상황 인식 응용을 위한 2-레벨 상황 적응 기법

김 수 중[†] · 윤 용 익[††]

요 약

유비쿼터스 환경에서의 응용은 사용자의 요구사항 변경과 사용자의 서비스 환경 변화를 적응시킴으로써 사용자에게 최선의 서비스를 제공할 수 있어야 하며 상황 변화에 따른 하부 플랫폼의 동작에 서비스의 동작을 적응시킬 수 있어야 한다. 본 논문에서는 유비쿼터스 컴퓨팅 환경에서 사용자의 요구사항을 능동적으로 반영하고 사용자에게 유연한 서비스를 제공하기 위하여 상황 변화에 대한 동적인 응용 적응성을 지원하는 2-레벨 상황 적응 기법을 제안한다. 상황 적응성을 지원하기 위하여 상황의 범위를 사용자 요구사항으로 확대하여 응용 서비스 제공에 있어서 사용자의 선호도를 반영할 수 있도록 하였으며, 응용의 적응성은 적응 정책을 사용하여 발생할 수 있는 특정 상황에 대한 적응 방법을 명시적으로 요구함으로써 실제 상황이 발생했을 때 응용이 능동적으로 대처할 수 있도록 하였다.

키워드 : 상황 인식, 동적 적응, 유비쿼터스 컴퓨팅

Two-Level Context Adaptation Method for Context-Aware Applications

Soo-Jong Ghim[†] · Yong Ik Yoon[††]

ABSTRACT

Applications in ubiquitous environments should provide the best services to users by considering the changes of users requirements and service environments, and should adapt service behaviors to the underlying platform's behaviors according to contextual changes. To reflect users requirements actively and provide more flexible services, we propose 2-level context adaptation method for supporting dynamic application adaptability in contextual changes. We extend the range of contexts to users requirements for supporting context adaptation. It can reflect users preferences in offering application services. For application adaptability, we use adaptation policies to allow applications require how they adapt to specific contexts, and to make them react actively on such situations.

Key Words : Context-awareness, Dynamic Adaptation, Ubiquitous Computing

1. 서 론

유비쿼터스 컴퓨팅은 물리적인 공간을 능동적이고 지능적인 환경으로 전환하는 대규모의 분산 시스템의 구축을 지원하며 HCI, 소프트웨어 에이전트, 인공지능, 모바일 기술 등의 여러 분야가 서로 융합되는 복합적인 분야로서 향후 우리의 일상생활에 큰 영향을 미칠 것으로 예상된다. 이러한 유비쿼터스 컴퓨팅의 응용 서비스에서 가장 핵심적인 기술은 상황 인식(context-awareness)이다. 상황은 실제 시스템이 사용자에게 서비스를 제공할 때 관련되는 모든 정보로서 상황 인식은 유비쿼터스 컴퓨팅 시스템이 갖추어야 할 필수적인 기능이다.

유비쿼터스 환경에서의 응용은 사용자의 요구사항 변경과 사용자의 서비스 환경 변화를 적응시킴으로써 사용자에게 최선의 서비스를 제공할 수 있어야 한다. 또한 상황 변화에 따른 하부 플랫폼의 동작에 서비스의 동작을 적응시킬 수 있어야 하며 시스템은 이러한 정보를 사용하여 시스템 자체의 동적인 구성이 유비쿼터스 형태로 이루어야 한다[3, 6, 7]. 특히, PDA부터 워크스테이션에 이르는 다양한 시스템이 연결된 환경에서 요구되는 적응성은 상위 사용자 레벨에서 하위 시스템 레벨까지 시스템의 모든 측면에 적용된다[8, 9, 10].

그러나 특정 레벨에 국한되는 적응성 지원은 몇 가지 문제점을 초래할 수 있다. 예를 들어, 운영체제에서의 적응성 지원은 무결성 및 성능 상의 문제를 발생시킬 수 있으므로 매우 주의 깊게 처리되어야 할 것이다. 또한, 이러한 경우 적응성 획득을 위해서는 반드시 운영체제에 의존해야 하기 때문에 응용 프로그램의 이식성이 손상될 수 있다. 반면, 응용 레벨에만 의존할 경우에는 적응성 지원 메커니즘을 응용 프로그램 내에서 모두 구현해야 하므로 개발에 대한 막대한 부담이 가중될 것이다[1, 4]. 응용이 상황을 사용하기 위해서는 응용의 하부 플랫폼에서 상황을 인식하고 응용에게 이를

※ 본 연구는 숙명여자대학교 2004년도 교내연구비 지원에 의해 수행되었음.
† 준 회 원 : 숙명여자대학교 대학원 컴퓨터과학과
†† 종신회원 : 숙명여자대학교 정보과학부 교수
논문접수 : 2004년 12월 15일, 심사완료 : 2005년 11월 17일

전달할 수 있어야 한다. 이를 위하여, 사용자의 선호도나 의도된 정보를 효율적으로 인식하여 지능적인 판단에 의한 최적화된 결정을 유도할 수 있는 연구가 요구된다.

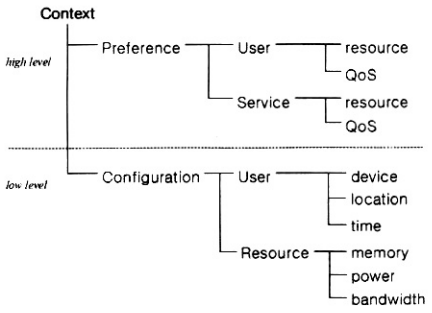
따라서 본 논문에서는 유비쿼터스 컴퓨팅 환경에서 선호도를 기반으로 사용자의 요구사항을 능동적으로 반영하고, 상황 인식을 통해 적응적인 응용 서비스를 제공하기 위하여 정책 기반의 응용 적응성을 지원하는 2-레벨 상황 적응 기법으로서 행동 결정 알고리즘을 제시하고 실험을 통한 성능을 평가한다.

2. 2-레벨 상황 적응 모델

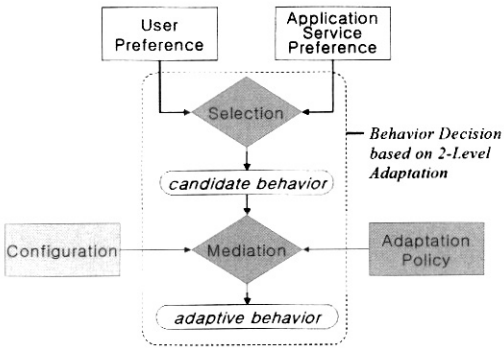
2.1 상황의 정의 및 분류

일반적으로 상황이란 적응 메커니즘의 수행을 유발하는 사용자와 시스템의 환경적인 정보이다. 본 논문에서는 상황을 (그림 1)과 같이 정의한다. 상황은 크게 상위 레벨 상황과 하위 레벨 상황으로 분류된다. 상위 레벨 상황은 선호도로서 사용자 선호도와 응용 서비스 선호도가 여기에 포함된다. 선호도는 각각 자원과 서비스 품질에 대한 선호도를 포함하는 명시적인 정보로서 여기서 자원 선호도란 사용자 또는 응용이 요구하는 자원의 사용 정도이고 서비스 품질 선호도는 해상도, 프레임률, 보안 등의 요소를 나타내는 것이다.

하위 레벨의 상황은 형상(configuration)으로서 다시 사용자 형상과 자원 형상으로 세분화된다. 사용자 형상은 사용자 장치, 위치, 시간 등을 포함하고 자원 형상은 메모리, 전원, 대역폭 등을 포함한다.



(그림 1) 상황 계층 구조



(그림 2) 2-레벨 상황 적응의 개념

지금까지 다수의 연구에서 사용자 선호도를 영화 장르, 쇼핑 정보 등과 같이 사용자가 원하는 정보의 종류나 내용으로 보고 유사도 측정 등에 의한 필터링을 통해 사용자에게 적합한 서비스를 제공하기 위한 연구가 진행되었다. 그러나 본 연구에서는 유비쿼터스 컴퓨팅의 특징에 보다 중점을 두고 사용자가 특정한 서비스를 이용하고 있는 시점에서 상황 변화가 발생할 경우 이에 따라 서비스 전달 방식을 동적으로 적응시키기 위하여 선호도를 '상황에 대한 명시적인 요구사항'으로 정의하는 것이다.

위와 같이 상황을 선호도와 형상 정보로 분류함으로써 상위 레벨의 명시적인 상황과 미들웨어가 수행하는 주기적인 모니터링을 통해 검출할 수 있는 하위 레벨 상황을 조정할 수 있으므로 보다 유연한 상황 적응을 지원할 수 있다. 본 연구에서는 이러한 개념을 2-레벨 상황 적응(2-Level Context Adaptation)이라 정의한다((그림 2) 참조). 2-레벨 상황 적응의 선택 단계에서는 선호도 우선순위에 기반 하여 사용자가 요청한 서비스에 대한 제공 방식 즉, 서비스 행동을 선택하며 이때 선택된 행동을 후보자 행동(candidate behavior)라 한다. 조정 단계에서는 현재의 형상과 후보자 행동의 적응 정책을 비교하여 최종적으로 적응될 행동을 결정한다.

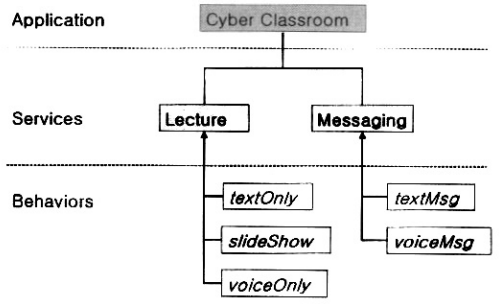
응용이 상황을 사용할 수 있기 위해서는 응용을 지원하는 미들웨어에서 상황을 인식하고 이를 전달할 수 있는 메커니즘이 요구된다. 현재까지는 주로 사용자나 장치의 위치 변화를 인식하기 위한 연구가 진행되었으나 보다 높은 수준의 상황 즉, 사용자의 선호도나 의도된 정보를 효율적으로 인식하여 지능적인 판단에 의한 최적화된 결정을 유도할 수 있는 연구가 필요하다.

적응 정책은 상황 변화에 따라서 응용 서비스의 행동을 변경하는데 사용되는 규칙으로서 본 연구에서의 트리거된 적응(triggered adaptation)을 위한 핵심적인 수단이다. 트리거된 적응은 상황에 대한 응용의 자발적인 적응 방법을 의미한다.

응용이 사용자가 원하는 서비스를 올바르게 제공하기 위해서는 우선적으로 사용자의 선호도에 대한 파악이 이루어져야한다. 사용자의 선호도란 언어를 수단으로 표현되는 불완전한 형태의 정보로서 대부분 경험적 방법(heuristic approach)에 의해 결정되며, 다양한 사용자 선호도는 하나의 목적을 지향하기 보다는 복합적인 여러 가지의 목적을 동시에 이루기 위한 것이다. 따라서 사용자 선호도에 기반 하여 적절한 서비스를 제공하기 위해서는 선호 정보들을 정량화하고 이로부터 사용자의 요구사항을 만족시키는 최적의 서비스를 선택할 수 있는 수학적 모델이 요구된다.

2.2 적응 정책

적응 정책은 상황 변화에 따라서 응용 서비스의 행동을 변경하는데 사용되는 규칙이다. (그림 3)과 같이 미들웨어 상에서 CyberClassroom이라는 응용이 실행된다고 가정하자. 이 응용은 Lecture 서비스와 Messaging 서비스를 제공할 수 있으며 각각의 서비스는 현재 상황에 맞는 행동을 수행



(그림 3) 응용구성 예 : CyberClassroom

하게 된다. 간단한 예를 들면, 사용자 장치의 메모리 가용량이 임계치 이하로 감소될 경우 Lecture 서비스가 textOnly 방식으로 제공되고 메모리 가용량이 충분할 경우에는 slideShow 방식으로 제공될 수 있다.

상황 인식을 지원하기 위해서는 주기적으로 환경을 모니터링하고 상황 변화를 검출하는 등의 하위 레벨 태스크의 처리가 필요한데, 미들웨어 계층에서 이러한 태스크를 수행함으로써 복잡성을 은폐하고 응용 개발자의 부담을 감소시킬 수 있다. 그러나 사용 및 응용의 요구사항과 상황의 다양성으로 인해 미들웨어가 변화를 미리 예측하기 어려우므로 응용은 상황 변화에 대해서 적응시켜야할 서비스를 식별하고 이러한 서비스를 제공하기 위해서 트리거해야 할 서비스 행동을 결정해야 한다. 본 연구에서는 이와 같이 응용 서비스에 의해 명시되는 사항들을 적응 정책이라 정의한다. 다시 말하면, 적응 정책이란 응용이 상황 변화에 따라서 미들웨어의 응용 서비스 제공 방식을 변경하도록 유도하는 규칙들의 집합이다.

적응 정책에 기술되는 사항은 응용을 구성하는 서비스들에게 중요한 형상 정보와 서비스 제공 시에 요구되는 형상의 상태 및 어떠한 형상에 변화가 발생했을 때 어떤 서비스 행동을 트리거해야 하는가에 대한 규칙이다. 응용 서비스가 요청될 때, 미들웨어는 적응 정책을 사용하여 현재의 상황에서 요청된 서비스를 제공하는데 있어서 어떤 규칙을 적용할 것인가를 결정하게 된다.

2.3 선호도 측정

이 절에서는 주어진 선호도 항목에 대해 선호의 정도를 분석하고 이를 수치 정보로 표현하여 다양한 선호도 사이의 논리적 및 산술적인 관계를 나타내기 위한 방법을 설명한다.

2.3.1 선호도의 관계

일반적으로 선호도는 의미가 모호하고 주관적인 정보를 포함하고 있으므로 본 논문에서는 선호도 정보를 정량화하기 위한 전처리 과정으로서 기존 사용한 개념을 보다 세분화하여 선호도를 나타내는 언어적 변수(linguistic variable)를 다음과 같이 분류한다[5].

그러나 위에서 나타낸 분류만으로는 언어적 변수를 통해 선호도에 대한 객관적이고 구체적인 상관관계를 파악하기 어렵다. 예를 들어, "A is more important than B"와 "A is more important than C"라는 선호도 표현을 보면, 세 개의 항목 중 A가 가장 중요한 것임은 알 수 있으나 B와 C의 관계에 있어서는 어떤 것이 보다 중요한가에 대한 정보가 포함되어 있지 않다. 그러므로 각 레벨의 언어적 변수에 대해 의미 있는 정량적 수치를 적용함으로써 여러 가지 선호도 항목들 사이의 명백한 상관관계와 전체적인 선호도에 대한 각 항목의 우선순위를 결정해야 한다. 언어적 변수에 대한 논리 연산 및 산술 연산을 수행하기 위하여 다음과 같이 관계 연산자를 정의한다.

〈표 1〉 언어적 변수 카테고리

	Description
1	Even less important
2	Less important
3	Much less important
4	Equally important
5	More important
6	Much more important
7	Even much important

정의 1. 임의의 선호도 항목 a, b에 대한 관계 연산자는 다음과 같다.

(1) $a \approx b$: a is equally important as b
(2) $a > b$: a is more important than b
(3) $a \gg b$: a is much more important than b
(4) $a \ggg b$: a is even much more important than b
(5) $\neg a$: a is not important
(6) $! a$: a is important

위의 (1)~(4)에서 정의한 이항 연산자는 b에 비해 a가 어느 정도 중요한가를 나타내는 것이므로, 예를 들어 "a is more important than b"라면 결과적으로 "b is less important than a"의 의미를 포함하게 된다.

2.3.2 선호도 우선순위

선호도의 우선순위를 결정함에 있어서 명시된 선호도 항목의 값으로 높은 우선순위를 할당하는 것은 다중의 목적을 성취하기 위한 방법으로 바람직하지 않다. 예를 들어, 사용자가 낮은 자원 사용과 높은 서비스 품질을 요구할 때 일정한 기준에 의하여 자원 선호도에 대해 낮은 우선순위를 할당한다면 서비스 품질에 치우치는 결과를 초래하기 때문에 사용자의 요구조건을 충분히 만족시킬 수 없다. 그러므로 자원 사용 대 서비스 품질과 같이 상호 충돌되는 선호도를 절충함으로써 사용자가 요구하는 최적 안을 제공할 수 있어야 한다.

따라서 본 논문에서는 선호도를 자원 선호도와 서비스 품질 선호도로 구분하고 선호도 항목 사이의 관계 연산을 통한 우선순위 결정을 수행하기 위하여 선호도의 집합을 다음과 같이 정의한다.

정의 2. 임의의 자원 선호도 항목을 r_i라 하고 r_i의 값을 rv_i라 할 때, 자원 선호도의 집합 R은 다음과 같다.
$R = \{(r_1, rv_1), (r_2, rv_2), \ldots, (r_n, rv_n)\}$,
$0 \leq rv_i \leq 1,\ rv_i \in \mathbb{R}$

정의 3. 임의의 서비스 품질 선호도 항목을 q_j라 하고 q_j의 값을 qv_i라 할 때, 서비스 품질 선호도의 집합 Q는 다음과 같다.
$Q = \{(q_1, qv_1), (q_2, qv_2), \ldots, (q_m, qv_m)\}$,
$0 \leq qv_i \leq 1,\ qv_i \in \mathbb{R}$

정의 4. 선호도에 대한 전체 집합 P는 집합 R과 집합 Q의 합집합이다.
$P = R \cup Q$
$= \{(p_1, pv_1), (p_2, pv_2), \ldots, (p_{n+m}, pv_{n+m})\}$

최대 선호도 값을 1이라 할 때, 선호도 우선순위 함수(Local Priority Function)는 다음의 식 1과 같다. 이 함수에서 p_k가 자원 선호도일 경우 p_k의 지역 우선순위는 선호도의 최대값(1)으로부터 pv_k를 뺀 값이고, p_k가 서비스 품질 선호도일 경우에는 지역 우선순위는 pv_k와 같다. 지역 우선순위 함수는 순차적으로 정렬된 선호도에 대해 단순히 순위를 매기는 것이 아니라 명시된 선호도 값에 따른 선호도 항목 사이의 상대적인 비율을 포함하도록 한다.

$$LP(p_k) = \begin{cases} 1-pv_k & \text{if } (p_k, pv_k) \in R, \\ pv_k & \text{if } (p_k, pv_k) \in Q \end{cases} \quad (식\ 1)$$
$$0 \leq pv_k \leq 1,\ pv_k \in \mathbb{R}$$

모바일 사용자는 제한된 자원의 사용을 최소화하면서 가능한 높은 서비스 품질을 제공받고자 한다. 자원 선호도의 경우 그 값이 낮을수록 선호도가 높은 것이고, 이에 반해 서비스 품질 선호도의 경우에는 값이 높을수록 선호도가 높은 것으로 볼 수 있다. 예를 들면, 사용자의 자원 선호도가 {(memory, 0.3), (power, 0.2), (bandwidth, 0.4)} 일 때 이들의 우선순위는 (memory, 0.7), (power, 0.8), (bandwidth, 0.6) 이다.

위의 식에 의하여 자원 선호도와 서비스 품질 선호도에 대한 지역 우선순위를 결정한 후, 식 2에 의한 전역 우선순위(Global Priority) 함수를 사용하여 선호도 전체에 대한 우선순위를 결정한다.

$$GP(p_k) = \frac{LP(p_k)}{\sum_{\forall (p_k, x) \in P} LP(p_k)} \quad (식\ 2)$$

전역 우선순위는 지역 우선순위와 같이 순차적으로 부여되는 고유한 순위 값이 아니라 전체 선호도 집합에 대해 각 선호도가 갖는 상대적인 비율을 나타내는 것이다. 위의 예에서 사용자의 서비스 품질 선호도가 {(frame rate, 0.7), (resolution, 0.8)} 이라면 이들의 지역 우선순위는 (frame rate, 0.7), (resolution, 0.8) 이므로, 식 2에 의해 전역 우선순위를 계산하면 (memory, 0.7/3.6), (power, 0.8/3.6), (bandwidth, 0.6/3.6), (frame rate, 0.7/3.6), (resolution, 0.8/3.6) 으로 memory와 frame rate, 그리고 power와 resolution이 각각 동일한 우선순위를 갖게 된다.

2.3.3 선호도 적합성

사용자와 각 응용 서비스의 행동이 선호도의 수치를 명시적으로 나타내었을 때, 이들 값의 차이는 사용자의 요구사항에 대한 응용 서비스의 적합한 정도를 의미한다. 사용자 선호도 집합과 응용 서비스 행동의 선호도 집합이 각각 정의 5, 정의 6과 같을 때, 사용자 선호도 항목 u_i와 응용 서비스 선호도 항목 s_j에 대한 거리 함수(Distance Function)를 간단히 식 3과 같이 정의한다.

정의 5. 임의의 사용자 선호도 항목을 u_i라 하고 u_i의 값을 uv_i라고할 때, 사용자 선호도의 집합 P_u를 다음과 같이 정의한다.
$P_u = \{(u_1, uv_1), (u_2, uv_2), \ldots, (u_n, uv_n)\}$,
$0 \leq uv_i \leq 1,\ uv_i \in \mathbb{R}$

정의 6. 임의의 응용 서비스 행동의 선호도 항목을 s_j라 하고 s_j의 sv_j값을 라고 할 때, 응용 서비스 선호도의 집합 P_s를 다음과 같이 정의한다.
$P_s = \{(s_1, sv_1), (s_2, sv_2), \ldots, (s_n, sv_n)\}$,
$0 \leq sv_j \leq 1,\ sv_j \in \mathbb{R}$

거리 함수 $d(u_i, s_j)$는 $(u_i, uv_i) \in P_u$, $(s_j, sv_j) \in P_s$인 u_i와 s_j에 대한 전역 우선순위의 차이를 나타낸다.

$$d(u_i, s_j) = GP(u_i) - GP(s_j) \quad (식\ 3)$$

집합 P_u와 집합 P_s에 대해 거리 함수를 계산하면 다음과 같은 n×n의 정방행렬 M_d를 구성할 수 있다.

정의 7. n×n의 가중치 행렬 M_d는
$M_d = [m_{ij}]$,
$m_{ij} = d(u_i, s_j)$,
$1 \leq i, j \leq n,\ u_i \in P_u,\ s_j \in P_s$

사용자의 요구조건을 최대한 만족시킬 수 있는 응용 서비스를 제공하기 위해서는 거리 함수 값이 최소인 서비스 행동을 선택하는 것이 바람직하다. 또한 사용자의 선호도과 서비스 행동의 선호도의 차이에 기반하여 우선적으로 만족되어야 할 선호도 항목을 결정해야 한다. 따라서 식 4는 거리 함수 값에 따라 임의의 응용 서비스 행동에 대해 사용자 선호도의 각 항목의 상대적인 중요도를 표현하기 위한 것으로서 응용 서비스가 사용자의 요구사항을 어느 정도 만족시킬 수 있는가에 대한 기준이 된다.

선호도의 구간은 0.0 ≤ui , sj ≤1.0 이고, 거리 함수의 구간은 $-1.0 \leq d(u_i, s_j) \leq 1.0$ 이므로, v가 언어적 변수의

수를 나타낼 때 각 관계 연산자가 갖게 되는 구간의 크기는 $2/v$ 이다. 위의 관계식에 따라 사용자 관점에서 응용 서비스 행동에 대한 각 선호도 항목의 상대적인 중요도가 결정되었을 때, 이를 보다 의미있는 수치로 정량화하기 위해서 선호도 관계에 근거하여 다음의 식과 같이 단계별 선호도에 대해 가중치(0, 0.5, α , β , γ, 1)를 할당한다.

$$
\begin{aligned}
&\text{if } -1.0 < d(u_i, s_j) \le -(1 - 2/v) \\
&\qquad \text{then } s_j \gg u_i \\
&\text{if } -(1 - 2/v) < d(u_i, s_j) \le -(1 - 4/v) \\
&\qquad \text{then } s_j \gg u_i \\
&\text{if } -(1 - 4/v) < d(u_i, s_j) \le -(1 - 6/v) \\
&\qquad \text{then } s_j > u_i \\
&\text{if } -(1 - 6/v) < d(u_i, s_j) < (1 - 6/v) \\
&\qquad \text{then } u_i \approx s_j \\
&\text{if } (1 - 6/v) \le d(u_i, s_j) < (1 - 4/v) \\
&\qquad \text{then } u_i > s_j \\
&\text{if } (1 - 4/v) \le d(u_i, s_j) < (1 - 2/v) \\
&\qquad \text{then } u_i \gg s_j \\
&\text{if } (1 - 2/v) \le d(u_i, s_j) < 1.0 \\
&\qquad \text{then } u_i \ggg s_j \\
&\text{if } d(u_i, s_j) = 1.0 \text{ then } !u_i \text{ and } \neg s_j \\
&\text{if } d(u_i, s_j) = -1.0 \text{ then } !s_j \text{ and } \neg u_i
\end{aligned}
\qquad \text{(식 4)}
$$

임의의 실수 a , β , γ가
$0 < 0.5 < a < \beta < \gamma < 1$ 일 때,

$$
\begin{aligned}
&\text{if } u_i \approx s_j \text{ then } w_{ij} = w_{ji} = 0.5 \\
&\text{if } u_i > s_j \text{ then } w_{ij} = a \text{ , } w_{ji} = 1- a \\
&\text{if } u_i \gg s_j \text{ then } w_{ij} = \beta \text{ , } w_{ji} = 1- \beta \\
&\text{if } u_i \ggg s_j \text{ then } w_{ij} = \gamma \text{ , } w_{ji} = 1- \gamma \\
&\text{if } !u_i \text{ then } w_{ij} = 1 \\
&\text{if } \neg u_i \text{ then } w_{ij} = 0
\end{aligned}
\qquad \text{(식 5)}
$$

따라서 위의 식을 사용하여 정의 7의 M_d로부터 다음과 같은 n×n의 정방행렬 M_w를 구성할 수 있다.

정의 8. n×n의 가중치 행렬 M_w를 다음과 같이 정의한다.

$M_w = [m_{ij}]$,

$m_{ij} = w_{ij}$, $\quad 1 \le i, j \le n$

예를 들어, $P_u = \{(u_1, uv_1), (u_2, uv_2), (u_3, uv_3)\}$이고 $P_s = \{(s_1, sv_1), (s_2, sv_2), (s_3, sv_3)\}$ 일 때 다음에 나타낸 선호도 관계가 성립된다고 가정하자.

$$
\begin{aligned}
&u_1 \approx s_1 \text{ , } u_1 > s_2 \text{ , } u_1 \approx s_3 \\
&u_2 \ll s_1 \text{ , } u_2 \approx s_2 \text{ , } u_2 > s_3 \\
&u_3 \ggg s_1 \text{ , } u_3 < s_2 \text{ , } u_3 > s_3
\end{aligned}
$$

선호도 관계의 예

이때, 위의 관계식에 대한 가중치 행렬 M_w를 계산하면 다음과 같다.

$$
M_w = \begin{bmatrix} 0.5 & a & 0.5 \\ 1-\beta & 0.5 & a \\ \gamma & 1-a & a \end{bmatrix}
$$

가중치 행렬 M_w의 예

3. 행동 결정 알고리즘

행동 결정 알고리즘(Behavior Decision Algorithm)은 선호도로 나타나는 사용자의 요구사항을 최대한 만족시키고 실행 시간에 동적으로 변화하는 상황에 대해 최적의 응용 서비스 전달 방식을 선택함으로써 적응을 유도할 수 있는 알고리즘이다. 2-레벨 적응 개념에 기반한 행동 결정 알고리즘은 선호도의 우선순위에 의해 행동을 선택하는 행동 선택 알고리즘(Behavior Selection Algorithm)과 적응 정책 및 형상 정보에 의해 상황 변화에 적응할 수 있는 행동을 선택하는 행동 조정 알고리즘(Behavior Mediation Algorithm)으로 구성된다.

3.1 행동 선택 알고리즘

사용자는 응용 서비스에 다양한 조건을 제시함으로써 동시에 다중의 목적이 이루어질 것을 요구한다. 일반적으로 사용자는 가능한 '최소의 자원 사용'과 '최고의 서비스 품질'을 제공하는 서비스를 요구하므로 본 논문의 행동 선택 알고리즘은 선호도로 표현되는 사용자의 요구사항에 대한 응용 서비스의 적합성을 분석하여 자원 사용이나 서비스 품질에 편중되지 않는 최적의 서비스 전달 방식의 선택을 목적으로 한다.

응용 서비스 행동의 적합성은 적합도 함수(Fitness Function)로 나타낼 수 있으며 이것은 사용자 선호도와 서비스 행동 선호도의 상관 관계에 대한 정규화된 비용 함수(Cost Function)로 계산된다. 여기서 적합도란 응용 서비스가 사용자 요구사항을 만족시킬 수 있는 정도를 의미하는 것이므로 적합도의 값이 높을수록 사용자에게 적당하다고 볼 수 있으므로 응용 서비스를 구성하는 행동 중 최대의 적합도 함수값을 갖는 행동을 선택한다((그림 4) 참조).

임의의 사용자에 대한 응용 서비스의 비용 함수는 각 사용자 선호도의 비용 함수의 합으로 정의한다. 즉, 비용 함수는 정의 7의 가중치 행렬 M_w의 m_{ii}를 각 열에 대한 행의 합으로 나눈 것의 총합이다. 다음의 식 6은 선호도의 비용 함수이다.

$$
c(u_i) = \frac{m_{ii}}{\sum_{1 \le j \le n} m_{ij}} \times GP(u_i) \qquad \text{(식 6)}
$$

따라서 위의 식을 사용하여 정의 8의 M_w로부터 각 선호도 항목의 비용을 나타내는 1×n의 행렬 M_c를 다음과 같이 구성할 수 있다.

정의 9. 1×n의 비용 행렬 M_c는 다음과 같다.

$M_c = [m_i]$,

$m_i = c(u_i)$, $\quad 1 \le i \le n$

그러므로 M_w로부터 M_c를 구하면 다음과 같다.

$$M_w = \begin{bmatrix} 0.5 & \alpha & 0.5 \\ 1-\beta & 0.5 & \alpha \\ \gamma & 1-\alpha & \alpha \end{bmatrix} \text{ 일 때,}$$

$$M_c = \begin{bmatrix} 0.5 \,/\, (1+\alpha)\times GP(u_i) \\ 0.5 \,/\, (1.5+\alpha-\beta) \times GP(u_i) \\ \alpha \,/\, (1+\gamma) \times GP(u_i) \end{bmatrix}$$

Behavior Selection Algorithm

```
BSA()
//step0: initialization
U_priori ← ∅
S_priori ← ∅
Max ← 0
Cost ← 0
// step1: global prirority computation
for all (u,uv) ∈ P_u
U_priori ← U_priori ∪GP(u)
for k ← 1 to numberOfBehavior
for all (s,sv) ∈ P_s
S_priori[k] ← S_priori[k] ∪ GP(s)
// step2: candidate behavior selection
for k ← 1 to numberOfBehavior
// step2a: matrix M_d computation
for all x ∈ U_priori and i ← 1 to numberOfItem
for y ∈ S_priori[k] and j ← 1 to numberOfItem
M_d[i,j] ← (x - y)
//step2b: matrix M_w computation
for i ← 1 to numberOfItem
for j ← i to numberOfItem
M_w[i,j] ← w_ij
M_w[j,i] ← w_ji
//step2c: matrix M_c computation
for all p ∈ U_priori and i ← 1 to numberOfItem
for j ← 1 to numberOfItem
M_c[i] ← M_w[i,j] / ∑M_w[i,j] ×p
//step2d: cost computation
for i ← 1 to numberOfItem
Cost ← ∑M_c[i]
   if(Max ≤ Cost) then
   Max ← Cost
   candidateBehavior ← Behavior
return candidateBehavior
```

(그림 4) 행동 선택 알고리즘

임의의 응용 서비스 행동 B_k의 비용은 M_c로부터 식 7에서 정의한 서비스 행동 비용 함수 $C(B_k)$를 사용하여 계산한다.

$$C(B_k) = \sum_{1\le i\le n} m_i \quad (m_i \in M_c) \qquad \text{(식 7)}$$
$$= \sum_{1\le i\le n} c(u_i)$$

모든 응용 서비스에 대한 비용 함수를 수행했을 때, 이 값이 최대인 응용 서비스의 행동을 현재 사용자 선호도에 가장 적합한 것으로 선택한다. 다음은 적합도 함수의 정의이다.

$$candidateBehavior \leftarrow f(B) = max\{C(B_k)\},$$
$$1 \le k \le m \qquad \text{(식 8)}$$

3.2 행동 조정 알고리즘

행동 선택 알고리즘은 각 선호도의 우선순위를 측정하여

Behavior Mediation Algorithm

```
BMA(candidateBehavior)
//step0: initialization
 minCost ←∞
 Solution ← ∅
 //step1: current configuration check
 if all CurrentConfig ≤upperBound(candidateBehavior)
    and CurrentConfig ≥lowerBound(candidateBehavior)
 then  adaptiveBehavior ← candidateBehavior
 //step2: mediation
 else
 //step2a: finding Behaviors satisfies CurrentConfig
 for all Behavior except candidateBehavior
 if all CurrentConfig ≤ upperBound(Behavior)
        and CurrentConfig ≥ lowerBound(Behavior)
 then   Solution ← Solution ∪ Behavior
 //step2b: explicit config conflict resolution
 for all Behavior ∈ Solution
 for all Config ∈ Policy(Behavior)
  mediationCost ←|median(
       [upperBound-lowerBound])-CurrentConfig|
  if minCost ≥ mediationCost then
     minCost ← mediationCost
     adaptiveBehavior ← Behavior
 return adaptiveBehavior
```

(그림 5) 행동 조정 알고리즘

사용자의 요구사항을 만족시킬 수 있는 응용 서비스 행동을 선택할 수 있다. 이때 선택된 후보자 행동을 실제로 실행하기 위해서는 현재 사용자 실행 환경의 형상 정보가 이 행동을 지원할 수 있어야 한다. 따라서 행동 조정 알고리즘(Behavior Mediation Algorithm)은 적응 정책에 명시된 *candidateBehavior*의 형상 임계치와 현재의 환경 형상을 비교하여 이 행동의 실행 여부를 결정한다((그림 5) 참조).

만약 현재 형상이 후보자 행동을 만족시키지 못할 경우에는 적응 정책으로부터 다른 행동을 선택해야 하는데, 이때 현재 형상에 대해 두 개 이상의 행동이 만족되는 경우가 발생할 수 있다. 이것은 적응 정책에서 응용 서비스 행동들 간의 자원 형상에 대한 임계치 구간이 중복됨으로써 발생하는 현상으로 본 논문에서는 이러한 현상을 명시적 형상 충돌(Explicit Configuration Conflict)이라 한다.

행동 조정 알고리즘에서는 이러한 충돌을 해결하기 위하여 각 행동의 형상 임계치 구간의 중간값(median)과 현재 형상 정보의 차에 대한 절대값을 취하여 그 최소값을 갖는 행동을 선택한다. 현재의 자원 형상에 대한 집합 G와 행동 B_j의 적응 정책에 명시된 형상 집합 Gb_j에 대한 정의가 각각 정의 10, 11과 같을 때, B_j의 조정 비용 *mediationCost*는 식 9와 같이 계산한다. 식 10의 적응 행동 *adaptiveBehavior*는 최소의 조정 비용을 갖는 것으로 선택한다.

정의 10. 현재 자원형상에 대한 집합 G는 자원형상 g_i와 가용성 a_i의 순서쌍인 (g_i, a_i)를 원소로 갖는다.
$G = \{(g_1, a_1), (g_2, a_2), ..., (g_n, a_n)\}$,
$0 \le a_i \le 1, a_i \in \mathbb{R}$

정의 11. 응용 서비스 행동 B_j의 형상에 대한 집합 Gb_j는 자원형상 gb_i, 상한 임계치 tu_i, 하한 임계치 tl_i의

순서쌍인 (gb_i, tu_i, tl_i)를 원소로 갖는다.
$Gb_j = \{(gb_1, tu_1, tl_1), (gb_2, tu_2, tl_2), ..., (gb_n, tu_n, tl_n)\}$,
$0 \le tu_i, tl_i \le 1$ and $tu_i, tl_i \in \mathbb{R}$

$$mediationCost(B_j) = \sum_{\forall(gbi,tui,tli)\in Gbj\ ,\ \forall(gi,ai)\in G} |median([tu_i, tl_i]) - a_i| \quad (식\ 9)$$

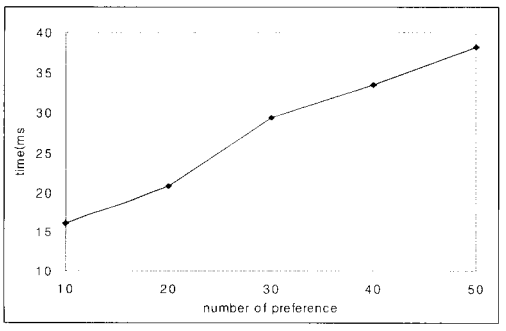
$$adaptiveBehavior \leftarrow min\{mediationCost(B_j)\},\ 1 \le j \le m \quad (식\ 10)$$

4. 성능 평가

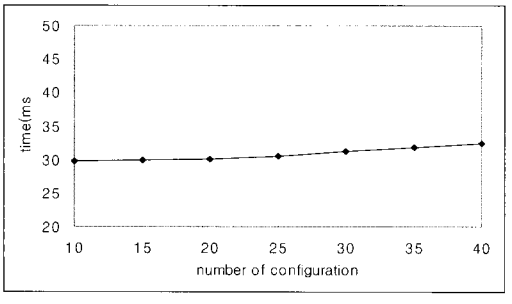
이 장에서는 3장에서 제시한 행동 결정 알고리즘의 성능 평가를 위해 수행한 실험 결과를 분석한다. 알고리즘의 성능 평가는 상황 인식과 적응성 측면에서의 효과를 측정하기 위하여 수행 경과 시간(elapsed time)을 비교하였다.

성능 평가를 위한 실험 환경은 2.80GHz의 Intel Pentium 4 프로세서, 512MB RAM을 갖춘 데스크탑 컴퓨터와 1GHz의 Intel Centrino 프로세서, 512MB DDR을 갖춘 무선 노트북 컴퓨터들로 구성하였고, 운영체제로는 마이크로소프트사의 Windows2000, WindowsXP를 사용하였으며 자바 가상 머쉰 1.4.1 상에서 실험을 수행하였다.

상황 인식의 효과는 상황 변화의 검출과 검출된 정보 획득 및 상황 요소의 수에 의해서 결정될 수 있다. 이 실험에서는 검출된 상황 정보가 로컬 서버에 존재한다고 가정하고 상황 요소의 수에 따른 수행 경과 시간으로 성능을 평가하였다. 본 논문에서는 상황을 선호도와 형상으로 분류하므로



(그림 6) 상황인식 실험: 선호도



(그림 7) 상황인식 실험:형상 개체

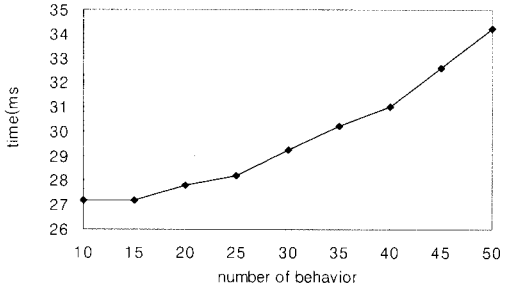
선호도 항목의 수와 형상 개체의 수를 성능의 변화 요인으로 볼 수 있다.

(그림 6)에서 선호도 항목의 수가 증가함에 따라 수행 시간이 증가됨을 알 수 있으나 본 논문에서의 선호도는 자원과 서비스 품질이므로 실제 사용자에게 제시될 수 있는 선호도 항목 수는 10~20개 정도가 일반적이다. (그림 7)은 선호도 항목 수를 20으로 고정하고 형상 개체의 수를 증가시켰을 때의 수행 시간 변화를 나타낸 것이다. 이 실험 결과를 보면 선호도에 대한 실험 결과에 비해 수행 시간의 변화가 비교적 낮은 것을 알 수 있다. 이는 선호도의 경우 모든 항목에 대한 연산을 수행해야하는 반면, 형상의 경우에는 서비스 행동의 형상이 현재의 형상을 모두 만족할 때에만 연산을 수행하기 때문이다.

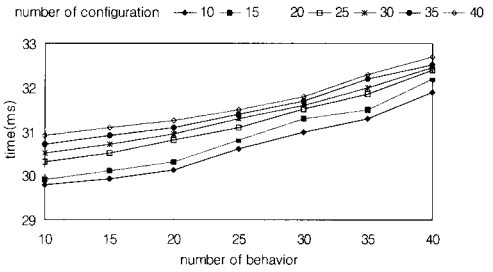
그러므로 본 논문의 행동 결정 알고리즘은 적절한 선호도 항목 수에 대해서 충분히 빠른 시간 내에 처리할 수 있으며 형상 개체 수의 변화에 대한 좋은 성능을 나타내므로 효율적인 상황 인식을 제공할 수 있는 알고리즘이다.

(그림 8)에서 서비스 행동의 수가 증가함에 따라 수행 시간의 증가폭이 높은 것을 알 수 있다. 이는 서비스 행동의 수가 행동 선택 단계의 연산 횟수와 행동 조정 단계의 연산 횟수에 모두 영향을 미치기 때문이다. 그러나 선호도의 경우와 같이 하나의 응용 서비스에 대해 제공될 수 있는 행동의 수가 10~20개 정도일 때에는 행동의 수로 인한 오버헤드가 발생하지 않을 것으로 예상된다.

적응 정책에 명시할 수 있는 서비스 행동의 형상 정보의 수는 최대 형상 개체의 수를 초과하지 않는다. 상황 인식 실험에서 보였듯이 행동 결정 알고리즘은 형상 개체 수에 대해 좋은 성능을 나타낸다. 그림 9에서는 행동의 수와 형상 정보의 수를 동시에 증가시켰을 때 수행 시간에 대한 실험 결과를 보였다. 이 결과를 보면 상황 인식 실험과 비교하여 수행 시간의 증가율이 높게 나타났으나 형상 정보의 수가 일정할 때의 실험과 비교했을 때에는 수행 시간의 증가율이 높지 않은 것을 알 수 있다. 이는 행동 조정 알고리즘에 의해서 현재 형상 조건을 만족하는 행동만 필터링되므로 행동의 수(n)와 형상 정보의 수(m)가 증가하더라도 수행 시간은 $n \times m$에 비례하지 않기 때문이다. 따라서 본 논문의 행동 결정 알고리즘은 상황 인식성과 적응성 측면에서 모두 좋은 성능을 나타내므로 상황 인식 컴퓨팅에 적용될 수 있는 적합한 알고리즘이다.



(그림 8) 적응성 실험: 서비스 행동



(그림 9) 적응성 실험: 행동/형상 정보

5. 결 론

유비쿼터스 컴퓨팅 환경을 위한 미들웨어는 다양한 상황 소스로부터 상황 정보를 용이하게 획득할 수 있어야 하고 이를 기반으로 상황 변화에 적응된 서비스를 제공할 수 있어야 한다. 이를 위하여 다양한 상황에 대해 적응적인 해결책을 제공할 수 있도록 상황 인식을 기반으로 미들웨어에서 수행되는 적응 메커니즘이 제공되어야 한다.

사용자의 요구사항을 반영하는 동시에 실행 환경의 변화를 인식하기 위하여 본 논문에서는 상황 정보를 선호도와 형상으로 분류하였다. 사용자는 선호도를 통해 응용 서비스에 대한 조건을 제시함으로써 동시에 다중의 목적이 이루어질 것을 요구한다. 또한 응용 서비스는 현재 사용자 상황이 허용하는 범위 내에서 최선의 형태로 제공되어야 한다. 이러한 두 가지 목적을 실현하기 위해서 본 논문에서는 행동 결정 알고리즘을 구현하고 실험을 통해 성능을 평가하였다. 행동 결정 알고리즘은 선호도의 정량적 분석을 기반으로 하는 행동 선택 알고리즘과 현재 상황의 분석과 적응 정책을 기반으로 하는 행동 조정 알고리즘으로 구성된다.

실험의 결과를 통해 본 논문의 행동 결정 알고리즘은 상황 인식 측면과 적응성 측면에서 좋은 성능을 나타내었으므로 상황 적응을 요구하는 응용 서비스 지원에 매우 적합하다고 볼 수 있다.

참 고 문 헌

[1] 김수중, 윤용익, "적응성 지원을 위한 메타 레벨 기반의 이동 에이전트 프레임워크", 정보처리학회논문지 제10-A권 제6호, pp.651-656, 2003.

[2] A. Ranganathan and R. H. Campbell. "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments," In ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, June, 16-20, 2003.

[3] L. Crowley, J. Coutaz, G. Rey, and P. Reignier. "Perceptual Components for Context Aware Computing," UBICOMP 2002, International Conference on Ubiquitous Computing, Goteborg, Sweden, September, 2002.

[4] Soojoong Ghim and Yongik Yoon, "A Reflective Approach to Dynamic Adaptation in Ubiquitous Computing Environment," The International Conference on Information Networking (ICOIN 2004), 2004.

[5] D. Cvetkovic and I. C. Parmee. "Preferences and their Application in Evolutionary Multiobjective Optimisation," IEEE Transactions on Evolutionary Computation, Vol.6, No.1, pp.42-57, February, 2002.

[6] J. Kenney and V. Cahill. "Chisel: A Policy-Driven, Context-Aware, Dynamic Adaptation Framework," In Proceedings of the Fourth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2003), Lake Como, Italy, June, 4-6, 2003.

[7] M. Roman, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. "Gaia: A Middleware Infrastructure to Enable Active Spaces," In IEEE Pervasive Computing, pp.74-83, Oct.-Dec., 2002.

[8] Benjamin Bappu, "Reflective Framework for Ubiquitous Mobile Computing," Ph.D Dissertation of Lancaster University, 2002.

[9] P. Bellavista, A.Corradi, R.Montanari, C.Stefanelli, "Context-aware Middleware for Resource Management in the wireless internet," IEEE trans. on Software Engineering, 2004.

[10] Licia Capra, Wolfgang Emmerich, and Cecilia Mascolo, "Exploiting Reflection and Metadata to build Mobile Computing Middleware," Proceedings of Mobile Computing Middleware, ACM SIGMOBILE Mobile Computing and Communications Review, Vol.6, No.4, pp.34-44, 2002.

[11] Dan Chalmers, Naranker Dulay, Morris Sloman, "Meta Data to Support Context Aware Mobile Applications," proceeding of MDM'04, pp.199-211, 2004.

김 수 중

e-mail : sjghim@sookmyung.ac.kr
1995년 숙명여자대학교 전산학과(이학사)
1998년 숙명여자대학교 컴퓨터과학과 (이학석사)
2005년 숙명여자대학교 컴퓨터과학과 (이학박사)
관심분야 : 미들웨어, 모바일 시스템

윤 용 익

e-mail : yiyoon@sookmyung.ac.kr
1985년 한국과학기술원 전산학과 (공학석사)
1994년 한국과학기술원 전산학과 (공학박사)
1985년~1997년 한국전자통신연구원 (책임연구원)
1997년~현재 숙명여자대학교 정보과학부 교수
관심분야 : 미들웨어, 멀티미디어 분산 시스템, 유비쿼터스 컴퓨팅, 임베디드 시스템, 실시간 시스템 등