다중 에이전트 기반의 유비쿼터스 프로그래밍 환경을 위한 실시간 스케줄링 서비스 지원

최형은°, 김태형
한양대학교 컴퓨터공학과
{hechoi°, tkim}@cse.hanyang.ac.kr

Providing Real-time Scheduling Services for Multi-agent Based Ubiquitous Programming Environments

Hyong-Eun Choi°, Tae-Hyung Kim
Dept. of Computer Science & Engineering, Hanyang University

요 약

유비쿼터스 환경 하에서 사용자가 원하는 정보나 서비스를 실시간으로 제공하는 프로그램을 효과적으로 제공하기 위해서는 에이전트와 같이 자율적 수행 능력을 갖는 시스템과 다중 에이전트들간의 실시간 스케줄링 기능을 미들웨어 단계에서 효과적으로 지원하는 기능이 필수적이다. 본 논문에서는 분산객체 미들웨어에 에이전트 서비스 계층을 제공하고, 이러한 계층에 다중 에이전트를 기반으로 한 실시간 스케줄링 서비스를 지원할 수 있는 구조를 제시한다.

1. 서 론

유비쿼터스 컴퓨팅[1] 환경에서 사용자에게 알맞은 서비스를 제공하기 위해서는 주변환경이나 데이터에 대해 능동적으로 인지하는 기능이 필요하다. 에이전트와 같이 자율적 수행능력을 갖는 시스템은 이러한 서비스를 제공하는데 있어 효과적인 방법으로 다중 에이전트 시스템의 경우 다수의 에이전트 간에 협업을 통해 서비스를 제공할 수 있다. 이때 사용자가 원하는 정보나 서비스를 실시간으로 제공하기 위해서는 실시간 에이전트 스케줄링[3] 기능이 제공되어야만 한다. 이에 본 논문에서는 다중 에이전트를 기반으로 한 실시간 스케줄링 서비스 구조를 설계하여 에이전트 서비스 계층[5]과 같은 미들웨어에서 제공되는 서비스에 실시간성을 보장하고자 한다.

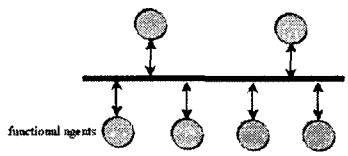
2. 관련연구

2.1 에이전트 시스템

에이전트란 사용자가 위탁한 일을 자율적으로 수행하는 컴퓨터 시스템[2]으로 자율적, 감시적, 지능적 특성을 지니며 변화된 환경에 대한 지각 능력을 통해 능동적으로 업무를 수행한다. 다중 에이전트란 하나의 에이전트로 해결하지 못하는 복잡한 문제를 여러 에이전트간의 통신과 협업을 통해 해결하는 에이전트 시스템을 말한다

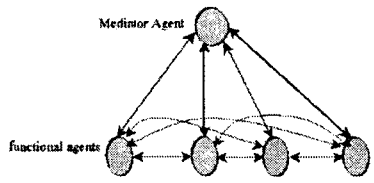
2.2 동적 스케줄링을 위한 다중 에이전트 구조

일반적인 시스템에서 사용되는 중앙 집중적이며 계층적인 스케줄링 방법과 달리 다중 에이전트를 기반으로 한 동적인 스케줄링 방법은 시스템의 확장과 복잡성의 증가에 유연한 구조를 갖고 있다. 동적 스케줄링을 위한 다중 에이전트의 구조에는 다음과 같은 두 가지 유형이 있다.



<그림 1. 자율적 구조>

<그림 1>은 자율적 구조의 형태를 나타낸 것으로 각각의 기능 에이전트는 지역내 스케줄링을 하고 에이전트간의 스케줄링 조정을 위해 협업 프로토콜을 사용한다. 이를 통해 전역적인 목표가 이루어지는데 에이전트 수가 많아 질수록 스케줄링 조정을 위한 에이전트간 통신이 잦아져서 시스템 전체의 성능을 저하시킬 수 있다.



<그림 2. 중재자 구조>

<그림 2>는 중재자 구조의 형태를 나타낸다. 자율적 구조가 갖는 단점을 보완한 것으로 중재자 에이전트를 두어서 기능 에이전트간의 스케줄링 조정 역할을 하도록 한다. 중

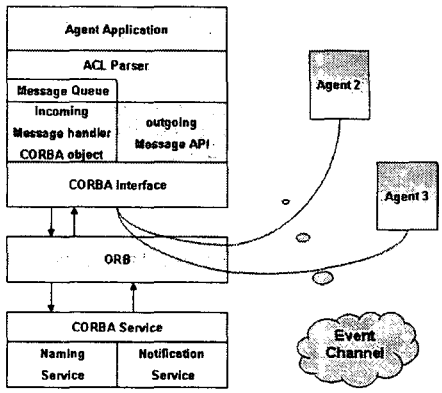
재자 에이전트는 기능 에이전트들이 등록하는 지역 스케줄링을 기반으로 전역적 목표를 이루기 위해 가장 효율적인 스케줄링을 판단하고 수정이 필요한 기능 에이전트에게 통보하여 지역 스케줄링을 변경하도록 한다.

2.3 실시간 에이전트 스케줄링

에이전트 서비스 제공에 있어 실시간성을 갖기 위해서는 시스템의 동작이 예측 가능해야만 한다. 이를 위해 Real-time AI 연구 분야에서 널리 알려진 방식으로는 Anytime 알고리즘과 Multiple method 방식[4]이 있다. Anytime 알고리즘은 초기에 문제 해결을 위한 해답을 빠르게 생성한 뒤 이를 시간이 지남에 따라 반복적으로 갱신하여 해답의 품질을 높여 나아가는 방식이다. 이 방식은 품질을 높여가는 과정 중간에 사용자가 원하는 시점에서 인터럽트를 발생시켜 결과를 얻을 수 있다는 장점이 있는 반면 인터럽트 발생시 결과 자체가 의미 없는 해답이 될 수 있는 정형화된 알고리즘에는 적용할 수 없다. Multiple method 방식은 문제 해결을 위해 여러 개의 해답을 이용하는 방식으로 각각의 해답은 서로 다른 특성을 갖는 알고리즘을 사용한다. 실시간 에이전트의 경우 문제 해결에 있어 시간적 제약 요소를 갖게 되는데 이러한 방법들을 통해서 실시간 서비스를 제공받을 수 있으며 품질과의 trade-off를 통해 제한된 시간 내에서 보다 나은 품질의 결과를 얻을 수 있다. 이와 관련된 기존의 연구로 Design-to-time real-time scheduling[3] 알고리즘이 있으며 이는 문제해결을 위해 최종 목표를 몇 개의 하위 태스크로 나누고 각 태스크 별로 Multiple method를 두는 계층적 모델을 제공한다. 이를 통해 생성된 스케줄링 목록으로 시간적 제약요소를 갖는 시스템 전반에 걸쳐 동적인 자원 할당이 가능하도록 한다.

2.4 에이전트 서비스 계층

에이전트 서비스 계층[5]은 CORBA를 기반으로 하고 있으며 에이전트 서비스 계층 상에서의 에이전트는 CORBA 객체로 표현된다. 에이전트 간의 메시지 전송은 이벤트 채널을 통한 통신 방법을 통해 객체간의 분리된 통신기법을 제공한다. <그림 3>는 CORBA를 기반으로 한 에이전트 서비스 계층 구조를 나타낸 것으로 에이전트 통신 언어를 처리하는 ACL Parser와 메시지 큐, 이벤트 채널을 관리하는 객체와 메시지 전송을 위한 함수들로 구성되어 있다. 이 모든 동작은 CORBA 인터페이스를 통해 ORB와 CORBA Service 기능을 이용하여 다른 에이전트들과의 상호작용을 가능하게 한다.



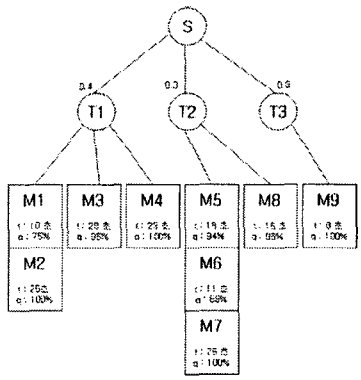
<그림 3. 에이전트 서비스 계층 구조>

3. 실시간 에이전트 스케줄링 서비스 구조

본 논문에서 설계하는 실시간 에이전트 스케줄링 서비스는 앞서 살펴본 에이전트 서비스 계층을 위한 것으로서 Multiple method 방식을 사용한다. 실시간 에이전트 서비스 제공을 위하여 시간, 품질과 같은 제약요소를 갖는 여러 개의 method들의 조합을 통해 스케줄링 리스트를 구성한 뒤 클라이언트의 요청에 따라 가장 적합한 스케줄링 리스트를 선택하여 실행 경로를 결정한다.

3.1 에이전트 태스크 모델

서비스는 하나 또는 그 이상의 에이전트간 협업을 통해 이루어질 수 있는데 이러한 서비스는 계층적 태스크 구조로 표현할 수 있다. <그림 4>는 이러한 계층적 태스크 구조의 예로서 S는 서비스를 나타내고, Ti는 서비스를 구성하는 하위 태스크, Mi는 해당 태스크에서 사용 가능한 method를 나타낸다.



<그림 4. 에이전트 태스크 계층 구조>

태스크마다 해당 태스크가 서비스에서 차지하는 비중을 표시하고 method가 제공 가능한 시간, 품질 값을 나타내어 태스크 구조 및 스케줄링 리스트 생성시 사용한다.

3.2 스케줄링 리스트 생성

<그림 4>의 에이전트 태스크 계층 구조를 이용하여 스케줄링 리스트를 생성하는데 기본적인 방법은 태스크별로 임의의 method를 선택하여 스케줄링 리스트를 만들고 사용자가 정의한 임계값을 만족하는 스케줄링을 사용한다. 이를 위해 method pool을 만드는데 태스크별 가중치에 따라 method의 갯수를 달리한다. 예를 들어 <그림 4>의 T1은 서비스에서 차지하는 비중이 0.4이고 T1이 갖고 있는 method는 4개이다. 4개의 method를 시간 요소에 따라 정렬하면 M1, M4, M2, M3 이 된다. 가장 낮은 method인 M3에 10을 할당하고 한 단계씩 높은 method에 대해 갯수를 할당할 때는 하위 단계를 모두 포함한 갯수에 0.4개 많은 갯수를 할당한다. 이 경우 M3은 10, M2는 14, M4는 34, M1은 81이 되므로 총 method pool의 갯수는 139가 다. 이렇게 생성된 method pool에서 임의로 method를 선택하여 스케줄링 리스트를 생성하고 총 소요시간과 태스크 비중에 따른 품질을 합산한다.

1. M1 - M7 - M9 : t = 33초, q = 90%
2. M3 - M8 - M9 : t = 52초, q = 96.5%
3. M1 - M6 - M9 : t = 29초, q = 86.4%

위와 같이 세 개의 리스트가 생성되고 사용자가 품질에 대해 임계값을 90%로 지정하였을 경우 해당 임계값을 넘지 못하는 3번 항목은 스케줄링 리스트에서 제거한다.

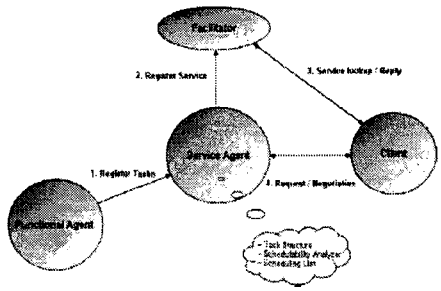
3.3 서비스 에이전트 동작 구조

에이전트 태스크 계층 구조에서 서비스는 서비스 에이전트로 method를 갖고 실제 기능을 수행하는 에이전트는 기능 에이전트로 표현되며 에이전트 타입에 따라 다르게 동작한다. 기능 에이전트는 다음과 같은 인터페이스를 사용하여 자신이 갖고 있는 method 내용을 해당 태스크 이름으로 서비스 에이전트에게 등록해야 한다.

```
Typedef struct _METHODINFO {
        string MethodName;
        int time;
        int qulity;
} METHODINFO;

RegisterMethods(string ServiceAgent, string TaskName,
METHODINFO* methods, int methodlen);
```

서비스 에이전트는 등록된 태스크 구조에 따라 스케줄링 리스트를 생성, 분석하는 기능을 수행하며 자신의 서비스를 facilitator에 등록한다. <그림 5>는 서비스 에이전트와 클라이언트 에이전트 간의 동작 구조를 나타낸 것으로 클라이언트는 자신이 필요로 하는 서비스를 facilitator를 통해 찾은 뒤 ACL메시지 내에 QoS 정보를 이용하여 서비스를 요청한다. 서비스 에이전트는 요청 받은 메시지가 스케줄링이 가능한지를 리스트에서 찾아본 뒤 만족하는 리스트의 실행 경로에 따라 서비스 동작을 수행한다.



<그림 5. 에이전트 동작 구조>

만일 클라이언트가 요구하는 제약조건을 만족하는 스케줄링 리스트를 찾지 못했을 경우에는 다른 제약 요소의 값을 낮추어 요구 조건을 만족시키는 스케줄링 리스트를 선택한 뒤 클라이언트에게 협상 메시지를 보내며 클라이언트는 서비스 에이전트가 보낸 협상 메시지에 대해 수락여부를 판단하여 서비스 에이전트에게 회신을 한다.

4. 결론 및 향후 과제

본 논문에서는 에이전트 서비스 제공에 있어서 실시간성을 보장하기 위해 실시간 에이전트 스케줄링 서비스 구조를 설계하였다. 이를 구현하고 에이전트 서비스 계층으로 적용하여 스케줄링 리스트에 대한 유효성 및 성능 평가가 필요하며 다양한 스케줄링 알고리즘을 적용할 수 있도록 사용자 정의 함수 추가를 통해 확장 가능한 서비스 구조를 갖도록 할 것이다.

5. 참고문헌

[1] Mark Weiser, "Hot Topics: Ubiquitous Computing" IEEE Computer, October 1993.

[2] Agent Platform Special Interest Group, " Agent Technology Green Paper " - http://www.objs.com/agent/agents_Green_Paper_v100.doc

[3] Alan Garvey, Victor Lesser. " Design-to-time Real-Time Scheduling." - IEEE Transactions on Systems, Man and Cybernetics - Special Issue on Planning, Scheduling and Control. vol. 23, no. 6, 1993.

[4] A. J. Garvey and V. R. Lesser. " A survey of research in deliberative real-time artificial intelligence." - Real-Time Systems, 6(3):317--47, 1994.

[5] 최형은, 김태형, " CORBA를 기반으로 한 분산 다중 에이전트 서비스 계층 설계" , 한국컴퓨터종합학술대회 2005 논문집 pp.292-294, 2005년 7월.