컨텍스트 온톨로지를 이용한 결함허용 커뮤니티 컴퓨팅

Fault Tolerant Community Computing using Context Ontology

김민수, Minsoo Kim, 김민구, Minkoo Kim
아주대학교 정보통신전문대학원, 인공지능연구실

요약 유비쿼터스 환경에 존재하는 스마트 객체들 사이의 협력으로 지능적인 서비스를 제공하는 시스템을 구축하기 위한 새로운 패러다임으로 커뮤니티 컴퓨팅이 제안되어, 활발히 연구되고 있다. 커뮤니티는 해결해야 할 목표를 가지며 목표 달성을 위해 필요한 멤버들을 구성하고, 멤버들은 목표 달성을 위해 상호작용한다. 이러한 상황에서 멤버의 결함은 멤버 사이의 상호작용을 중단시키는데, 원활한 멤버 사이의 상호작용이 커뮤니티의 목표를 달성하는 가장 중요한 문제이므로, 결국 멤버의 결함은 커뮤니티의 목표 달성의 실패를 가져온다. 이 문제를 해결하기 위해, 본 논문에서는 컨텍스트 온톨로지를 이용한 멤버의 복제 방법 및 복구 알고리즘을 제안한다. 각 멤버는 자신의 상태를 커뮤니티에 전송하며, 커뮤니티는 모든 멤버의 상태를 온톨로지로서 유지한다. 멤버의 결함이 발견되면, 커뮤니티는 결함이 발생한 멤버의 상태 온톨로지를 추론하여 이 멤버를 대신할 새로운 멤버를 커뮤니티로 끌어들인다. 커뮤니티에 들어온 새로운 멤버는 다른 멤버들과 즉시 상호작용을 할 수 없기 때문에, 커뮤니티로부터 결함이 발생한 멤버의 온톨로지를 전송 받아 추론하여 자신의 상태를 결함이 발생한 멤버의 상태로 전환시킨다. 이 과정이 끝나면, 새로운 멤버는 자연스럽게 다른 멤버들과 상호작용을 할 수 있으며, 커뮤니티는 목표 달성을 위해 계속적으로 나아간다. 본 논문에서는 이 알고리즘의 빠르고 효율적인 수행을 위해 OWL(Web Ontology Language)로 기술된 컨텍스트 온톨로지를 사용하였으며, Jade 에이전트 플랫폼을 이용하여 제안한 방법을 실험, 분석 하였다.

핵심어: Community Computing, Fault Tolerant System, Replication, Context

1. 서론

현재, 유비쿼터스 컴퓨팅과 그에 관한 연구는 그 의미 그대로 어느 곳에나 존재하며, 각 분야에서 활발하게 진행 중이다. 그 연구의 일환으로 커뮤니티 컴퓨팅[1,2]은 유비쿼터스 환경에 존재하는 컴퓨터, PDA 등 스마트 객체들을 이용하여 지능적인 서비스를 제공하는 시스템 구축을 위한 새로운 패러다임으로 제안되었다. 유비쿼터스 환경에서 발생하는 문제 중 많은 부분은 객체들 사이의 협력을 통해 해결 가능하며, 이러한 시스템을 구축하는 것은 단일 시스템을 구축하는 것보다 많은 비용과 노력이 요구된다. 따라서 커뮤니티 컴퓨팅에서 제공되는 협력 시스템 구축을 위한 방식 및 개발 도구는 이러한 요구를 반영한 긍정적인 연구 성과로 판단되며, 꾸준히 연구되고 있다.

커뮤니티 컴퓨팅에서 커뮤니티는 달성해야 할 목표를 가지고 있으며, 이 목표 달성을 위해 멤버들을 커뮤니티로 구성하고, 이 멤버들은 목표달성을 위해 서로 상호작용한다. 상호작용 과정에서 한 멤버의 결함이 발생한다면, 상호작용이 끊기게 되고 결국 커뮤니티의 목표달성은 실패하게 된다. 따라서 멤버 결함을 극복하고, 끊김 없는 상호작용을 가능하게

하기 위해서는 멤버의 결함을 허용하는 커뮤니티 컴퓨팅에 관한 연구가 필수적이다.

협력 시스템에서 결함 허용에 관한 연구는 멀티에이전트 시스템에 관한 연구에서 찾아볼 수 있다[4-8]. 이들 연구는 소프트웨어 결함 연구에서 가장 효율적인 방법으로 평가되고 있는 Replication 기법[3]을 응용하여 에이전트 시스템의 결함을 극복하고 있다. 이러한 연구들은 커뮤니티 컴퓨팅의 결함 허용 연구에 활용 수 있으나, 몇 가지 차이점이 존재한다. 가장 중요하게 커뮤니티 컴퓨팅은 정적인 공간에 존재하는 에이전트를 가정하는 것이 아니라, 모바일 장치 등 동적인 멤버를 가정하기 때문에 존재하는 Replication 기법만으로는 커뮤니티 컴퓨팅에서의 멤버 결함을 극복할 수 없다.

본 논문에서는 이 문제를 해결하기 위하여 커뮤니티 컴퓨팅에서의 컨텍스트 온톨로지[11]를 이용한 Replication 방법을 제안한다. 컨텍스트 온톨로지는 커뮤니티를 구성하는 객

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

체 및 시스템에 필요한 컨텍스트를 표현하고 관리하기 위한 온톨로지로서 OWL(Web Ontology Language)[2]로 기술되어 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 멀티에이전트 시스템에서의 결함 허용 방법들을 살펴보고, 3장에서는 커뮤니티 컴퓨팅 및 커뮤니티 컴퓨팅에서의 결함에 대해 기술한다. 4장은 본 논문에서 제안하는 컨텍스트 온톨로지를 사용한 Replication 방법에 대해 상세히 기술한다. 5장에서는 제안한 Replication 방법을 시뮬레이션을 통해 평가한 후 마지막으로 6장에서 본 연구의 결론을 기술하도록 하겠다.

2. 멀티에이전트 시스템에서의 결함 허용

제안하는 결함허용 방법을 기술하기 전에, Replication 기법에 기반한 멀티에이전트 시스템(Multi-Agent System, MAS) 에서의 결함허용 방법을 간략하게 살펴보기로 한다.

2.1 멀티에이전트 시스템에서의 결함허용

협력 시스템에서의 결함허용 연구는 MAS 시스템을 이용한 다양한 결함허용 연구 및 평가에서 찾아볼 수 있다. MAS 시스템에서 결함 허용은 Replication 기법을 중심으로 연구되어 왔으며[3], 아래에서 설명할 세 연구 결과는 MAS에서 대체 기법을 활용한 대표적인 결함 허용 방법을 보여준다. 이들은 에이전트의 연속적인 행동의 보장에 초점을 맞추고 있으며, 동기화, 자원 관리 등에 관한 연구를 포함한다. FATMOS(fault-tolerant mobile agent system)[4,5]는 시스템의 한 부분이나 특정 에이전트에서 결함이 발생할 때 나타나는 blocking 현상을 방지하기 위한 방법으로 'agreement problem의 연속으로서 에이전트를 실행하는 방법을 제안하였다. DARX[6]는 동적으로 에이전트 결함 복구를 가능하게 하기 위한 프레임워크로, 에이전트 시스템의 그룹 정보를 유지하고, 시스템 재시작 전략 등을 제공한다. 본 논문에서 제안하는 대체(Replication) 방법은 DARX의 그룹 정보 유지와 유사한 전략을 사용한다. 마지막으로 에 CHAMELEON[7,8] 소프트웨어나 하드웨어의 결함 정보를 유지하여, 결함을 복구하는 방법을 제공하고 있다.

MAS에서 결함 복구를 가능하게 하는 연구는 협력 시스템에 적용하여, 협력 과정의 결함을 극복하여 목표를 달성하게 하는 데 도움을 줄 수 있다. 그러나 기존의 연구들에서 다루는 에이전트는 대부분 정적인 특성을 지니고 있다. 즉, 한 시스템에 고정되어 있거나, 모바일 디바이스 등 이동성을 갖는 장치에 존재한다 하더라도, 장치 내에서의 결함 극복에 초점을 둔다는 제한이 있다. 그러나 다음 절에서 설명할 커뮤니티 컴퓨팅에서는 정적인 에이전트를 가정하는 것이 아니라, 모바일 디바이스 그 자체의 성격 그대로 동적인 성격을 갖는다. 즉 커뮤니티에서 에이전트의 결함은 에이전트가 포함된 장치에서 복구될 수 없다. 따라서 본 절에서 살펴본, Replication의 방법만으로는 커뮤니티 컴퓨팅에서의 동적인 결함 복구를 완성할 수 없다. 본 연구에서 제안하는 Replication 방법의 평가에 있어, 위 연구들에서 자신들이 제안한 방법의 효율성을 입증하기 위해 도입한 항목들을 반영하여 평가하도록 하겠다.

[2] http://www.w3c.org/2004/OWL/

3. 커뮤니티 컴퓨팅

3.1 커뮤니티 컴퓨팅의 개요

커뮤니티 컴퓨팅은 유비쿼터스 환경에서 다양한 스마트 장치들을 이용한 협력 시스템을 구축하기 위한 새로운 방법으로 제안되었다. 미래의 유비쿼터스 환경에 존재하게 될 장치들은 스스로 지능적인 서비스를 제공할 수 있지만, 하나의 장치만이 아니라 여러 장치들의 협력으로 문제를 해결하고 서비스를 제공해야 할 필요가 있다. 커뮤니티 컴퓨팅은 이러한 시스템을 구축함에 있어 커뮤니티라는 메타포를 사용하여 서비스를 제공함을 목표로 하고 있다. 커뮤니티 컴퓨팅은 표 1과 같은 개념을 정의하고, 이를 이용하여 커뮤니티 컴퓨팅 시스템을 구성한다.

표 1 커뮤니티 컴퓨팅에서 사용되는 개념

용어	설명
Society	시스템이 개발되는 공간으로서 유비쿼터스 환경을 가정하며, 커뮤니티와 멤버를 포함
Community	달성해야 할 목표를 가지는 협력체로, 멤버들을 포함하며, 이 멤버들이 수행해야 할 협력 과정을 포함
Member	유비쿼터스 환경에 존재하는 스마트 객체로서 커뮤니티의 목표를 달성해 가는 주체
Role	커뮤니티로부터 멤버에게 할당되는 역할로서 각 역할들의 협력으로 커뮤니티의 목표가 달성됨
Goal	커뮤니티가 달성해야 할 목표
Protocol	목표달성을 위해 멤버들이 협력해야 할 과정

커뮤니티 컴퓨팅 시스템은 위와 같은 개념을 사용하여, 유비쿼터스 환경에서 발생하는 문제를 동적으로 해결하는 방법을 제공한다. Society는 일반적인 유비쿼터스 환경을 가리키며, 다양한 문제가 발생하고 서비스가 요구되는 공간이다. 문제가 발생하면 Society는 문제를 해결할 수 있는 Community를 동적으로 생성한다. 생성된 Community는 자신이 해결해야 할 목표를 가지며, 이 목표를 해결하기 위해 필요한 멤버들을 커뮤니티로 불러들인다. 즉 커뮤니티의 목표를 해결하기 위해 특정 역할을 수행할 멤버들을 불러 들이며, 역할을 맡은 멤버들에게 각자가 해야 할 일(Protocol)을 전송한다. 멤버들이 해야 할 일은 멤버 각자의 독립된 행동이 아니라, 커뮤니티의 관점에서 목표를 달성하는 협업의 개념이 된다. 멤버들은 자신의 서비스를 수행함과 동시에 다른 멤버들과 통신을 해가며 커뮤니티의 목표를 향해 나아간다. 이러한 맥락에서 멤버들의 협력은 커뮤니티의 목표 달성을 위한 중요한 요소이다. 그림 1은 이와 같은 과정을 Flow

Diagram으로 표현한 것이다.

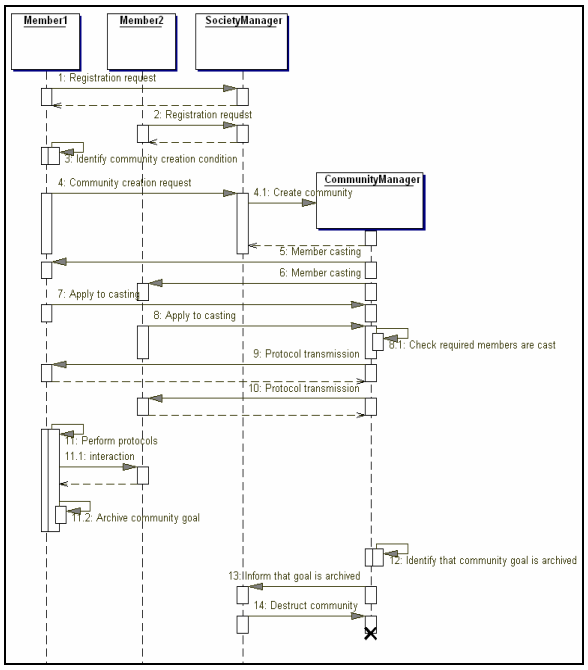

그림 1 커뮤니티 컴퓨팅 시스템 수행 과정

3.2 커뮤니티 컴퓨팅에서의 결함

커뮤니티 컴퓨팅 응용 시스템에서의 결함은 그림 1에 나타난 어느 과정에서라도 발생할 수 있다. 작업을 수행하는 객체의 관점에서 살펴보면, 먼저 커뮤니티와 커뮤니티에 속한 모든 멤버를 관리하는 커뮤니티 서버 혹은 커뮤니티 관리자의 결함이다. 커뮤니티 관리자의 결함은 의미 그대로 커뮤니티에서 일어나는 모든 작업의 중단을 의미한다. 각 멤버들이 협업을 한다 해도, 커뮤니티 관리자의 결함은 커뮤니티의 목표 달성 여부를 판단할 수 없을 뿐 아니라, 커뮤니티에 속해 있는 멤버의 상태도 변경할 수 없기 때문에 중요한 결함으로 작용한다. 두 번째로, 커뮤니티에 참여하는 멤버의 결함이다. 멤버의 결함은 단순히 멤버 역할을 하는 에이전트 혹은 객체의 결함을 의미하는 것이 아니라, 멤버 역할을 하는 디바이스 자체의 결함을 의미한다. 전기적 이유 혹은 통신과 관련된 이유로 멤버의 결함이 발생하면, 커뮤니티에서는 그 멤버와 통신할 수 없고 결국 멤버의 협업으로 목표를 달성하게 되는 커뮤니티는 제 역할을 할 수 없게 된다. 이처럼 멤버의 결함 역시 커뮤니티 목표달성을 가로막는 중대한 요인이 된다.

한편 커뮤니티 관리자의 결함은 이전 장에서 살펴본 기존의 Replication 방법으로도 해결 가능함을 알 수 있다. 커뮤니티 관리자는 이동성이 있거나 동적으로 변하는 성격을 지니는 것이 아니라, 정적인 영역, 이를 테면 서버 컴퓨터에 존재하여 커뮤니티를 관리하는 역할을 수행하게 된다. 따라서 존재하는 Replication 방법을 응용하여 커뮤니티 관리자의 결함을 극복해 낼 수 있다. 그러나 위 장에서 언급한대로 멤버의 결함은 기존의 Replication 방법만으로는 극복될 수 없다. 따라서 본 논문에서는 커뮤니티 관리자의 결함은 제외하고, 커뮤니티에 참여하는 멤버 결함의 극복 방안에 대해서만 다루고자 한다.

4. 컨텍스트 온톨로지를 이용한 Replication 알고리즘

본 장에서는 이미 설명한 멤버의 결함을 극복하기 위한 Replication 방법에 대해서 기술하도록 한다. Replication은 커뮤니티 컴퓨팅 시스템에서 응용되는 컨텍스트 온톨로지를 기반으로 하고 있는데, 컨텍스트 온톨로지는 커뮤니티에 존재하는 각 객체 및 협업에 관한 개념을 포함하고 있다. 먼저 컨텍스트 온톨로지에 대해 간략히 살펴본 후, 제안하는 Replication 방법에 대해 설명하고자 한다.

4.1 컨텍스트 온톨로지

컨텍스트에 관한 연구는 지식표현 연구의 하나로 오랜 기간 연구되어 왔다[10,16]. 특히 최근 유비쿼터스 환경 구축에 관한 연구가 활기를 띠면서, 컨텍스트 관리 및 컨텍스트 인지는 유비쿼터스 컴퓨팅 시스템의 필수적인 요소로서 주목받고 있으며, 활발히 연구되고 있다. 컨텍스트 관련 연구는는 컨텍스트 표현 및 관리를 위한 모델, 아키텍쳐, 인지 시스템 등 다양한 주제로 광범위하게 진행되고 있으며, 특히 시맨틱 웹에 관한 연구에서 새로이 부각된 온톨로지를 활용한 연구가 큰 성과를 보이고 있다[12-15]. SOUPA[12,13], GAS[14] 등은 유비쿼터스 환경에 적합한 컨텍스트 온톨로지로 주목 받고 활용되고 있다.

커뮤니티 컴퓨팅 연구의 일환으로 저자는 커뮤니티 컴퓨팅을 위한 컨텍스트에 관한 연구를 진행한바 있는데[11], 컨텍스트 온톨로지에는 커뮤니티 모델에 속한 각 객체 및 협업을 기술하기 위한 개념들을 포함하고 있다. 또한 실제 시스템 구축에 필요한 시간, 사람, 공간 등의 개념을 포함하기 위해서 SOUPA, GAS 등에서 정의한 개념들을 재사용하였다. 그림 2는 이 컨텍스트 온톨로지의 주요 개념들과 관계들을 보여주고 있으며, 점선 원으로 표현된 부분은 다음 절에서 설명한 Replication에 사용되는 개념들을 나타낸다.

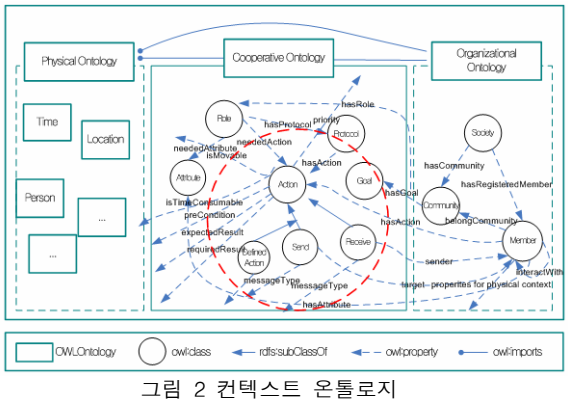

그림 2 컨텍스트 온톨로지

4.2 컨텍스트 온톨로지를 활용한 Replication

본 논문에서 다루는 멤버의 결함은 간단한 방법으로 발견

될 수 있다고 가정한다. 이를 테면, 두 멤버가 통신을 하고 있을 때, 한 멤버가 다른 멤버의 신호를 일정 시간이 지나도록 받지 못하는 경우, 이 멤버에 결함이 있다고 커뮤니티에 보고 하는 방법을 생각할 수 있다. 물론 지능적인 감시 시스템 등을 고안하여 활용할 수 있으나, 이 문제는 본 논문의 주요 주제가 아니므로 다루지 않기로 한다.

Replication 방법에는 위 절에서 기술한 컨텍스트 온톨로지가 활용된다. 컨텍스트 온톨로지에는 객체를 표현하기 위한 개념 외에도, 멤버들의 협력이나 멤버가 수행해야 할 행동들을 기술할 수 있는 개념들이 포함되어 있는데, 이 개념들을 활용해 멤버의 상태를 저장하는 것이다. 그림 3은 Replication에 사용되는 주요 개념들을 OWL로 쓰여진 예를 보여주고 있다.

```
<owl:Class rdf:ID="Step"/>
  <owl:Class rdf:ID="Receive">
    <rdfs:subClassOf rdf:resource="#Action"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="sender">
    <rdfs:domain rdf:resource="#Receive"/>
    <rdfs:range rdf:resource="#Person"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="receiver">
    <rdfs:range rdf:resource="#Person"/>
    <rdfs:domain rdf:resource="#Send"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="sentMessage">
    <rdfs:domain rdf:resource="#Send"/>
    <rdfs:range rdf:resource="#Message"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="receivedMessage">
    <rdfs:range rdf:resource="#Message"/>
    <rdfs:domain rdf:resource="#Receive"/>
  </owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="content">
    <rdfs:range rdf:resource="…/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Message"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=
"isTimeConsumable">
    <rdfs:range rdf:resource="…/XMLSchema#boolean"/>
    <rdfs:domain rdf:resource="#Action"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="isMovable">
<rdfs:range rdf:resource="…/XMLSchema#boolean"/>
    <rdfs:domain rdf:resource="#Action"/>
</owl:DatatypeProperty>
```

그림 3 Replication 에 사용되는 개념들의 OWL 표현

이제 Replication의 방법의 주요 내용을 설명하고자 한다. 표기 및 알고리즘은 그림 4을 참고하기 바란다.

(1) Replication을 위해서 커뮤니티는 모든 멤버들의 상태를 컨텍스트 온톨로지 형태로 유지 한다. 즉 각 멤버는 자신이 특정 행동을 한 후 그 내용을 컨텍스트 온톨로지를 활용하여 작성한 후 커뮤니티에 전송하게 되고, 커뮤니티는 이 정보를 유지, 갱신 함으로서 각 멤버의 Replica를 유지한다.

(2) 특정 멤버의 결함을 발견한 멤버는 커뮤니티에 멤버의 결함을 알린다.

(3) 멤버의 결함을 보고 받은 커뮤니티는 이 정보를 커뮤니티의 모든 멤버에게 알리게 되고,

(4) 정보를 전송 받은 멤버들은 모든 행동을 멈추게 된다. 한 멤버의 결함이 커뮤니티 전체에 즉각적인 영향을 주는 것은 아니다. 왜냐하면, 결함이 발생한 멤버와 통신을 하지 않는 멤버는 아무 문제 없이 자신의 일을 해 나갈 수 있기 때문이다. 그러나 통신을 하지 않는다고 하더라도 문제는 발생될 수 있다. 결함이 발생한 멤버로부터 자유로운 멤버라 하더라도, 결함이 발생한 멤버에 의존하는 또 다른 멤버와 통신을 한다면 다른 행동을 하고 결과가 다르게 나올 수 있다. 또한 커뮤니티 전체에서 공통적으로 사용하는 자원이 결함이 발생한 멤버에 강한 관계를 갖는 경우라면 그 자원을 사용하는 모든 멤버는 다른 결과를 낼 수 있다. 따라서 본 연구에서는 멤버의 결함이 발생하면 커뮤니티에 속한 모든 멤버의 행동을 중지하도록 하였다. 물론 이 방법은 멤버의 행동에 제약을 가하여 다른 문제를 발생시킬 수 있으나, 이는 차후 의존 그래프 (Dependency Graph)등의 방법으로 해결 가능하리라 여겨진다.

(5) 커뮤니티는 결함이 발생할 멤버의 역할을 수행할 또 다른 멤버를 찾게 된다. 대신할 멤버를 구하는 것은 초기 커뮤니티가 구성될 때와 마찬가지의 방법으로 진행된다.

(6) 적당한 멤버가 구해지면, Replication을 위해 커뮤니티는 자신이 유지하고 있는 결함이 발생한 멤버의 온톨로지, 즉 Replica를 새로운 멤버에게 전송한다.

(7) Replica를 전송 받은 새로운 멤버는 온톨로지를 해석하여 자신의 상태를 변경한다. 새로운 멤버는 결함이 발생한 멤버를 대신할 수 있는 능력이 있다 하더라도, 내부 상태나 컨텍스트가 동일할 수 없다. 위치나 내부 변수 등이 다를 것이기 때문에, 이 상태로는 커뮤니티에 바로 참여하여 다른 멤버들과 협업할 수 없다. 따라서 새로운 멤버는 전송 받은 온톨로지를 해석하여 변경할 필요가 있는 자신의 상태를 바꾸게 된다. 주목할 점은 이 과정이 다름 아닌 협업의 일부라는 것이다. 즉 다른 멤버와의 통신이나 특별한 입력 없이, 결함이 발생한 멤버가 협업을 했던 방식 그대로 행동하는 것이며, 통신이나 기타 행동에 소요되는 시간을 없앨 수 있기 때문에 빠른 시간에 진행 될 수 있다.

(8) 새로운 멤버의 상태가 바뀌었다는 보고를 받은 커뮤니티는 모든 멤버들에게 새로운 멤버의 정보를 전송하고,

(9) 새로운 멤버를 포함한 모든 멤버는 이제 계속적으로 협업하여 커뮤니티의 목표를 향해 나아간다.

Definition

M_i: a Member in Community

M_f: a Member that meets failure

M_r : Replica (as a new cast member)
C : Community consists of M_i $(0<i\leq n)$
O_i: an Ontology for M_i

Replication Process

(1) For all i $(0 < i \leq n)$, M_i sends process information when each process finishes and C receives them and updates O_i.
(2) C detects a failure of M_f $(0 < f \leq n)$ M_j $(0 < j \leq n, i \neq f)$ who occurs time-out on interacting with M_f reports about failure M_f to C
(3) C advertises 'M_f failure' to all members in C
(4) M_i $(0 < i \leq n, i \neq f)$ stops all actions.
(5) C casts M_r who can act exactly same as M_f and sends role.
(6) C sends O_f to M_r
(7) M_r does needed actions by interpreting O_f.
(8) C broadcasts information of M_r to M_i $(0 < i \leq n, i \neq f)$ in C
(9) C and M_i $(0 < i \leq n, i \neq f)$ including M_r work forward their goal

그림 4 Replication 알고리즘

5. 시뮬레이션

본 장은 제안한 Replication 방법을 평가하기 위한 시뮬레이션에 관한 내용을 담고 있다. 시뮬레이션을 위하여 실 세계에서 가능한 시나리오를 생성하고, 시나리오에서 나타난 문제를 해결하기 위한 커뮤니티 컴퓨팅 시스템을 구현한 후 이 시스템의 특정 멤버에 결함을 주입하여 해결하는 과정을 평가하였다.

5.1 시나리오

아파트 놀이터에서 한 아이가 놀고 있다가, 예쁜 나비를 보고 쫓아가기 시작한다. 한참을 쫓아가던 아이는 자신이 처음 와본 장소에 위치하게 되고, 아이는 집으로 돌아가는 길을 몰라 헤매고 있다. 이 상황에서 아이가 차고 있는 스마트 벨트는 아이의 위치를 파악하여 위험한 상황임을 판단하고 아파트 관리소에 아이를 집으로 데려다 줄 것을 요청한다. 요청을 받은 관리소는 아이를 집으로 데려가기 위한 커뮤니티를 생성한다. 생성된 커뮤니티는 목표 달성을 위해 아이의 엄마, 이웃들 및 아파트에 설치되어 있는 CCTV 등 스마트 객체들을 커뮤니티 멤버로 불러들인다. 멤버들은 정해진 대로 행동하여 아이들을 무사히 집으로 데려오게 되고, 커뮤니티는 목표를 달성하고 해체된다.

5.2 시뮬레이션을 통한 평가

위의 시나리오를 커뮤니티 컴퓨팅 시스템으로 구현하기 위해 본 논문에서는 JADE 에이전트 플랫폼을 이용하였다. 커뮤니티 및 각 멤버는 JADE 에이전트로 구현하고, 이들의 행동은 에이전트의 Behavior로 구현하였다. 시스템을 구현한 후 이웃 멤버에게 시스템 시작 후 각각 12초(#1), 14초(#2), 20초(#3), 그리고 32초(#4)에 결함을 발생시켰다. 그림 5는 결함이 발생하지 않은 경우(#0)와 각각 다른 시간에 결함을 발생시킨 다섯 번의 시뮬레이션 한 결과를 총 소비 시간으로 비교한 것이다. #0부터 #3의 그래프는 유사한 총 수행시간을 보여주고 있다. 이는 결함이 발생한 경우 결함을 복구하고 시작한 시간이 길게 걸리지 않음을 의미한다. 그러나 #4의 경우 다른 결과와 비교를 했을 때 10초 이상의 수행 시간의 차이를 보였다. 실험 결과를 분석한 결과 #4의 경우 결함이 발생하기 전에 이웃 멤버가 거리를 이동한 행동을 취하였다. 즉 결함이 발생하고 이를 대체할 멤버 역시 거리를 이동할 행동을 취하여야 했고, 이는 온톨로지를 해석함으로써 빠르게 결함을 복구할 수 없고 실제 시간을 투자하여 복구될 수 있는 성격을 갖는다.

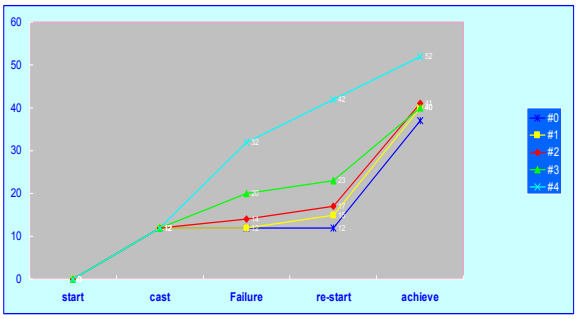

그림 5 시뮬레이션 결과

시뮬레이션 결과 온톨로지를 이용한 결함 복구는 많은 시간을 필요로 하지 않았으며, 복구 된 후 멤버들의 협업은 끊김 없이 이어짐을 확인하였다. 그러나 이동을 하는 행동이나 시간을 투자하여야 하는 행동에 대해서는 제안하는 방법으로는 어쩔 수 없이 복구에 같은 시간을 필요로 한다. 표 2는 제안한 방법을 MAS 사용하는 평가항목들을 이용하여 비교한 결과이다.

표 2 제안한 Replication 방법 분석 결과

issue	description
non-blocking	- yes
exactly-once property	- yes, in community point of view
object of replication	- all members in community
architecture for fault tolerance	- architecture with ontology interpreter, reasoning engine with replicator, and context-related components - strongly based on context ontology
replication cost	- low - interpreting ontology and recovering - some recovery delay (time-consumable and/or movable actions)
multi-replication	- no

6. 결론 및 향후 연구

커뮤니티 컴퓨팅에서의 멤버 결함은 멤버 사이의 협업을

중단시켜 커뮤니티 목표 달성을 가로막는 중요한 문제이다. 본 논문에서는 이러한 멤버 결함을 극복하기 위해 컨텍스트 온톨로지를 이용한 Replication 방법을 제안하였다. Replication은 각 멤버들의 컨텍스트 온톨로지를 커뮤니티가 유지함으로써 멤버의 정보를 저장하고, 결함이 발생하였을 때 결함이 발생한 멤버를 대체할 멤버에게 온톨로지를 전송하여 결함을 복구하게 된다. 이 과정은 온톨로지를 해석하여 빠르게 수행되고, 끊김 없는 멤버들의 협력을 보장한다.

그러나 본 연구는 커뮤니티 컴퓨팅에서의 멤버결함에 관한 초기 연구로서, 실제 시간이 소비되는 행동에 대한 처리 등 보완해야 할 문제가 존재한다. 또한 제안한 Replication 방법을 확장하여 일반적인 MAS 시스템으로의 확장도 향후 연구로 남긴다.

참고문헌

[1] Yuna Jung, Jungtae Lee, Minkoo Kim, "Multi-agent based Community Computing System Development with the Model Driven Architecture", In Proceedings of Fifth International Joint Conference on Autonomous agents and Multiagents Systems(AAMAS-06), Hakodate, Japan, 2006

[2] Youna Jung, Jungtae Lee and Minkoo Kim, "Community based Ubiquitous System Development in Multi-agent Environment", CAiSE'06 Workshop Ubiquitous Mobile Information and Collaboration Systems(UMICS 2006), Luxembourg, Grand-Duchy of Luxembourg, 2006

[3] R. Guerraoui and A. Schiper, "Software-based replication for fault tolerance", IEEE Computer, 30(4):68-74, April, 1997

[4] Stefan Pleisch, Andre Schiper , "FATOMAS-A Fault-Tolerant Mobile Agent System Based on the Agent-Dependent Approach", The International Conference on Dependable Systems and Networks (DSN'01), pp. 215, July 2001

[5] Pleisch, S. Schiper, A., "Fault-tolerant mobile agent execution" Computers, IEEE Transactions on Volume 52, Issue 2, pp. 209-222, Feb. 2003

[6] Olivier Marin, Pierre Sens, Jean-Pierre Briot and Zahia Guessoum, "Towards Adaptive Fault Tolerance for Distributed Multi-Agent Systems", European Research Seminar on Advances in Distributed Systems (Ersads2001), , Bertinoro (Forlì), Italy May 2001

[7] Benno Overeinder, Frances Brazier, Olivier Marin, "Fault Tolerance in Scalable Agent Support Systems: Integrating DARX in the AgentScape Framework," ccgrid, p. 688, 3rd International Symposium on Cluster Computing and the Grid, 2003.

[8] S. Bagchi, K. Whisnant, Z. Kalbarczyk, R.K. Iyer, "Chameleon: A Software Infrastructure for Adaptive Fault Tolerance", IEEE Transactions on Parallel and Distributed Systems, 10(6):560-579, 1999

[9] G. Cabri, L. Ferrari, L. Leonardi. "Agent Role-based Collaboration and Coordination: a Survey About Existing Approaches", In Proceedings of the 2004 IEEE systems, Man and Cybernetics Conference, Netherlands, 2004

[10] Anind K. Dey, "Understanding and Using Context", Personal and Ubiquitous Computing, Special Issue on Situated Interaction and Ubiquitous Computing 5, 1, 200

[11] Minsoo Kim, Youna Jung, Jungtae Lee, Minkoo Kim, "Context-based cooperation architecture in ubiquitous environment", In proceedings of International Symposium on Ubiquitous Computing System (UCS2006), Seoul, Korea

[12] Harry Chen, Tim Finin, and Anupam Joshi, "The SOUPA Ontology for Pervasive Computing", Ontologies for Agents: Theory and Experiences, Springer, July 2005

[13] Xiao Hang Wang, Tao Gu, Da Qing Zhang, Hung Keng Pung, "Ontology Based Context Modeling and Reasoning using OWL", In Proceedings of Workshop on Context Modeling and Reasoning(CoMoRea 2004), In conjunction with the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 2004), Orlando, Florida USA, March 2004

[14] Christopoulou E., Kameas A., "GAS Ontology: an ontology for collaboration among ubiquitous computing devices", Protege special issue of the International Journal of Human - Computer Studies, 2004

[15] T. Strang and C. Linnhoff-popien, "A Context Modeling Survey", In Proceedings of 1st International workshop on Advanced Context modeling, Reasoning and Management UbiComp2004, 2004