다중 컨텍스트 환경의 충돌 해결 방법[1)]

이건수° 김민구
아주대학교 정보통신 전문 대학원
lks7256@ajou.ac.kr°
아주대학교 정보통신대학 정보 및 컴퓨터공학부
minkoo@ajou.ac.kr

Conflict Resolution Method for Multi-Contexts Environment

Keonsoo Lee° Monkoo Kim
Graduate School of Information and Communication, Ajou University°
College of Information & Computer Engineering, Ajou University

요 약

상황 인식(Context-Aware) 시스템은 현재 자신이 처한 상황 정보를 인식하여 그에 맞는 행동을 결정하고 그 결과에 따라 동작하는 시스템을 일컫는다. 이러한 상황 인식 시스템은, 유비쿼터스 컴퓨팅을 비롯해 특정 작업을 수행하기 위해 사용자로부터 주어지는 입력 값 이외의 환경 정보를 필요로 하는 분야에서 주로 사용된다. 이때 시스템이 인지하는 상황은 자신의 작업을 수행하기 위해 필요한 환경 정보로 구성되어 있다. 가령, 온도조절 서비스를 담당하는 에이전트가 있다고 할 때, 이 에이전트는 사용자의 요청 온도를 맞춰주기 위해 현재 실내 온도를 측정할 수 있어야 한다. 그래야 요청 온도에 도달했을 때, 동작을 멈출 수 있기 때문이다. 이처럼 각 에이전트가 자신이 필요로 하는 정보를 인지하고 그에 따라 작업을 수행할 때, 단일 에이전트 혼자 그 공간에 존재한다면, 자신이 필요한 정보만을 인지하는 것으로 적절한 작업 수행을 기대할 수 있지만, 둘 이상의 에이전트가 동일 공간에 존재하고, 각 에이전트의 작업 과정이 서로의 수행 과정에 영향을 미칠 가능성 즉, 에이전트 사이의 충돌 발생 가능성이 존재한다. 이미 오디오가 동작하고 있는 방안에서 TV를 작동시키게 된다면, 음악을 들을 수도, TV를 볼 수도 없는 상황이 발생한다. 이에 본 논문에서는 동일 공간에 존재하는 상황 인식 에이전트들 사이에 발생하는 충돌을 해결하기 위한 방법을 제안한다.

1. 서 론

유비쿼터스 컴퓨팅은 대표적인 상황인지 시스템 분야이다. 상황 인지 시스템이란, 일종의 반응 시스템으로 시스템이 사용자로부터 주어지는 명확한 입력 데이터 이외의 자신이 처한 주변 상황을 감지하여, 필요한 정보를 추출해 내고, 이 정보들과 사용자로부터의 입력 데이터를 바탕으로 현재 가장 적합한 행동을 선택하여 수행하는 시스템을 의미한다. 이러한 상황 인지 시스템은 현재 상황과 사용자의 상태를 인지하여 사용자로부터 불명확한 명령이 들어오더라도, 그 상황에서의 최적의 행동을 수행할 수 있다. 가령, 사용자가 온도 조절 에이전트에게 "24도로 온도를 맞춰라"라고 명령을 내렸을 때, 에이전트는 현재 온도를 인지하고, 요청 온도와의 차이를 비교하여 낮다면 높여주고, 높다면 낮춰주는 동작을 수행하게 된다. 또한 그 사용자가 선호하는 특정 온도를 알고 있다면, 직접적인 명령을 받지 못하였더라도, 사용자가 공간에 들어오는 순간, 그 사용자의 선호 온도를 맞춰줄 수도 있다. 이러한 동작을 수행하기 위해 이 온도조절 에이전트는 그 공간의 전체 상황을 이해할 필요는 없다. 온도를 조절하는 작업을 수행함에 있어, 그 공간의 조명 상태, 다른 가전기기들의 작동 상황을 인지할 필요는 없다. 즉, 모든 에이전트는 인지해야 하는 주변 상황 정보의 범위를 갖는다. 이러한 에이전트별 인지 상황의 범위는 보다 빠른 작업 결과를 보장해 주지만, 다중 컨텍스트가 발생하는 공간, 즉 동일 환경에서 여러 에이전트들이 자신의 인지 범위를 갖고 그 환경을 모델링 하는 경우에서 에이전트간 충돌 발생의 가능성이 존재한다.

가령, 사용자 A에 의해 특정 공간의 오디오 시스템이 동작하고 있다고 하자. 이때, 사용자 B가 TV에 대한 서비스 요청하고 이로 인해 TV가 켜진다면, 그 공간에 "음향"이라는 자원에 대한 충돌이 발생하게 된다. 이는 TV와 오디오가 현재 공간의 "음향" 상태를 인지하고 있다면 피할 수 있는 충돌이지만, 오디오가 동작하기 위해 현재 공간이 얼마나 조용한지, 조용하지 않다면, 단순한 소음인지 아니면 다른 서비스의 과정 때문인지를 판단하기란 쉬운 일이 아니다. 이에 본 논문에서는 이처럼, 단일 환경에 존재하는 각각의 시스템들이 자신만의 상황정보를 인지하는 경우, 제안된 서비스들 사이에서 발생 가능한 충돌을 방지하기 위한 방법을 제안한다.

2. 본 론

2.1 Service Conflict

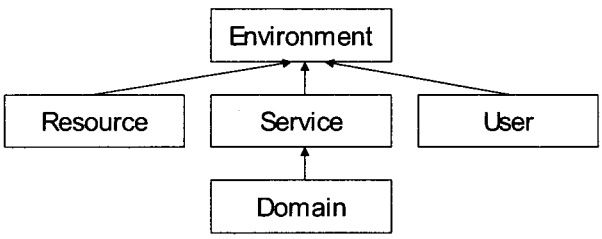
상황 인지 과정을 수행하는 에이전트들 사이에서 충돌이 발생하는 원인은 크게 두 가지로 나뉜다. 첫째, 각 에이전

1) 본 논문은 유비쿼터스 컴퓨팅 및 네트워크 원천기반 기술 개발사업 내 uT-DB 기반의 지능성장 학습 엔진 사업의 일환으로 지원 받아 수행되었음 (과제번호 M103KT010007-04K2001-00713)

트들이 갖고 있는 현재 환경 모델이 각기 다르다. 모든 에이전트가 전체 상황에 대한 완벽한 인지를 한다면, 충돌을 원천적으로 봉쇄할 수 있지만, 이는 네트워크 상의 대역을 낭비시킬 수 있고, 실재 에이전트의 동작에 과중한 노드를 주게 된다. 실내 온도를 맞추는데, 실내조명 상태, 소음의 양 또는 다른 가전기기의 사용상태를 알 필요는 없다. 그러나 이러한 서비스 실행 과정에 있어 사용되는 자원의 종류와 서비스 자체의 실행 결과는 다른 에이전트가 인지하는 환경을 변화시킬 수 있다. 협소한 공간에서 많은 가전 기기의 사용은 실내 온도를 높일 수 있지만, 각 기기들의 상태를 온도 관리 에이전트가 체크하지는 않는다. 이 경우, 불필요한 기기를 멈추기보다 오히려 에어컨을 실행시키는 결과를 갖게 되고, 이는 적절한 서비스 해결 방법이라고 할 수 없다. 둘째, 실재 사용되는 자원은 하나의 요청에 대해 독점적으로 점유된다. 실내 온도를 맞추기 위해 온도 조절 에이전트는 에어컨, 선풍기, 온풍기, 난로 등을 사용할 수 있다. 그러나, 만약 선풍기가 다른 사용자에 의해 머리를 말리기 위해 사용되고 있다면, 그 자원은 온도 관리 에이전트의 통제를 벗어난 상황으로, 선풍기 이외의 자원을 사용하여 서비스를 수행해야 한다. 즉, 하나의 자원은 동시에 둘 이상의 서비스에 사용될 수 없다. 이상의 두 가지 요소가 충돌을 발생시키는 가장 큰 요인이고, 본 연구에서는 이 요소들의 조정을 통하여 충돌을 해결하는 방법을 제안한다.

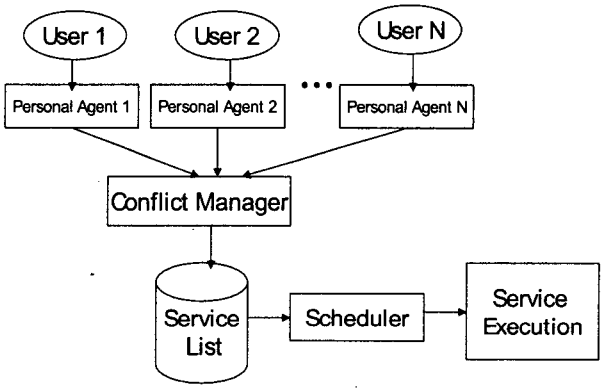
2.2 제안 방법

각 에이전트들이 환경을 인식하는 과정에 있어서, 전체 환경을 인식하지 못한다는 단점은 실제 에이전트들이 주어진 서비스를 실행하는 과정이 다른 에이전트와 얽힌 경우, 서비스 충돌을 일으키는 주된 원인이다. 이를 해결하기 위해서는 전체 환경을 인지하고 있는 중계자가 각각의 에이전트들이 수행하려는 서비스가 다른 서비스들과 충돌이 일어나지 않는지를 검증해주거나, 실재 서비스가 수행될 때마다, 각 에이전트들이 다른 에이전트와 직접 통신을 통해 충돌이 발생하지 않는 다는 것을 확인하는 방법이 있다. 전체 환경을 인지하는 중계자가 있는 경우, 에이전트가 증가함에 따라 중계자의 짐이 많아지게 되고, 유비쿼터스 컴퓨팅 도메인처럼, 환경을 모델링에 소요되는 요소가 많은 경우, 전체 환경을 모델링 하는 것 자체가 불가능하다. 또한 중계자 없이 에이전트들 스스로 협력/협상하는 방법으로는 네트워크 상의 트래픽 증가와 함께, 모든 에이전트들에 추가적인 협력 모듈이 필요하다.



<그림 1> Key Features for modeling environment.

본 연구에서는 미들웨어를 사용한 충돌 해결 방법을 제안한다. 미들웨어가 관리하는 전체 환경은 <그림 1>과 같이 도메인에 따라 달라지는 서비스와 자원 그리고 사용자 정보를 기반으로 모델링 된다. 서비스 정보는 현재 공간에서 제공 받을 수 있는 서비스의 목록과, 각 서비스별 카테고리 정보, 그리고 서비스가 수행되기 위해 필요로 하는 자원들의 순서 목록 정보가 담겨져 있다. 리소스 정보는 현재 공간에 존재하는 자원들의 속성 정보들로 구성되어 있지만, 가장 중요한 정보는 각 자원들 사이에 대체자원 관계이다. 가령 데스크 탑과 노트북이 있는 환경에서 이 두 자원들은 서로 대체관계를 갖고 있다. 사용자 정보는 사용자의 개별 프로파일과 그 사용자가 갖고 있는 환경에서의 중요도 정보로 구성되어 있다. 본 모델에서 채용하고 있는 미들웨어는 Conflict Manager와 Scheduler로 구성되어 있는데 이들 미들웨어가 다른 에이전트들과 연계하여 동작하는 구조는 <그림 2>에 나타나 있다.



<그림 2> Conflict Managing Architecture

개별 사용자들은 자신의 프로파일 정보에 특성화된 Personal Agent를 갖고 있다. 이 에이전트는 현재 사용자의 특성에 맞는 서비스를 탐색하고 그 서비스의 실행을 요청한다. 이들 요청은 Conflict Manager에게 전달된다. Conflict Manager는 전체 환경을 인식하고 있고, 이 인식 정보를 바탕으로 새로이 요청된 서비스가 요청된 시간에 다른 서비스들과의 충돌 없이 실행가능한지를 검증한다.

<표 1> Request Service Format

Service ID		
Service Starting Time		
Service Ending Time		
Acceptable Delay Time		
Requester ID (Service Owner)		
Resource Sequences		
Resource ID	Seized Time	Effected State

<표 1>은 Conflict Manager가 받는 서비스 요청 포맷이다. Conflict Manager는 전체 환경을 리소스들의 상태로

인식하기 때문에 지금 요청된 서비스가 기존 서비스가 실행되기 위해 필요한 자원을 요청하는지, 새로운 서비스의 실행을 위한 자원의 사용이 기존 서비스가 변경시켜 놓은 자원의 상태를 변화시키는지를 확인할 수 있다. 이렇게 검증이 이루어진 서비스는 서비스 리스트에 기록되고 검증 결과 충돌이 발견된 서비스는 다음의 방법을 통해 처리된다. 첫째, 두 요청 서비스 사이의 중요도가 크게 차이가 난다면, 보다 중요한 서비스가 선택되고 그렇지 않은 서비스는 거절된다. 둘째, 충돌이 발생하는 두 요청 사이의 중요도가 비슷하다면, Conflict Manager는 서비스를 요청한 두 에이전트를 연결시켜 둘 사이에서 직접 자원 협상이 일어날 수 있는 통신라인을 구축해준다. 셋째, 앞의 두 조건에 맞지 않은 서비스 요청에 대해서는 거절한다.

일단 Conflict Manager에 의해 서비스가 검증되고 리스트에 저장되면, Scheduler는 그 서비스 리스트를 확인하면서 현재 수행되어야 할 서비스를 찾아 실행시킨다. 이때, 수행 시간이 겹치거나 자원 할당에 문제가 발생하는 경우, 서비스의 중요도 (서비스 계급에 의해 결정)와 서비스 요청자의 중요도를 참조해 보다 중요한 서비스가 먼저 수행되고, 보다 좋은 자원을 선정하여 사용할 수 있게 해준다. 현재 수행하려는 서비스가 필요로 하는 자원이 다른 서비스에 의해 선점되어 있다면, 리소스 온톨로지 (그림 1)를 참조해 그 리소스의 대체 자원을 찾고 그 자원을 할당시켜준다.

이처럼, 각 에이전트의 서비스 요청을 현재 환경 정보에 근거해 실행 가능한지를 판단해 주는 Conflict Manager와 등록된 서비스 요청을 실제 선택해 수행하도록 해주는 Scheduler를 통해 에이전트들 사이에 발생 가능한 충돌 상황을 회피하고, 발생시 Conflict Manager의 조정 기능을 통해 해결할 수 있다.

2.3 시뮬레이션

본 방법을 평가하기 위해 사용된 시뮬레이션 시나리오는 다음과 같다. 두 명의 사용자가 공간에서 존재하는 TV, 오디오, 조명에 대한 서비스를 요청하는 과정에서 발생하는 충돌 상황을 가정하고, 시스템이 그 충돌을 해결하는 과정을 확인하였다.

<표 2> Simulation Scenario

공간	TV, 오디오, 조명이 있는 방에 두 명의 사용자가 존재한다.
가능 서비스	조명, TV, 오디오
사용자	2명
서비스 시나리오	
1) 사용자 A가 방에 들어와 조명을 끄고 TV를 시청한다. 2) 사용자 B가 방에 들어와 불을 켠다. 3) 사용자 B가 오디오를 듣기를 희망한다. 4) 사용자 A가 방에서 나간다.	

처음 서비스 요청을 하는 사용자 A의 경우 충돌 없이 서비스가 수행된다. 두 번째 사용자 B의 요청의 경우, 조명에 대한 충돌이 발생하고 이 때, Conflict Manager는 충돌을 감지하고, 두 서비스 요청에 대한 중요도를 비교한다. 사용자와 서비스 자체의 중요도 차는 사용되는 온톨로지 정보를 바탕으로 모델링된 환경 정보에 기초한다. 공간에 진입하며 첫 번째 서비스로 조명을 요청하는 것에 대한 중요도를 높였기 때문에 이 상황에서는 사용자 B의 요청이 채택된다. 사용자 B의 오디오 청취 요청은 이전의 TV와 대체 관계에 있기 때문에 충돌이 일어나고, 사용자 A와 B사이의 중요도 차가 없는 경우, 이미 진행 중인 서비스를 중지시키는 것보다 새로운 요청을 거부한다는 규칙(모델링 정보에 담긴 내용으로 실제 관리자에 따라 달라질 수 있다)에 의해 오디오는 실행되지 않는다. 그 후 사용자 A가 방을 나가면서, TV 서비스는 중지되고 남아있는 사용자 B를 위해 오디오 서비스가 실현된다. 이때, 사용자 A가 나가는 시간은 오디오 서비스의 "Acceptable Delay" 시간 이전에 발생해야 한다.

3. 결 론

본 연구에서는 동일 공간 안에 여러 서비스 에이전트들이 존재할 때, 발생 가능한 서비스 사이의 충돌을 해결하기 위해 전체 환경을 자원들의 상태 집합으로 인지하고 발생하는 서비스들 간의 충돌을 감독하며 서비스 리스트를 관리하는 Conflict Manager와 현재 서비스 리스트에서 서비스를 선택해 실행시키는 Scheduler를 미들웨어로 갖고 있는 충돌 해결 시스템을 제안하였다. 미들웨어는 전체 환경을 자원들의 상태집합으로 인지하고, 각 서비스를 수행하는 과정에서 변화가 일어나는 자원의 상태들 사이에서 모순이 발생하는지를 검사함으로써 충돌을 예상하고, 충돌 서비스를 거부할 수 있다.

참 고 문 헌

[1] Nicholas Hanssens, Ajay Kulkarni, Rattapoom Tuchinda, and Tyler Horton, "Building Agent-Based Intelligent Workspaces," In ABA Conference Proceedings, June 2002.

[2] S. S. Yau, F. Karim, Y. Wang, B. Wang, and S. Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing," IEEE Pervasive Computing, July-September 2002, pp.33-40.

[3] Maedche, A., Motik, B., Stojanovic, L., Studer, R. and Volz, R., Ontologies for Enterprise Knowledge Management, IEEE Intelligent Systems, 11/12, 2002.

[4] Dey, A.K. et al. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction Journal 16, 24(2001), 97-166.

[5] S. S. Yau, F. Karim, Y. Wang, B. Wang, and S. Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing," IEEE Pervasive Computing, July-September 2002, pp.33-40.