특집

임베디스 시스템 미들웨어의 기술 동향

김수중, 윤용익 (숙명여자대학교, 정보과학부)

I. 서 론

모바일 컴퓨팅(Mobile Computing)과 더 나아가 유비쿼터스 컴퓨팅(Ubiquitous Computing)의 등장은 사용자가 시간과 장소에 구애받지 않고 자신이 원하는 정보와 서비스에 접근할 수 있는 가능성을 제공하고 있다. 그러나 이처럼 컴퓨터가 광범위하고 다양한 경우에 사용될 때 사용자는 서로 다른 입장에서 동일한 서비스로부터 별개의 정보를 획득하려 하기 때문에 이로 인한 새로운 문제점과 상황 처리에 대한 요구가 제기되고 있다.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 이루어지도록 해야한다. 특히, PDA부터 워크스테이션에 이르는 다양한 시스템이 연결된 환경에서 요구되는 적응성은 상위 사용자 레벨에서 하위 시스템 레벨까지 시스템의 모든 측면에 적용된다. 이러한 적응성은 통신 성능, 자원 사용, 위치, 응용 서비스 및 사용자의 요구사항 등을 포함한다.

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. 이러한 문제를 해결하기 위하여 미들웨어 레벨에서의 적응성 지원이 제기되면서 이미 리플렉티브 미들웨어에 대한 연구가 진행되었다.

유비쿼터스 컴퓨팅 환경을 위한 미들웨어는 객체 간의 통신을 투명하게 관리할 수 있도록 상황 인식이 가능한 구조이어야 한다. 또한 미들웨어가 응용 프로그램으로 하여금 상황 정보를 용이하게 획득할 수 있도록 하고 여러 가지 논리를 사용하여 추론하며 상황 변화에 적응할 수 있어야 한다. 유비쿼터스 환경을 고려해 볼 때, 미들

웨어가 갖추어야 할 주요한 요건에는 적응성 지원 이외에 서비스 코드의 이동성(mobility)에 대한 관리를 들 수 있다. 유비쿼터스 응용 서비스에서는 사용자와의 원활한 상호작용 뿐만 아니라 사용자 요구사항에 대한 반영이 중요시되며, 다양한 형태의 사용자 시스템들에게 유연한 서비스 제공이 이루어져야 한다.

본 논문에서는 유비쿼터스 컴퓨팅 환경에서 사용자의 요구사항을 능동적으로 반영하고 사용자에게 유연한 서비스를 제공하기 위해서 유비쿼터스 컴퓨팅 환경을 구성하는 다양하고 동적인 요소 즉, 상황의 변화를 인식하고 이에 적응할 수 있는 상황인식 미들웨어 현 연구 동향을 제시한다.

II. 적응 시스템

임베디드 시스템 소프트웨어는 실제로 외부에서 동적으로 변화하는 사용자와 관련된 사항들을 실시간으로 변화된 정보를 수집하여 시스템 내에서 반영 시켜, 시스템이 다음 응용을 지원하기 편이하게 유지하고 있어야 한다. 이러한 적응성은 통신상의 성능, 자원 사용, 위치, 응용 서비스의 선호도 등을 포함한다. 이러한 기능이 적응성 미들웨어의 기능이다. 본 장에서는 현재까지 연구되고 있는 적응성 지원 시스템들에 대한 기술 동향이다.

1. Odyssey

Odyssey는 다양한 모바일 정보 응용 서비스를 지원하기 위해 설계된 플랫폼이다. Odyssey는 음성이나 비디오와 같은 멀티미디어에 적합하도록 응용 인식 적응(application-aware adaptation)을 지원하며 이러한 방법은 자원의 사용 가능한 정도에 따라 멀티미디어 데이터의 품질을 조절할 수 있도록 한다. Odyssey에서는 자원 소비와 성능에 맞춘 데이터 품질도를 fidelity라고 정의한다. 예를 들면, 대역폭이 현저히 감소했을 때 웹 브라우저에서 요구하는 이미지의 압축률이나 비디오 재생기가 요구하는 프레임율과 같은 것이다. Odyssey의 응용 인식 적응은 운영체제와 응용 간의 협력 모델을 통해 제공되는데, 이 협력 모델에서 운영체제는 상세한 자원 가용성을 결정하고 응용은 적합한 적응 정책을 결정하여 이에 따라 적응을 수행한다. 이러한 결정 과정은 fidelity 선택, 변화 검출, 응용에 대한 통지의 단계로 구성된다. 시스템으로부터 자원 변화 통지를 받은 응용은 데이터에 대한 접근 패턴을 적응시켜야 한다. 그러나 이와 같은 통지에 의한 적응 방법은 적응 정책 조정을 지원하지 않기 때문에 응용 개발자의 부담을 가중시키게 되므로 비효율적이다.

2. GUIDE

GUIDE는 Lancaster University에서 수행 중인 프로젝트로서 그 주요 목표는 Lancaster시의 방문객들에게 제공할 Context-Sensitive 모바일 멀티미디어 가이드의 개발이다. GUIDE 시스템의 응용은 상황 인식 정보에 대한 접근, 관광 일정의 맞춤 구성, 인터랙티브 서비스에 대한 접근, 메시지 송수신 기능을 제공한다. 방문객은 GUIDE 시스템을 사용하여 현재 위치(현재 상황)에 따라 웹 기반의 하이퍼텍스트 정보를 검색할 수 있고 GUIDE 시스템이 제공하는 카테고리로부터 방문을 원하는 장소를 선택할 수 있다. 시스템은 방문객이 선택한 장소들에 대한 방문

순서를 제시하여 방문객이 관광 일정을 구성할 수 있도록 한다. 또한 인터랙티브 서비스를 통해 호텔 숙박을 예약하거나 극장 좌석을 예매할 수 있다. 메시지 송수신 기능은 서로 다른 장소를 방문하는 방문객 그룹의 구성원들 사이에 메시지로 연락을 취할 수 있도록 하는 기능이다. GUIDE 시스템의 정보 제공은 무선 연결 상태에 대해 종속적이다. 따라서 방문객에게 연결 상태에 대한 피드백을 제공하여 연결이 해제된 동안 방문객이 사용할 수 있는 기능이 제한된다는 것을 알린다. 또한 방문객의 장치가 네트워크로부터 연결 해제되었을 경우에는 최근 위치 정보를 이용할 수 없으므로 사용자 인터페이스를 통해 시스템의 위치 정보 수신 상태를 알린다.

III. 리플렉티브 미들웨어

리플렉션(reflection) 기술은 프로그래밍 언어 분야에서 보다 개방적이고 확장성 있는 언어 설계를 지원하기 위해 사용되기 시작하였고 지금까지 운영체제와 분산 시스템, 미들웨어 등의 여러 다른 분야에서도 적용되고 있다. 리플렉션이란 스스로 추론하고 행동할 수 있는 시스템의 능력으로서 리플렉티브 시스템은 검사(inspection)와 적응(adaptation)에 따르는 시스템의 행동에 대한 표현을 제공하고 하부의 동작에 대해 Casually connected되어 있는 시스템이다.

리플렉티브 시스템은 메타 레벨과 베이스 레벨로 구성된다. 메타 레벨은 하위 레벨에 위치하는 객체에 대한 연산을 수행하고, 베이스 레벨은 응용 도메인 개체에 대한 연산을 수행한다. 또한 런타임에 하부 구현(베이스 레벨)에 대한 검사와 적응을 지원하는 메타-인터페이스(meta-interface)를 제공하고, 메타-객체 프로토콜(meta-object protocol:MOP)을 제공하여 메타-레벨에서 사용할 수 있는 서비스를 정의한다. 리플렉티브 미들웨어는 기존의 미들웨어 플랫폼의 문제를 극복하기 위한 차세대 미들웨어 플랫폼으로서 멀티미디어와 모바일 컴퓨팅 등의 분야에 적용하기 위해서는 다음과 같은 조건들이 요구된다.

- 형상(Configurability): 응용 도메인의 요구사항을 충족시킬 수 있도록 구성 가능해야 한다.
- 동적 재형상(Dynamic Reconfigurability): 플랫폼이 환경의 변화에 응답할 수 있도록 동적으로 재구성할 수 있어야 한다.
- 요구사항의 변화에 따라 발전된 플랫폼 설계를 지원해야 한다.

1. OpenCorba

OpenCorba는 리플렉티브 CORBA 브로커로서 원격 호출(remote invocation), IDL 타입 검사, 인터페이스 저장소 오류 처리 등과 같은 ORB의 런타임 동작에 동적인 적응성을 제공한다[2]. OpenCorba의 구조는 베이스 레벨과 메타 레벨 사이의 명확한 분리를 제공하기 위하여 메타 클래스 프로그래밍을 지원한다. 베이스 레벨은 인스턴스의 동작을 나타내는 클래스로 구성되는 반면, 메타 레벨은 객체의 생성, 캡슐화, 상속 규칙, 메시지 처리 등과 관련된 클래스의 동작을 정의하는 메타 클래스로 구성된다. OpenCorba의 주요한 리플렉티브 관점은 프록시 클래스를 통한 원격 호출 메커니즘을 동적으로 수정하는 것이다. OpenCorba의 IDL 컴파일러는 프록시 클래스를 생성하여 이것을 서비스 요청을 가로채는 메타 클래스에 연결한 후 실제 서버 객체

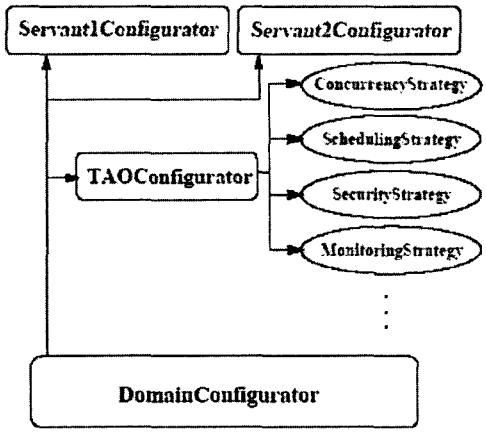
로의 원격 호출을 발생시킨다. 호출 시맨틱스는 런타임에 프록시와 다른 메타클래스를 연관함으로써 재정의할 수 있다. 이러한 스킴에 의해 객체 이동(object migration)이나 복제와 같은 분산 시스템의 성능을 향상시킬 수 있는 새로운 메커니즘이 가능하게 되었다.

2. dynamicTAO

dynamicTAO는 ORB 엔진의 런타임 재구성과 비-CORBA 응용을 지원하기 위하여 TAO를 확장한 리플렉티브 CORBA ORB이다[3]. dynamicTAO는 메타-인터페이스를 사용하여 분산 시스템을 통해 컴포넌트를 전송하고 런타임에 시스템으로 모듈을 로딩하거나 시스템으로부터 모듈을 업로딩하며 ORB 구성 상태에 대한 검사와 변경을 수행한다.

dynamicTAO에서 내부 시스템은 컴포넌트 형상자(component configurator)를 통해 나타내어진다. 컴포넌트 형상자는 ORB 컴포넌트 간의 종속성과 ORB와 응용 컴포넌트 간의 종속성을 저장한다. dynamicTAO ORB를 수행하는 각 프로세스는 컴포넌트 형상자의 인스턴스인 DomainConfigurator를 포함한다. DomainConfigurator는 ORB 인스턴스와 프로세스 도메인 안에서 수행되는 서번트(servant)에 대한 레퍼런스를 유지한다. ORB의 각 인스턴스는 맞춤 구성된 컴포넌트 형상자인 TAO Configurator를 포함한다. TAOConfigurator는 스케쥴링, 보안, 모니터링 등의 여러 전략에 대한 훅(hook)를 가지고 있는데 이러한 전략들은 동적으로 로딩할 수 있는 라이브러리 형태로써 실제화 된다.

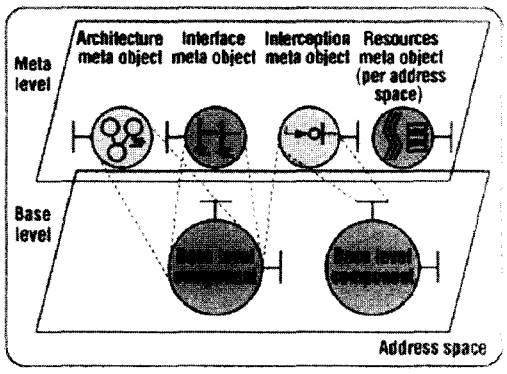
dynamicTAO의 구조적인 프레임워크는 그림 1에서 나타내었다. Persistant Repository는 로컬 파일 시스템 내에 구현된 카테고리를 저장하고, Network Broker는 네트워크로부터 재구성 요청을 받아 Dynamic Service Configurator에게 전달한다. Dynamic Service Configurator는 DomainConfigurator를 포함하며 런타임에 컴포넌트를 동적으로 재구성하기 위한 공통 오퍼레이션을 제공한다.



〈그림 1〉Reifying dynamicTAO structure

3. Open ORB

Open ORB는 Lancaster 대학에서 개발한 컴포넌트 기반의 리플렉티브 미들웨어 플랫폼으로서 응용 프로그램의 동적인 요구사항, 예를 들면 멀티미디어, 실시간, 이동성 등을 지원하기 위하여 동적으로 재구성할 수 있는 플랫폼을 목표로 설계 및 구현되었다[4]. Open ORB는 다양한 미들웨어 기능을 구현하기 위하여 다중 인터페이스를 갖는 컴포넌트 모델을 기반으로 구축되었다. 각각의 컴포넌트는 하나의 메타 공간(meta-



〈그림 2〉 Open ORB의 메타-공간 구조

space)과 연관되며, 이 메타 공간은 미들웨어 구현에 대한 서로 다른 관점을 다루는 여러 메타 모델(meta-model)로 분할된다. 현재의 Open ORB는 그림 2와 같이 인터페이스, 구조, 인터셉션, 자원 메타 모델을 지원한다.

각 메타 모델에 대한 접근은 메타 객체 프로토콜(MOP)을 통해 이루어지고 메타 객체 프로토콜은 메타 모델이 제공하는 서비스를 정의한다. 구조적인 리플렉션은 인터페이스 메타 모델(interface meta-model)과 구조 메타 모델(architectural meta-model)에 의해 지원되고 행동적인 리플렉션은 인터셉션 메타 모델(interception meta-model)과 자원 메타 모델(resource meta-model)에 의해 지원된다. 인터페이스 메타 모델은 컴포넌트의 외부 표현(external representation)을 다루며 MOP는 이러한 인터페이스 정의를 통해 특정 컴포넌트가 제공하는 서비스를 동적으로 탐색한다. 구조 메타 모델은 컴포넌트 그래프와 구조적인 제약 조건으로 구성되는 컴포넌트 구현에 대한 접근을 제공한다. 컴포넌트 그래프(component graph)는 로컬 바인딩(local binding)에 의해 연결되는 컴포넌트의 집합으로 표현되는 것으로, 여기서 로컬 바인딩은 제공되는 인터페이스와 요구되는 인터페이스를 단일 주소 공간에서 매핑하는 것이다.

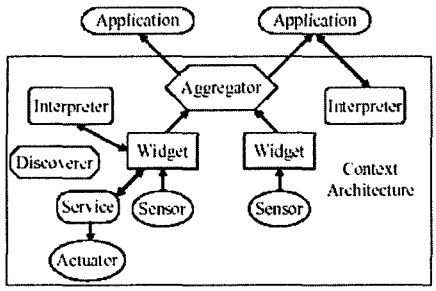
인터셉션 메타 모델은 인터셉터(interceptor)의 동적인 삽입을 가능하게 한다. 이러한 인터셉터는 인터페이스와 연관되어 있으며 선행 동작 및 후행 동작을 삽입할 수 있다. 또한 인터셉터는 보안 검사나 동시성 제어와 같은 비-기능적인(non-functional)인 동작을 부가적으로 도입할 수 있다. 자원 메타 모델은 하부 자원 및 자원 관리에 대한 접근을 제공하고 자원과 태스크의 추상화에 기반한다.

IV. 상황 인식 미들웨어

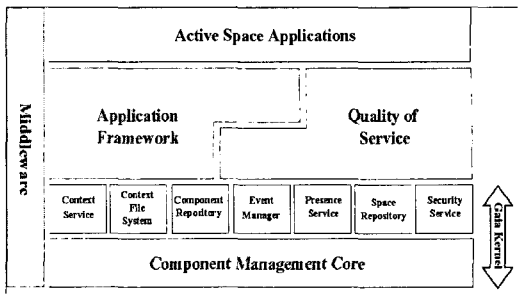
1. Context Toolkit

Anind Dey는 그의 연구에서 상황 인식 컴퓨팅을 정의하고 상황 인식 응용의 구축 및 이러한 응용을 신속하게 프로토타이핑 할 수 있는 툴킷 개발에 요구되는 지원 사항을 설명하였다[5, 6]. 이 연구에서는 개념적인 프레임워크를 제안함으로써 상황 인식 응용의 설계와 개발을 위한 기반을 마련하였다. 제안된 개념적 프레임워크는 상황 획득과 응용에서의 상황 정보 사용의 관계를 분리하여 응용에 독립적인 방식으로 상황을 획득, 수집 및 관리할 수 있는 추상화를 제공하고 이에 해당하는 소프트웨어 컴포넌트를 식별한다.

Dey는 특정한 상황 오퍼레이션을 처리하기 위하여 개념적인 프레임워크의 컴포넌트들을 5개의 카테고리로 분류하였는데 그 각각은 상황 위젯(context widget), 인터프리터, 집합자(aggregator), 서비스, 탐색자(discoverer)이다. 이 컴포넌트들은 그림 3과 같은 형태로 구성될 수 있다.



〈그림 3〉 Context Toolkit의 컴포넌트 구성



〈그림 4〉 Gaia 구조

2. Gaia Project

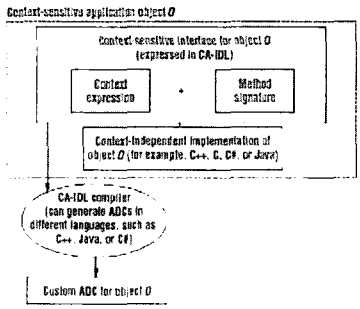
Gaia 미들웨어는 상황에 대한 술어 모델(predicate model)을 기반으로 하며 상황 모델과 미들웨어는 1차 논리(first order logic), 일시적 논리(temporal logic), 퍼지 논리(fuzzy logic) 등의 여러 가지 추론 메커니즘을 제공하여 에이전트가 서로 다른 상황에서 추론하고 행동을 결정할 수 있도록 한다. 에이전트는 각기 다른 상황에서 여러 행동을 학습하기 위하여 베이지언 학습(Bayesian learning), 보강 학습(reinforcement learning)과 같은 학습 메커니즘을 사용할 수 있다.

Gaia의 구조는 그림 4와 같다. Gaia 미들웨어에서의 에이전트는 분산 능력에 따라 미들웨어 컴포넌트와 응용(context consumer) 을 포함한다. 에이전트는 추론 메커니즘이나 학습 메커니즘을 사용하여 현재 상황에 관한 다양한 조건들을 추론하고 논리적인 질의에 응답하며 각기 다른 상황에서 행동 방식을 적응시킨다.

3. RCSM(Reflective Context-Sensitive Middleware)

RCSM은 상황 인식과 ad hoc 통신을 요구하는 응용 개발을 위해 설계된 미들웨어로서 응용이 상황에 근거하여 활성화될 메소드를 결정하기보다는 미들웨어가 응용 내의 적절한 행동을 직접 트리거(trigger)하는 새로운 방안이다. 이 연구의 동기는 새로운 상황 소스를 추가함으로써 기존의 상황 인식 응용을 확장하고 동시에 발생하는 다중의 상황들이 특정한 행동을 쉽게 유발하도록 하기 위한 것이다.

CORBA와 COM과 같은 기존의 미들웨어와 유사하게 RCSM은 상황 인식 응용을 위해 객체 기반 프레임워크를 제공한다. RCSM은 그림 5와 같이 상황 인식 응용 소프트웨어를 상황 인식 객체로 모델링 하였으며 이는 상황 인식 인터페이스와 상황에 독립적인 구현으로 구성된다. 상황 인식 인터페이스는 응용의 상황 인식에 대한 기술을 캡슐화한다. 응용의 상황 인식을 구체적으로 언급하자면, 응용이 사용하는 상황와 응용이 제공하는 기능들, 그리고 명시된 상황와 기능 사이의 매핑 정보이다. 상황에 독립적인 구현은 응용 소프트웨어가 반드시 제공해야 할 기능에 대한 실제적인 구현이다. 이러한 구조의 중요한 특성은 구현이 상황 명세로부터 완전히 분리되기 때문에 상황에 독립적이라는 것이다. RCSM은 상황 인식 인터페이스를 사용하여 어떠한 상



〈그림 5〉 RCSM의 상황 인식 모델

황를 모니터링할 것인지와 명시된 상황가 유효화될 때마다 응용의 어떠한 행동을 활성화시킬 것인지를 결정한다.

R-ORB가 센서와 운영체제로부터 데이터를 수집하면 ADC는 R-ORB를 통하여 가공되지 않은 필요한 데이터를 주기적으로 수집한다. 초기에 각 ADC는 R-ORB에 등록함으로써 상황에 대한 요구를 나타내고 이에 상응하는 상황 인식 인터페이스를 알린다. 미들웨어는 런타임 동안 각각의 ADC를 추가하거나 삭제할 수 있도록 함으로써 RCSM 내부의 다른 런타임 오퍼레이션에 영향을 주지 않고 새로운 또는 기존의 상황 인식 응용 객체를 관리하기 때문에 RCSM은 재구성 가능한 시스템이라 할 수 있다.

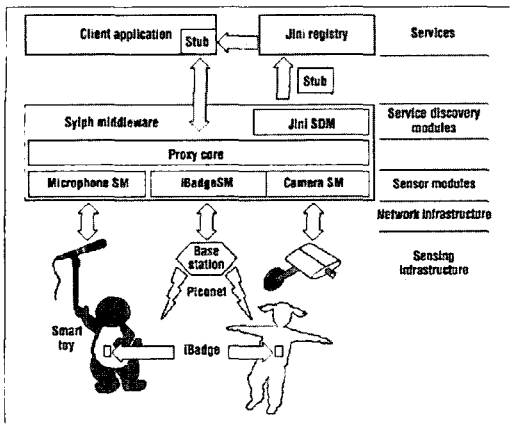
Context Toolkit 미들웨어 및 다른 관련 프로젝트와 비교해 볼 때, RCSM은 상황 인식을 충분히 지원하는 보다 진보한 미들웨어라 할 수 있다. RCSM은 상황 정보를 동적으로 처리하고 응용 개발자들이 상황 정보 획득에 대한 상세한 사항에 구애받지 않고 선호하는 언어를 사용하여 기능들을 구현할 수 있도록 한다.

4. Smart Kindergarten

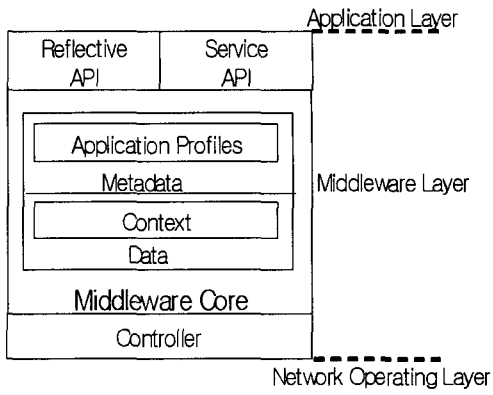
Smart Kindergarten(SmartKG)은 UCLA에서 수행 중인 공동 프로젝트로서 아동 초기 교육을 위하여 스마트 환경에서 임베디드 처리와 무선 통신 및 센싱 기술을 시험하는 시스템이다. SmartKG는 어린 아동들이 그들의 환경 안에 있는 장난감과 같은 객체와의 상호작용을 통해 학습할 수 있는 응용을 제공한다. 이 프로젝트에서 궁극적으로 계획하고 있는 시스템은 상황에 적응하고 여러 아동들의 행동을 조화시키며 교사에 의한 계속적인 학습 과정 평가를 가능하게 하는 각 아동마다 개별화된 환경을 아동에게 제공함으로써 교육 과정을 개선하기 위한 것이다. 이는 센서가 강화된 무선 네트워크 상의 장난감과 백엔드(backend) 미들웨어 서비스 및 데이터베이스 기술을 갖춘 강의실 객체에 의해 이루어진다. SmartKG는 유선 하부구조 내의 상황 데이터를 분석 및 저장하고 이후의 사용을 위해 상황 정보를 영구적인 데이터로 변환한다. 그림 6은 SmartKG의 시스템 구조이다.

SmartKG 프로젝트에서는 미들웨어 하부구조인 Sylph가 구현되었다. Sylph는 각기 다른 센서에 대해 Jini 코드를 반복하지 않고 MUSE에 새로운 센서를 도입하기 위한 미들웨어 시스템으로서 서비스 탐색 처리와 SmartKG의 변환기에 대한 접근을 중재하기 위한 경량의 확장 가능한 프록시 서비스를 수행한다. Sylph는 미들웨어 기능을 제공하기 위하여 다음의 세 가지 컴포넌트를 사용한다.

- 센서 모듈 : 센서 하드웨어와의 통신 기능 수행
- 서비스 탐색 모듈 : 룩업 서비스를 통해 센



〈그림 6〉 Smart Kindergarten



〈그림 7〉 XMIDDLE 구조

서 통지

- 프록시 코어 : 응용 질의 관리 및 두 타입의 모듈 사이의 중계

Sylph의 모듈화 특징으로 인해 센서 개발자들은 단순한 하드웨어 인터페이스에 초점을 두고 프록시 코어는 질의 실행과 자원 관리의 복잡성을 처리할 수 있다. 이와 유사하게, 서비스 탐색 모듈은 Sylph가 다양한 탐색 메커니즘을 통하여 서비스를 export할 수 있도록 한다. 각 추상화 레벨에서 확장 가능한 모듈을 사용함으로써, Sylph는 많은 기술들은 응집력 있는 정보 시스템으로 통합할 수 있다.

5. XMIDDLE

위치기반 상황인식 서비스는 모바일 컴퓨팅 환경에서 수시로 이동하는 단말과 사용자의 위치를 인식하여 다양한 위치 인식 응용에 활용할 수 있도록 하는 것으로, MS의 UPnP(Universal Plug and Play), Sun의 Jini, Oracle의 9iASWE (Oracle9i Advanced Search Wireless Enabling), 카네기 멜론 대학교(CMU)의 Coda, 런던대학교(UCL)의 XMIDDLE(Information Sharing Middleware for Mobile Environment) 등이 있다.

이 중, XMIDDLE은 LIME모델의 데이터 구조 부분을 보완하여 데이터의 반복(replication)과 일치(reconciliation)에 중점을 둔 모델이다. 데이터의 일치는 호스트나 호스트 그룹의 연결이 끊어진 상태에서 정보가 변경될 때, 심각한 문제가 발생한다. XMIDDLE은 이 작업을 application-dependent 방식으로 지원하는 미들웨어를 제공한다. 그림 7은 XMIDDLE의 구조이며 응용 프로파일(application profiles)에서 메타데이터를 관리하고, 외부의 변화를 컨텍스트 데이터로 다루는 미들웨어 코어의 구조를 볼 수 있다.

XMIDDLE의 개념은 정적인 데이터와 동적인 데이터를 분리하여 Application Profile과 Context로 표현하는 간단한 구조의 개념이지만, Reflective API는 미들웨어 코어나 서비스로 전환하는 메커니즘을 제공하지 못한다.

V. 결론

본 논문에서는 임베디드 시스템 시스템의 새로운 추세인 모바일 컴퓨팅 환경을 포함한 유비쿼터스 컴퓨팅 환경에서 모바일 단말기와 응용 서버 간에 지속적인 사용자의 서비스를 지원하기 위한 미들웨어의 기술 동향에 대하여 언급하였다. 이러한 미들웨어는 초기의 일반적인 유선 환경의 분산 시스템을 지원하기 위한 구조에서 많은 변화가 있었던 것을 알 수 있다.

최근 이동성이 강한 사용자들을 지원하기 위하여 사용자의 이동성, 형상 등의 정보를 시스템에 반영하기 위한 적응성 기법의 연구와 동적으로 수집된 사용자 정보들을 실제 시스템에 반영하여 시스템의 지원 능력을 능동적으로 향상 시키는 리플렉티브 미들웨어 기술은 이미 확보되었다고 볼 수 있다.

앞으로 실제 환경에서 모바일 디바이스와 응용 서버 간에 지속적은 응용 서비스들을 지원하기 위한 상황 인식 미들웨어 개발이 가시화 되고 있다. 상황인식 미들웨어의 핵심 부분에는 기 연구되었던 적응성 기법과 리플렉티브 미들웨어의 기술이 활용되고 있다. 그러나, 실제로 이러한 기술들을 상황 인식 미들웨어에 적용하여 실제 환경에 적용하기 위해서는 각 응용 서비스의 요구 사항을 충분히 반영하여 개발되어야 하는 특징이 있기 때문에 다양한 분야에서 다양한 형태의 미들웨어가 연구 개발되고 있다.

참고문헌

[1] A. Friday, N. Davies, G.S. Blair, K.W.J. Cheverst. "Developing Adaptive Applications: The MOST Experience", Journal of Integrated Computer-Aided Engineering, 6(2):143-157, 1999.

[2] T. Ledoux. "OpenCorba: a Reflective Open Broker. In Reflection' 99, volume 1616 of LNCS. Springer Verlag, 1999.

[3] F. Kon, M. Roman, P. Liu, J. Mao, T. Yamane, L. C. Magalhaes, and R. H. Campbell. "Monitoring, Security, and Dynamic Configuration with the dynamicTAO Reflective ORB", IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware' 2000), New York, April 3-7, 2000.

[4] G. S. Blair, G. Coulson, A. Anderson, L. Blair, M. Clarke, F. Costa, H. Duran-Limon, T. Fitzpatrick, L. Johnston, R. Moreira, N.Parlavantzas, and K. Saikoski. "The Design and Implementation of OpenORB v2", IEEE DS Online, Special Issue on Reflective Middleware, Vol. 2, No. 6, 2001.

[5] A. K. Dey and Gregory D. Abowd. "The Context Toolkit: Aiding the Development of Context-Aware Applications", In the Workshop on Software Engineering for Wearable and Pervasive Computing, Limerick, Ireland, June 6, 2000.

[6] A. K. Dey. "Providing Architectural Support for Building Context-Aware Applications", PhD thesis, College of Computing, Georgia Institute of Technology, December 2000

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

[8] 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.

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

저자소개

윤 용 익

1983년 동국대학교 통계학과 학사
1985년 한국과학기술원 전산학과 석사
1994년 한국과학기술원 전산학과 박사
1985년-1997년 한국전자통신연구원 책임연구원
1997년-현 재 숙명여자대학교 정보과학부 부교수
1997년-현 재 한국 정보 통신 대학교 부교수
주관심분야 미들웨어, 실시간 분산 시스템, 임베디드 시스템 소프트웨어,멀티미디어 분산 시스템

김 수 중

1995년 숙명여자대학교, 전산학과 학사
1998년 숙명여자대학교, 전산학과 석사
2004년 숙명여자대학교, 컴퓨터과학과 박사
주관심분야 상황 인식 미들웨어, 유비쿼터스 컴퓨팅