, Yoo-Hun Won*
Department of Computer Engineering, Hong-Ik University*
Department of Computer Software, Chung Kang College of Cultural Industries**

요 약

컴퓨터 환경이 집중화에서 분산화로 변화함에 따라 새로운 분야들이 나타나게 되었다. 이러한 것에는 Jini, 이동 에이전트 등과 같은 것들이 있고, 이 두 분야는 신기술을 채용했지만 Jini는 서비스에 부하를 많이 준다는 점에서, 이동 에이전트는 정해진 경로만 이동하거나, 새로운 경로의 추가와 제거가 쉽지 않다는 점에서 문제점을 가지고 있다. 따라서 이 논문에서는 Jini 기반의 이동 에이전트를 설계하여 동적으로 경로를 설정하고, 서비스에 부하를 덜 주면서 클라이언트에게 서비스를 제공하는 시스템을 제안한다.

1. 서 론

컴퓨터가 나타나기 시작한 후부터 이것을 활용하고 발전시키기 위한 많은 연구가 진행되어 왔다. 초기에는 대형 메인 프레임 컴퓨터를 두고 터미널을 이용한 작업을 하였고, 이 때의 작업 방식은 대형 컴퓨터를 중심으로 한 집중형 작업이었다. 그러나 이러한 방식은 대형 컴퓨터에 많은 부하를 주게 되었고, 컴퓨터의 특성상 많은 비용이 들게 되었다. 그래서 이러한 문제점을 해결하기 위해 나온 방식이 각 부서별 미니컴퓨터를 두거나 LAN을 이용하여 각 컴퓨터들을 연결하는 방식이었다. 또한 개인용 컴퓨터의 발전으로 인하여 시스템 환경이 기존의 집중형에서 분산형으로 바뀌게 되었다. 그러나 이러한 분산형 시스템들은 각자가 자신의 시스템을 관리해야 했으므로 지나치게 높은 소유 비용(TCO :Total Cost of Ownership)이 발생하게 되었다. 한편 네트워크의 발전으로 인해 많은 시스템들이 네트워크에 연결되게 되었고, 인터넷의 폭발적인 성장으로 인해 기존의 프로그래밍적 패러다임이 변화를 요구받게 되었다. 그래서 분산환경에 알맞은 Jini, 이동 에이전트와 같은 신기술이 나오게 되었다. 이러한 기술들을 이용하면 네트워크에 연결된 기기들에 대한 관리가 가능하므로 TCO를 낮출 수 있게 되었다[1,5]. 그러나 새로운 기술들은 독자적으로 발전하였기 때문에 다른 기술의 장점들을 받아들이는데 소홀하였고, 단점이 많이 부각되었다.

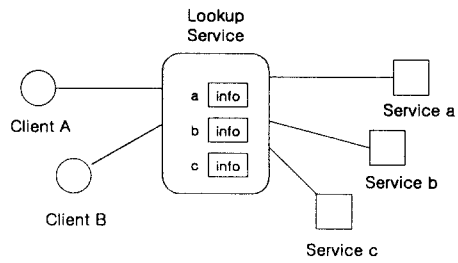
예를 들어 Jini와 같은 경우는 클라이언트가 자기가 원하는 서비스를 찾아서 서비스를 얻을 수 있다는 장점을 가지고 있지만 서비스에 서비스가 필요치 않는 클라이언트가 많이 몰릴 수 있다는 단점을 가지고 있다[4]. 또한 이동 에이전트는 자기가 이동해야 하는 경로를 미리 정해 놓고 가기 때문에 동적으로 구성하기 어렵다는 단점을 가지고 있었다[6].

본 논문에서는 이러한 단점을 개선시킬 수 있는 시스템을 제안한다. 논문의 1장은 서론이며, 2장은 연구 배경으로 Jini와 이동 에이전트를 소개하고 문제점을 제시한다. 3장은 이 논문에서 제안한 Jini 기반 이동 에이전트 시스템 구조를 설명하며, 4장은 이 시스템을 이용한 예로써 소프트웨어 관리 시스템 소개할 것이며, 그리고 5장은 결론 및 향후과제에 대해 서술한다.

2. 연구 배경

2.1 Jini

Jini는 Sun Microsystems 제안한 기술로서 기존의 분산환경에서 서비스를 필요로 하는 사용자가 자기의 기기를 네트워크에 접속하기만 하면 필요한 서비스를 받을 수 있는 기술이다(Plug & Work).

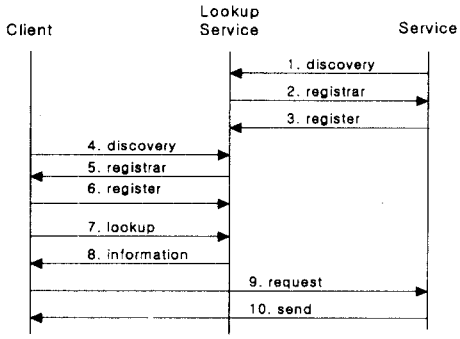


[그림 1] Jini 시스템

그림 1에서는 Jini 기술에 대한 대략적인 그림을 나타내고 있다. 여기서 클라이언트는 자바 가상 기계(Java Virtual Machine)를 가지고 있고 네트워크에 접속할 수

있는 컴퓨터, PDA, 셀룰러폰과 같은 기기를 말하고, 서비스는 클라이언트에게 서비스를 제공하는 역할을 하는 네트워크에 접속되어 있는 프린터 서비스, 파일 저장 서비스 등과 같은 것을 말한다. 또한 룩업 서비스는 클라이언트와 서비스를 연결하는 역할을 하는 것으로 서비스가 어디에 위치하고 있고, 속성은 무엇이며, 서비스를 이용하기 위해 어떻게 해야 하는지에 관한 정보를 가지고 있다[1].

그림 2는 Jini 시스템에서의 클라이언트, 룩업 서비스, 서비스와의 동작과정을 나타내고 있다.



[그림 2] Jini 시스템 동작과정

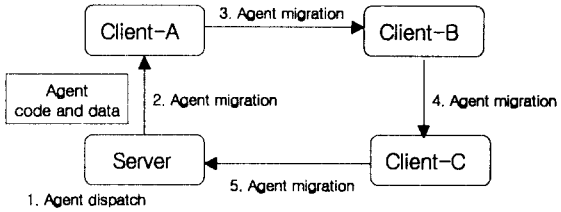
서비스는 먼저 룩업 서비스를 찾아야 한다. 여기서 서비스는 디스커버리와 조인프로토콜을 사용하여 룩업 서비스를 찾게 된다. 서비스가 룩업 서비스를 찾게 되면, 룩업 서비스에서는 서비스를 등록시킬 수 있는 Registrar 인터페이스를 서비스에게 건네주게 된다. 그러면 서비스는 이 인터페이스를 이용해서 자기의 속성이나 프록시를 집어넣은 후에 룩업 서비스에게 보내게 된다. 여기서 룩업 서비스는 서비스를 등록한다.

네트워크에 접속한 클라이언트는 자기에게 필요한 서비스를 찾기 위해서 먼저 룩업 서비스를 찾게된다. 여기서 디스커버리 프로토콜을 사용하여 찾고, 룩업 서비스는 클라이언트를 등록시키기 위해 Registrar 인터페이스를 건네준다. 클라이언트는 이 인터페이스를 통해 등록하고, 찾고자 하는 서비스에 관한 정보를 룩업 서비스에게 보낸다. 그러면 룩업 서비스는 자신의 데이터베이스를 검색하여 필요한 서비스에 대한 정보를 클라이언트에게 보낸다. 클라이언트는 이 정보를 가지고 서비스에 접근해서 필요한 서비스를 얻게 된다[2,3,4].

2.2 이동 에이전트

이동 에이전트는 분산 환경에서 사용자를 대신하여 주어진 문제의 해결을 위해 어떤 장소로 이동해야 하며, 어떠한 일을 해야 하는지를 스스로 결정할 수 있는 자율적인 소프트웨어 객체이다. 즉 이동 에이전트는 정해진 호스트를 순차적으로 방문하여, 각 플랫폼 위에서 정해진 작업을 수행 할 수 있는 역할을 한다.

그림 3은 이동 에이전트 시스템의 대략적인 구성을 보여준다. 서버와 모든 클라이언트들은 연결되어 있고, 서버는 에이전트를 활성화시키고 미리 정해진 경로에 의해 에이전트를 이동하게 한다. 그리고 에이전트는 경로를 이동하면서 주어진 일을 하게되며, 클라이언트는 이동 에이전트의 플랫폼을 제공한다[5].



[그림 3] 이동 에이전트 시스템

2.3 Jini와 이동 에이전트의 문제점

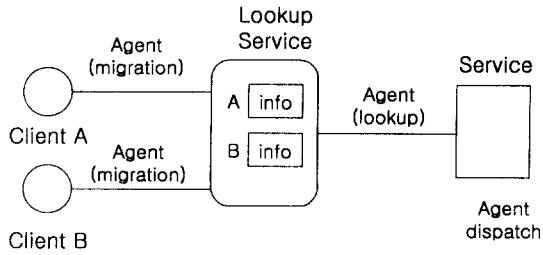
위의 두 기술은 분산 환경에 적합한 기술이지만, 문제점을 가지고 있다.

Jini에서 클라이언트는 자기가 서비스를 받아야 할지, 받지 말아야 할지 모를 경우가 있다. 그러나 기존의 Jini 방식에서는 받을 필요가 없을 경우라도, 룩업 서비스에서 서비스를 찾아 직접 접근에 의해 그것을 확인하게 된다. 여기서 많은 부하가 생긴다. 확인하는 과정을 거치지 않게 하기 위해서는 룩업 서비스에 많은 정보를 저장해야 하는데 이것은 룩업 서비스에게 많은 부담을 가중시킨다. 또한 서비스는 클라이언트에게 서비스를 제공하기 위해서는 룩업 서비스에 자신에 관한 정보를 남겨야만 한다. 그런데 서비스의 정보를 남겨둠으로 인하여 유효하지 않은 클라이언트가 이러한 정보를 가지고 오용할 수 있다.

이동 에이전트는 정해진 경로를 따라서 이동한다. 그런데 에이전트가 클라이언트의 상태를 알지 못하므로 기존의 경로로 이동하는데, 클라이언트가 꺼져 있을 때는 필요 없는 경로를 이동하게 되고, 클라이언트가 추가되었을 경우에는 이동해야 하는 경로를 이동하지 못하게 된다.

3. Jini 기반의 이동 에이전트 시스템

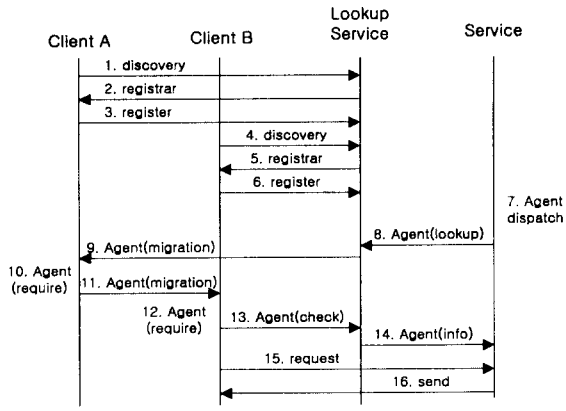
위의 문제점으로 인하여 이 논문에서는 Jini 기반 이동 에이전트를 설계하게 되었다.



[그림 4] Jini 기반 이동 에이전트 시스템

그림 4에서 보면 클라이언트 A, B가 룩업 서비스에 등록되어 있다. 여기서 룩업 서비스는 클라이언트 A, B에 관한 정보를 가지고 있고, 서비스는 에이전트를 만들 수 있다. 그리고 에이전트는 룩업 서비스에서 클라이언트

A, B에 관한 정보를 찾아서 클라이언트 A, B로 이동하게 된다.

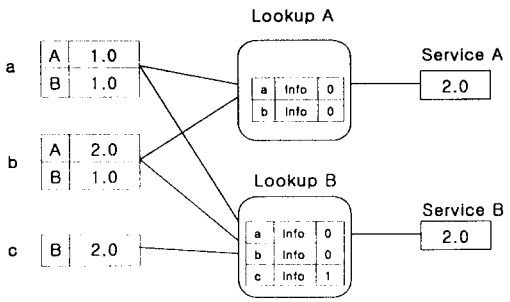


[그림 5] Jini 기반 이동 에이전트 동작과정

그림 5에서 클라이언트 A는 룩업 서비스에 등록시키기 위해 디스커버리 요청을 한다. 요청을 받은 룩업 서비스는 클라이언트에게 Registrar 인터페이스를 제공하고 클라이언트 A는 등록한다. 마찬가지 방식으로 클라이언트 B도 등록한다. 서비스는 클라이언트에게 전달할 서비스가 있으면 에이전트를 활성화하고 이동시킨다. 에이전트는 룩업 서비스에서 이동해야 할 클라이언트의 정보를 얻게 되고 에이전트는 먼저 클라이언트 A로 이동한다. 클라이언트 A로 이동한 후 서비스가 필요한 것인지를 확인 후 B로 이동한다. 그림 5에서는 클라이언트 A는 서비스를 필요로 하지 않고, B는 서비스를 필요로 한다. 그래서 에이전트는 클라이언트 B에게 서비스에 접근할 수 있는 정보를 주고 룩업 서비스로 이동한다. 여기서 에이전트가 이동한 곳을 표시한 후 서비스로 돌아와 이동하면서 얻은 정보를 서비스에게 제공한다. 한편 서비스가 필요한 B는 얻은 정보를 바탕으로 서비스에 요청하게 되고, 서비스는 클라이언트에게 정보를 제공하게 된다.

4. Jini 기반 이동 에이전트의 예

이 시스템을 바탕으로 여러 분야에 적용시킬 수가 있다. 여기서는 소프트웨어 관리 시스템을 예로 들기로 한다.



[그림 6] 소프트웨어 관리 시스템

그림 6에서는 소프트웨어 관리 시스템을 보여주고 있다.

컴퓨터 a, b는 A, B라는 소프트웨어를 c는 B라는 소프트웨어를 가지고 있다. 그리고 각 룩업 서비스에는 등록된 컴퓨터의 정보와 소프트웨어 새 버전이 나온 후 에이전트가 컴퓨터를 다녀갔는지 검사하는 필드가 있다. 그리고 서비스는 2.0버전을 가지고 있다. 컴퓨터 a, b, c를 켜자마자 Jini의 Plug&Work 기능에 의해 룩업 서비스에 등록된다. 서비스 A는 룩업 서비스에 등록된 사용자의 소프트웨어가 최신 버전인지를 확인하기 위해 에이전트를 보내고, 에이전트는 룩업 서비스의 검사필드가 체크되지 않은 등록된 사용자의 정보를 바탕으로 경로를 이동하게 된다(A->B). 이 때 에이전트는 컴퓨터를 이동하면서 버전이 낮은 소프트웨어는 서비스에 접근할 수 있는 정보를 제공하고 버전이 같은 소프트웨어는 그냥 지나치게 된다. 그리고 에이전트는 룩업 서비스로 돌아와 이동한 컴퓨터의 검사필드를 체크한다. 서비스 B도 마찬가지의 역할을 한다. 그래서 a 컴퓨터의 A, B 소프트웨어와 b 컴퓨터의 B 소프트웨어가 변경되게 된다.

5. 결론 및 향후과제

이 논문에서는 Jini 기반의 분산환경에서 이동 에이전트를 설계하고 그것을 응용하는 예를 설명하였다. 그러나 이 시스템에도 문제점을 가지고 있는데 기기들이 자바 가상 기계(Java Virtual Machine)를 기본적으로 가지고 있어야 하고 클라이언트가 서비스를 받기 위해서는 에이전트가 이동해 올 동안 기다려야 한다. 만약 룩업 서비스에 많은 클라이언트가 등록이 되면 에이전트가 이동하는 데에 많은 시간이 걸리게 된다. 이 때에는 에이전트를 여러 개 사용해서 보내야 되는데 몇 개를 보내고, 어떻게 나눌 것인지가 중요한 문제가 될 것이다. 그리고 룩업 서비스에 많은 클라이언트 접근시 많은 부하가 생길 수 있는데 이 룩업 서비스를 어떻게 분산시킬 것인지도 중요한 문제가 될 것이다.

참고 문헌

[1] Sun microsystems, "Jini Architecture Specification", http://www.sun.com/jini/specs/, 2000
[2] W. K. Edwards, "Core Jini", Sun microsystems press, 1999
[3] Jan Newmatch, "Jini Tutorial", http://pandonia.canberra.edu.au/java/JINI/tutorial/, 2000
[4] Danny Ayers, et al., "Professional Java Server Programming", WROX, 2000
[5] P. Mates, "Agent that Reduce Work and Information Overload", CACM, 1994
[6] M.J. Wooldridge, et al., "Software Engineering with Agents: Pitfalls and Pratfalls", IEEE Internet Computing, 1999
[7] Dejan S. Milojicic, et al., "Mobile Objects and Agents", OSF Research Institute, 1996
[8] 조영상 외, "XML 기반의 지능 에이전트 시스템의 설계 및 구현", 정보과학회, 1999