유비쿼터스 환경을 위한 상황인지 미들웨어의 모니터링 시스템 설계와 구현

이정아[O1], 임성화[1], 김재훈[1], 조위덕[2]
[1]아주대학교 정보통신전문대학원
{lja0519[O], holyfire, jaikim}@ajou.ac.kr
[2]아주대학교 유비쿼터스시스템 연구센터
chowd@ajou.ac.kr

Design and Implementation of Monitoring System on Context-aware Middleware In Ubiquitous Environment

Jeong-Ah Lee[O1], Sung-Hwa Lim[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 컴퓨팅은 환경 및 사용자의 상황을 필요로 하는 곳에 센서 노드들을 부착해 환경 정보를 자율적으로 수집하고, 수집된 정보를 관리 및 제어하여 사용자에게 적합한 서비스를 제공하는 기술이다. 상황인지 미들웨어는 무선 센서들로부터 습득한 정보를 기반으로 사용자게 어떤 서비스를 제공할지 결정한다. 이를 통해 사용자는 적절하게 필요로 하는 서비스를 받을 수 있다. 따라서 현재 유비쿼터스 환경을 제공하기 위해서는 유비쿼터스 컴퓨팅 미들웨어에 대한 연구가 활발히 진행 중이다. 우리는 실제 환경을 신속히 판단할 수 있도록 다양한 기기로부터 들어오는 환경 정보들을 수집하여 컴퓨팅 시스템에게 적합한 컨텍스트로 가공하여 관리 및 전달하는 방법에 대해 연구하였다. 본 논문에서는 다양한 환경 정보를 수집하는 센서, 무선 및 유선 디바이스로부터 전달받은 정보를 컴퓨팅 시스템이 사용하기 적절하도록 수집, 관리, 여과, 통합하는 모니터링 시스템에 대한 설계 및 구현에 대해 기술하였다. 이는 다양한 양식의 정보를 컴퓨팅 시스템이 인식하기 최적의 형태로 변경하여 상황판단 컴퓨팅 시스템이 신속하고 정확하게 상황판단을 할 수 있도록 설계되어있다. 마지막으로 홈 및 지하주차장 시나리오에 기반하여 구현한 상황인지 미들웨어의 환경 모니터링 시스템을 검증하였다.

1. 서론

현재 사람들의 삶과 안전 및 건강을 위한 유비쿼터스 프로젝트들이 활발히 진행되고 있다. 유비쿼터스 환경에서 사용자는 어디에 있어도 아무런 제약 없이 네트워크의 존재를 의식하지 않고 원하는 서비스를 제공 받을 수 있다. 이를 위해 유비쿼터스 컴퓨팅 분야에서는 환경 및 사용자의 상황 등을 필요로 하는 곳에 센서 노드들을 부착해 빛, 소리, 압력, 온도 등과 같은 환경 정보를 자율적으로 수집하고 수집된 정보를 관리 및 제어하는 시스템이 활발히 개발되고 있다.

그 중에서 사용자가 처한 상황을 인지하기 위해서는 사용자와 주변 환경에 대한 정확한 정보 수집이 선행 되어야 한다. 따라서 유비쿼터스 미들웨어는 필요로 하는 정보를 수집하기 위해 센서, 무선 기기, PC 등 다 기종 서비스를 연동하는 기능이 중요한 이슈이다. 또한 수집된 실제 환경의 컨텍스트를 컴퓨팅 환경에 맞게 변환 및 적용해야 한다. 이를 위해 수집된 정보를 가공하여 상황인지 및 판단 미들웨어가 가용할 수 있는 상태의 컨텍스트로 변환하는 기능이 요구된다.

우리는 유비쿼터스 프로젝트를 바탕으로 실시간으로 전송되는 환경 및 사용자 정보를 이용해 컴퓨팅 시스템 인 추론엔진이 현재 상황을 정확하게 인지 및 제공해야 하는 서비스를 판단할 수 있도록 기반 모니터링 시스템을 설계 및 구현하였다. 즉, 유비쿼터스 미들웨어의 요소 중 하나인 모니터링 시스템에서는 센서 및 유무선 디바이스로부터 다양한 형태의 환경 정보를 제공 받아 컴퓨팅 시스템이 사용할 수 있는 컨텍스트로 가공하여 상황정보를 구성한다. 이로써 사용자의 행동패턴이나 주변환경의 상황을 예측하기 위한 기본정보를 제공하여 상황 판단 시스템이 현재 사용자 및 환경을 추론을 통해 신속히 판단할 수 있도록 한다. 모니터링 시스템은 다양한 종류의 센서 및 디바이스가 쉽게 등록할 수 있도록 아답터 기능을 제공하고, 다양한 포맷으로 들어오는 정보를 일정한 포맷으로 변형시키는 컨버터 기능 및 수집한 정보를 컴퓨팅 시스템이 원하는 정보로 가공하는 필터링 및 어그리게이션 기능, 추론엔진에게 환경 정보들을 전달하고 데이터베이스에 저장하는 기능으로 구성되어 있다. 우리는 이러한 기능을 가진 모니터링 시스템을 컨텍스트 위젯이라 부르고, 설계 및 구현 하였다.

2. 관련 연구

통신의 발전과 함께 유비쿼터스 환경을 구축하기 위한 유비쿼터스 컴퓨팅 구축에 대한 연구가 활발하게 진행되어오고 있다. 특히 실제 환경을 컴퓨팅 시스템이

* 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스 컴퓨팅및네트워크원천기반기술개발사업의 지원에 의한 것임

정확하게 판단할 수 있도록 다양한 정보를 획득 및 통합하기 위한 미들웨어에 대한 연구가 증가하고 있다.

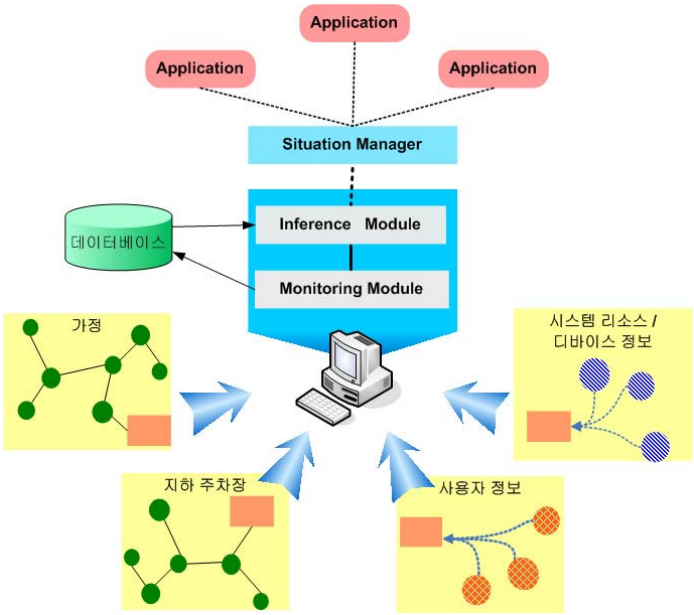
Gaia OS는 다양한 디바이스에 비해 이러한 디바이스들을 연결해주는 메타운영 시스템이다[1]. 이 시스템은 분산 미들웨어 구조와 비슷하게 구성 및 디자인되어 있어서 다양한 프로그램 및 디바이스를 이식 및 실행시킬 수 있다. 따라서 사용자는 Gaia를 통해 다양한 디바이스 및 서비스를 동시에 받을 수 있다. 이 외에도 Gaia의 컴포넌트 관리, I/O 수행, 파일시스템 기능, 통신, 에러감지, 리소스 관리 등의 기능도 제공한다. [2]는 다양한 범위의 환경 및 다수의 사용자 및 이 기종의 디바이스 정보를 동시에 처리하기 위한 소프트웨어 인프라구조를 제시한다. 유비쿼터스 환경을 구축하는 디바이스들은 구성이 자주 바뀌어 디바이스들 간의 연계가 중요하다고 강조한다. 따라서 디바이스들이 서로 협동 및 상호작용을 하는 등 기기들이 서로 유연하게 연계할 수 있도록 유비쿼터스 컴퓨팅 환경의 소프트웨어 인프라를 연구 및 구축하였다. RCSM[3]은 다 기종 어플리케이션들이 투명하게 상호작용을 할 수 있도록 오브젝트 기반 컨텍스트 프로세싱 시스템을 제공하는 미들웨어이다. 이는 오브젝트 적응 컨테이너와 오브젝트에 따른 브로커를 두어 다양한 오브젝트들을 쉽게 확장 연결시킬 수 있게 하였고 다양한 컨텍스트에 대한 인터페이스들을 제공하였다. [4]는 상황인지 시스템을 위한 온톨로지를 제공한다. 특히 웹 언어들을 하나로 통합하여 정보를 공유하고 이들의 사실적인 정보들을 의미로 바꾸어 제공해주는 서비스를 구현하였다. [5]는 유비쿼터스 상황인지 시스템을 컨텍스트 온톨로지 및 추론엔진, 컨텍스트 레벨조절 모듈, 컨텍스트 관리모듈 등 3가지 구성요소로 나누었다. 컨텍스트 레벨조절 모듈에서 유저의 요구수준에 따라 컨텍스트 정보 가공 수준을 조절한 뒤 컨텍스트 관리모듈에서 조절한 수준에 따라 환경 정보들을 모니터하고 가공 및 관리한다. 컨텍스트 관리모듈에 의해 생성된 컨텍스트는 추론엔진에게 전송되고 인공지능 알고리즘을 통해 의미 있는 정보로 재생산된다. 이러한 상황인지 시스템은 컴퓨팅 시스템이 구분 할 수 없는 환경정보를 명확한 뜻을 담는 컨텍스트로 만든다.

우리는 유비쿼터스 프로젝트의 일환으로 상황인지 미들웨어 시스템 중 모니터링 시스템을 개발하였다. 이 논문에서는 다양한 기기로부터 들어오는 환경 정보를 수집하여 컨텍스트 처리 레벨 조절 기능을 가진 컴퓨팅 시스템에 적합하게 가공하는 모니터링 시스템에 대해 설명한다.

3. 상황인지 미들웨어

컨텍스트 위젯이 포함되어 있는 상황인지 미들웨어는 사용자와 주변 정보를 모니터링 하여 사용자의 상황을 추론하고 그에 알맞은 서비스를 찾아 제공한다. 상황은 매우 다양하게 변하고 사용자가 원하는 유비쿼터스 환경이 다르므로 사용자 중심의 유비쿼터스 서비스를 제공하기 위해 사용자의 상황과 환경을 동시에 고려해 상황판단을 정확히 하기 위해 상황인지 미들웨어에서는 지능형 예측/추론 기법이 도입되었고 이를 추론엔진이라고 부른다. 상황인지 미들웨어는 크게 모니터링 시스템, 추론엔진, 상황 매니저로 구성되어 있고 개괄적인 구조는 <그림 1>과 같다.

컨텍스트 위젯은 환경을 모니터링 하는 미들웨어로 유비쿼터스 환경에 산재되어있는 다양한 종류의 센서를 등록하여 센싱된 환경 정보를 수집한다. 이렇게 수집된 다양한 형태의 정보를 추론엔진 및 상황 매니저가 사용할 수 있는 형태로 가공한다. 또한 사용자와 컴퓨터 간의 상호작용이 극도로 배제되는 유비쿼터스 환경에서 최적의 서비스를 제공하기 위해서는 사용자의 행동이나 액션에 대한 히스토리를 저장하여 차후 발생할 수 있는 해당 양식에 대해 미리 예측할 수 있어야 한다. 이러한 예측을 위해 컨텍스트 위젯에서 필터링 및 어그리게이션 단계를 거친 컨텍스트는 데이터베이스에 저장된다. 이로써 사용자의 행동 패턴이나 주변 환경의 상황을 예측하기 위한 기본 정보를 제공할 수 있다.



<그림 1> 상황인지 미들웨어 구조

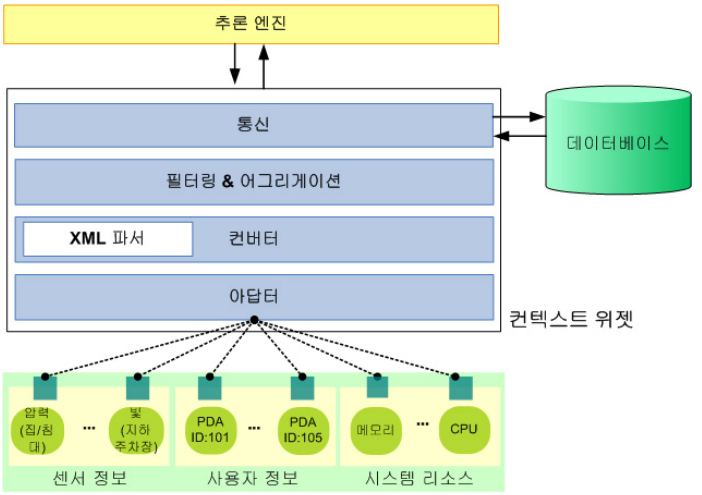
추론엔진에서는 모니터링 된 환경 컨텍스트를 이용하여 상황을 판단하고 사용자나 주변환경의 패턴을 추론한다. 더욱 정확한 추론을 위해 사용자 정보, 디바이스 및 센싱정보 등을 다양하게 활용하며 그에 맞는 추론 알고리즘을 적용하고 있다. 또한 각 사용자에게 필요한 상황을 추론하기 위해 동적 룰을 생성하여 환경 적응적인 추론을 하도록 하였다.

추론된 상황은 상황 매니저에게 전달되어 상황에 따라 적절한 제어를 수행한다. 인지 가능한 상황이 발생되면 그에 따른 제어 신호를 발생시켜 이를 이용하여 제어를 담당하는 시스템이 실제 장비를 제어하거나 어플리케이션에서 이를 이용하여 상황 적응형 서비스를 제공한다.

4. 컨텍스트 위젯의 설계

컨텍스트 위젯은 크게 4가지 모듈로 나뉜다. 다양한

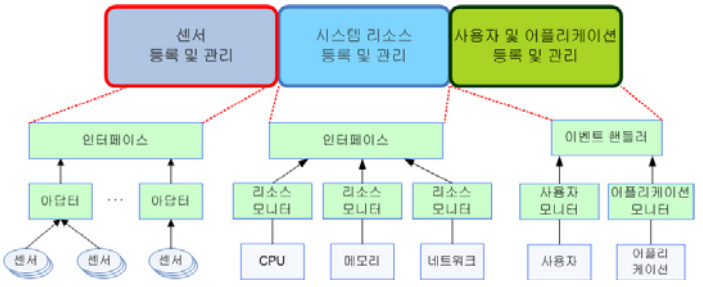
센서로부터 오는 수많은 환경 정보와 무선기기로부터 전송되는 사용자 정보, 오디오 및 PC 등 유선 디바이스 정보를 아답터 모듈을 통해 수집된다. 다양한 포맷의 정보는 빠른 정보해석 및 처리를 위해 일정한 양식으로 변환될 필요가 있다. 컨버터 모듈에서는 여러 종류의 포맷을 컨텍스트 위젯이 필터링 및 어그리게이션 하기 좋은 포맷으로 변경한다. 이러한 컨텍스트에는 추론에 전혀 사용되지 않거나 현재 사용되지 않지만 나중에 사용할 필요가 있거나 잘못된 정보 등 다양한 정보가 들어오기 때문에 추론엔진에게 불필요한 정보는 걸러낼 필요가 있다. 또한 환경 및 사용자의 정보가 다양한 독립적인 포맷으로 들어오기 때문에 유비쿼터스 컴퓨팅에 필요한 포맷으로 가공해야 할 필요가 있기 때문에 필터링 및 어그리게이션 모듈에서 처리하도록 하였다. 이렇게 처리된 정보는 통신 모듈을 통해 추론엔진에게 전송되고 차후 데이터의 사용을 위해 데이터베이스에 저장한다. 따라서 이러한 고려사항을 중심으로 컨텍스트 위젯을 <그림 2>와 같이 설계하였다.



<그림 2> 컨텍스트 위젯 구조

4.1 아답터

아답터는 센서 및 시스템 리소스 등을 등록하고 관리한다. 아답터는 다양한 센서로부터 오는 수많은 환경 정보와 무선기기로부터 전송되는 사용자 정보, 오디오 및 PC 등 유선으로 전송되는 디바이스 정보를 수집한다. 유비쿼터스 컴퓨팅 환경을 구성하는 다양한 기종의 디바이스 및 센서들은 상황에 따라 추가 및 삭제 될 수 있다. 따라서 다양한 이 기종 기기와 접속이 유연하도록 interface를 만들어 정의하고 기기들과 쉽게 연동이 가능하도록 설계하였다.



<그림 3> 아답터의 구조

우선 각 정보는 크게 세가지로 분류된다. 환경 정보를 수집하는 센서정보, 사용자 정보를 수집하는 무선 기기, 서비스를 제공하는 어플리케이션 시스템 리소스로 나누어 연결된 기기들의 리스트를 관리하도록 한다. 아답터는 <그림 3>과 같이 통신 기기의 종류에 따라 인터페이스를 제공하고 각 센서 및 사용자와 디바이스 정보를 <표 1>과 같은 포맷으로 전송 받는다. 또한 아답터모듈에서 기초적인 에러 검출이 이루어 진다. 포맷 형태가 다르거나 수집시간오류 등을 검출하여 이를 로그에 남긴 후, 삭제한다.

<표 1> 환경, 사용자, 디바이스 및 어플리케이션의 패킷 포맷

환경 정보	
위치	센서타입 | 센서ID | 모바일ID | x좌표 | y좌표 | 수집시간
압력	센서타입 | 센서ID | 값 | 존속기간 | 수집시간
조도 / 소음	센서타입 | 센서ID | 값 | 수집시간
행동	센서타입 | 센서ID | 행동 | x좌표 | y좌표 | 수집시간
사용자 정보	
모바일	타입 | 모바일ID | 상태 | 수집시간
디바이스 / 어플리케이션 정보	
"HW" | "CPU" | <CPU 사용량> | "Mem" | <Memory 사용량>| "NW" | <네트워크 사용량> | 어플리케이션ID1 | 어플리케이션상태1 | 서비스ID1 | 서비스상태1 | 어플리케이션ID2 | 어플리케이션상태2 | 서비스ID2 | 서비스상태2 | 수집시간	

4.2 컨버터

다양한 형태의 센싱된 정보를 시스템이 사용할 수 있는 공통된 포맷으로 변환하는 작업이 필요하다. 또한 센서 및 디바이스 정보는 string 형태 이외에 xml의 형태로도 받을 수 있어야 하기 때문에 컨버터 안에 xml 파서를 두어 Xml문서는 따로 관리하도록 설계하였다. xml파서는 xml문서를 분석 한 뒤, 필요한 정보만 추출하여 string포맷으로 재구성한다. 이렇게 추출된 정보는 다른 포맷의 정보들과 함께 다음 단계에서 필터링 및 어그리게이션 된다. <그림 4>는 디바이스 정보가 담긴 xml문서의 내용으로 어플리케이션 아이디, 서비스 아이디, 상태 값 등의 정보가 들어있다.

```
<?xml version="1.0"? encoding="UTF-8"?>
<xmlns:uss="http://dmc.ajou.ac.kr/uss/uss-monitor/>
<uss:MessageName>HeartBeatReq</uss:MessageName>
<uss:TransID>238fq949tu68@dmc.ajou.ac.kr</uss:TransID>
<uss:AgentID>93kdkgal284</uss:AgentID>
<uss:Hardware>
<uss:CPUusage>30</uss:CPUusage>
<uss:MemoryUsage>10</uss:MemoryUsage>
<uss:NetworkUsage>20</uss:NetworkUsage>
</uss:Hardware>
<uss:Software>
<uss:Application id=1245>
<uss:ApplicationID>23@dmc.ajou.ac.kr</uss:ApplicationID>
<uss:ApplicationStatus>Active</uss:ApplicationStatus>
<uss:DeviceID>Light01</uss:DeviceID>
<uss:DeviceStatus>100</uss:DeviceStatus>
</uss:Application>
<uss:Application id=1248>
<uss:ApplicationID>27@cmd.ajou.ac.kr</uss:ApplicationID>
        <uss:ApplicationStatus>Running</uss:ApplicationStatus>
<uss:/Application>
</uss:Software>
```

☐ ApplicationID – 서비스 아이디
☐ ApplicationStatus – 서비스 상태(Active, DeActive, Fail)
☐ DeviceID – 디바이스 아이디
☐ DeviceStatus – 디바이스 상태값

<그림 4> 디바이스 정보가 담긴 xml 파일

4.3 필터링 및 어그리게이션

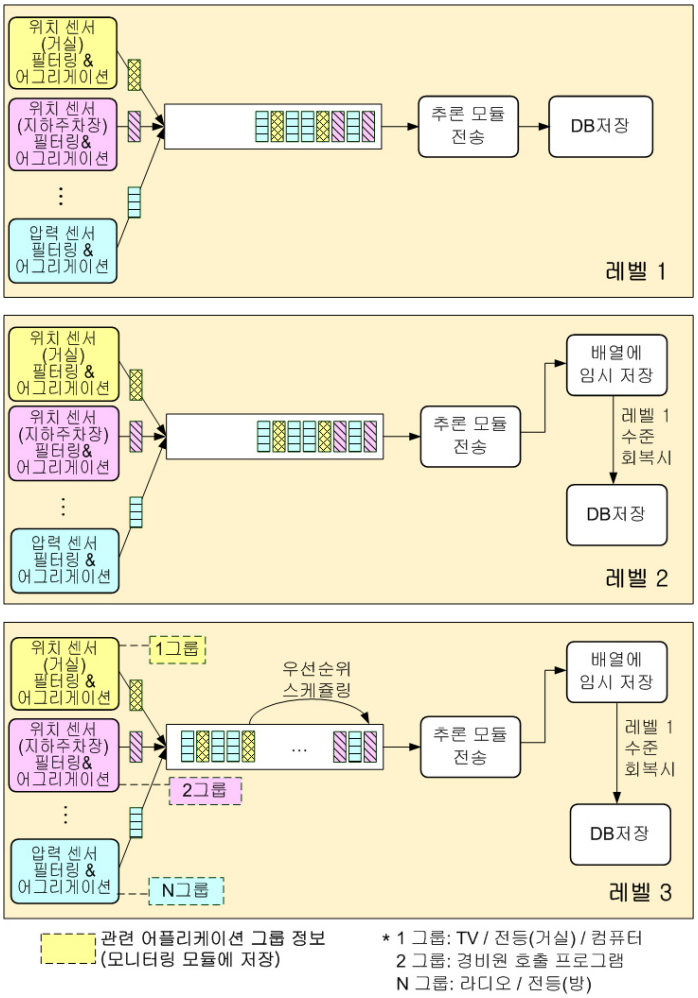
컨버터에서 xml 및 string 포맷을 string 포맷으로 통일한 뒤, 필터링 및 어그리게이션 단계에서 각 정보는 추론엔진에 적합한 형태로 가공된다. 가공되지 않은 원천정보들은 추론엔진에게 불필요한 정보도 함께 들어있다. 따라서 원천정보를 그대로 추론엔진으로 전송하면 추론엔진의 처리 속도에 비해 처리해야 할 정보의 많아져 지연시간이 증가하게 되고 환경정보들은 적체된다. 이러한 적체가 지속되면 지연시간은 기하급수적으로 증가하게 된다. 유비쿼터스 환경을 위한 유비쿼터스 컴퓨팅에서 사용자의 상황에 신속히 반응하는 기능은 매우 중요하다. 만약 센서를 통해 감지된 환경변화 인지 시각과 어플리케이션의 서비스 제공시간 사이의 지연이 크면 사용자에게 적합한 서비스를 제공해줄 수 없어 사용자는 오히려 불편을 느끼게 된다. 예를 들어 사용자가 방에 들어온 후 전등이 수초 이상 후에 켜진다면 사용자는 불편함을 느끼게 될 것이고, 이는 컴퓨팅 환경의 질적 저하를 의미한다. 또한 이때 전등이 켜지기 전 사용자가 다시 밖으로 나갔을 경우, 그 후 전등이 켜졌다 꺼지는 현상은 사용자가 어플리케이션의 오작동으로 오인하게 된다. 따라서 많은 양의 정보 중에 추론엔진이 필요로 하지 않는 정보를 필터링 하여 추론엔진이 처리해야 할 정보의 양을 줄여, 처리부담을 줄여주어야 한다. 또한 독립적으로 오는 센서의 위치, 시간 등 정보 각각과 사용자 정보 및 시스템이 미리 정의한 정보(ID) 들과 혼합하여 어그리게이션을 하여, 추론엔진이 신속하고 정확하게 추론할 수 있는 컨텍스트로 변환시킨다.

필터링은 크게 2가지의 기능을 수행한다. 첫째, JESS를 적용하고 있는 추론엔진의 다양한 룰에 필요한 정보만 걸러낸다. 만약 '데이터의 변화가 있을 경우'의 정보를 원하는 룰일 경우 변하지 않는 데이터를 추론엔진에게 넘겨 줄 필요가 없다. 따라서 추론엔진의 요구에 따라 이러한 메시지들은 데이터를 체크한 뒤, 로그를 남기고 삭제한다. 둘째, 추론엔진 또는 어플리케이션의 통신장애 및 처리속도보다 입력 데이터 량의 입력 양이 많을 경우 늦게 오는 데이터는 상황에 따라 필요 없게 된다. 따라서 필터링에서 지연 되는 정도를 파악하여 추론엔진에게 넘겨주는 데이터의 양을 줄여야 한다. 필터링 객체는 큐의 입력 양과 출력 양을 비교하여 필터링 수준을 조절한다. 필터링 수준은 추론엔진으로 데이터를 보내기 위한 큐에 적체된 메시지의 양 및 처리 속도를 보고 수준을 판단하여 <그림 5>와 같이 총 3수준으로 나누고 기본적으로 레벨1의 필터링을 제공한다. 이는 <표 2>에 정리하였다.

<표 2> 필터링 단계에서 여과 수준

	상황	필터링 & 어그리게이션	데이터베이스
레벨 1	정보의 입력 속도 ≤추론 처리속도	이벤트 발생시 추론엔진에게 전송	추론엔진 전송 후 데이터베이스에 저장
레벨 2	정보의 입력 속도 ≥ 추론 처리속도	레벨 1과 동일	배열에 임시 저장 레벨1 수준에 도달 시 정보저장
레벨 3	정보의 입력 속도 ≫ 추론 처리속도	정보에 우선순위 설정 후 중요도에 따라 우선 전송	레벨 2와 동일

레벨 1에서는 정상적인 속도로 정보가 처리되고 있는 상태일 때 적용되는 수준으로 상위인 추론엔진에게 필요하지 않은 정보들을 걸러낸다. 예를 들어 룰 기반인 추론엔진이 '데이터의 변화가 왔을 시'라는 룰로만 적용될 때가 있다. 이 경우 변하지 않은 데이터의 변화를 보내주어도 추론엔진의 처리 업무만 늘어날 뿐, 상황 정황은 변하지 않는다. 따라서 추론엔진의 상황에 따라 적절한 필터링을 해 주면 미들웨어의 성능을 높일 수 있다. 따라서 이 경우 여러 기기로부터 받은 정보들은 이전 데이터의 변화가 없을 시에 데이터를 필터링하고 어그리게이션 하여 추론엔진에게 전송한다. 즉, 같은 센서로부터 이전과 같은 값의 데이터가 왔을 경우 그 데이터를 추론엔진에 보내지 않고 삭제하는 대신, 필터링 모듈에서 정보를 기억하면서 이전 데이터와 시간을 통합하여 로그를 남긴다. 그리고 변화된 값이나 에러 패킷이 오는 경우, 또는 전송이 끊기면 추론엔진에게 값 변환이라는 컨텍스트를 생성하여 추론엔진에게 전송한다. 필터링을 마친 데이터는 어그리게이션 과정을 거친 후 상황정보로 가공되어 추론엔진으로 전송된 후 데이터베이스에 저장된다.

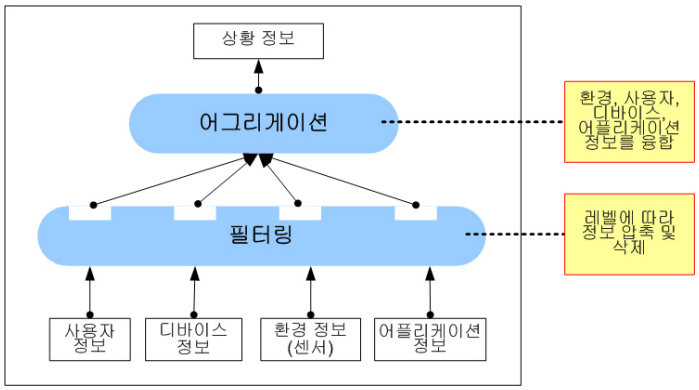


<그림 5> 레벨에 따른 필터링 및 어그리게이션 방법

레벨 2에서는 정보의 입력속도가 추론엔진의 처리 속도보다 빠른 경우로, 레벨1 처리방법에서 추론

엔진으로 전송 후 처리하였던 데이터베이스 전송을 일시 중지한다. 유비쿼터스 컴퓨팅 환경에서 사용자에게 신속한 서비스를 제공하는 것이 중요한 요소이기 때문에 데이터베이스에 저장되어있던 과거 데이터를 재사용 하는 것 보다 현재 들어오고 있는 정보의 사용이 훨씬 중요하고 빈번하게 일어난다. 따라서 데이터베이스에 접속 및 전송하는 시간이 다른 처리시간보다 많이 걸리기 때문에 배열 형태로 정보를 임시 저장 큐에 쌓아두었다가 큐에 들어있는 정보의 양이 줄어들면 그 동안 기억하고 있던 정보들을 데이터베이스로 전송한다.

레벨 3에서는 너무 많은 양의 데이터가 입력되어 추론엔진에서 중요시 되는 데이터가 제시간에 파악이 되지 않는 상태일 경우로, 전송 정보에 우선순위를 두어 서비스의 지연을 감소시킨다. 우선 순위를 결정하는 방법은 다음과 같다. 만약 어플리케이션이 서비스를 제공하지 못하는 경우, 상황이 인지되어도 서비스를 제공하지 못한다. 그러므로 이보다 다른 서비스를 제공할 수 있는 상황인지가 우선적으로 이뤄져야 한다. 그러므로 추론엔진 전송 우선순위는 센싱 시각과 현재 시각 지연 및 서비스를 제공하는 어플리케이션의 동작 여부 따라 결정된다.



<그림6> 원천정보의 가공

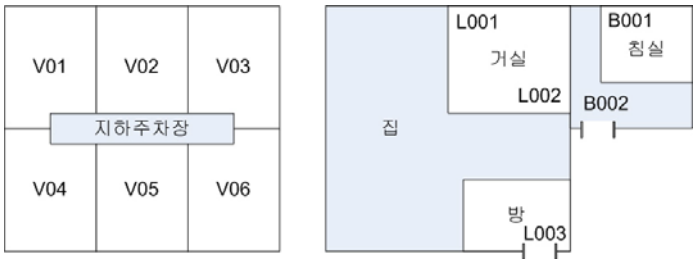
어그리게이션 단계에서는 디바이스 정보 및 사용자 정보와 센서 등 다양한 데이터의 정보를 혼합한다 <그림 6>. 추론엔진에서는 JESS라는 에이전트를 사용하기 때문에 각 센서 상태들의 많은 정보를 가지고 추론하려면 많은 부하가 들어 처리 속도가 떨어진다. 그러므로 모니터링 시스템에서 추론엔진의 부담을 덜어주기 위해 이러한 정보들을 혼합하여 보내주어야 한다. 사용자 정보 및 디바이스 정보는 거의 변하지 않고 이벤트 식으로 가끔씩 정보가 변하기 때문에 추론엔진에게 계속 알릴 필요가 없다. 그러므로 센서로부터 환경 정보가 들어왔을 때 사용자 정보 및 디바이스 정보, 센서정보를 혼합하여 추론엔진에게 보내주어 추론엔진이 최소의 비용으로 상황을 추론할 수 있도록 최적의 컨텍스트 형태로 보내준다.

4.4 정보 전송 및 데이터베이스 저장

센싱된 정보를 시스템이 사용할 수 있는 컨텍스트로 만든 뒤, 전송 단계에서 추론엔진과 데이터베이스에 가공된 정보를 전달한다. 데이터베이스에는 아답터에서 받은 정보들을 필터링 후, 어그리게이션을 하기 전의 포맷을 저장한다. 어그리게이션 단계에서는 추론엔진에게 최적의 컨텍스트를 전송해 주기 위해 추론엔진이 필요로 하는 정보만 추려서 보내지만 현재 상황 판단 외에 사후 관리를 위해서는 센서 및 디바이스, 사용자의 자세한 정보가 필요할 수 있기 때문이다. 만약 모니터링 시스템이 추론엔진에게 보낸 상황정보를 알고 싶다면 데이터베이스에 저장한 정보를 다시 혼합하여 만들 수 있다. 그러나 처음에 센서로부터 들어온 정보를 모두 저장하면 나중에 정보를 찾고자 할 경우, 방대한 양의 정보 때문에 효율성이 떨어지므로 필터링 이후의 정보를 저장한다.

5. 시나리오 및 구현

본 논문은 프로젝트 일환으로 지하 주차장과 집안을 대상으로 5명의 사람에게 필요한 서비스를 제공해 주는 유비쿼터스 컴퓨팅 환경을 조성한 시나리오를 기반으로 모니터링 시스템을 검증하였다. 예를 들어 <그림 7>과 같이 공간은 거실, 침실, 지하주차장으로 설정하여 센서를 통해 위치 정보를 얻게 된다.



<그림 7> 위치 센서 구성도

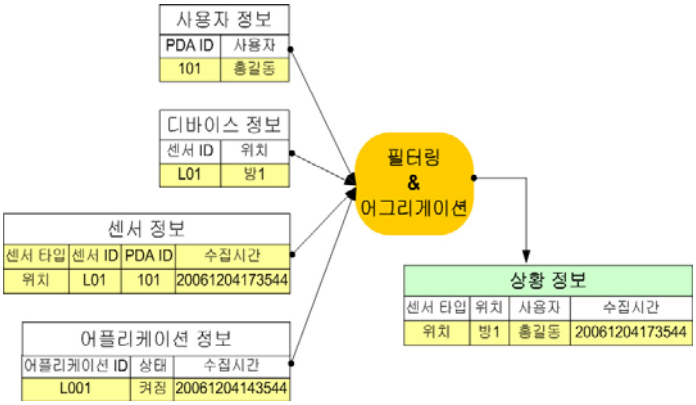
또한 사용자 정보는 5명으로 감시자 그룹으로 불리는 경비원과 서비스를 이용하는 사용자로 나누었다. 센서의 종류는 총 6종류로 위치, 동작, 조도, 모바일 패드, 압력, 소음으로 나뉘어 들어오게 된다. 디바이스는 CPU, 메모리, 네트워크 정보가 들어오게 된다. 이를 위해 아답터에 다양한 기기가 등록을 하고 연결을 맺는다. 컨텍스트 위젯은 기본적으로 <표 3>과 같은 기본 환경 정보를 갖고 있다. 또한 이 정보에서 사용자 및 디바이스, 어플리케이션 정보가 바뀔 때 마다 이벤트 형식으로 컨텍스트 위젯에게 교체된 정보를 보내면 위젯은 기존 정보에 대한 로그를 남기고 새로운 정보로 교체한다. 이러한 디바이스 및 사용자 정보는 추론엔진에게 전송하기 위한 어그리게이션 단계에서 주기적으로 들어오는 센서정보와 통합 된 뒤 추론엔진에게 보내게 된다.

<표 3> 모니터링 시스템이 가지고 있는 기본 정보

센서ID-공간		사용자ID-그룹		모바일ID-사용자	
L001~003	거실 1~6	101	감시자	1	홍길동
B001~002	침실 1~2	102	사용자	2	고길동
V01~06	지하주차장 1~6			3	경비원1
				4	경비원2
				5	모니터요원

각 센서는 전송 빈도가 다르다. 예를 들어 위치

센서는 0.5초마다, 압력센서는 2초마다 전송된다. 이러한 전송 빈도가 낮으면 컨텍스트 위젯의 필터링 및 어그리게이션은 레벨1 단계로 처리하고 높으면 레벨3단계로 처리하였다. 본 시나리오에서는 입력 정보량을 변화시켜 레벨1에서부터 3까지 테스트 하였다. 컨텍스트 위젯은 입력 정보를 빠르게 필터링 및 가공하여 정보를 이용하는 추론엔진에게 적절한 상황정보를 제공함으로써 추론엔진의 성능향상에 도움을 주었다.



<그림 8> 다양한 기기로부터 전송된 정보와
필터링 및 어그리게이션을 거친 후 가공된 정보 포맷

<그림 8>은 홍길동이라는 사용자가 방에 들어왔을 때 각 센서정보, 디바이스 정보, 어플리케이션 정보 및 사용자 정보의 필터링 및 통합 과정을 나타낸다. ID 'L01'을 가진 위치 센서는 PDA에 달린 RFID를 읽어 PDA ID가 101임을 감지하고 수집시간과 자신의 ID, 모바일 ID를 담아 모니터링 시스템에게 보낸다. 이 센서정보는 기존에 등록했던 디바이스 정보와 매치되고 ID가 L01인 센서가 방1에 존재하고, ID 가 101인 PDA를 가지고 있는 사람이 홍길동이라는 사실을 알고 있다. 따라서 추론엔진에게 필요 없는 ID를 제외하고, 추론엔진이 원하는 상황정보를 담아 추론엔진에게 보낸다. 즉, 원천정보들을 상위 추론 엔진이 필요한 정보로만 선택, 취합하여 새로운 상황정보 컨텍스트를 만든다. 이렇게 각 정보에서 필요한 정보만을 추출하고 상위 처리시스템이 인식하는 정보만 전달함으로써 데이터 전송에 드는 비용을 감소시킨다.

컨텍스트 위젯은 상황 판단을 하는 추론엔진에게 적합한 컨텍스트로 가공하여 환경 정보를 상황정보를 만들어 전송한 후, SQL을 사용해 데이터베이스에 저장하였다. 데이터베이스 저장 시 각 센서 및 디바이스, 어플리케이션에서 받은 정보를 필터링 후 의 각 정보를 추론엔진에게 전송하는 단계가 끝난 후 데이터베이스에 저장하였다. 단, 어그리게이션 과정에서 ID등의 정보가 삭제되어 컨텍스트가 생성되기 때문에 추후에 정보를 얻을 때 정보의 손실이 있을 수 있기 때문에 단 통합된 정보가 아닌 어그리게이션 이전 정보들을 추론엔진에게 전송할 때까지 가지고 있다가 저장하였다. 각 센서, 디바이스 및 사용자 정보는 다른 시스템이나 어플리케이션이 어떻게 사용하는지에 따라 SQL 구문으로 내용을 얻을 수 있다. 따라서 데이터 베이스에서 내용을 획득하면 컨텍스트 위젯이 추론 엔진에게 어떠한 정보를 보냈는지 전송한 환경 정보에 대한 내용도 알 수 있다.

6. 결론

유비쿼터스 환경을 이루기 위해 사용자에게 상황에 맞는 적합한 서비스를 찾아 제공하기 위해서는 유비쿼터스 환경에 설치된 장치들의 컨텍스트를 신속하게 수집하고 해석해야 한다. 그러나 유비쿼터스 컴퓨팅 환경은 다양하고 많은 독립적인 장치로 구성되어 있다. 이러한 각각의 장치는 다양한 컨텍스트 정보와 포맷을 가진다. 이렇게 다양한 장치들로부터 수집된 정보는 서로 다른 포맷으로 되어 있기 때문에 수집된 정보들을 하나의 시스템이 사용하려면 이러한 정보를 공통적인 포맷으로 변환해야 한다. 또한 정보 처리 속도 보다 많은 양의 정보가 들어올 경우 환경 인지 속도가 느려져 이는 곧 서비스의 지연을 가져오는 문제가 있다. 따라서 컨텍스트 위젯은 이러한 유비쿼터스의 2가지 요구사항을 반영하여 4가지 모듈로 나누어 설계 및 구현되었다. 또한 이러한 환경 및 사용자, 제공자의 정보를 저장하여 차후 발생 할 수 있는 해당 양식에 대해 예측 할 수 있고, 유비쿼터스 컴퓨팅에서 에러가 생겼을 때 확인 요소가 될 수 있다. 프로젝트의 일환으로 구현된 모니터링 시스템으로 약 7종류의 센서정보와 2종류의 사용자 정보, 3가지의 디바이스 정보를 모니터링 하고, 이를 필터링 및 어그리게이션 하여 다른 시스템에게 적합한 형태의 컨텍스트로 제공하였다. 컨텍스트 위젯은 더 많은 종류의 정보들을 쉽게 추가 할 수 있는 구성으로 구현되었다. 또한 종류에 따라 처리 하는 아답터 및 필터링을 하고 있기 때문에 더 많은 양의 정보를 신속하고 정확하게 가공, 위 시스템에 전달 할 수 있다.

향후 연구과제로는 많은 양의 데이터를 데이터베이스에 효율적으로 저장하기 위해 컨텍스트 정보 포맷을 결정하는 연구가 필요하며 상황에 따른 환경 모니터링의 효과적인 빈도 조절 연구가 필요하다. 또한 본 논문에서 구축된 상황인지 모니터링 시스템을 광역적인 유비쿼터스 환경에서 다양한 통신 방법에 쉽게 적용가능하고 및 많은 양의 데이터 정보를 처리 가능하도록 확장 연구 및 개발이 필요하다.

7. 참고 문헌

[1] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, K. Nahrstedt, "a middleware infrastructure for active spaces", in Pervasive Computing, IEEE (2002)

[2] T. Peter, "software infrastructure for ubiquitous computing environments: supporting synchronous collaboration with heterogeneous devices" in Proc. UbiComp (2001)

[3] S.S. Yau, F. Karim, W. Yu, W. Bin, S.K.S. Gupta, "reconfigurable context-sensitive middleware for pervasive computing", in Pervasive Computing, IEEE (2002)

[4] C. Harry, F. Tim, "an ontology for context-aware pervasive computing environments" in Proc. Workshop Ontologies and Distributed Systems, IJCAI Press (2003)

[5] M. Khedr, A. Karmouch, "negotiating context information in context-aware systems", in IEEE Intelligent Systems 19(6): p21-29 (2004)