커뮤니티 컴퓨팅을 위한 정책기반 충돌 해결 방안 연구*

조용석° 정유나 김민구[1]
아주대학교 정보통신 전문대학원
°{cho, serazade}@ceai.ajou.ac.kr
[1]minkoo@ajou.ac.kr

Policy-Based Conflict Resolution in Community Computing

°Yongsuk Cho, Yuna Jung, Minkoo Kim
Graduate school of information and communication, Ajou university

요 약

커뮤니티 컴퓨팅 (Community Computing)은 유비쿼터스 환경을 하나의 사회(Society)로 보고 이 사회 내에서 특정한 목표(Goal)가 생겼을 경우 이를 달성시키기 위해 커뮤니티(Community)라는 개념을 이용하는 컴퓨팅 방법이다. 커뮤니티 컴퓨팅에서 목표가 발생함에 따라 커뮤니티는 동적으로 생성되며, 한 커뮤니티 내에 속하는 구성원(Member)들은 그 커뮤니티의 목표 달성을 위해 상호 작용한다. 기존의 커뮤니티 컴퓨팅 연구에서는 커뮤니티를 생성하고 멤버를 관리하는데 있어 서로 다른 커뮤니티 혹은 멤버 간에 충돌이 발생할 수 있다. 본 논문에서는 커뮤니티 컴퓨팅에서 발생할 수 있는 충돌요인을 제시하고 이의 해결을 위하여 커뮤니티 정책(Community Policy)에 기반한 방법을 제안하였다. 이의 구현 방법론으로 정책 표현을 위하여 많이 사용되는 Ponder 언어를 적용하는 사례를 제시하였다.

1. 서 론

유비쿼터스 환경은 복잡한 분산 환경중 하나로 생각할 수 있다. 일반적인 분산 환경의 특징 외에 유비쿼터스 환경은 서비스 주체의 동적인 배치, 목표 지향의 구성 등의 유비쿼터스 환경 자체만의 특징을 가진다[1]. 기존의 모델링 방법[2],[3]들은 이러한 유비쿼터스 환경의 특징에 대해 고려할 수 없었기 때문에 모델링시 이 특징을 충분히 반영하지 못한다. 커뮤니티 컴퓨팅[1]은 기존의 방법들에 비해 유비쿼터스 시스템의 특징을 반영한 모델링 기법이다. [1]커뮤니티 컴퓨팅은 유비쿼터스 환경을 하나의 사회(Society)로 보고 이 사회 내에서 발생하는 목표(Goal)에 따라 커뮤니티(Community)라고 하는 가상의 조직을 구성하여 목표를 달성하려는 방법이다. 커뮤니티 컴퓨팅을 통해 유비쿼터스 환경의 특징을 반영한 모델링을 할 수 있으며, 유비쿼터스 환경을 구성하는 요소들 간의 관계를 설명할 수 있다. 그러나 현재의 커뮤니티 컴퓨팅은 시스템의 실행 시에 발생할 수 있는 충돌 요인을 고려하지 않고 있으며, 이는 시스템의 수행시 예상치 못한 문제를 발생시킨다. 따라서 커뮤니티 컴퓨팅에서 발생하는 충돌 요인을 분석하고, 이를 모델에 적용시키는 방안에 대한 연구가 필요하다.

본 논문에서는 분산 환경에서의 충돌해결을 위한 방법 중 하나인 정책(Policy)을 커뮤니티 컴퓨팅에 적용하기 위해 커뮤니티 컴퓨팅에서의 충돌 요인을 분석하고 이에 맞는 커뮤니티 정책을 제시한다. 또한 이를 Ponder 언어로 기술해 봄으로써 실제 구현을 위한 적용방법론을 제시한다.

* "본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 지원에 의한 것임"

2. 관련연구

2.1 커뮤니티 컴퓨팅

커뮤니티 컴퓨팅은 유비쿼터스 환경의 특징을 반영할 수 있는 모델링을 위해 제안된 기법이다. 이 방법은 특정 유비쿼터스 도메인을 하나의 사회로 보고 이 사회 내에서 목표가 동적으로 발생하는 환경으로 생각한다. 커뮤니티 컴퓨팅에서는 동적으로 발생하는 목표의 달성을 위해 커뮤니티라는 개념을 이용하여 접근하고 있다. 커뮤니티는 목표 달성을 위한 서비스 주체들의 집합이며, 필요에 따라 동적으로 생성되고 소멸된다. 따라서 커뮤니티 컴퓨팅의 모델에는 사회와 사회를 구성하는 구성원(Member)에 대해 기술한 후 목표 달성을 위해 생성되어야 하는 커뮤니티와 이 커뮤니티를 구성하기 위해 필요한 사회의 구성원, 그리고 목표 달성을 위한 커뮤니티 내의 각 구성원 간의 상호작용(Interaction)에 대해 기술한다. 이를 통해 유비쿼터스 환경에서 동적으로 배치되는 컴퓨팅 요소들과 목표지향적인 특징을 반영할 수 있게 된다. 하지만 현재의 커뮤니티 컴퓨팅은 구성요소들이 실행될 때 발생할 수 있는 충돌에 대한 고려가 되어 있지 않다. 분산 환경에서 상호작용하는 요소들 간에는 충돌이 발생하며, 충돌에 대한 처리는 필수적인 것이다[4]. 커뮤니티 컴퓨팅에서는 이러한 충돌에 대한 처리가 개발자의 몫으로 남겨져 있어 새로운 충돌요인이 발생되면 컴퓨팅 요소의 내부를 변화시켜야 하는 문제점이 발생한다. 이는 효율적인 시스템 개발의 저해요인이 된다.

2.2 분산 환경에서의 충돌 해결

분산 환경에서 각 컴퓨팅 요소들이 독립적인 행동을 수행하다보면 요소 간의 충돌이 발생하는 상황이 발생한다. 지난 수년간 이러한 충돌 해결을 위한 많은 연구들이 진행되어 왔으며 접근 방법에 따라 여러 가지로 구분

된다. 대표적으로, 협상(Negotiation), 중재(Arbitration), 투표(Voting) 등을 들 수 있으며, 이 방법들을 실제 운용함에 있어 크게 알고리즘 기반으로 수행하거나 혹은 정책 기반으로 수행하는 방법이 있다[4]. 알고리즘 기반의 수행은 빠른 수행속도가 장점인 반면 충돌 해결을 위한 방법이 바뀔 때마다 컴퓨팅 요소의 내부를 변경해야하는 문제점을 지닌다. 또한 컴퓨팅 요소의 수가 많아진 상태에서 알고리즘의 변화가 생기면 각 요소에 대한 변환 작업은 많은 시간을 필요로 하게 된다. 이는 컴퓨팅 요소들이 고정되어 있는 분산 환경에 적합한 방법이다. 이에 비해 정책 기반의 운용은 정책을 수행하는 부분이 한번 만들어 지면 거의 변화가 없으며 충돌 해결 방법이 바뀌어도 정책을 수정하기만 하면 되기 때문에 에이전트 내부 코드에 대한 수정을 고려하지 않아도 되는 장점이 있다. 이는 컴퓨팅 요소들이 가변적인 환경에 적합한 방법이다. 유비쿼터스 환경은 컴퓨팅 요소들이 가변적인 환경이므로 알고리즘에 기반한 방법보다 정책에 기반한 접근 방법이 더 적합하다.

2.2.1 정책 (Policy)

정책은 시스템의 행동을 변화시키는데 이용될 수 있는 정보로 정의된다[5]. 일반적으로 정책은 주체 (Subject)와 자원 (Target resource) 간에 필요한 권한 (Authorization)과 의무 (Obligation) 그리고 이러한 정책을 수행하면서 발생할 수 있는 충돌 (Conflict)을 해결하기 위한 부분으로 구성된다. 분산 환경의 경우 이에 덧붙여 위임 (Delegation)을 위한 정책이 추가적으로 필요하다. 위임을 통해 정책을 분산시키면 중앙 집중 형으로 정책을 수행할 때의 오버헤드를 줄일 수 있기 때문이다[6]. 커뮤니티 컴퓨팅을 위해서는 의무과 위임 그리고 정책간의 충돌을 해결할 수 있는 메타정책(Meta-Policy)에 대해 연구가 필요하다. 권한은 리소스에 대한 접근 제어를 위해 이용되는데 커뮤니티 컴퓨팅에서는 자신의 리소스를 가지고 있는 구성원 간의 상호 작용에 대해 기술하고 있기 때문에 권한에 관한 정책은 고려하지 않는다.

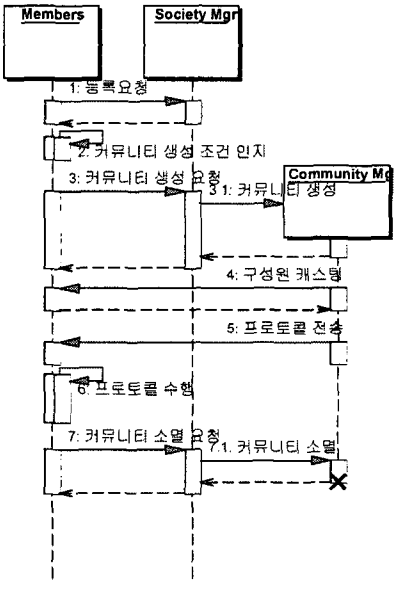
2.2.2 정책 언어 (Policy Languages)

정책의 표현을 위해서는 정책언어가 필수적이다. 정책 표현을 위한 표준 언어는 아직 존재하지 않지만 정책의 필요성에 의해 개별적인 연구가 진행되고 있다. 현재 연구 중인 정책 언어로는 Rei[7]와 Ponder[5]가 있다. Rei는 분산 환경에서의 보안 정책 표현을 위한 정책 언어로 분산 환경을 위해 필요한 위임에 관한 정책을 다양하게 표현할 수 있도록 지원하고 있다. 하지만 보안 정책에 초점이 맞추어져 있어 일반적인 용도로 사용하기에는 적절하지 않다. Ponder는 일반적인 목적으로 개발된 정책 언어로 모든 도메인에서 사용할 수 있도록 다양한 표현력을 제공한다. 위임 정책에 대해서는 가장 기본적인 위임에 관한 표현력을 제공한다.

3. 커뮤니티 컴퓨팅에서의 정책

3.1 커뮤니티 컴퓨팅의 수행

[그림 1]은 커뮤니티 컴퓨팅의 수행 단계를 나타낸다. 초기화 단계에는 사회 관리자와 어느 사회에도 속하지 않은 구성원들이 생성되며, 구성원들의 사회로의 등록이 이루어진다. 등록이 완료되면 커뮤니티 생성 조건이 발생하지 않는 한 구성원들은 각자의 행동(Own Action)을 수행한다. 커뮤니티 생성 조건이 발생하면 이를 인지한 특정 구성원이 사회 관리자에게 커뮤니티 생성을 요청하고, 이 요청에 따라 사회 관리자는 해당 커뮤니티를 위한 커뮤니티 관리자를 생성하고 커뮤니티에 목표를 전달한다. 목표를 받은 커뮤니티 관리자는 커뮤니티의 목표 달성을 위해 필요한 구성원들을 캐스팅하여 각 구성원의 역할에 필요한 프로토콜을 전송한다. 프로토콜이 모두 전달된 후 목표 달성을 위한 초기 행동을 시작으로 커뮤니티 구성원들 간의 통신을 통해 각 프로토콜을 수행한다. 프로토콜의 수행이 모두 끝나면 목표가 달성되었다는 뜻이며, 목표 달성을 인지한 구성원은 사회 관리자에게 커뮤니티 소멸을 요청한다. 커뮤니티가 소멸되면 초기에 사회 관리자와 구성원들이 존재하던 상태로 돌아간다. 이러한 과정을 반복하며 기술된 목표를 위한 조건이 생길 때마다 커뮤니티가 동적으로 생성되거나 소멸된다.



[그림 1] 커뮤니티 컴퓨팅의 수행 단계

3.2 커뮤니티 컴퓨팅에서의 충돌

커뮤니티 컴퓨팅에서 발생하는 충돌은 커뮤니티 컴퓨팅 구성요소에 따라 크게 사회, 커뮤니티, 구성원에서 발생할 수 있는 충돌로 나눌 수 있다. 각 구성요소에 따른 충돌을 나열하면 다음과 같다.

① 사회에서의 충돌
- 구성원의 무조건적인 등록으로 인한 구성원간의 충돌
- 서로 공존해서는 안 되는 커뮤니티의 생성에 따른 커뮤니티 간 충돌
- 두 멤버로부터 동시에 커뮤니티 생성 요청을 받았을 때 생성 될 커뮤니티의 우선순위 문제
- 구성원의 기본 행동(Own Action) 간에 발생하는 충돌

② 커뮤니티에서의 충돌
- 서로 상반되는 역할을 하는 구성원의 캐스팅으로 인한 충돌
- 다른 커뮤니티에 속한 구성원의 캐스팅으로 인한 역할 충돌
- 캐스팅 된 멤버의 이상으로 인한 통신 문제시의 목표 달성 불가능

③ 구성원에서의 충돌

· 구성원의 기본 행동과 사회 관리자로부터 전달 받은 프로토콜의 상충
· 전달받은 프로토콜이 다른 구성원의 프로토콜과 충돌

위에서 제시한 충돌은 커뮤니티 컴퓨팅에서 일반적으로 발생할 수 있는 충돌을 나열한 것이다. 응용 영역에 따라 발생할 수 있는 충돌은 해당 도메인에 따라 달라질 수 있다.

3.3 충돌 해결을 위한 정책 및 표현

분석된 충돌을 해결하기 위하여 필요한 정책의 예는 다음과 같다.

① 사회정책
- 사회 구성원은 구성원의 현재 상태가 커뮤니티에 등록되어 있지 않고, 다른 사회구성원과의 충돌이 없을 경우에 한해 등록한다.
- 구성원으로부터 커뮤니티 생성 요청시 현재 존재하는 커뮤니티와 공존할 수 없는 커뮤니티의 경우 생성을 보류한다.
- 구성원의 종류(Type)별로 우선순위를 두어 구성원의 기본 행동을 수행한다.
- 커뮤니티 생성시에 우선순위를 두어 동시에 커뮤니티 생성 요청이 발생할 경우 우선순위에 따라 생성시킨다.

② 커뮤니티 정책
- 구성원 캐스팅시 구성원의 타입에 따라 우선순위를 두어 캐스팅하며, 우선순위가 낮은 구성원이 우선순위가 높은 구성원과 충돌이 예상되면, 같은 타입내의 다른 구성원을 캐스팅한다.
- 다른 커뮤니티에 속해 있는 구성원의 경우 캐스팅 대상에서 제외시킨다.
- 캐스팅 된 구성원과의 통신 문제 발생시 3회 시도 후 구성원에 대한 캐스팅 작업을 다시 수행한다.

③ 구성원 정책
- 기본 행동과 전달 받은 프로토콜이 같이 수행될 수 있는 지의 여부를 판단하여 같이 수행될 수 없다면 기본행동을 중지하고 프로토콜을 수행한다.
- 다른 구성원의 프로토콜과의 충돌 시 우선순위가 낮은 구성원의 프로토콜 수행을 중지시킨다.

다음은 도출된 정책의 적용하기 위하여 정책 사회구성원의 등록, 커뮤니티 구성원의 캐스팅, 사회 구성원과의 통신 문제시의 정책을 기존의 정책 언어 중 Ponder를 이용하여 기술해 본 예이다.

```
type oblig registeration(suject s, target t){
        on memberRegisteration(society)
        do society->s.mail("reject registeration");
        when s.state = negative;
}

type oblig selectMember (subject s, domain T){
        on castingConfusion (community)
        target T->select(t | t.state = 'uncasted');
        do community->t.mail("cast");
        when max(t.priority);
}

type oblig failCommunication(subject s, target t){
        do s->reject(protocol);
        when(t.cfailCount = 3);
}
```

[그림 2] 정책 표현의 예

4. 결론 및 향후과제

유비쿼터스 시스템을 모델링하고 수행함에 있어 충돌 요인에 대한 고려는 필수적이며, 동적인 분산 환경에서의 충돌해결을 위해서는 정책을 이용하는 방법이 효율적이다. 본 논문에서는 유비쿼터스 환경을 모델링하기 위한 커뮤니티 컴퓨팅에서 발생할 수 있는 충돌요인을 분석하고 이의 해결을 위한 정책을 제시하였다. 또한 기존의 정책언어를 통해 정책을 표현해 봄으로써 실제 적용을 위한 방법론을 제시하였다. 하지만 실제 정책의 적용을 위해서는 모델이 정책 수행을 위해 필요한 부가적인 정보를 제공해 줄 수 있어야 한다. 그리고 현재는 기존의 정책언어를 이용해 정책을 표현했지만 커뮤니티 컴퓨팅에 맞는 정책 언어와 정책 수행기의 개발이 필요하다.

5. 참고문헌

[1] Yuna Jung, Jungtae Lee, Minkoo Kim. "Multi-agent based Community Computing System Development with the Model Driven Architecture" AAMAS, 2006.

[2] Michael Wooldridge, Nicholas R. J, David K. The Gaia Methodology for Agent-oriented Analysis and Design, Autonomous Agents and Multi-Agent Systems, 3, 2000, 285-312

[3] Mohan Kumar, Behrooz A. Shirazi, Sajal K. Das, Byung Y. Sung, et. al. PICO: A Middleware Framework for Pervasive Computing, Pervasive Computing, 1536-1268, 2003, 72-79

[4] T.H.Liu, A.Goel, C.E.Martin, K.S.Barber "Classification and Representation of Conflict in Multi-Agent Systems", Technical Report of The University of Texas at Austin, Jan, 1998

[5] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The ponder policy specification language. In Morris Sloman, editor, Proc. of Policy Worshop, 2001, Bristol UK, January 2001

[6] Patwardhan, A., Korolev, V., Kagal, L., Joshi, A.: Enforcing policies in Pervasive Environments. International Conference on Mobile and Ubiquitous Systems: Networking and Services. (2004)

[7] Kagal. L., Finin. T., Josh. A., "A Policy Language for a Pervasive Computing Environment" Proceedings of Policy '03, June 2003, Lake Como, Italy p63-77

[8] Keven F., Karl Q, David L, Declan O, Vincent W, "Relationship-driven Policy Engineering for Autonomic Organisations" Proceedings of POLICY'05, June 2005, Stockholm, Sweden

[9] Anand T., Devdatta K., Tanvir A., "Policy-Driven Configuration and Management of Agent Based Distributed Systems", Proceedings of the fourth international workshop on Software engineering for large-scale multi-agent systems