자기 상관계수를 이용한 우선순위 기반의 정황인지 협동 시스템 충돌 해결 모델

채희서° 이동현 김도훈 인 호
고려대학교 컴퓨터학과 임베디드소프트웨어 공학 연구실
{royalhs°, tellmehenry, karmy01, hoh_in}@korea.ac.kr

Conflict Resolution Model based on Priority Using Auto-Correlation Coefficient of binary variables in Situation Aware Collaboration System

Heeseo Chae° , DongHyun Lee, Do-Hoon Kim Hoh peter In
Korea Univ. Department of Computer Science, Embedded S/W Engineering Lab

요 약

Situation awareness는 유비쿼터스 환경에서 사용자의 상황에 맞는 적절한 서비스를 제공하기 위한 방법으로, 다양한 context와 action이력을 기술하여 협동 시스템을 표현할 수 있는 Situation Aware-Interface Definition Language(SA-IDL) 접근방법이 유용하다. 하지만, SA-IDL로 다수의 상황인지 서비스를 만들게 될 때 서비스 단독으로는 문제가 없을지라도 여러 개의 서비스들 사이에서는 충돌문제가 발생할 수 있다. 이런 충돌을 검출해내고 해결하기 위해서, 올 수학적으로 모델링된 SA-IDL을 통해, 충돌을 수학적으로 정의하여 기술하고, graph theory적인 접근 방법과 자기 상관계수를 통한 마크호프 예측 기법으로 해결하고자 한다.

1. 서 론

유비쿼터스 환경의 목표는 여러 개의 개별적으로 개발된 시스템들을 상황에 맞게 통합하고 사용자의 편의를 위하여 맞춤형 서비스를 제공하는 것이다[1]. 기존의 환경에는 electronic device들을 사용하기 위해서 각각의 장치들을 따로 조작할 수 밖에 없었고, 여러 기기들을 동시에 이용하려면 기기간의 조율을 사용자가 일일이 다루어야 하는 번거로움이 있었다. 하지만, 모든 electronic device들이 컴퓨팅 기능을 갖거나, 컴퓨터와 연동될 수 있는 유비쿼터스 환경에서는 모든 기기들이 Main Frame로 접근할 수 있기 때문에, 기기들이 어떻게 상호 협동하고 동작할 지만 결정해주면 사용자의 상황에 맞는 디지털 환경을 제공하는 것이 가능해졌다.
따라서 협동 시스템이란 이기종간의 기기에서 사용자의 요구에 맞게 원활한 서비스를 제공하는 플랫폼을 의미하며, 이러한 협동 시스템을 정황인지 접근법으로 접근한 것이 정황인지 협동 시스템이다.

정황 인지를 통하여 기기들의 상호협동을 기술할 수 있는 Situation Aware Interface Definition Language(SA-IDL)은 다중의 Context 이력을 검사할 수 있으며, 수행했던 Action 의 이력도 검사하여 상황인지 서비스를 가능하게 해준다. 또한, 실행할 Action 의 시간을 지정할 수 있어, 보다 정확한 상황포착에 능한 장점이 있다.

사용자가 이러한 SA-IDL로 상황인지 서비스를 기술할 때, 서비스 내부의 충돌과 서비스 사이의 충돌 문제가 발생할 수 있는데, 동일한 상황인지 시기에, 같은 디바이스에서 서로 상이한 명령을 실행 시키는 경우가 충돌이라고 정의할 수 있다.

이 충돌 문제를 확인하기 위해서는 하나의 서비스를 만들 때만 검사하는 것이 아니라 같이 실행되는 다른 모든 서비스와 같이 충돌문제가 발생하는지 확인해야 한다. 하나의 서비스에서는 충돌 문제가 없다고 하더라도, 다른 서비스와는 같은 상황인지 시기에 충돌되는 명령어를 실행 할 수 있기 때문이다.

이와 같은 충돌 문제를 정의하기 위해서, 우선 SA-IDL을 수학적으로 모델링하고, 충돌 문제를 수학 집합 개념으로 정의한다. 또한 서비스간의 충돌을 풀기 위해서, graph theory의 접근방법을 도입하여 노드와 edge로 이루어진 서비스간의 충돌 그래프를 그리고, 이렇게 표현된 서비스간의 충돌 관계를 priority를 기준으로 한 conflict resolution algorithm으로 해결한다. 이러한 알고리즘의 구현은 이진 자기상관계수(Auto-Correlation Coefficient of binary variables)를 이용한 고차 마코프체인(Higher Order Markov Chains)을 도입하여, 사용자의 선호도나 리소스 할당률 등의 서로 다른 변량 사이의 상관관계가 시간에 따라 결정되는 priority의 예측값에 신뢰성을 부가함으로써 그 해법을 제시하게 된다.

2. 정황인지 컴퓨팅(Situation-Aware Computing)

2.1 Overview of Situation Awareness

정황인지(SA, Situation-Awareness)란 행위(Actions)와 다중 문맥(Multiple Contexts)간의 관계성을 시간에 따라 변화 추세 등을 분석하고 이해할 수 있는 특성을 의미한다. 즉, 주변 상황에 따라 디바이스가 자동적으로 적시에 행동을 취할 수 있어야 한다는 개념이다. 다시 말해서 정황인지는 일련의 프로세스가 정해진 시간간격을 두고

이루어지는 것이 아니라 상황에 따라 이루어짐을 의미한다. 따라서 정황인지란 시간, 장소, 또는 사용자에 종속되어 동작하는 것이 아닌 인지된 상황을 추론하여 이 결과를 바탕으로 동작이 이루어진다는 것을 뜻하고 이러한 개념은 잘 알려진 Context Awareness 와 비교함으로써 더욱 분명해진다.

- Context-Awareness: 작동 중인 장치의 현재 문맥과 변화를 문맥상의 전후 관계 데이터를 통해서 발견하는 기능.
- Situation-Awareness: 다수의 문맥관계와 그에 해당되는 시간에 따른 실행 간의 관계를 분석하고 감지하는 기능.

2.2 정황인지 협동 시스템

유비쿼터스 환경에서는 기술이 발전할수록 Device 들 간의 상호 협동이 상황에 따른 사용자의 요구에 부합하기 위해 점점 더 중요해 진다. 각각의 장치는 situation expression을 통해 서비스를 제공하고, 관련 있는 device간의 협동관계를 통해서 다음과 같은 Situation-Aware Collaboration System 을 구성하게 된다[2].

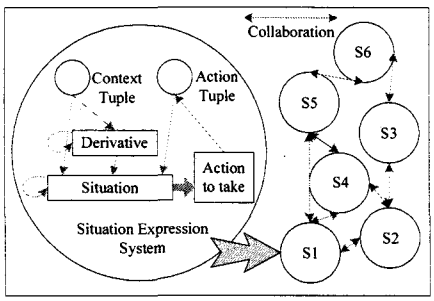

Fig1. Situation-Aware Collaboration System

그림 1과 같은 협동 시스템에서 각각의 Situation Expression System은 context tuple[3], action tuple, situation등으로 이루어져있고 이러한 협동시스템의 예는 다음과 같다. 집안에서 청소기를 작동 시킬 경우, 환기를 위하여 창문이 자동으로 열리는 청소-환기 시스템을 간단히 생각해볼 수 있다. 여기서 서로 다른 기종인 청소기와 환기시스템이 사용자의 상황에 맞추어서 서로 협동하여 동작하는 것을 확인할 수 있지만 이러한 협동관계에서 난방시스템이 추가될 경우 문제가 발생되게 된다. 집안의 난방을 위해서는 자동으로 창문이 닫혀야 되는데 반하여 청소를 위해서는 창문이 열려야 되고, 이렇게 서로 다른 서비스의 목표 때문에 창문이라는 하나의 장치 내에서 서로 다른 서비스간의 충돌이 일어날 수 있는 것이다. 따라서 원활한 collaboration system을 위하여, 이러한 서비스간의 충돌을 확인하고 그 충돌을 해결할 수 있는 방법을 제시해야 하고 각각의 시스템 (청소기, 환기 시스템, 난방 시스템)은 각각의 Situation들과 매칭이 되며 각각의 서비스들은 정황 표현 시스템 안에서 실행되고 각각의 tuple과 action들로 산출된다. 그리고 이러한 Situation들은 환기시스템의 예처럼 충돌이 일어날 수 있으며, 이러한 충돌 관계를 해결할 수 있는 정형기법적인 접근이 다음 절에 제시된다.

3. 정황인지 협동 시스템 충돌 모델

3.1 SA-IDL Model

상황인지를 기술하고, 상황인지 서비스를 만들 수 있는 일반적인 개발 방법론을 제시하기 위해서 SA-IDL 이 사용된다. 앞에서 언급했지만 SA-IDL은 다중의 Context 이력을 검사할 수 있으며, 수행했던 Action 의 이력도 검사하여 상황인지 서비스를 가능하게 해준다. 또한, 실행할 Action 의 시간을 지정할 수 있어, 보다 정확한 상황포착에 능한 장점이 있다. [4]

SA-IDL 은 Context Tuple, Action Tuple, Derived Context의 구성을 갖고 있고, 각각은 다음과 같다. [2]

Action := (Time, Device, State, StateValue)
Context := (Timestamp, Device, State, StateValue)
A := {x | x is an Action}
C := {x | x is a Context}
DerivedContext := P(C) → {true, false}
Situation := (DerivedContext, P(C), A)

Table 1.SA-IDL model

Situation 는 특정상황이 만족하면 일련의 명령을 실행하는 서비스를 기술한다. 이 Situation 을 모델링 하기 위해서 튜플 (), 함수 ->, 집합 {}, power set P() 개념을 사용하였다.

- **Action tuple**: action은 Time, Device, State, StateValue로 구성된 튜플로 나타낼 수 있다. Action 은 Time 시각에 Device 의 State 값을 StateValue로 치환하는 방법으로 표현된다. 예를 들어, 5초 후에 PDA 의 전원을 끄는 Action 은 (5, PDA, Power, Off) 로 나타낼 수 있다. Action 을 튜플로 표현하면, Action 을 전달하기 위해서 tuple space를 사용 할 수 있어서, Action을 heterogeneous 한 Device 들 사이에도 쉽게 공유할 수 있는 장점이 있다.
- **Context tuple**: Context 는 특정 시각에 특정 Device 의 일련의 상태들을 나타낸다. Context 는 또한 실행된 Action 의 이력을 표현한다. 예를 들어, 0초 시각에 PDA 를 켜고, 5초 시각에 program.exe 를 실행시켰다면, 다음과 같이 ActionContext 로 표현된다. (0, PDA, Power, On), (5, PDA, Execute, program.exe)
- **DerivedContext**: DerivedContext 는 특정상황이 만족하는지 기술하는 proper function (total and single-valued) 함수이다. 이 함수의 Domain 은 Context 의 멱집합이다. Range 는 true 또는 false 가 된다. 즉, DerivedContext 는 Context 집합을 입력 받아서, 특정 상황이 유발되었는지 확인하기 위해서 true 또는 false 값을 반환하는 함수이다.
- **Situation tuple**: Situation 는 DerivedContext, Context 파워 집합, Action집합으로 구성된 tuple이다. DerivedContext 가 특정 상황이 만족되는지 표현하며, A는 상황이 만족될 때 실행될 일련의 Action 들의 집합이다.

3.2 충돌의 정의

서로 다른 두개의 Situation 에서 발생할 수 있는 Conflict 는 튜플 (C, Action, Action) 로 표현될 수 있다.

두 개의 Action 은 서로 충돌이 되는 행동이며, C 는 이 두 Action 을 실행시키는 Context 집합이다. Conflict 의 자세한 조건은 수학의 집합 개념을 이용하여 아래와 같이 정의하였다.

```
Conflict := (C, Action, Action)
{∃(c, x, y): Conflict; ∀a, b: Situation |
   c ∈ a.P(C) ∩ b.P(C)
   and c ≠ Ø
   and a.DerivedContext (c)
       = true and b.DerivedContext (c) = true
   and x ∈ a.A and y ∈ b.A
   and x.Time = y.Time
   and x.Device = y.Device
   and x.State = y.State
   and x.NewState ≠ y.NewState
}
```

Table 2.Definition of Conflicts

Conflict 는 하나의 Context 집합과 두 개의 Action으로 구성된 tuple이다. 이 세 tuple구성원이 크게 두 가지 조건으로 정의된다. 첫 번째는, Situation a, b 의 상황을 만족시키는 Context 집합을 찾는 부분이다. 두 번째는, 서로 상반된 두 Action 을 찾는 부분이다.

Conflict (c, x, y) 는 임의의 Situation a, b 에서 위와 같은 조건이 만족한다면 존재한다. Context 집합 c 는 Situation a 의 DerivedContext 의 도메인에 속하며, Situation b 의 DerivedContext 의 도메인에 속하는 공집합이 아닌 원소이다. 그리고 c 가 발생했을 때, DerivedContext 는 상황을 만족시킨다. c 는 충돌되는 두 Action x, y 를 실행시킨다. x 는 a 상황의 Action 집합의 원소이고, y 는 b 상황의 Action 집합의 원소이다. 두 Action x, y 는 같은 시각에 같은 Device 의 같은 상태를 서로 다른 값으로 바꾼다.

4. 충돌 해결

사용자는 다양한 device들을 협동시켜 SA-IDL로 기술된 service를 제공하고 이 service들은 하나하나 보았을 때에는 무리 없이 실행될 수 있지만, 이러한 무결점의 service도 통합되면 service 간의 논리적인 오류로 인해 충돌이 발생할 수 있다. 이러한 충돌을 define 하고 해결하기 위해서 각각의 situation를 node로 설정하고 서비스간에 발생되는 conflict를 edge로 표현하는 graph적인 기법을 도입하도록 한다. 이러한 충돌 해결의 자세한 접근법과 해결 알고리즘을 이번 절에서 제시한다.

4.1 충돌 확인

SA-IDL의 수학적 모델링을 통해서 표현된 Conflict는 다음과 같은 graph적인 방법으로 표현될 수 있다[5]. 개별적인 situation은 각각의 node로, situation 간의 conflict는 node를 연결하는 edge로 표현 할 수 있다. 이러한 표현기법은 situation간의 논리적 오류로 인해 발생되는 conflict를 분명하게 잘 표현해 주고 있고, 하나의 situation과 situation 간의 conflict이나 여러 situation의 조합으로 일어나는 conflict관계 또한 명시적으로 보여준다.

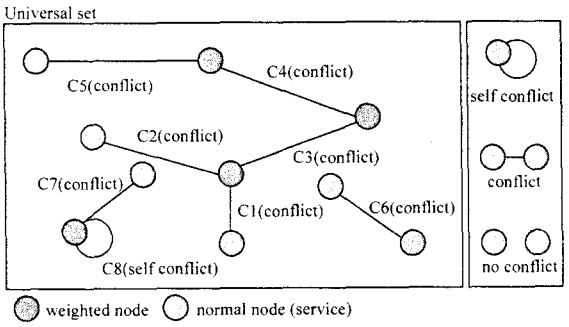

Fig2. Conflict Graph

그림 2과 같은 conflict identification은 다음과 같은 절차를 통해 표현된다.

- Situation 과 Node 의 Mapping
  main device 상의 활성화된 situation를 표현하는 방법은 새롭게 진입한 service를 하나의 normal node라고 보고, Main device를 뜻하는Universal set안 쪽에 하나의 node를 나타내는 형태이다.
- Situation 간의 Conflict 표현
  Exhausted search 접근으로 2개의service간, 혹은 그 이상의 service 간의 SA-IDL문법에 어긋나는 논리적인 conflict 부분을 하나씩 찾아내고 conflict이 발생된 node를 서로 연결하여 그 사이의 conflict을 edge형태로 표현한다.
- Node 의 구별
  Conflict의 복잡도를 구분 지어 conflict의 해결을 쉽게 하기 위해서 주변 node들과 2개 이상의 conflict를 발생하는 node를 weighted node라고 명명한다. 이러한 node는 conflict을 많이 발생시키지 않는 normal node와 비교 되고 보다 효과적인 conflict 해결에 많은 도움을 준다.

이렇게 표현된 충돌은 이것을 해결할 수 있는 모델과 알고리즘을 제시하는데 이론적인 도움을 준다. 특히, graph 형태로 나타낸 situation conflict model은 graph theory에 적용 시켜 edge로 표현된 conflict를 제거할 수 있는 학문적인 기반을 마련할 수 있다

4.2 충돌 해결 모델

어떤 방법으로 충돌을 찾아내고 발견된 그 충돌을 해결하기 위해 어떤 해결책을 제공할지는 개연성 있는 상황에 따라 달라지게 된다. 그러한 상황은 충돌이 표현된 graph자체만을 고려하여 가장 conflict 발생 정도가 심한 weighted node를 제거하는 것으로 생각해 볼 수 있고[6], node 마다 할당된 graph외부의 값을 반영하여 해당 node를 제거하는 것으로도 생각해 볼 수 있다. 물론 여기서 외부의 값이라는 것은 node마다 다르게 평가된 우선순위(priority)이고, 이것은 일종의 메타 데이터로써 node에 할당된 resource나 중요도 등을 모두 포함하는 개념이다. 이처럼 상황을 구분하고 그에 맞는 기준과 모델을 제시하는 것은 앞으로 기술될

conflict resolution의 사전 단계로서 매우 중요한 의미를 가진다.

앞에서 보여주었던 그림2을 통해서 충돌 해결 모델을 구축하고 해결방안에 대안 접근 방법 별로 그것을 표현할 수 있다. 이미 언급했지만 주요한 접근 모델은 우선순위를 반영한 모델로서 그 기반이 되는 weight based 알고리즘[6]을 발전시킨 향상된 모델 및 알고리즘이다.

- Priority Node Model

이 model은 conflicts identification graph 자체의 위상 관계뿐 만 아니라 외부조건을 고려하여 priority를 반영한 model이다. 기존과 같이 표현된 node간의 위상관계에 priority value를 반영하여 priority가 낮은 것부터 차례로 node를 제거하는 model이라고 할 수 있다. Priority는 외부 조건의 모든 value, 즉 해당 situation의 중요도나 resource 할당 율 등을 모두 포함하고 있는 meta data이고 그러한 메타 데이터는 누적된 data로부터 얻을 수 있는 사용자의 선호도를 평가하며, 이런 값들을 통해 고차 마코프체인(Higher Order Markov Chains)연산을 실행함으로써 Next Priority값을 산출하게 된다. 또한 이러한 예측 값에 신뢰성을 부가하기 위해 바이너리 값을 이용한 자기상관계수를 초기 백터에 적용한다.

- Priority에 따른 Node 제거

그림 3에서 볼 수 있듯이, Priority에 따른 node 제거는 priority가 낮은 순서로 이루어진다. Subset A는 priority가 1인 node들로 구성되어 있고, subset A의 conflict들이 사라지게 되면 그 다음 순서인 subset B에 대한 처리가 이루어진다. Subset B에 priority가 2인 C7coflict를 발생시켰던 node가 포함되지 않은 이유는 이미 근접 node의 제거로 edge가 자연스럽게 사라졌기 때문이다. 이런 식으로 priority에 따른 node제거가 이루어지고 priority가 같을 경우에는 이전에 먼저 언급된 Weighted Node Model에 기초하여 model이 구성된다.

4.3 자기 상관 계수

파라메터의 수(전이확률)가 기하급수적으로 증가하였을 때 고차 마코프체인을 이용하여 다음 상태를 예측 할 수 있으며 이를 통해 예상되는 priority값을 구할 수 있다. 또한, 이와 같은 자기 상관계수를 통해 예측되는 priority에는 다음과 같은 가정사항이 포함되어있다.

Assumption	Contents
파라메터 수	연결된 situation간, 기존에 적용된 priority의 사용 가능성 참작 - 전이확률의 기하급수적 증가
마코프체인	SA-Personalization[9]의 기존연구에서 사용된 마코프체인의 특수한 종류인 고차 마코프체인을 사용. 기하급수적으로 증가하는 파라메터에 적용

Table 3.Assumption of Auto-Correlation Coefficients

- 자기 상관(auto-correlation coefficients)

X(t)를 하나의 임의의 프로세스로 하여 시각 t_1일 때의 값 $X(t_1)$과 시각 t_2일 때의 값 $X(t_2)$ 사이에 존재하는 상관. 서로 다른 2가지 변량 사이의 상관 관계는 상호 상관관계라고 하고 자기 상관을 표시하는 것으로 자기 상관 함수라는 것이 있으며 $X(t_1) * X(t_2)$의 조화 평균으로 정의된다.

- 이진 자기 상관계수의 parameter

이 논문에서 고려하는 자기 상관계수의 파라메터는 다음과 같다.

Parameter	Contents
BSF	발생되는 Situation의 빈도수를 binary value로 측정 (Binary Situation Frequency)
BRQ	각각의 서비스나 situation에 해당되는 resource 할당 율을 binary value로 측정 (Binary Resource Quotation)

Table 4.Parameter of Auto-Correlation Coefficients

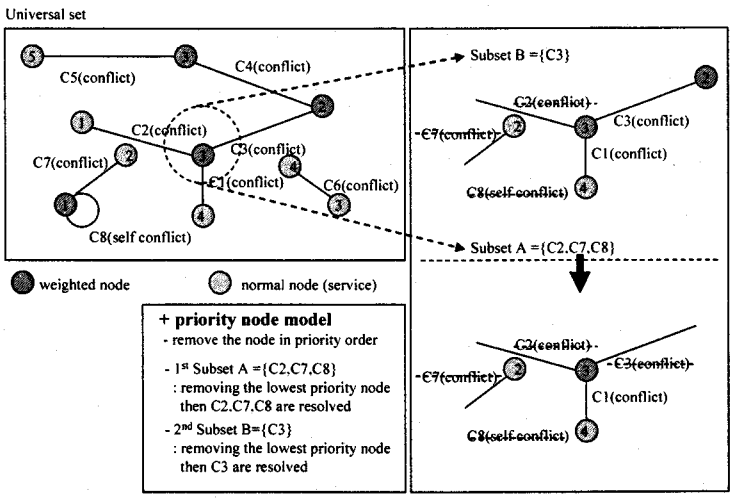

Fig3. Priority Node Model

4.4 이진 자기 상관 계수를 통한 우선순위 설정 방법

우선순위의 설정은 이전의 연구에서 진행되었던 마크호프 체인을 통한 사용자의 선호도 예측 알고리즘[7]을 기반으로 진행된다. 또한 그 기반 위에 이진 자기 상관계수를 적용한 고차 마크호프 체인이 사용자의 사용내력과 리소스 할당에 따른 발전된 사용자 선호도를 산출해 내며, 이때 쓰인 파라메터들의 상관계수는 특별히 다음과 같은 이진 상관계수를 통하여 마크호프체인의 초기 백터에 영향을 미치게 된다.

$$r = \frac{n_{11}n_{22} - n_{12}n_{21}}{\sqrt{n_{1+} * n_{2+} * n_{+1} * n_{+2}}}$$

Fig4. Transitional Frequency Matrix

그림 4에서 n은 각각의 바이너리 자기 상관 계수로 이루어진 행렬의 원소와 대응되며, n_{12} 와 n_{12}는 시스템의 현상유지를 위한 state를 표현하며, 반면에 n_{12}와 n_{21}은 state의 변화를 의미한다.

4.5 충돌 해결 알고리즘

Graph형태로 Modeling된 conflict과 situation의 관계를 토대로 일반화된 conflict resolution을 제시할 수 있다. 기본적으로는 edge를 포함하는 node를 제거함으로써 conflict를 하나씩 수정해나가는 것이 주요 접근방법이며, 본 절에서는 conflict resolution model에서 제시한 우선순위 설정법을 바탕으로 해당되는 resolution algorithm을 논의한다.

- 우선순위 모델을 통한 해결 알고리즘

충돌확인 그래프 자체의 위상 관계뿐 만 아니라 외부조건인 priority를 반영하여 충돌을 해결하고자 하는 것이 이 알고리즘의 핵심이다.

Node의 weight 보다 priority value가 우선적으로 반영되어 우선순위가 낮은 것부터 차례로 node가 제거되고, 우선순위가 같은 값일 경우 차선책으로 weight를 적용[6]하여 node를 관리한다. 이러한 알고리즘을 통해서 남게 되는 node는 많은 수의 node가 아니라 꼭 필요한 node를 위주로 존재하게 된다. 이러한 알고리즘은 그림5에서 나타나듯이 다음과 같은 단계를 통해 진행된다.

- Setp1. node의 priority 설정
  각각의 node들은 외부 조건의 모든 value의 meta data 값, 즉 해당 situation의 중요도나 resource 할당 율 등을 모두 고려하여 자신의 priority를 보여준다. 물론 Priority Node Model Resolution Algorithm 에서도 그 필요성에 의해 weighted node를 색으로 구분하여 표기는 하지만 node에 할당된 숫자는 weight가 아니라 priority value이다.
- Step2. Priority subset 구성
  Priority 별로 subset을 구성하고 가장 낮은 priority를 가지는 node로 이루어진 subset부터 차례로 node를 제거한다. 이러한 node 제거는 가장 낮은 단계의 priority subset에서 edge에 연결된 node가 없어 질 때까지 계속된다.
- Step3. 동등한 Priority일 경우 weight 비교
  동등한 priority를 가진 priority subset안에서는 node와 연결된 edge의 수로 weight를 산출하고, 그 값을 반영하여 subset 내부에서 높은 weight를 가진 node를 우선적으로 제거한다.

$$priority.A = priority.B \ (A,B \in S \ ; \ S \ is \ subset)$$
$$\sum weight_A > \sum weight_B \rightarrow S - A(A \in S)$$

Priority는 낮지만 인접 node의 제거로 Edge가 사라진 node는 priority가 낮음에도 불구하고 제거하지는 않는다.

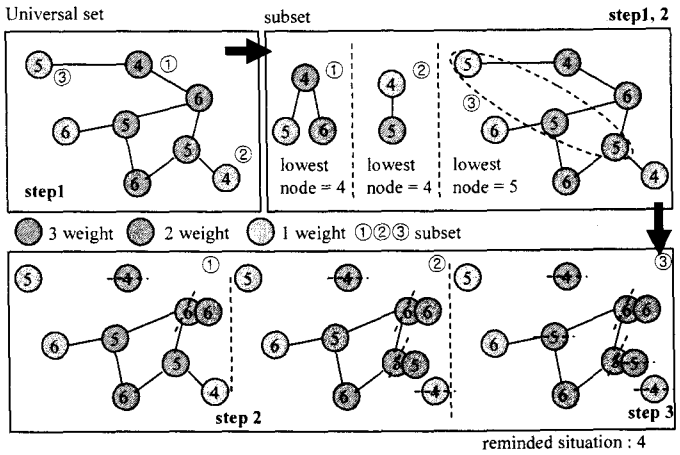

Fig5. Priority Node Model Resolution Algorithm

- Step4. node value 비교 반복
  앞 단계의 node 제거로 변화된 노드를 다시 비교하고 Step3의 단계를 edge가 없어질 때까지 반복한다.

6. 결론

유비쿼터스 환경에서는 모든 기기들이 Main Frame으로 접근할 수 있고 상호 협동을 통해 사용자의 요구를 만족시키게 된다. 하지만 이러한 협동 모델에서도 device 간situation에 대한 논리적 오류로 인해 충돌은 예상된다. 이러한 서비스간의 충돌 문제를 수학적으로 모델링 한 SA-IDL로 정의하고, 충돌을 일으키는 서비스를 graph theory적인 접근 방법으로 검출하여 해결하는 내용을 본 논문에서 다루어보았다. 또한 이전 연구에서 진행되었던 Weighted Node Model Resolution Algorithm[6]을 기반으로 한 Priority Node Model Resolution Algorithm을 자기 상관계수를 적용하여 해결함으로써 보다 향상된 우선순위 설정을 기대할 수 있게 되었다. .

이 논문에서 제시한 충돌 모델링 기법과 해결 알고리즘은 기기간 혹은 서비스간의 복잡도가 기하급수적으로 증가될 앞으로의 유비쿼터스 환경에서 협동 시스템을 동작하고자 하는 사람은 누구나 고려하지 않을 수 없는 중요한 요소가 될 것이다. 이러한 contribution을 확고히 하고 연구를 보완하기 위해 앞으로 좀더 복잡도가 높은 유비쿼터스 환경에서의 알고리즘 적용이 필요하다. 또한 이러한 연구 결과를 토대로 우선순위 설정에 영향을 미칠 수 있는 QoS등의 새로운 상관 파라메터등의 확장으로 보다 신뢰성있는 개선된 연구로, 그 범위를 확장시켜 나가야 할 것이다.

7. Acknowledge
This work was supported in part by MIC & IITA through IT Leading R&D Support Project.

Reference

[1] Dongwon Jeong, Heeseo Chae, and Hoh Peter In. " The Performance Estimation of the Situation Awareness RFID System from Ubiquitous Environment Scenario" International Conference EUC 2005, Nagasaki, Japan, December 6-9, 2005. Proceedings p. 987 - 995
[2] Yau, S.S.; Yu Wang; Dazhi Huang; In, H.P. " Situation-aware contract specification language for middleware for ubiquitous computting.Distributed Computing Systems" FTDCS 2003.Proceedings. The Ninth IEEE Workshop on Future Trends of 28-30 May 2003 Page(s):93 - 99
[3] G. Bollella, S. Graham, and T. Lehman, " Real-time TSpaces" , IEEE 25th Industrial Electron ics Society (IECON 1999), pp. 837-842, 1999.
[4] S. Yau, Y. Wang, and F. Karim, " Developing Situation-Awareness in Middleware for Ubicomp Environments," Proc. 26th Int'l Computer Software and Applications Conference (2002) 233-238
[5] Robin J.Wilson," Introduction to Graph Theory" Prentice Hall (1996)
[6] Heeseo Chae, Taeyoun Kim, Donghyun Lee and Hoh Peter In, " Conflict Resolution Model based on weight in Situation Aware CollaborationSystem" ,(2006), http://embedded.korea.ac.kr/files/research_sa.html#publication_SA
[7] Heeseo Chae, Do-Hoon Kim, Dongwon Jeong, Hoh Peter In, " A Situation Aware Personalization in Ubiquitous Mobile Computing Environment" , (EUC 2006), Seoul, Korea, 1-4 Aug.