# 유비쿼터스 시스템을 위한 실시간 모니터링 에이전트

## (Real-Time Monitoring Agent for Ubiquitous System)

권 성 현 † 이 병 훈 †
(Sung-Hyun Kwon) (Byoung-Hoon Lee)
김 재 훈 †† 조 위 덕 †††
(Jai-Hoon Kim) (We-Duke Cho)

**요 약** 유비쿼터스 미들웨어는 다양한 서비스들의 집합체로 구성되어 있다. 서비스들은 서로 다른 언어로 작성 될 수 있고, 다양한 서비스들의 추가 및 변경이 동적으로 발생하는 환경에서 서비스 협업이 이루어진다. 이러한 환경에서 서비스하는 유비쿼터스 시스템은 실시간 시스템에 적합한 응답성과 신뢰성이 요구된다. 본 논문에서는 실시간으로 추가되거나 제거되는 서비스들 사이에서 모니터링 시스템이 런-타임 환경에서 지속적인 모니터링을 하고, 어플리케이션과의 결합으로 발생하는 오버헤드를 최대한 줄여서 어플리케이션의 성능 및 신뢰성을 보장할 수 있는 모니터링 에이전트를 제안한다. 제안된 기법은 시나리오 상황에 맞는 데모를 보이고 실험을 통해 서비스 어플리케이션에 대한 모니터링 기능과 성능을 검증하였다.

**키워드** : 유비쿼터스 시스템, 모니터링 시스템, 응답성, 신뢰성

***Abstract*** The ubiquitous middleware configurated an aggregation of the various services. The services are made in different languages and the various services that are dynamically changing environment are carried out in collaboration service. The ubiquitous system that services in the these environment is required appropriate response to real-time system and reliability. In this paper, we suggest the monitoring agent that a monitoring system added or removed the services in real-time is made continuous monitoring in run-time environment and guaranteed performance and reliability of a application by maximally reducing overhead that combined with applications is occurred. The suggested scheme shows the demonstration to fit the scenario situation and verifies function and performance of monitoring about the service applications through the experiment.

**Key words** : Ubiquitous System, Monitoring System, Performance, Reliability

## 1. 서 론

유비쿼터스 시스템은 전 세계적으로 차세대 동력 산업 또는 미래 산업으로 부각되고 있는 추세이다. 그리고 풍부한 정보통신 인프라가 갖추어져 있는 우리나라는 유비쿼터스 환경에 최적의 조건을 갖추고 있다. 또한 우리나라는 유비쿼터스 기술이 사람들에게 편리하고 쾌적한 생활을 할 수 있게 도와주는 미래 환경을 기축하는 것으로 인식하고 국가적인 차원에서 유비쿼터스 환경 개발진행을 하고 있다. 유비쿼터스 시스템은 주거환경을 개선하는 것뿐만 아니라 물류, 재난/환경, 정부 및 비즈니스 시스템등 많은 곳에 응용될 전망이다. 그런데 지금까지 편리한 유비쿼터스 시스템을 구성하기 위한 기술적인 구현 위주로만 연구를 했다. 그래서 유비쿼터스 시스템의 확장된 응용환경과 사용자에게 편리하고 안정적인 서비스를 제공하기 위해서 유비쿼터스적인 모니터링 시스템에 대한 연구가 필요하다. 일반적으로 모니터링을 하기위해서 호스트 플랫폼 자원과 어플리케이션 자원, 네트워크 상황을 모니터링 한다. 엔터프라이즈 모니터링 환경 같은 경우 IT인프라를 단위 자원별로 관리하지 않고 비즈니스 관점에서 통합관리 한다. 그리고 어플리케이션 성능을 모니터링하기 위해 어플리케이션 운영환경, 개발 단계 및 평가에 걸친 어플리케이션 라이플 사이클에 관한 모든 영역을 감시한다. 그러나 기존의 모니터링 기법은 다음과 같이 유비쿼터스 모니터링 요소를 지원하기에는 문제가 있다.

- **협업 어플리케이션의 응답성과 신뢰성 보장:** 개별 어플리케이션의 모음인 유비쿼터스 서비스에서 어플리케이션 개별성능 분석으로는 협업하는 어플리케이션 전체의 성능을 알 수 없다. 그래서 각 어플리케이션의 데이타 및 단위 실행 상황 분석과 자원상황 등을 모

· 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 지식경제부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 08B3-S2-10M 과제로 지원된 것임
· 이 논문은 2008 한국컴퓨터종합학술대회에서 '유비쿼터스 시스템을 위한 실시간 모니터링 에이전트'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 아주대학교 정보통신전문대학원
liebeym@gmail.com
componer@ajou.ac.kr
†† 정 회 원 : 아주대학교 정보통신전문대학원 교수
jaikim@ajou.ac.kr
††† 종신회원 : 아주대학교 전자공학과 교수
chowd@ajou.ac.kr

논문접수 : 2008년 9월 3일
심사완료 : 2008년 10월 20일

니터링해서 종합적인 판단으로 응답성과 신뢰성을 보장해야 한다.

- **실시간 모니터링:** 유비쿼터스 시스템 환경은 사용자가 필요한 서비스를 신속히 제공하기위해 필요한 어플리케이션이 동적으로 협업해야한다. 그래서 유비쿼터스 모니터링 시스템은 협업된 서비스가 사용자에게 만족하는 서비스를 하고 있는지 실시간으로 판단해야 한다. 또한 새롭게 추가된 어플리케이션을 모니터링하기 위해 모니터링 시스템은 재시작 없이 지속적인 모니터링을 지원해야 하고 서비스 상태를 인식할 수 있는 요구시간 안에 모니터링 데이타를 전달해야한다.
- **유연한 모니터링 시스템:** 일반적으로 어플리케이션의 실행 상태를 모니터링 하기위해 모니터링 프레임워크에 맞추어 어플리케이션을 개발한다. 이것은 어플리케이션과 모니터링 시스템이 밀접하게 연결되면서 모니터링의 오류로 인해 어플리케이션 실행에 문제가 발생할 수 있고 신뢰성에 문제가 될 수 있다. 따라서 서비스 어플리케이션과 모니터링 시스템과의 관계를 최소화해서 어플리케이션 개발이 자유롭게하고 모니터링으로 인한 성능저하를 발생 시키지 않아야 한다.

본 논문에서 제안하는 모니터링 에이전트는 위의 3가지를 지원하기 위해서 어플리케이션의 동작 상태를 메서드 단위로 성능 분석하고 변수 할당 상태를 실시간으로 모니터링 하고 모니터링 시스템과 어플리케이션이 비동기적으로 동작한다. 또한 어플리케이션과 결합시에는 간단한 모니터링 API를 제공해 결합을 최소화 했다.

## 2. 관련 연구

### 2.1 분산 응용환경에서의 계층적 모니터링 [1]

유비쿼터스 환경에서는 이기종의 분산 환경에서 다양한 종류의 자원이 유동적으로 변화하는 것을 모니터링해야 한다. 분산 환경에서 다양한 자원과 이질적인 통신 프로토콜 상황을 갖게 되며 이를 지원하기 위한 계층적 모니터링은 다음과 같은 특징을 가진다.

- 컴포넌트 기반으로 시스템 구성을 용이하게 한다.
- 통신 프로토콜등을 다양하게 지원해서 모니터링 하려는 값은 SQL구문을 사용해 자원 정보를 보여준다.

### 2.2 HP Openview [2]

HP Openview는 다음과 같은 특징을 가지고 있다.

- IT 인프라를 단위 자원별로 관리하지 않고 비즈니스 관점에서 통합 관리하는 시스템이다.
- 관리 시스템은 운영 중인 전산환경에 적합하게 구축되며 다른 관리 시스템과 연동 또한 쉽게 지원한다.

### 2.3 CoSMos [3]

어플리케이션 성능을 다음과 같은 방식으로 개선한다.

- 어플리케이션의 코드분석 데이타, 호스트 및 네트워크 자원, OS성능 분석 데이타를 모아서 SQL서버에 저장
- 자동 또는 수동적인 측정을 통해 기능들을 컨트롤한다.
- 분석된 데이타를 가지고 어플리케이션의 소스 레벨 성능 개선에 사용한다.

기존의 모니터링은 호스트의 자원 정보를 모니터링하거나 비즈니스 통합적인 차원의 모니터링으로 단위 자원별 모니터링에 취약하다. 또한 모니터링 시스템과 소스 결합을 통해 어플리케이션 모니터링을 함으로써 모니터링과 어플리케이션과의 결합도가 높고 저장된 모니터링 데이타를 분석해서 상황 정보를 획득함으로써 실시간 실행상황 정보를 얻기 힘들다. 그러나 유비쿼터스 환경의 지능공간을 모니터링 하기 위해서 자원 상태뿐만 아니라 지능공간에서 사용자에게 서비스를 해주는 어플리케이션 협업 상황을 실시간으로 모니터링 해줘야 한다. 그래서 모니터링 시스템과 어플리케이션과의 결합부분을 최소화해서 서비스간의 상호 데이타 전달상황, 서비스 플랫폼 및 서비스 어플리케이션의 자원상황등 서비스들의 용·복합적인 상황을 종합적으로 판단 할 수 있는 요소를 효과적으로 수집하는 것이 필요하다.

## 3. 본 론

### 3.1 배경 및 모니터링 시스템 구조도

유비쿼터스 웰빙 시스템, 공공 안전 시스템 등은 사용자가 의식 하지 않고 자동으로 사용자의 상태를 감지하고 적시 적소에 필요한 서비스를 수행한다. 유비쿼터스 시스템을 위한 서비스 어플리케이션은 다양한 종류로 사용자에게 하나의 서비스 또는 복수개의 서비스를 제공한다. 서비스들은 지속적이고 정확하게 제공되야한다. 이를 위해 실시간 모니터링 시스템은 사용자 서비스의 정확성을 평가하고 서비스의 신뢰성을 검사하고 유비쿼터스 시스템에 피드백을 준다.

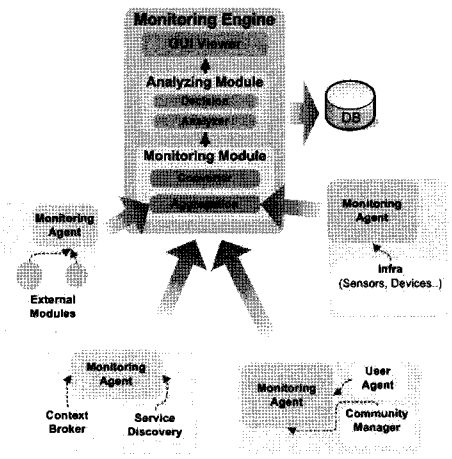


그림 1 모니터링 시스템 구조도

실행 중 새로운 서비스가 추가 및 제거 시에 시스템 중단 없이 지속적으로 서비스하기 위해 모니터링 시스템은 어플리케이션 독립적으로 실행한다. 또한 분석, 판단을 하는 모니터링 엔진을 서버로 만들고 어플리케이션의 정보 및 호스트의 정보를 모으는 모니터링 에이전트를 클라이언트로 만들어 비동기적인 서버, 클라이언트 시스템으로 구성함으로써 지속적인 모니터링을 가능하게 한다. 그림 1은 모니터링 시스템 구조를 보여준다. 모니터링 엔진은 각 컴퓨터에서 수행되고 모니터링 에이전트로부터 서비스 모듈의 상황 정보를 받아서 원격에 있는 모니터링 서버에게 모니터링된 데이타를 보내준다. 모니터링 모듈이 각 전달 받은 데이타를 필터링하고 분석 모듈은 각 수집된 데이타를 가지고 서비스 모듈의 상황을 분석하고 판단한다. 분석한 결과를 사용자가 보기 쉽게 GUI로 화면에 나타낸다.

### 3.2 제안하는 모니터링 에이전트

3.2.1 구조도

어플리케이션과의 연관성을 최소화하기 위해서 어플리케이션과 코드 결합이 필요한 변수 모니터링, 어플리케이션 자원 모니터링은 간단한 API를 제공함으로써 코드 변경을 최소화 했다. 그리고 호스트 자원 분석 모니터링, 어플리케이션 메서드 호출 모니터링은 어플리케이션 외부에서 모니터링 할 수 있도록 해서 어플리케이션과 결합을 하지 않도록 했다. 모니터링 에이전트는 JAVA로 구현이 되었다. 그리고 JMX(Java Management eXtensions)[4]를 사용해 IPC(Inter Process Communication)기능과 데이타 저장을 하고 서버와 연결한다. 그리고 JDI(Java Debug Interface)[5]를 사용해서 메서드 분석을 하고 JNI(Java Native Interface)[6]를 사용해 호스트 자원 정보를 획득한다.

- **Module Monitor**는 Module Field Observer와 Module Method Observer기능을 갖는다. Module Field Observer는 어플리케이션이 실행될 때 변수의 상태변화를 감시고 어플리케이션의 상태를 알리는 메시지를 생성을 한다. 그리고 필요에 따라 긴급 메시지를 생성하기도 한다. Module Method Observer는 어플리케이션이 동작할 때 각 메서드의 이름과 호출된 시간, 호출이 끝난 시간을 통해 어플리케이션의 단위 실행시간과 전체 실행시간을 감시한다. 그리고 각 메서드의 오류를 감시한다.
- **Resource Monitor**는 Module Resource Observer와 System Resource Observer의 기능을 한다. Module Resource Observer는 어플리케이션이 동작할 때 어플리케이션의 프로세스ID를 획득해서 호스트 내에서 어플리케이션이 사용하는 CPU, 메모리 자원 사용 상황을 실시간으로 획득한다. System Resource Observer는 호스트 전체의 CPU, 메모리 자원 상황을 실시간 획득한다.
- **Information Processor**는 모니터링된 데이타를 모니터링 에이전트의 정보와 함께 프로토콜에서 정의한 메시지로 만들며 메시지는 각 메시지 버퍼에 저장이 된다.

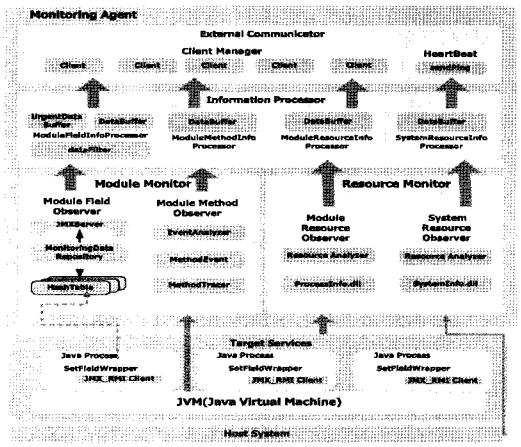


그림 2 모니터링 에이전트 구조도

- **External Communicator**는 각 버퍼에 하나의 클라이언트를 할당해 동시에 병렬로 메시지를 보낼 수 있고 긴급 메시지 전달 클라이언트를 사용해서 모니터링 엔진은 즉시 판단이 필요한 어플리케이션의 상태를 인지함으로써 요구 시간 내에 실시간으로 어플리케이션 상황을 모니터링할 수 있다. 모니터링 엔진 재시작시 모니터링 에이전트는 재시작 없이 즉시 접속한다. 그리고 모니터링 엔진에 하트비트 핑을 보냄으로써 모니터링 에이전트의 접속 상태를 검사한다.

다음은 모니터링 에이전트 핵심기능에 대해 설명한다.

- **Module Field Observer**

변수 모니터링은 그림 3과 같이 실행되면 SetField Wrapper 클래스는 정적 생성되면서 내부적으로 모니터링 에이전트에 자동으로 접속 한다. 어플리케이션에서 모니터링이 필요한 변수를 SetField Wrapper.setField()에 할당하면 이 데이타는 Module Field Observer에 전송이 되고 전송된 데이타는 해쉬 테이블에 필드명은 Key값으로 변수 값은 데이타로 저장이 된다. 신규 및 변경된 데이타는 원격 모니터링 엔진에 전송이 된다.

```
public class AppTest {
    public static void main(String[] arg) throws Exception {
        SetProcessInforWrapper.setProcessInfo(null);
        String str="init";   /*생성시 PID획득후 모니터링 에이전트로 전송*/
        while(true) {
        str=changeString();
        SetFieldWrapper.setField("TestMA.AppTest.str",str);
        } /*생성시 모니터링 에이전트로 입력된 데이터 전송*/
        SetFieldWrapper.close();
    }
}
```

그림 3 어플리케이션 모니터링을 위한 코드 추가

표 1은 setField메서드에 관한 설명이다. fieldName은 변수의 데이타를 기술하기위해서 사용한다. "어플리케이션이 속한 모니터링 에이전트이름, 어플리케이션 이름, 변수 명" 형식으로 기술한다. 변수의 상태를 긴급하게 전송할 필요가 있을 경우 모니터링 에이전트의 긴급 메시지 리스트에 등록한다. 그래서 변수 모니터링 메시지와 함께 적채 되지 않고 바로 모니터링 엔진에 보냄으로써 긴급 상태를 실시간으로 판단한다.

표 1 setField 매소드

| public void setField(String fieldName, Object fieldData) | |
|---|---|
| fieldName | 모니터링할 필드 데이타의 기술 |
| fieldData | 모니터링 데이타 |

**• Module Resource Observer**

어플리케이션 자원을 모니터링 하기 위해 C library함수를 사용해서 자바 어플리케이션의 자원정보를 획득한다. JNI를 사용하면 C Native함수를 Java에서 호출할 수 있다. 그림 4에서 SetProcessInfoWrapper 클래스가 정적으로 생성되면서 모니터링 에이전트에 접속한다. 그리고 setProcessInfo메서드가 호출될 때 JNI로 만들어진 getpid.dll 파일이 호출되면서 어플리케이션의 PID를 획득한다. 이정보는 모니터링 에이전트로 전송이 된다. 획득된 PID로 어플리케이션 자원 정보를 수집하는 ProcessInfo.dll을 호출해서 주기적으로 모니터링 한다.

**• Module Method Observer**

자바의 메서드를 모니터링을 하기 위해서 동작중인 어플리케이션의 JVM(Java Virtual Machine)에서 정보를 획득해야한다. 그림 4는 어플리케이션이 실행되면 JVM이 모니터링 에이전트의 Module Method Observer에 접속한다. 어플리케이션이 동작할 때 타깃 어플리케이션 JVM의 debug interface를 통해 모니터링 에이전트는 메서드 단위로 메서드 이름, 호출 시간, 호출이 끝난 시간, 에러 정보, 어플리케이션 전체 실행 시간 등 필요한 정보를 얻을 수 있다.

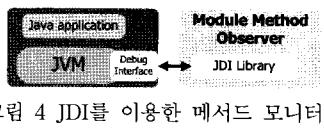


그림 4 JDI를 이용한 메서드 모니터링

## 4. 데 모

### 4.1 데모 환경

데모환경은 4대의 컴퓨터에서 행해졌다. 그림 5는 웰빙 서비스를 위한 데모환경으로 사용자의 환경을 쾌적하게 하기 위해서 집안에 부착된 각 센서들이 정보를

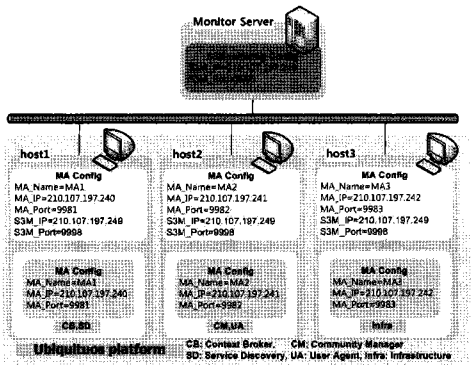


그림 5 데모환경 도식

얻어서 현재 적용 가능한 룰 정보에 찾는다. 그리고 알맞은 룰이 수행됨으로써 등록된 커뮤니티 서비스 중 사용자에게 필요한 서비스를 찾아서 수행한다.

### 4.2 모니터링 데모 결과

그림 6은 각 서비스 어플리케이션들이 구동이 되고 모니터링 에이전트에 의해 모니터링된 데이타를 모니터링 엔진에 전송했을 때 전체 어플리케이션들의 모니터링 상태를 간략히 보여준다. 그리고 모니터 그림의 On/Off로 모니터링 접속 유무를 판별해서 각 어플리케이션들의 실행 상태를 검사한다. 그리고 각 변수의 상태를 색깔별로 표시해서 각 서비스 어플리케이션의 현재 동작 상태를 판별한다.

그림 7은 모니터그림 클릭시에 세부 변수들의 동작 상태를 출력하는 화면으로 각 어플리케이션의 실시간

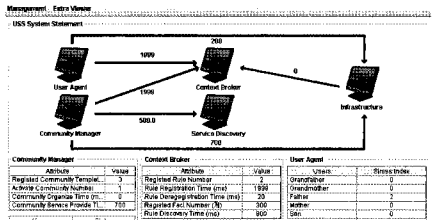


그림 6 Main Monitoring 화면

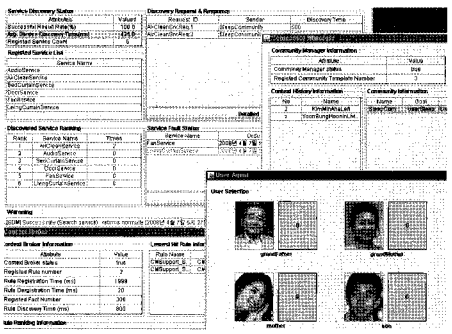

그림 7 어플리케이션의 변수 모니터링

데이타 변경상태를 파악 할 수 있다. 그리고 데이타 값이 성능 요구조건에 따른 색상 구분으로 어플리케이션의 데이타 성능을 평가한다. 그림 8은 각 어플리케이션과 호스트의 자원정보 그리고 각 메서드 분기를 분석한 결과를 보여준다. 그림 9는 긴급메시지전송메커니즘과 일반 메시지와 긴급메시지를 같은 레벨로 전송하는 통합메시지전송메커니즘의 최대 처리시간(Sec.)을 비교한다. 시뮬레이션 조건은 표 2와 같다.

표 2 시뮬레이션 조건

| 구분 | 내용 |
|---|---|
| 메시지 개수 | 어플리케이션 당 최대 100개/Sec |
| 어플리케이션 수/데드라인 | 5개/2초 |
| 긴급메시지 발생빈도 | 총 메시지에서 10%(정규분포 생성) |

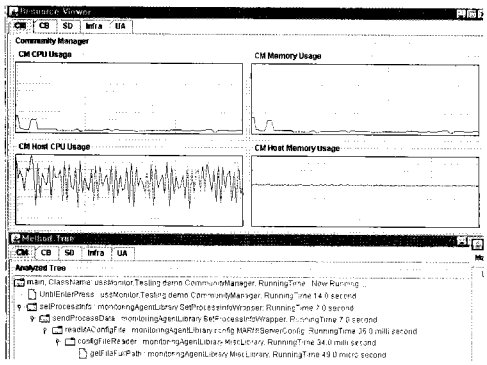


그림 8 Method 모니터링과 리소스 모니터링

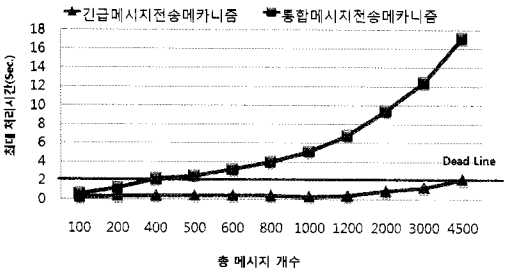


그림 9 메시지 발생량별 처리시간 비교

모니터링 메시지가가 도착했을 때 통합메시지전송메커니즘의 경우는 순차적으로 메시지를 읽으면서 처리한다. 그래서 모니터링할 어플리케이션의 개수가 증가되어 메시지가 약 450개 이상이 한 번에 모니터링 엔진에 들어 올 때 긴급 메시지에 대해 실시간 모니터링을 할 수 없다. 그러나 긴급메시지전송메커니즘을 사용할 경우 긴급메시지가 모니터링 되었을 때 긴급전송 데이타로 분류되어 전용 전송 클라이언트에 메시지를 전송하게 한다. 그래서 긴급메시지 발생 비율이 450개/Sec. 이상 전송되지 않는 한도 내에 긴급 메시지 모니터링 데이타는 데드라인을 만족한다.

## 5. 결 론

유비쿼터스 커뮤니티 컴퓨팅 서비스[7], 상황인지서비스[8]등 최근 유비쿼터스 시스템을 구현하기 위해 많은 연구들이 진행되고 있다. 그러나 이들 서비스의 신뢰성, 응답성을 보장하는 모니터링 관련 연구 분야는 아직 부족하다. 그리고 기존 모니터링 방법론으로는 유비쿼터스적인 모니터링을 지원하기에는 역부족이다. 이러한 문제점을 해결하기위해서 본 논문에서는 분산 환경에서 동작하는 유비쿼터스 모니터링 시스템을 지원하는 모니터링 에이전트를 제안했다.

분산 환경에서 실시간 동작하는 서비스 어플리케이션들이 안정적이고 응답성 있게 동작하도록 모니터링 에이전트는 최소한의 모니터링 코드로 어플리케이션을 효과적으로 모니터링 할 수 있게 했다. 그리고 모니터링 엔진과 독립된 동작으로 모니터링 할 서비스 어플리케이션의 재시작 또는 추가, 제거에 상관없이 지속적인 모니터링을 할 수 있도록 했다. 향후 네트워크 대역폭 측정, Module Method Observer의 분석 성능 향상 및 C, C++,VB등으로 개발된 어플리케이션을 지원하는 모니터링 에이전트를 연구할 것이다.

## 참 고 문 헌

[1] 권성주, 최재영, 이지수, "계층형 구조를 기반으로 한 모니터링 시스템," 정보과학회논문지: 시스템 및 이론, 제33권, 제7호, pp. 440-447, 2006.
[2] HP Openview, http://www.openview.hp.com.
[3] C. Steigner and J.Wilke, "Multi-Source Performance Analysis of Distributed Software," the Communication Networks and Distributed Systems Modeling and Simulation Conference 2002, January 2002.
[4] JMX 1.4 specification, http://java.sun.com/javase/6/docs /technotes/guides/jmx/index.html.
[5] JDI Specification, http://java.sun.com/javase/6/docs /jdk/api/ jpda/jdi/index.html.
[6] JNI 6.0 Specification, http://java.sun.com/javase/6/docs /technotes/guides/jni/index.html.
[7] Y.N. Lee, J.T. Lee and M.K. Kim, "Multi-agent based Community Computing System Development with the Model Driven Architecture," Autonomous Agents and Multi-Agent Systems, May 2006.
[8] D.Zheng, et al., "Deployment of Context-Aware Component-Based Application Based on Middleware," Ubiquitous Intelligence and Computing 2007, pp.908-918, 2007.