# 경량 모바일 미들웨어 원시 시스템 설계

## Design of Lightweight Mobile Middleware Naive System

양승일, 이태규, 박성훈
충북대학교 컴퓨터공학과

Seung-Il Yang(ysipaul@yahoo.co.kr), Tae-Gyu Lee(tigerlee88@empal.com),
Sung-Hoon Park(spark@cbnu.ac.kr)

**요약**

기존의 미들웨어 시스템은 유선 환경을 중심으로 개발되었기 때문에 미들웨어 및 관련 응용 프로그램들의 규모 또는 운용성이 크다. 따라서 이동 단말의 CPU 및 메모리 등 하드웨어 자원의 취약성과 네트워크의 낮은 대역폭을 가진 무선 이동성 환경에 적용하기 어렵다는 문제가 발생한다. 본 논문은 이러한 문제점을 극복하기 위해서 미들웨어 응용 및 실행 단위가 모바일 환경에 적합하게 최소 및 최적화되고 이동시에도 끊어짐 없는 서비스를 제공하기 위한 경량 모바일 미들웨어 시스템인 "LightMware"시스템을 설계 및 제안한다.

■ 중심어 : | 미들웨어 | 경량 | 모바일 |

**Abstract**

A conventional middleware system is optimized for wired computing environments. The wireless mobile devices have several disadvantages of low-speed processors, small memory and narrow bandwidth of wireless network. To overcome those problem issues in wireless mobile environments, middleware and many middleware-related applications have to be changed of the small sized components. In this paper, we design a lightweight mobile middleware system called "LightMware" which is optimized for mobile environment and middleware applications

■ keyword : | Middleware | Lightweight | Mobile |

## I. 서 론

최근 IT 환경은 모바일 사용자들을 지원하는 유비쿼터스 환경으로 진화하고 있다. 유비쿼터스 환경에서는 PDA, notebook, smart phone 등의 모바일 기기들을 사용하므로 CPU, memory, network 자원이 취약하다. 기존 미들웨어 시스템은 유선 환경을 중심으로 개발되었기 때문에 미들웨어 및 관련 응용 프로그램들의 규모 및 운용성이 크다. 따라서 유비쿼터스 환경에서 이동사용자의 무선 이동성을 지원하기 위해서는 실행프로세스 경량화 및 무선이동성이 요구된다. 이러한 문제점들을 극복하기 위해서 본 연구는 미들웨어 응용 및 실행 단위가 모바일 환경에 적합하게 최소 및 최적화되어야 한다는 데 그 필요성을 인식하고 있다. 이러한 문제점들을 극복한 미들웨어 시스템을 본 연구는 "경량 모바일 미들웨어 시스템 (Lightweight Mobile Middleware

* 본 연구는 교육과학기술부와 한국산업기술재단의 지역혁신인력양성사업으로 수행된 연구결과임.

접수번호 : #090730-006
접수일자 : 2009년 07월 30일
심사완료일 : 2009년 09월 03일
교신저자 : 이태규, e-mail : tigerlee88@empal.com

Systems)" 또는 "LightMware"라고 한다.

본 경량 모바일 미들웨어 시스템은 모바일 클라이언트의 운영환경에 맞게 미들웨어 및 그 응용 실행 단위를 최소화하는 것이 요구된다. 이러한 목적을 실현하기 위해서 미들웨어 전체 시스템의 구성을 기능 및 사용자 응용 단위로 나누어서 컴포넌트(Component) 단위로 구성하는 것이 요구된다. 그러므로 본 논문은 이러한 관점을 기반으로 하여 다음과 같이 구성된다. 2장에서는 관련 연구를 알아보고, 3장에서는 경량모바일미들웨어 시스템을 설계하고, 4장에서는 결론을 맺는다.

## II. 관련연구

미들웨어란 운영체제와 응용프로그램 사이에 존재하는 소프트웨어 계층으로 사용자에게 하부의 하드웨어, 운영체제, 네트워크에 상관없이 분산 컴퓨팅 환경, 원격 프로시저 콜, ORB(Object Request Broker) 메시징과 같은 서비스를 제공할 수 있도록 도와주는 소프트웨어로 정의할 수 있다[1].

미들웨어의 종류는 기능과 응용분야에 따라서 분류된다. 미국의 IEEE 분산시스템 온라인 커뮤니티에서는 이러한 미들웨어를 크게 네 가지로 나눈다[2]. 첫째는 리플렉티브 미들웨어(Reflective Middleware)이다. 리플렉티브 미들웨어는 자신의 컴퓨팅 환경에서 실제적인 행동과 상태의 원인에 따라 스스로가 문제를 판단하고 적용시킬 수 있는 CCSR(Casually Connected Self Representation)을 통해 최적의 컴퓨팅 환경을 만들어 주는 미들웨어이다. 둘째는 이벤트기반 미들웨어로서 개념, 설계, 수행능력, 서비스 또는 컴포넌트들의 응용에 중점을 둔 미들웨어이다. 셋째로는 객체지향형 프로그래밍 패러다임에서 분산시스템으로 확장된 객체지향형 미들웨어이다. 넷째는 메시지 지향 미들웨어(Message Oriented Middleware)이다. 이러한 미들웨어 중에서 주위 환경의 변화에 맞추어 스스로 변화하고 적응할 수 있는 자가 지식 성장이 가능하고 이동성을 가진 유비쿼터스 환경에 가장 적합하고 발전 가능성이 있는 미들웨어는 리플렉티브 미들웨어이다. 그 핵심 기능에 따라 어댑티브 미들웨어(Adaptive Middleware), 상황인지 미들웨어(Context Aware Middleware), 리플렉티브 미들웨어로 나눈다[3]. 리플렉티브 미들웨어는 단단한 구조의 블랙박스처럼 구성된 이러한 기존 미들웨어들과는 달리 상호 협업이 가능한 컴포넌트들의 그룹으로 구성된다. 이러한 특성으로 인해 리플렉티브 미들웨어는 기존의 미들웨어와 상호작용이 가능하고 모바일 환경에 적합한 최소형 미들웨어 엔진 구성이 가능하다[4]. 이러한 리플렉티브 미들웨어는 지속적으로 발전하고 있으며, OpenCorba, dynamicTAO, OpenORB (Lancaster Univ) 등이 있다[5].

본 논문은 유비쿼터스 환경에서 적응적 리플렉티브 미들웨어에 기초한다. 다음은 리플렉티브 미들에어 정의(definition)와 종류(classification)에 따른 모바일 컴퓨팅을 지원하기 위한 이슈들(issues)과 문제점들을 기술한다.

### 1. 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 따른 시스템의 행동에 대한 표현을 제공하고 하부의 동작에 대해 인과관계로 연결(casually connected) 되어 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( MOP : Meta-Object Protocol)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 다음 조

건들이 요구된다.

- 형상(Configurability): 응용 도메인의 요구사항을 충족시킬 수 있도록 구성 가능해야한다.
- 동적 재형상(Dynamic Reconfigurability): 플랫폼이 환경의 변화에 응답할 수 있도록 동적으로 재구성할 수 있어야 한다.
- 요구사항의 변화에 따라 발전된 플랫폼 설계를 지원해야 한다[6].

## 2. 리플렉티브 미들웨어 종류

### 2.1 OpenCorba

OpenCorba는 Smalltalk로 구현된 리플렉티브 코바(CORBA)로서 원격 호출(Remote Invocation), IDL 타입 검사, 인터페이스 저장소 오류처리 등과 같은 ORB의 런타임 동작에 동적인 적응성을 제공한다[7][11]. 구조는 베이스 레벨과 메타 레벨 사이의 명확한 분리를 제공하기 위하여 메타 클래스 프로그램을 지원한다. 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 객체로의 원격 호출을 발생시킨다. 객체 이동이나 복제와 같은 분산 시스템의 성능을 향상시킬 수 있는 새로운 메커니즘이 가능하게 되었다.

메타클래스들을 이용하여 미들웨어의 행동을 수정할 수 있다는 아이디어를 이용하지만 컴파일 시 모든 메소들에 대해 컴파일 함으로 실행 시 특정 메소드의 메시지 뿐 아니라 불필요한 메소드들의 메시지 전송을 발생시키므로 최적 경량화를 실현하기 위한 한계점을 가지고 있다.

### 2.2 dynamicTAO

ORB 엔진의 런타임 재구성과 TAO를 확장한 리플렉티브 코바(CORBA) ORB이다. 메타 인터페이스를 사용하여 분산 시스템 콜을 통해 컴포넌트를 전송하고 런타임에 시스템으로 모듈을 로딩(loading)하거나 시스템으로 부터 모듈을 언로딩(unloading)하며 ORB 구성 상태에 대한 검사와 변경을 수행한다[8][11].

DynamicTAO에서 내부 시스템은 컴포넌트 형상자(component configurator)를 통해 나타내어진다. 컴포넌트 형상자는 ORB 컴포넌트 간의 종속성과 ORB와 응용 컴포넌트 간의 종속성을 저장한다. 구조적인 프레임워크는 Persistent Repository와 Network Broker, Dynamic Service Configurator로 구성되는데 Persistent 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.

DynamicTAO는 두 가지 문제점이 있다. 첫째, 전략(Strategy)전환 시 이전의 전략이 존재하지 않는데 사용자에서 호출이 오면 시스템은 다운된다. 둘째 상태값을 전환 전부터 계속 현재 까지 유지시켜야 하므로 툴들이 도입이 되는데 이는 핵심 컴포넌트의 사이즈를 점점 늘어나게 하는 원인이 된다. 그러므로 시스템이 무거워 이동 환경에는 적합하지 않다.

### 2.3 OpenORB

Lancaster 대학에서 개발한 컴포넌트 기반의 리플렉티브 미들웨어 플랫폼으로서 Python으로 구현된 응용 프로그램의 동적인 요구사항을 지원하기 위하여 동적으로 재구성할 수 있는 플랫폼을 목표로 설계 및 구현되었다[9][11]. OpenORB는 다양한 미들웨어 기능을 구현하기 위하여 다중 인터페이스를 갖는 컴포넌트 모델을 기반으로 구축되었다. 각각의 컴포넌트는 하나의 메타 공간(Meta-space)과 연관되며, 이 메타 공간은 미들웨어 구현에 대한 서로 다른 관점을 다루는 여러 메타 모델로 분할된다. 현재의 OpenORB는 인터페이스, 구조, 인터셉션, 자원 메타 모델을 지원한다. 각 메타 모델에 대한 접근은 메타 객체 프로토콜(Meta Object Protocol)을 통해 이루어지고 메타 객체 프로토콜은 메타 모델이 제공하는 서비스를 정의 한다.

OpenORB는 멀티미디어를 지원하는 관점에서 코바의 제한성을 극복할 수 있는 점과 견고한 차세대 미들웨어 설계를 위해서는 유망한 시스템이다. 그러나 코바

가 구현된 위에 만들어져야 하므로 컴퓨팅(computing) 부하가 많이 발생하여 모바일에는 적합하지 않다.

**표 1. 리플렉티브 미들웨어예**

| 종류 | 주요특징 및 기능 | 단점 | tool |
|---|---|---|---|
| OpenCorba | ◦ORB의 런타임시 동적인 적응성 제공<br>◦프록시 클래스를 통한 원격 호출 메커니즘을 통하여 메타 클래스 동적전환 가능 | 불필요한 오버헤드 메시지 발생 | Smalltalk |
| dynamicTAO | ◦런타임시 ORB엔진과 NON-CORBA 응용프로그램의 재구성을 지원 | 핵심 컴포넌트 사이즈증가 | TAO확장 |
| OpenORB | ◦응용 프로그램의 동적인 요구사항 지원<br>◦유망한 차세대 미들웨어설계 가능 | 코바 구현 위에 만들어지므로 compu-ting 부하가 많음 | Python |

[표 1]은 현재까지 제시된 미들웨어를 비교한 표이다. 표에서 제시하는 것 같이 개발 구현 시스템들은 모바일 환경에 적합하지 않다.

본 연구는 이러한 미들웨어의 문제점을 극복하여 개발자 및 사용자에게 용이한 개발 및 응용 인터페이스를 지원하는 동시에 이동 컴퓨팅 응용 환경 및 사용자 시나리오에 기초한 경량화 및 이동성 지원 인터페이스(API)를 제공하는 미들웨어 시스템을 설계하고자 한다.

## III. 경량 모바일 미들웨어 원시 시스템 설계

본 논문은 리플렉티브 미들웨어를 기반으로 설계한 경량 모바일 미들웨어 시스템이다[10]. 산업현장에서 기존 미들웨어들를 사용하여 모바일환경에 적용하는 것은 적합하지 않으므로 이 문제를 해결하고자 경량 미들웨어 시스템을 설계하였다. 제안시스템은 모바일 환경에 적합한 실행 컴포넌트 크기로 다양한 응용프로그램을 제공하고 이동시에도 끊어짐 없이 서비스되는 것을 핵심으로 하고 있다. 모바일 기기들의 메모리공간은 작은 사이즈의 실행 파일들이 로드(load)되어 상황에 맞는 응용프로그램을 실행하는데 이때 실행 파일의 사이즈를 경량화 하는 것이 설계의 주안점을 두었다. 본 설계는 경량화에 대한 원시 시스템의 상황인지(CA : Context Awareness)에 대한 시간과 공간의 경량화 모델을 먼저 제시하고 다른 각 요소의 컴포넌트를 설계 및 구성하고 실행 환경 과 응용 시스템에 따라 동적으로 구성하도록 미들웨어를 설계 하는 것을 목적으로 둔다.

### 1. 시스템 설계 모델

시스템설계모델은 [그림 1]과 같이 분산네트워크시스템모델이다. 클라이언트는 유선과 무선을 통하여 연결될 수 있는데 유선을 통하여 연결되는 경우 클라이언트-서버환경이고 무선을 통하여 연결되는 경우 분산에이전트 서버 환경이다.

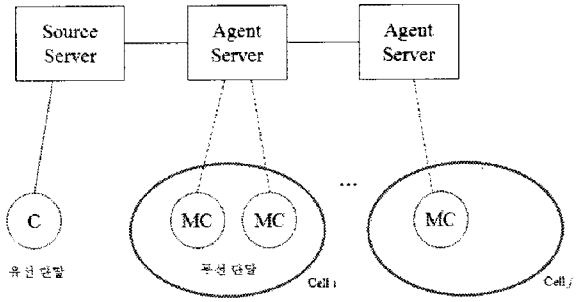


**그림 1. 시스템 모델**

본 논문은 이러한 시스템 모델을 기반으로 경량화(Lightweight) 모델과 이동성(Mobility) 모델에 대한 설계를 하였다. 경량화 모델은 시간기반 시나리오와 공간기반 시나리오를 기반으로 설계하였고, 이동성모델은 셀룰라(Cellular)모델과 셀기반(Cell-based) 위치관리 기법에 기초하여 설계한다.

### 2. 시스템 설계 환경

시스템 설계환경은 다음 [그림 2]와 같다.

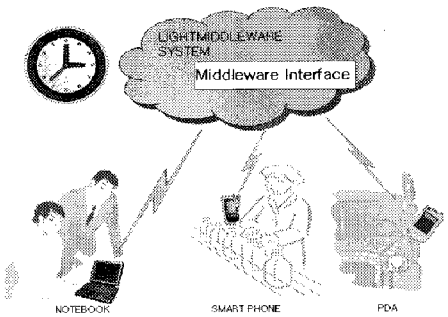


**그림 2. 미들웨어시스템 환경**

- 모바일 단말 : 노트북, 피디에이(PDA), 스마트폰(smart phone)등 이동성을 갖는 시스템으로 구축되었다.
- 데이터베이스 : 마이에스큐엘(MySql)
- 개발 툴 : 비주얼 닷넷을 이용한 코바기반의 솔루션
- 이동성지원 환경 : 유선과 무선의 네트워크 환경, 무선허브 또는 무선 공유기
- 서버 : 리눅스 미들웨어 시스템 구축

위와 같은 환경으로 경량 모바일 미들웨어 원시 시스템개발을 위하여 환경을 갖춘다.

## 3. 구성요소

[그림 3]은 본 논문에서 설계한 핵심적인 4가지 경량 모바일미들웨어 기본 요소에 대한 기본 구성을 보여준다.

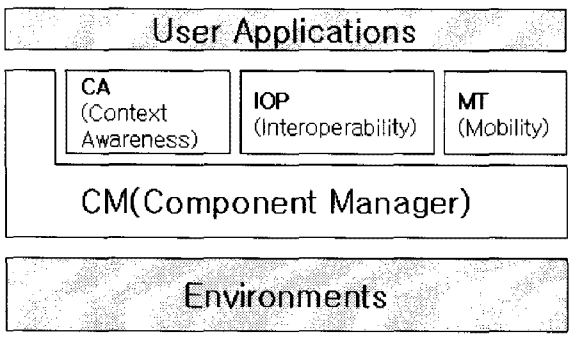


그림 3. 기본구성

### 3.1 컴포넌트 매니저(CM: Component Manager)

전체 컴포넌트(component)들을 관리하는 미들웨어 시스템 플랫폼(platform) 및 컴포넌트 관리자 인터페이스를 제공한다. 이를 위해서 컴포넌트 설계(design), 컴포넌트 추가(add), 컴포넌트 삭제(delete), 컴포넌트 수정(편집), 컴포넌트 현황 및 실행 관리, 컴포넌트 로깅(logging)과 같은 기능들이 요구된다. 이러한 여러 컴포넌트들은 관련성 정도에 따라 그룹단위로 묶여 관리의 효율성을 제공한다.

### 3.2 상황 인지

미들웨어 기반 응용 서비스 자원 및 환경을 감지하여 개발 및 운용 모듈 및 인터페이스(API)를 제공하는 모듈로 하드웨어 자원과 소프트웨어 자원, 네트워크 자원 등 다양한 환경 자원을 인식 하고 이벤트 정보를 응용 인터페이스(API)와 컴포넌트 매니저 사이에서 관리한다. 하드웨어 자원관리는 단말, 서버, 네트워크 자원 등의 위치 관리, 전원 관리, 생존 관리 서비스를 지원하기 위한 위치 관리 맵(map)을 제공하고 자원들 사이의 관계를 트리 구조에 기반 한 자원관리 서비스를 제공한다. 클라이언트 자원 관리는 위치 관리, 파워 관리, 생존 관리를 지원하고, 서버 자원 관리는 중앙처리장치(CPU) 사용 현황, 메모리(memory)사용 현황, 입출력(I/O) 인터페이스 현황을 관리한다. 또한 네트워크 자원 관리는 유선자원과 무선자원이 있는데 유선 자원 관리는 네트워크의 연결성 관리, 전송속도 관리, 네트워크 범주 등에 관한 관리 서비스를 지원한다. 무선 자원 관리는 무선 단말의 무선 네트워크 접속 시 응용프로그램이 무선 환경에 맞도록 경량화 되고 이동성을 지원하여 단절이 없이 서비스되도록 관리해 준다.

소프트웨어 자원관리는 미들웨어 컴포넌트(component) 관리, 응용프로그램 컴포넌트 관리, 외부 미들웨어 컴포넌트 관리가 있는데 외부 미들웨어 컴포넌트는 모바일 미들웨어 응용 서비스 관리를 포함한다. 또한 모바일 웹서비스, 모바일 파일전송(mobile ftp) 서비스, 모바일 텔레매틱스 서비스 등 다양한 응용프로그램을 컴포넌트 관리자가 관리해 준다. 또한 이러한 하드웨어관리와 소프트웨어 관리를 통하여 각종 모바일 서비스인 철도예약서비스, 항공예약서비스, 고객관리예약서비스 등 서비스산업에서 주로 사용하고 관리하는 서비스를 정형화하여 이를 경량화 시켜 서비스를 제공한다.

상황인지 모듈은 하드웨어자원과 소프트웨어자원 그리고 네트워크자원을 고려한 효율적인 서비스를 제공하기 위하여 시나리오를 기초로 한 인터페이스(API)를 제공한다.

### 3.3 상호연동성(Interoperability)

경량 모바일 미들웨어는 지원하는 응용시스템들 사이에 데이터(data) 교환 및 인터페이스(interface) 교환을 제공하는 상호연동성을 지원한다. 상호연동성은 경량 모바일 미들웨어 와 동일한 미들웨어를 사용하는 내

부 상호연동성과 서로 다른 미들웨어 또는 미들웨어를 사용하지 않는 제3의 응용시스템과 연동을 지원하기 위한 외부 상호 연동성으로 구분된다. 외부 상호연동성은 향후 상호 호환성(Inter-compatability) 모듈로 제공된다.

### 3.4 이동성(Mobility)

경량 모바일 미들웨어는 이동성을 지원하는데 시스템 이동성, 소프트웨어 이동성, 네트워크 이동성 등을 지원한다. 시스템 이동성은 이동성이 잦은 단말을 비롯한 이동성이 낮은 서버의 이동성을 지원한다. 소프트웨어 이동성은 응용 소프트웨어 제거 및 추가, 데이터베이스 이동성(데이터 이동성 포함) 등을 지원한다. 네트워크 이동성은 기존 네트워크의 제거, 재구성, 추가 등으로 발생하는 동적 네트워크 구성을 지원한다. 이러한 이동성 지원을 위하여 이동성 인터페이스 제공하고, 단말의 위치를 모니터링 하여 단말의 위치를 관리한다. 또한 이동성을 지원하면서 원활하게 이동성을 지원하기 위해서는 모바일의 위치 백업과 단절관리가 중요한데 백업은 응용 서비스의 다양한 롤백(rollback) 기능을 지원하기 위해 이미 진행된 응용서비스에 대한 기록을 보전하고 단절관리는 서비스 연결 복원 시, 서비스 복구(recovery)를 지원하기 위한 단절 연산을 지원한다.

## 4. 미들웨어 설계의 경량화 및 이동성

### 4.1 시간 및 공간에 대한 원시 경량화 모델

다음 [그림 4]는 시간과 공간에 대한 간단한 경량화 모델을 나타내 준다.

시나리오는 다음과 같다.

- 클라이언트는 클라이언트용 미들웨어가 설치되어 있어 환경자원을 모니터링 하여 서버 측의 미들웨어에 전송한다.
- 미들웨어 서버는 시간과 공간정보를 입력받아 상황합수(context function)를 통하여 상황리스트들(context spec lists)을 출력한다.
- 미들웨어서버는 모바일 환경에 맞게 상황리스트들과 응용프로그램들의 실행정보를 입력받아 경량화 프로세스를 실행시킨다.

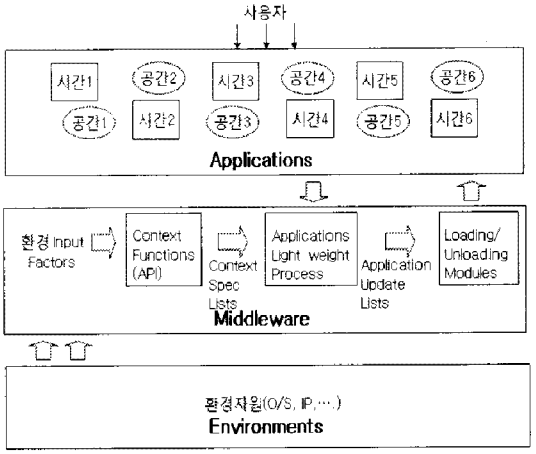


**그림 4. 경량화 모델**

- 미들웨어는 출력된 응용프로그램수정목록들(application update lists)을 이용하여 실행 모듈을 메모리에 로딩/언로딩(Loading/Unloading) 한다.

본 논문에서 중점적으로 다루는 모바일 환경에 맞는 경량화된 미들웨어에 대한 전체적인 설계를 진행하였다.

### 4.2 공간에 대한 이동성 모델

이동성은 위치관리와 백업관리 그리고 단절관리의 중요한 세 가지의 이동성 특징을 지원하는 것이다.

- 위치관리 : 이동단말의 실제 위치를 파악하여 공간을 관리하면서 위치에 맞는 서비스를 제공하고 공간정보와 연동하여 백업관리와 단절관리를 제공한다.
- 백업(Backup) 관리 : 이동단말의 장애(failure)가 발생했을 때 복구를 지원하는 기능을 말하며 데이터의 안전성(Safety)을 위해 필수적인 요소이다.
- 단절(Disconnection) 관리 : 단절은 이동 단말이 고정된 공간에서의 단절과 핸드오프(handoff)시의 단절로 나눌 수 있는데 어떤 단절이 발생한 경우라도 사용자가 인식하지 못하고 작업을 원활하게 수행할 수 있도록 지원하는 기능을 말한다.

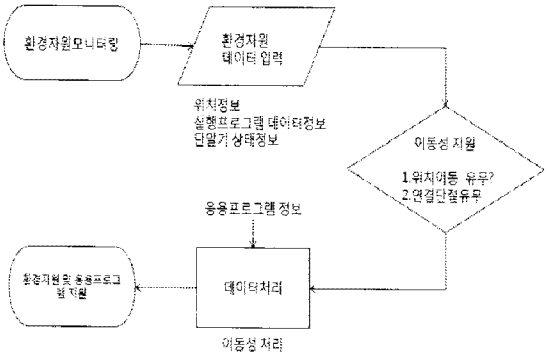


그림 5. 이동성 처리순서

[그림 5]는 이동성의 처리 순서를 나타내는 것으로 경량모바일미들웨어가 환경자원을 모니터링하면서 환경자원을 입력받는다. 이때 모바일 단말의 위치와 상태, 그리고 실행되고 있는 응용프로그램들의 상태를 입력받는다. 경량모바일미들웨어의 이동성을 지원하는 함수는 입력정보를 이용하여 지원에 필요한 목록을 만든다. 위치정보를 이용하여 위치 이동유무를 판단하여 로밍 및 백업위치 지점을 결정 한다. 또한 연결단절유무 상태를 확인하여 백업 및 복구를 지원해 준다. 경량모바일미들웨어는 위치와 연결 상태에 따라서 필요한 서비스를 지원하여 응용프로그램이 최적의 상태에서 이동성 서비스가 원활하게 지원되도록 한다.

## 5. 시스템 설계

경량 모바일 미들웨어 시스템의 처리 흐름도는 [그림 6]과 같다. [그림 6]은 경량 모바일 시스템이 실행되는 전체 흐름도를 나타낸다. 경량모바일 시스템의 전체 흐름을 보면 먼저 사용자 또는 관리자가 환경 및 자원을 등록한다. 또한 컴포넌트 관리자(Component Manager)가 환경 및 자원, 사용자 관리, 응용프로그램 실행등 미들웨어(LightMware)를 관리해 준다. 환경 및 자원등록이 사용자에 의해 등록되기도 하지만 때로는 시스템에서 자동으로 미들웨어에게 등록된다. 그러면 미들웨어의 각 모듈들은 사용자의 환경과 자원을 모니터링 하여 상황인지나 이동성, 상호호환성의 모듈들이 실행이 된다. 모듈들이 실행되면 각각의 모듈들 중에서 기존의 미들웨어 모듈들을 통해 들어온 환경이나 자원들에 대한 상태를 재정립하여 시간이나 공간에 대한 모듈을 실행하

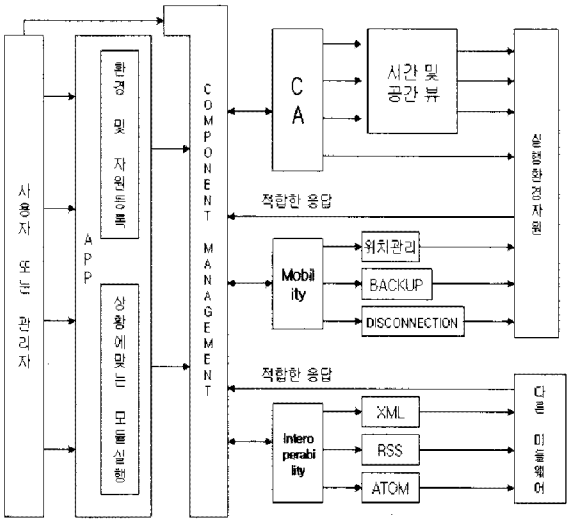


그림 6. 경량 모바일 시스템의 처리 흐름도

여 사용자의 환경에 적합한 응용프로그램이 실행되도록 함으로써 사용자에게 경량화 되고 이동성을 지원 서비스를 제공한다.

경량 모바일 시스템은 클라이언트의 환경과 상황을 정확히 파악하여 모바일 환경에 적합한 최적의 서비스를 위하여 컴포넌트관리자가 모든 모듈을 모니터링 및 관리해 준다. 중요한 모듈은 상황인지모듈과 상호호환성을 관리하는 모듈, 이동성을 지원하는 모듈들로 본 경량 모바일 미들웨어의 핵심 모듈이다.

### 5.1 원시 시스템 메뉴 구성

경량 모바일 시스템의 원시 시스템 구현을 위해서는 다음 [표 2]와 같은 메뉴 구조를 구성하였다.

표 2. 메뉴 구성 현황

| 메뉴 | 내용 |
|---|---|
| 파일 | 미들웨어 프로젝트 관리 데이터를 저장 관리 |
| 자원관리 환경 | 하드웨어, 소프트웨어, 네트워크 자원 관리 |
| Dynamics | 시나리오기반 동적 컴포넌트 관리 서비스를 지원 |
| 상호연동 | 미들웨어 상호연동성을 지원 |
| 이동성지원 | 시스템, 네트워크, 데이터 이동성 등을 지원 |
| 도움말 | LightMware 경량 모바일 미들웨어 시스템 프로젝트 및 구현에 관한 도움말을 제공 |

### 5.2 구현 인터페이스 뷰(view)

경량 모바일 미들웨어는 인터페이스가 구현되어 서

버측 인터페이스와 클라이언트 인터페이스로 구현되는데 서버측 인터페이스를 통하여 컴포넌트 관리자가 실행되는데 이 뷰를 통하여 관리자가 상황에 맞는 메뉴를 관리하고 필요한 권한을 관리하고 컴포넌트들을 관리한다.

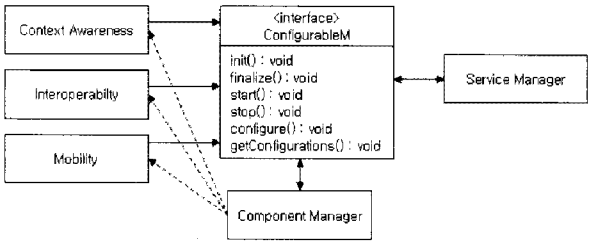


**그림 7. 웹 클라이언트 응용 인터페이스**

경량 모바일 미들웨어의 클라이언트 측의 인터페이스 뷰는 사용자가 등록한 환경 및 자원에 맞는 컴포넌트를 제공해주고 불필요한 컴포넌트는 제거한다. 클라이언트 인터페이스는 윈도우 클라이언트 응용 인터페이스와 웹 클라이언트 응용 인터페이스로 개발되는데 [그림 7]은 웹클라이언트의 응용 인터페이스를 보여준다.

### 5.3 경량 모바일 시스템 프로세스 및 테이블 설계

본 논문에서 제안한 경량 모바일 시스템에 대한 프로세스 및 테이블을 설계하였다. 각 모듈은 자원관리와 컴포넌트관리, 응용서비스관리, 상황인지 처리 프로세스 관리로 구성되어 있다. 자원관리는 응용소프트웨어와 미들웨어서 소프트웨어를 관리하는 프로세스가 있다. 컴포넌트 관리에는 컴포넌트 등록관리가 있고 응용서비스관리와 상황인지 처리프로세스 관리가 있다. [그림 8]은 상황인지 처리프로세스관리를 나타내는 것으로 먼저 클라이언트가 접속되었는지를 확인하고 클라이언트가 접속되었으면 클라이언트의 자원과 클라이언트의 위치해 있는 장소와 시간을 파악하여 상황에 맞는 프로세스를 선택하여 메모리 공간에 할당해 준다. 이때 불필요한 프로세스들은 모두 제거한다. 이러한 일련의 과정을 본 논문은 rule-base로 처리 하도록 설계하였다.

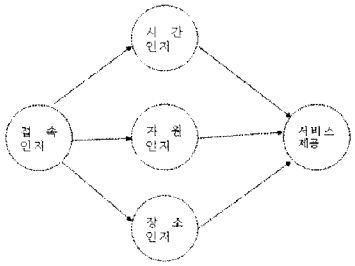


**그림 8. 상황인지 처리프로세스 관리**

이러한 프로세스를 기반으로 하여 본 논문에서는 시스템을 구현하기 위하여 필요한 테이블을 설계하였다. [표 3]은 본 논문에서 설계한 핵심테이블과 테이블의 기능을 보여주고 있다. [표 3]에서는 경량 모바일 미들웨어 원시 시스템을 구현하기 위해 설계한 테이블 목록이다.

**표 3. 설계 테이블 목록**

| 테이블명 | 비고 |
|---|---|
| Context Awareness | 상황인지 |
| 모바일 웹서비스 | 모바일 서비스의 예 |
| 모바일 예약서비스 | 모바일 서비스의 예 |
| 모바일 FTP서비스 | 모바일 서비스의 예 |
| 컴포넌트 추가(Add) | 미들웨어 컴포넌트 관리 |
| 컴포넌트 현황 및 실행 관리(Situation) | 미들웨어 컴포넌트 관리 |
| 컴포넌트 삭제(Delete) | 미들웨어 컴포컨트 관리 |
| 컴포넌트 설계(Design) | 미들웨어 컴포넌트 관리 |
| 컴포넌트 수정(Edit) | 미들웨어 컴포넌트 관리 |
| 컴포넌트 로깅(Logging) | 미들웨어 컴포넌트 관리 |
| 미들웨어 COMPONENT | 미들웨어 컴포넌트 총괄 |
| 응용프로그램 COMPONENT | 응용프로그램 관리 |

위의 테이블 설계 목록 중 한 예를 다음 [그림 9]에서 보여준다. 본 논문은 위의 설계를 기초로 하여 미들웨어 시스템의 경량화를 설계하였으며, 이동성 환경에서도 서비스를 지속해서 공급할 수 있는 미들웨어 시스템을 설계하였다. 설계프로세스와 설계테이블의 예에서 보여주는 것처럼 본 논문은 시간 및 공간 시나리오에 기초한 시스템 경량화와 무선 이동 공간에 대한 이동성에 중점을 두고 경량 모바일 미들웨어 원시 시스템을 설계하였다.

| 테이블 정보 | | | |
|---|---|---|---|
| 테이블명 | Middleware component | | 사용 DB |
| 설계자 | | | 수정일자 |
| 사용용도 | 미들웨어 컴포넌트 관리 테이블 | | 관련ASP |
| 테이블 상세 설계 | | | |
| COLUMNAME | DESCRIPTION | TYPE | 길이 |
| Server_ID | 서버 ID | VARCHAR | 16 |
| Server_IP | 서버 IP | VARCHAR | 16 |
| Client_ID | 클라이언트 ID | VARCHAR | 16 |
| Client_IP | 클라이언트 IP | VARCHAR | 16 |
| Request_Component_ID | 요청 컴포넌트 ID | VARCHAR | 16 |
| Request_Time | 요청 시간 | Time | 8 |
| LogIn_ID | 접속 ID | VARCHAR | 16 |
| LogIn_Time | 로그인 시간 | Time | 8 |
| LogOut_Time | 로그아웃 시간 | Time | 8 |
| Component_Condition | 컴포넌트의 상태 | VARCHAR | 16 |
| Network_Value | 네트워크 상태 | VARCHAR | 16 |
| Transfer_Protocol | 전송 프로토콜 | INT | 16 |

그림 9. 설계테이블 예

## Ⅳ. 결론 및 연구과제

본 논문에서는 경량 모바일 환경에 맞는 미들웨어 시스템을 설계하였다. 설계의 주안점은 상황인지처리에 대한 경량화와 미들웨어 시스템의 이동성이다. 즉 경량 모바일 미들웨어(LightMware)는 모바일 미들웨어 시스템 환경에서 여러 환경정보를 입력받아 시간과 공간에 따라 변화하는 상황을 인식하고 시간과 장소에 적합한 업무내용 및 스케줄 등의 경량 최적 적응서비스(Just In Application Service)를 제공한다. 이동 사용자의 장소가 변경되거나 또는 단절될 때 서비스가 단절되지 않고 지속적인 연결 서비스를 제공하는 이동성을 지원하도록 설계하였다.

향후 우리는 각 구성요소에 대한 세부적인 설계와 구현에 대한 연구를 진행 할 것이다. 유비쿼터스 환경으로 진화하는 컴퓨팅 환경에 적합한 경량화된 모바일 미들웨어에 대한 지속적인 연구가 필요하다.

## 참 고 문 헌


[1] P. A. Bernstein, "Middleware : A Model for Distributed System Services," Communications of the ACM, Vol.39, pp.86-98, 1996.

[2] http://dsonline.computer.org/portal/site/dsonline/menuitem.20d6846e1c7ed783f1a516106bbe36ec/index.jsp?&pName=dso_level1_home&path=dsonline/topics/middleware&file=index.xml&xsl=generic.xsl

[3] 윤여봉, 윤희용, 정윤철 "유비쿼터스 미들웨어 기술", 한국 인터넷 정보학회, 제5권, 제3호, pp.49-57, 2004.

[4] http://computer.org/dsonline/middleware/RMarticle1.htm

[5] 한승욱, 송성근, 윤희용, "커뮤니티 컴퓨팅을 위한 에이전트 기반 지능형 미들웨어", 정보과학회지 제23권, 제9호, pp.50-58, 2005.

[6] 김수중, 윤용익 "임베디스 시스템 미들웨어의 기술 동향", 전자공학회지, 제31권, 제11호, pp.1455-1464, 2004.

[7] Ledoux. "OpenCorba: a Reflective Open Broker," Lecture Notes in Computer Science, Vol.1616, pp.197-214, 1999.

[8] F. Kon, M. Roman, P. Liu.J. Mao, and T. Yamane, "Monitoring, Security, and Dynamic configuration with the dynamicTAO Reflective ORB," IFIP/ACP International Conference on Distributed Systems Platforms and Open Distributed Processing, New York, 2000.

[9] G. S. Blair, G. Coulson, and A, Andersion, "The Design and Implementation of OpenORB v2," IEEE DS Online, Special Issue on Reflective Middleware, Vol.2, No.16, 2001.

[10] 양승일, 이태규, 박성훈, "경량 모바일 미들웨어 시스템 설계", 2009년도 한국콘텐츠학회 춘계논문집, 7권, 1호, pp.659-662, 2009.

[11] A Gaddah and T. Kunz, "A Survery of Middleware Paradigms for Mobile Computing," Technical Report SCE-03-16, Carleton University Systems and Computing Engineering, 2003(7).

## 저 자 소 개


**양 승 일(Seung-Il Yang)** 정회원

- 1996년 2월 : 호서대학교 정보통신공학과 (공학사)
- 2004년 8월 : 중앙대학교 컴퓨터소프트웨어학과 (공학석사)
- 2005년 3월 ~ 현재 : 충북대학교 컴퓨터공학과 박사과정

<관심분야> : 미들웨어, 유비쿼터스, 분산알고리즘, 모바일, 상호배제

**이 태 규(Tae-Gyu Lee)** 정회원

- 1992년 2월 : 군산대학교 전자계산학과 (이학사)
- 1999년 8월 : 숭실대학교 컴퓨터공학(공학석사)
- 2006년 2월 : 고려대학교 컴퓨터학과(이학박사)
- 2007년 3월 ~ 현재 : 한국산업기술대학교 겸임교수

<관심분야> : 분산시스템, 네트워크, 미들웨어, 유비쿼터스, 로봇컴퓨팅

**박 성 훈(Sung-Hoon Park)** 종신회원

- 1982년 2월 : 고려대학교 정경대학 통계학과(경제학사)
- 1993년 2월 : 인디애나대학교 대학원 컴퓨터학과(공학석사)
- 1997년 2월 : 고려대학교 컴퓨터학과(공학박사)
- 2004년 9월 ~ 현재 : 충북대학교 전기전자컴퓨터공학부 부교수

<관심분야> : 분산, 모바일, 유비쿼터스 시스템, 알고리즘