유비쿼터스 환경에서의 JMX를 이용한 모니터링 구조 설계

이효남[O1], 이병훈[1], 김재훈[1], 조위덕[2]
[1]아주대학교 정보통신공학과
{ narsisse[O], componer, jaikim }@ajou.ac.kr
[2]아주대학교 유비쿼터스 시스템 연구센터
chowd@ajou.ac.kr

Monitoring Architecture Design Using JMX in Ubiquitous Smart Space

Hyo-Nam Lee[O1], Byoung-Hoon Lee[1], Jai-Hoon Kim[1], We-Duke Cho[2]
[1]Graduate School of Information and Communication, Ajou University
[2]Center of excellence for Ubiquitous System, Ajou University

요 약

유비쿼터스 환경에서 중요한 이슈는 사용자에게 현재 상황에 적절한 서비스를 실시간으로 제공하는 것이며, 이를 위해서 상황 인식(Context Aware) 기법에 대한 연구가 널리 진행되고 있다. 사용자에게 편리한 환경을 제공하기 위해서는 상황 인식 환경이 구축되어야 한다. 상황 인식 환경이 구축이 되면 센서가 환경을 감지하여 사용자에게 맞는 특정한 애플리케이션을 실행하거나, 시스템을 재구성 하는 서비스를 제공할 수 있다. 자동적으로 상황을 인식하여 서비스를 제공하기 위해서 애플리케이션의 모니터링이 매우 중요하다. 애플리케이션의 기능, 성능, 상태를 모니터링 함으로써 서비스의 질을 높일 수 있고 사용자에게 최적의 서비스를 제공할 수 있기 때문이다. 본 논문에서는 시스템의 CPU, Memory 사용량 등의 측정에 이용이 되었던 JMX(Java Management Extension)를 이용하여 유비쿼터스 환경의 상황 인식 애플리케이션에 적합한 모니터링 기법을 설계하고 적용방안을 제시하였다.

1. 서 론

유비쿼터스 컴퓨팅은 일상생활 속에 편재해 있는 컴퓨팅 자원을 이용하여 사용자가 언제 어디서나 동적인 서비스를 받을 수 있는 환경을 제공한다. 조용한 컴퓨팅, 보이지 않는 컴퓨팅, 사라지는 컴퓨팅 등의 용어는 유비쿼터스 컴퓨팅에 관한 사용자 인터페이스 관점을 잘 설명해주고 있다.

일반적인 상황 정보는 사용자 상황, 물리적 환경 상황, 컴퓨팅 시스템 상황, 사용자-컴퓨터, 상호 작용, 기타 상황으로 분류할 수 있다. 사용자의 현재 상황에 적절한 서비스를 제공하기 위해 상황을 이용하는 것을 상황 인식(Context-Awareness)이라한다 [1, 2].

유비쿼터스 환경은 기존 컴퓨팅 환경에서 사용자와 컴퓨터간의 대화형 상호작용이 아닌 물리적인 환경, 상황 등을 시스템이 인식한다. 이를 기반으로 사용자와 상호작용을 지원하는 상황 인식 기술이 필수적인 요소로 자리 잡고 있다. 또한 상황인식 서비스는 다양한 상황 정보 수집, 해석, 인식, 추론과정을 거친다. 사용자가 제어하는 기존의 수동적인 서비스에서 벗어나 사용자 명령 없이도 자동으로 실행되는 지능형 서비스를 지원하며,

각 사용자에게 맞춰진 개인화된 서비스 등을 제공한다. 상황 인식 서비스는 의료, 교육, 재난, 구호, 쇼핑 등 사회 전 분야에 걸쳐 응용될 수 있어 사회 전반에 걸쳐 많은 영향을 줄 것이다 [1].

유비쿼터스 컴퓨팅의 3가지 핵심 이슈는 자동 수집(Autonomic Sensing), 상황 인식(Context-Aware), 자가 성장(Self-growing)이다. RFID나 Smart Dust와 같은 센서가 주변의 상황정보를 인식하고(Autonomic Sensing), 이를 잘 분석해서(Context-Aware), 적절한 산출물을 출력한다. 이 과정에서 센서는 스스로 학습하면서 지능화

* 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스 컴퓨팅 및 네트워크 원천기반기술 개발사업의 지원에 의한 것임

됨으로써(Self-growing) 다음 인식과 분석 시에 더 적합한 산출물을 출력해주게 된다.

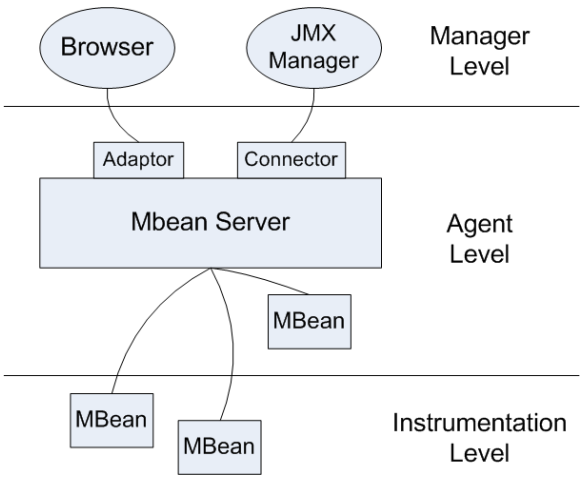
사용자에게 최적화된 상황 인식 서비스를 제공하기 위해여 애플리케이션의 모니터링은 필수적이다. 애플리케이션을 모니터링 하기 위해 여러 툴이 있지만 본 논문에서는 JMX를 이용한다. 기존에는 JMX를 시스템의 CPU, 메모리 사용량, 등의 자원을 모니터링 하는 도구로 사용하였다. 본 논문에서는 기존의 방법과는 다르게 JMX를 이용하여 유비쿼터스 환경에서의 상황 인식 애플리케이션의 모니터링 시스템을 설계하였다.

JMX를 자바 코드에 삽입하여 에이전트를 만들고, 분산 및 관리 미들웨어와 관리 계층을 구현한다. 이 솔루션들은 관리 및 모니터링 시스템으로 적절하게 통합하여 최적의 관리 수단을 제공한다 [3, 4].

2. 관련 연구

2.1 JMX (Java Management Extension)

JMX는 프로그래머들에게 자바 애플리케이션의 모니터링과 관리 기능을 제공한다. JMX 기술을 사용하면 MBean(Managed Bean)으로 알려진 하나 이상의 사용자 JavaBean 오브젝트를 통해 한 기계 안의 애플리케이션, 디바이스, 또는 서비스를 원격으로 제어할 수 있다. [그림 1]과 같이 세 개의 계층의 조합은 완전한 관리 솔루션을 설계하고, 개발하는데 필요한 아키텍처를 제공한다. 따라서 JMX 기술은 이식성, 관리 기능의 on-demand 배치, 동적 및 이동성 서비스, 그리고 보안을 모두 제공해 준다. [그림 1]에서 보듯이 JMX는 기본적으로 세계의 계층으로 되어있다 [3, 4, 5, 6].



[그림 1] JMX 구조

1) 매개 계층 (Instrumentation Level)

애플리케이션, 디바이스, 서비스와 같은 리소스들은 Manage Bean(MBean)이라고 불리는 자바 오브젝트를 이용하여 설치된다. MBean은 원격으로 관리하고 모니터링 하는 JMX 에이전트를 통해 속성과 연산으로 구성된 관리 인터페이스(속성과 조작)를 보여준다.

2) 에이전트 계층 (Agent Level)

JMX 에이전트의 주요 컴포넌트는 MBean 서버이다. MBean 서버는 MBean이 등록 되어 있는 코어 관리 에이전트 오브젝트 서버이다. JMX 에이전트 또한 MBean을 핸들링하기 위한 서비스들을 포함하고 있다. JMX 에이전트는 직접적으로 리소스를 제어하여 원격 관리 에이전트로 이용할 수 있도록 한다.

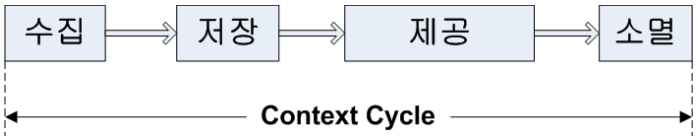
3) 관리 계층 (Manager Level)

프로토콜 어댑터(protocol adaptor)와 표준 커넥터(connector)에 의한다. 에이전트의 Java 가상 머신(VM) 외부에 있는 원격 관리 애플리케이션으로부터 접근 가능한 JMX 에이전트를 만드는 프로토콜 어댑터(adaptor)와 커넥터(connector)를 정의한다.

2.2 상황 인식(Context Aware)

사용자가 컴퓨터를 제어할 때는 상황 정보를 사용할 수 없다. 이런 점을 보완해서 컴퓨터가 상황을 이해하게 만듦으로써 사용자와 컴퓨터 간의 커뮤니케이션을 더욱 효과적으로 만든다. 또 사용성이 더욱 뛰어난 컴퓨팅을 가능하게 하는 것을 목적으로 하는 분야가 'Context Aware Computing'이다.

상황 인식 서비스(Context Aware Service)를 구성하는 방법론은 컨텍스트(context)의 수집, 저장, 제공, 소멸로 크게 4가지 단계로 나누어진다 [7].



[그림 2] 상황 인식 서비스의 구성 요인

1) 수집 단계

컨텍스트(context)의 수집은 사용자가 사용하는 장치(device) 혹은 애플리케이션에서 얻어 낼 수 있는 사용자의 환경 정보를 모으는 것이다. 센서가 얻어 낸 정보와

그 센서와 연결되는 객체(object)와의 상관관계 분석을 통한 정보 취합의 단계이다.

2) 저장 단계

이 단계에서 컨텍스트(context) 정보의 저장은 중앙집중식의 저장 형태로 구성되어야 한다. 즉 여러 경로로 유입된 사용자의 컨텍스트 정보를 각 수집단말과는 분리해서 저장하게 된다. 이는 사용자의 컨텍스트 정보의 독립성을 유지하고 저장된 컨텍스트를 서로 다른 단말에서도 사용할 수 있도록 하기 위함이다.

3) 제공 단계

컨텍스트(context)의 제공 단계는 실제 정보나 서비스가 제공되는 단말에서 사용자의 환경에 맞는 서비스가 제공된다. 이 때, 저장된 컨텍스트 정보에서 현재 사용자에게 최적의 서비스를 할 수 있도록 기반 컨텍스트 정보를 제공해 주는 단계이다. 이 단계는 사용자의 요구에 응답하는 '사용자 주도형'이 될 수 있고, '시스템이 먼저 반응하는 '시스템 주도형'이 될 수 있다.

4) 소멸 단계

컨텍스트(context)는 한번 생성된 것이 영속적으로 사용자의 주변에 존재하는 것이 아니다. 컨텍스트의 속성, 시간, 혹은 다른 컨텍스트에 따라 소멸되거나 다른 형태로 변화하게 된다. 사용자가 끊임없이 사고하고, 행동함에 따라 주어진 상황이 바뀌기 때문이다.

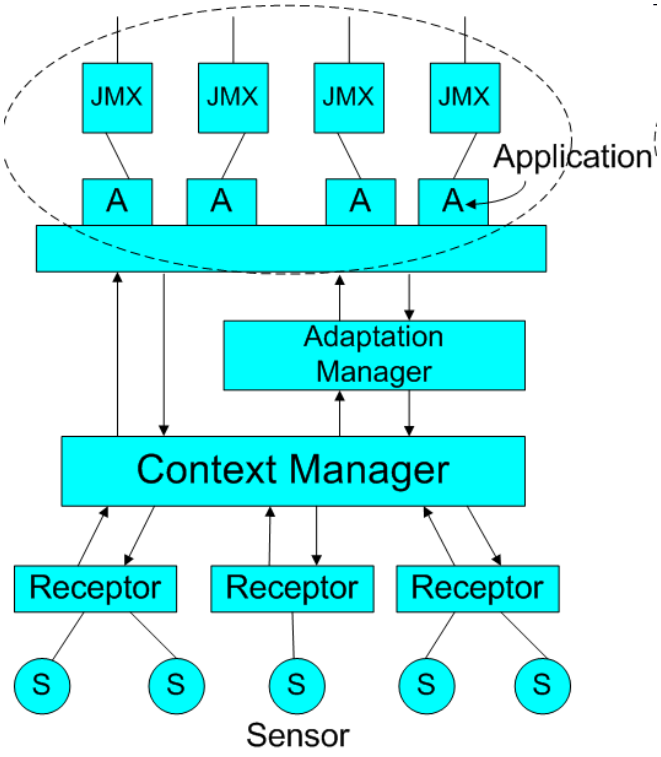
[표 1]을 보면 소멸과 관련한 컨텍스트의 종류는 영속성 컨텍스트, 소멸형 컨텍스트, 변화형 컨텍스트로 3가지의 종류로 나눌 수 있다.

	설 명	대표적 예
영속형	변하지 않는 영속적인 데이터이다. 즉, 고유 속성 등 이다	사용자의 profile (생년월일 등)
소멸형	시간 혹은 타 컨텍스트에 따라 자체가 소멸하는 컨텍스트로서 수집된 컨텍스트가 일정한 조건이 되면 소멸하는 컨텍스트이다.	병원에 입원하는 컨텍스트는 퇴원을 하게 되면 사라진다.
변화형	수집된 컨텍스트가 자체적 혹은 다른 컨텍스트로 인해 계속 변화하게 되는 것이다.	위치정보

[표 1] 소멸단계의 컨텍스트(context) 분류

3. 상황 인식 애플리케이션 모니터링 시스템 설계

3.1 구조 설계



[그림 3] 상황 인식 환경에서의 JMX 적용 구조

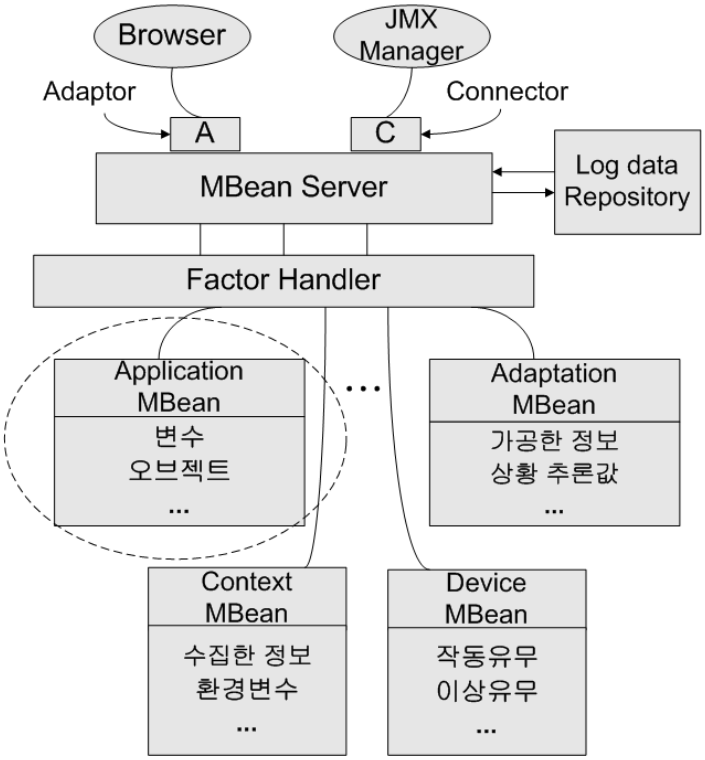
[그림 3]은 상황 인식 환경에서의 JMX 적용 구조이다. 사용자 주위 환경에 설치 되어있는 센서들이 사용자 환경에 맞는 정보들을 수집한다. 각각의 센서들은 수집한 정보들을 수집기(Receptor)에게 전달을 하고, 수집기(Receptor)는 센서에게서 받은 정보들을 컨텍스트 매니저(Context Manager)에게 전달한다 [8].

컨텍스트 매니저는 받은 정보들을 모델링하고 각 상황에 의한 정보들을 저장한다. 센서에게서 받은 정보 중 복잡한 추론을 필요로 하지 않는 간단한 정보들은 곧바로 애플리케이션을 실행하여 처리 실행 시간을 줄인다. 보다 나은 상황의 서비스를 제공하기 위해서는 어댑테이션 매니저(Adaptation Manager)에게 정보들을 전달한다.

어댑테이션 매니저(Adaptation Manager)는 이 정보들을 이용하여 보다 사용자 상황에 맞는, 보다 나은 서비스를 추론하는 과정을 거쳐 사용자에게 가장 적합한 애플리케이션을 실행하거나 시스템을 재구성한다.

이런 상황 인식 시스템위에 JMX를 적용하여 유비쿼터스 지능형 공간에서의 모니터링 방법을 제안한다.

[그림 4]는 본 논문에서 제안하는 유비쿼터스 지능형 공간에서의 모니터링 구조이다.



[그림 4] 유비쿼터스 지능형 공간에서의 모니터링 구조 설계

유비쿼터스 환경의 지능형 공간에서 모니터링 할 요소들은 굉장히 많다. 이는 JMX를 이용하여 애플리케이션, 서비스, 디바이스, 유저 등 다양한 타입의 리소스를 관리할 수 있다 [9, 10].

정보 수집 단계에서 센서들이 사용자 주위에서 수집한 정보를 수집기(Receptor)에 보낸다. 이렇게 보낸 정보들을 하나의 MBean으로 만들어 모니터링 할 수 있다. 예를 들어 사용자 주위에 설치되어 있는 온도 센서가 수집하는 온도, 사용자 침대의 압력 센서가 수집하는 압력, 조도 센서가 수집하는 조도의 수치 등이 요소로서 모니터링 할 수 있다.

사용자 주위의 각종 장치(device)도 모니터링 요소가 될 수 있다. 유비쿼터스 환경에서 사용자의 주위에는 컨텍스트 매니저와 통신하는 각종 장치가 있다. JMX의 외부 관리 애플리케이션에서는 MBean이 보내는 통지(notification)의 등록과 수신이 가능하다. 사용자가 휴대하는 PDA나 각종 장치들이 전송하는 정보들을 MBean화하여 모니터링 함으로서 각종 장치 및 센서들의 이상유무를 판단할 수 있다.

이제 하위 계층이 아닌 정보들을 가공하고 애플리케이션을 실행하는 상위 계층을 살펴본다.

어댑테이션 매니저(Adaptation Manager)는 컨텍스트 매니저(Context Manager)에서 정보를 받아 이용자에게 맞는 상황을 추론한다. 여기서 어댑테이션 매니저는 하위 계층에서 계속 변하는 상황 정보들을 받는다. 변하는 정보들을 계속 받기 때문에 상황 추론 값도 계속 변하게 된다. 어댑테이션 매니저의 변하는 요소들을 파악하고, MBean화하여 MBean 서버에 등록을 한다. MBean 서버는 등록 된 요소를 어댑터(adaptor)나 커넥터(connector)를 통하여 뷰어로 모니터링 할 수 있다.

어댑테이션 매니저(Adaptation manager)는 추론한 값을 가지고 사용자에게 가장 적합한 애플리케이션을 실행한다. 여기서 애플리케이션은 지능형 공간에서 사용자에게 제공하는 서비스를 말한다. 사용자에게 제공하는 서비스가 제대로 작동하는지와 실행하는 각각의 애플리케이션들 간의 충돌이 일어나는지에 관하여 모니터링은 필수적이다. 각각의 애플리케이션의 변수, 오브젝트 그리고 메소드 같은 모니터링 요소들을 MBean화하여 MBean 서버에 등록을 하여 각 요소들의 변화 값을 모니터링 한다. 이러한 요소들을 모니터링 하면 사용자에게 제공되는 서비스 애플리케이션이 제대로 작동을 하는지, 다른 애플리케이션과 충돌이 생겨 오작동을 하는지에 대해 알 수 있게 된다.

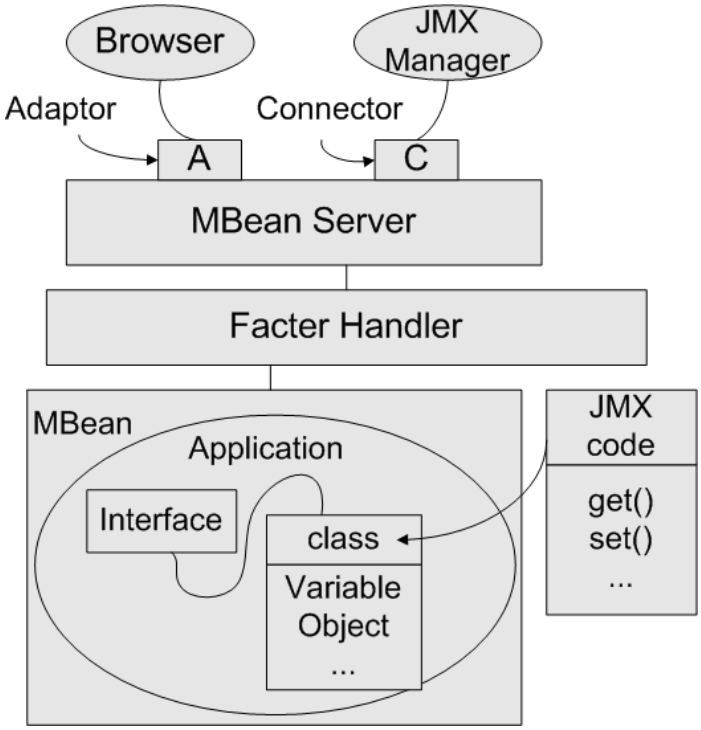
이렇게 각 부분들의 모니터링 요소를 각각 MBean으로 만들어 MBean 서버에 등록을 한다. MBean 서버에 등록을 하기 전에 각각의 MBean들은 요소 핸들러(Factor Handler)라는 계층을 거쳐 MBean 서버에 등록이 된다. 모니터링을 할 각각의 MBean들은 MBean 서버에 등록을 하여 관리를 하게 된다.

유비쿼터스 상황 인식 환경에서 모니터링 해야 하는 요소는 굉장히 많다. 즉, MBean 서버에 등록해야 하는 MBean들이 많아진다. MBean 서버에 등록할 MBean들이 많아지면 그 만큼 지연시간이 발생 할뿐만 아니라 부화가 걸리기 쉽다. 이러한 문제점을 방지하기 위해서 요소 핸들러(Factor Handler)라는 계층을 두었다. 이 핸들러는 관리자가 모니터링 할 항목만 MBean 서버에 등록을 해주는 역할을 한다. 요소 핸들러가 상황에 맞게 모니터링 할 MBean만 MBean 서버에 등록을 하기 때문에 모든 MBean을 MBean 서버에 등록을 할 경우보다 지연시간과 시스템의 부하를 줄일 수 있다.

유비쿼터스 지능형 공간은 각 부분들이 연계되어 복잡

한 구조를 지닌다. 각각의 부분들을 모니터링 함으로써 사용자에게 상황 인식(Context Aware) 환경을 제공하는 유비쿼터스 지능형 공간의 원할 한 서비스를 제공할 수 있다.

[그림 5]는 [그림 4]의 하나의 MBean인 애플리케이션과 JMX의 관계를 나타낸 구조이다.



[그림 5] JMX와 애플리케이션의 구조

[그림 5]의 구조를 살펴보면 변수나 오브젝트가 필요한 경우 MBean 인터페이스(interface)를 구현하여 MBean을 생성한다. 생성된 MBean을 MBean 서버에 등록한다. 이렇게 등록된 MBean을 커넥터나 어댑터를 통하여 브라우저로 모니터링 한다.

리소스의 설치에 접근을 구현하기 위해서는 MBean을 생성해야한다. MBean의 인터페이스(interface)는 일반적으로 읽고 쓸수 있는 이름별, 타입별 속성들과 활성화될 수 있는 이름별, 타입별 작용들로 구성되어있다.

다음 단계는 하나의 클래스를 생성을 한다. 클래스를 생성 한 후에 MBean 인터페이스의 구현을 생성해야한다. 클래스에 JMX에서 제공하는 get(), set()와 같은 메소드를 개입시켜 클래스의 모니터링 할 오브젝트에 대해 레퍼런스(reference)를 얻는다.

MBean은 MBean 서버로 불리는 코어 관리 대상 오브젝트 서버에 등록된다. MBean 서버에 등록되어 있는 모든 에이전트가 관리 애플리케이션에 공개되어 관리 할 수 있게 된다. 다만, MBean 서버는 MBean의 관리 인터페이스만을 공개하고 오브젝트의 직접 참조는 공개하지 않는다.

리소스가 MBean에 의해 설치 된 후에는 JMX 에이전트를 통해 관리할 수 있다. 에이전트는 서버에 위치한 서비스들로부터 도움을 받는다. 에이전트 서비스는 MBean 서버에 등록된 MBean에서의 관리 작용을 수행할 수 있는 오브젝트이다. 에이전트 서비스도 MBean이 되는 일이 있는데, 이때에는 에이전트 서비스와 그 기능을 MBean 서버로부터 제어할 수 있다. 에이전트 서비스는 모니터로 MBean 속성의 수치 또는 변화 값을 관찰해 각종의 변경을 다른 오브젝트로 통지할 수 있다. 또한 타이머에 의해 스케줄링 기구를 제공해 사전에 정의한 각종의 통지를 송신할 수 있다.

모니터링 요소를 표시하는 방법은 여러 가지가 있다. 기존의 관리 프로토콜이나 독자적인 프로토콜에 의한 다양한 방법으로 액세스 된다. 에이전트의 자바 가상 머신 외부에 있는 원격 관리 애플리케이션으로부터 JMX 에이전트에 액세스 할 수 있도록 하기 위해서, MBean 서버는 프로토콜 어댑터(adaptor)나 커넥터(connector)를 사용한다.

프로토콜 어댑터는 지정된 프로토콜을 개입시킨 JMX 에이전트의 관리 뷰를 제공한다. MBean의 조작 및 MBean 서버가 지정된 프로토콜의 형식으로 변환한다. 관리 애플리케이션은 JMX 에이전트에 대해서 MBean 서버의 원격 형식에서 액세스 하는 것이 아니라, MBean 서버의 매핑(mapping)된 조작으로 액세스한다. 어떤 어댑터도 MBean 서버에 등록된 모든 MBean 특유의 프로토콜을 사용해 뷰를 제공한다. 이는 브라우저나 J manager 등이 해당된다. 외부 관리 애플리케이션에서는 MBean 속성의 취득 또는 설정을 할 수 있고, 새로운 MBean의 인스턴스를 등록할 수 있다. 이렇게 다양한 프로토콜에 의하여 모니터링 가능하다.

커넥터는 에이전트와 원격 관리 애플리케이션을 접속하기 위해서 사용된다. 이 경우, 원격 관리 애플리케이션은 JMX의 분산 서비스를 사용해 개발된 관리 애플리케이션이다. 커넥터를 이용하는 통신은 에이전트 내의 커넥터 서버나, 매니저 내의 커넥터 클라이언트가 실행한다. 모든 커넥터는 같은 원격 인터페이스를 갖추고 있기 때문에, 관리 애플리케이션에서는 어떤 커넥터도 똑같이 사용할 수가 있다.

3.2 시나리오 적용 계획

유비쿼터스 스마트 홈에서 취침 시나리오를 가정하여 설계한 상황 인신 애플리케이션을 모니터링 하는 방법을 적용해보겠다. 사용자가 침대에 누워 잠을 자는 경우, 애플리케이션이 상황을 인지하여 방의 형광등이나 TV, 라디오를 끄는 서비스의 시나리오이다.

사용자가 침대에 누우면 침대에 설치되어 있는 압력 센서가 사용자의 취침여부를 감지한다. 여기서 센서는 압력과 사용자의 뒤척임의 압력 변화와 움직이지 않는 시간 등의 정보를 수집하여 미들웨어와 애플리케이션에 정보를 전달한다. 컨텍스트 매니저(Context Manager)와 어댑테이션 매니저(Adaptation Manager)는 받은 정보를 가지고 추론 과정을 거쳐 상황에 맞는 사용자에게 가장 적합한 애플리케이션을 실행한다.

컨택스트 매니저와 어댑테이션 매니저는 사용자가 잠을 잔다고 추론을 하여 판단하게 된다. 또한, 사용자 주위에 있는 온도 센서, 조도 센서들 같은 센서에게서 정보를 받아 사용자에가 편안히 잠을 잘 수 있는 추가적인 애플리케이션을 실행하게 된다. 온도 센서가 감지한 정보들을 가지고 방의 온도를 조절하고, 조도 센서가 감지한 정보들을 가지고 방의 조도를 조절한다. 이렇게 사용자가 취침을 하는데 있어서 방해 요소를 제거하는 서비스를 제공한다.

이러한 시나리오를 가지고 JMX를 이용하여 모니터링을 하면, 각각의 애플리케이션의 변화하는 값을 모니터링 할 수 있다. 선행적으로 각각의 애플리케이션에 JMX에서 지원하는 코드를 삽입 후, 압력 센서가 받는 압력의 변화, 온도를 측정하는 센서의 온도 변화와 같이 모니터링 할 변수나 객체를 파악하여 코드를 삽입한다. MBean을 생성하기 위해 클래스에 JMX에서 제공하는 get(), set() 등의 메소드(method)를 이용하여 클래스를 생성하고, MBean 인터페이스를 구현하여 생성된 MBean을 MBean 서버에 등록을 한다. 그 후 애플리케이션을 실행하고 어댑터나 커넥터를 통해 관리 브라우저로 모니터링 할 변수나 객체의 값을 모니터링 할 수 있다.

4. 결론

유비쿼터스 컴퓨팅의 자동수집(Autonomic Sensing), 상황 인식(Context-Aware), 자가 성장(Self-growing) 중 가장 이슈인 상황 인지(Context Aware)를 설명하고, 서비스되는 애플리케이션의 모니터링 시스템을 설계하였다. 기존의 시스템에서 CPU, Memory 사용량 등을 모니터링 하는데 사용했던 JMX를 유비쿼터스 환경에 접목시켜 애플리케이션을 모니터링 구조를 설계하였다.

JMX의 구조에 요소 핸들러(Factor Handler)를 추가하였다. 모든 MBean을 MBean 서버에 등록을 기존의 방법과는 달리 요소 핸들러를 둠으로써 관리자가 모니터링 할 MBean만 MBean 서버에 등록을 한다. 이로써 지연시간과 시스템의 부하를 줄일 수 있다.

본 논문은 유비쿼터스 지능형 공간에서 상황을 인식하고 추론하기 위해 수집되는 정보들의 변화를 실시간으로 모니터링 하는 시스템을 설계하였다. 이러한 모니터링 시스템의 설계로 상황 인지를 위한 정보들을 제공하여 사용자에게 최적의 서비스를 제공할 수 있다.

5. 참고문헌

[1] Anind K.Dey, Gregory D. Aborod, and Daniel Salber, "A conceptual Framework and a Toolkit for supporting the Rapid Prototypeing of Context-Aware Application," Human-Computer Interaction(HCI) Journal, vol.16, 97-166, 2001.

[2] Harry Chen, Tim Finn, and Anupam Joshi, "An Intelligent Broker for Context-Aware Systems," Adjunct Proceedings of Ubicomp, 12-15, October 2003

[3] Sun Microsystems Inc. "Java Management Extension white Paper," revision 01, June 1999

[4] BG Sullins, and MB Whipple, "JMX In Action," Manning Publications, 2002

[5] JMX Hompage, http://java.sun.com/products/JavaManagement/index.html

[6] SUN Homepage http://kr.sun.com/developers/techtips/e2005_0222.html#1

[7] David & Danny's Column http://www.davidndanny.com

[8] Roseman, Michael, and Recker, "Context-aware process design: Exploring the extrinsic drivers for process flexibility," 18th international Conference on advanced information system engineering, 149-158, Jan 2006

[9] MH Guiagoussou, R Boutaba, and M Kadoch, "A Java API for advanced faults management," IEEE Communications Magazine, 14-18, 483-498, May 2002

[10] Patton, S.J, "Using Java management extension standards as the basis for IceCube's experiment control," IEEE Communications Magazine, 4-10, 4, June 2005