센서와 미들웨간의 통신을 위한 아키텍쳐 설계

정종태*, 정국상*, 최덕재**
전남대학교 전산학과
e-mail : {northstar,handeum}iat.chonnam.ac.kr*, dchoi@chonnam.ac.kr**

Design of the Architecture for Communication between Sensors and Middleware

Jongtae Jeong, Kugsang Jeong, Deokjai Choi
Dept. of Computer Science, Chonnam National University

요 약

유비쿼터스 컴퓨팅 환경의 인프라는 센서, 미들웨어, 그리고 응용 프로그램으로 구성된다. 유비쿼스터 컴퓨팅 환경이 실현되기 위해서는 이 세 요소들은 상호 유기적으로 메시지를 전달해야 한다. 특히 센서와 미들웨어간의 통신은 이 점에서 중요한 역할을 한다. 본 논문에서는 센서와 미들웨어간의 통신 메커니즘을 지원하는 아키텍쳐를 제안한다. 기존의 시스템에서는 센서와 미들웨어 사이에서 통신 기능을 담당하는 컴포넌트는 데이터와 이를 처리하는 기능이 함께 존재했다. 그렇기 때문에 데이터를 처리 기능이 같을지라도 받아 들이는 데이터가 다르고 처리하는 정보가 다르다면 센서 수와 같은 컴포넌트가 존재해야 한다. 또한 센서와 미들웨어간의 통신 기능을 담당하는 컴포넌트를 만들기 위해서는 미들웨어와 센서에서 제공하는 API 를 이용하여 개발자가 직접 코딩작업을 해야 한다. 이럴 경우 개발자의 시간과 노력이 많이 필요로 한다. 두 문제점을 해결하기 위하여 먼저 데이터와 이 데이터를 처리하는 부분을 분리시킨다. 이러한 메커니즘은 SNMP 에서 도입하였다. SNMP 를 구성하는 요소 중에서 자료를 처리하는 부분은 에이전트가 맡고, 자료를 저장하는 부분은 MIB 이 맡는다. 그 결과 해당 컴포넌트의 재사용이 가능하게 된다. 또한 MIB 과 에이전트의 개발 시간을 단축하기 위해서 SNMP 를 이용한 툴킷을 이용한다. 이렇게 함으로써 센서측과 미들웨어 사이에 통신하는 컴포넌트를 개발하는 시간과 개발자의 노력의 효율성을 증대 시킬 수 있다.

1. 서론

최근 IT 패러다임이 노매딕 컴퓨팅[1]에서 유비쿼터스 컴퓨팅으로 변화하면서 사람들은 사회적으로 많은 변화를 느끼고 있다. 유비쿼터스 컴퓨팅 환경은 인간의 조작에 의해 환경을 변화시키는 것이 아니라 환경이 인간의 상태와 주변 환경의 요소에 따라서 최적인 환경을 스스로 구축하는 것이다.

이런 유비쿼터스 컴퓨팅 환경을 구축하기 위해서는 크게 세 부분으로 구성되어 있다. 사용자들이 흔히 접할 수 있는 사물이나 사람들을 감지할 수 있는 많은 종류와 수량의 센서 부분과 센서에서 발생된 데이터를 받아 들이는 미들웨어 부분, 그리고 이러한 데이터를 사용자가 직접 이용할 수 있도록 응용 프로그램이 있다. 이와 같이 유비쿼터스 컴퓨팅 환경의 인프라를 구성하고 있는 세 가지 요소들은 유기적으로 동작하는 것 중요하다. 즉 센서에서 감지된 데이터는 미들웨어에 잘 전달이 되어야 하고, 미들웨어는 사용자에게 많은 데이터를 응용 프로그램에게 전달해야만 한다. 결국 중요한 것은 세 요소간의 메시지를 전달

* 본 연구보고서는 정보통신부 정보통신연구진흥원에서 지원하고 있는 정보통신기초연구지원사업의 연구결과입니다.

하는 각 구성요소들(센서-미들웨어, 미들웨어를 구성하는 요소, 미들웨어-응용 프로그램)간의 통신이다. 이 중에서 1 차적으로 센서가 감지한 주변 환경 정보를 미들웨어로 전달하는 것이 무엇보다 중요하다. 따라서 본 논문에서는 센서와 미들웨어간의 통신방식을 기술하려 한다.

현재까지 미들웨어에 대한 연구는 Dey 의 Context 미들웨어[2], CALAIS[3], Gaia[4], 그리고 Oxygen[5] 등의 프로젝트가 진행되어 왔다. 이 미들웨어들은 센서와 통신하기 위한 독자적인 컴포넌트를 가지고 있다. 하지만 이들 미들웨어도 센서와 통신하기 위한 방법으로 크게 두 가지로 분류가 가능하다. 첫 번째는 센서에 이벤트가 발생되어 미들웨어에 통보하는 방식과 두 번째는 미들웨어에서 필요한 정보를 센서 측으로부터 얻는 방식이다.

정보를 생성하는 센서 측과 정보를 수집하고 가공하는 미들웨어는 서로 통신할 수 있는 기능을 가진 컴포넌트가 존재해야 한다. 대표적으로 Dey 의 프레임워크(CKT)에서는 위젯이라 명명했다[2]. 이 위젯은 각 센서와 일대일 대응을 이루고 있다. 왜냐 하면, 센서와 통신하는 컴포넌트가 동일한 계산 기능을 가지고 있더라도 센서에서 받은 정보가 다르기 때문에 다른 센서에서 사용할 수가 없다. 또한 위젯은 센서와 미들웨어에서 API 를 제공한다고 할 지라도 개발자가 손수 코딩작업을 해야 하기 때문에 개발의 용이성이 떨어진다.

센서에 대해 위젯과 같은 호환성 문제를 해결하기 위해서 기존에 망 관리에서 사용한 SNMP 개념을 적용한다. 왜냐하면, SNMP 는 망 관리분야에서 오랜 시간 사용하여 정립이 잘 되어 있다. 특히 이벤트 처리하는 부분이 센서에서 이벤트를 처리하는 것과 유사하기 때문이다. SNMP 를 구성하고 있는 요소는 MIB, 에이전트, 그리고 매니저가 있다. 그 중 에이전트는 네트워크 장비들과 통신 하면서 관리자가 필요한 정보를 획득, 특정 값을 설정, 그리고 측정된 값이 일정 범위를 벗어날 경우 통보한다. 여기서 에이전트는 획득한 정보를 가지고 있지 않고, MIB 에 저장하게 된다. 이처럼 정보를 저장하는 부분과 계산하는 부분이 분리가 가능하다. 두 기능이 분리가 되고, 그 결과 센서간의 호환성 문제를 극복할 수 있다.

SNMP 는 오랜 시간 동안 잘 정립되어 왔다. 그리고 SNMP 를 지원하는 많은 툴들이 현존하고 있다. 따라서 이런 툴을 이용하면 SNMP 의 에이전트를 만드는 것이 쉽다는 것이다. 다시 말하면 위젯은 미들웨어가 제공하는 API 를 바탕으로 개발자가 손수 코딩작업을 해야 했다. 이 때문에 컴포넌트를 만드는 일에 많은 시간과 노력이 필요했다. 하지만 관련 툴을 이용함으로써 센서와 미들웨어 사이를 담당하는 컴포넌트를 개발하는 시간과 노력을 절약하게 된다.

본 논문의 2 장에서는 제안한 시스템의 구조에 대해 기술하며 3 장에서는 메시지를 중점으로 본 시스템의 동작 과정을 기술한다. 마지막으로 결론에 대해 기술한다.

2. 시스템 구조

본 논문에서 제안한 시스템은 망 관리에서 사용되는 SNMP 에 기초하여 센서와 미들웨어간의 통신을 지원하는 것을 목적으로 한다. 이 시스템은 SNMP 의 구성요소인 MIB, 에이전트, 그리고 매니저를 포함하고 있다. 이 시스템은 기본적으로 MIB 과 에이전트가 독립적으로 존재한다. 따라서 자연스럽게 데이터를 저장하는 부분과 데이터를 처리하는 부분이 분리가 된다. 즉, 센서에서 감지한 데이터는 에이전트를 거쳐서 MIB 으로 저장된다. 또한 에이전트는 매니저가 요청한 데이터에 대해서는 MIB 에 접근하여 데이터를 전달한다.

2.1. 구성 요소

그림 1 은 본 아키텍쳐를 간략하게 도식화 하였다. 본 시스템은 크게 3 계층, 6 개의 구성 요소로 구성되어 있다. 다음은 6 개의 구성 요소의 역할을 살펴본다.

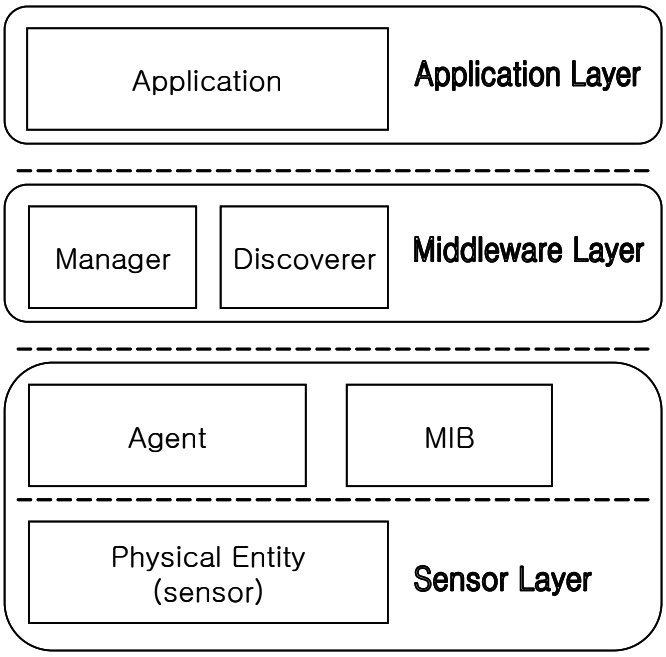

그림 1. Functional 아키텍쳐

• Physical Entity: 주위 환경에 존재하는 센서와 컴퓨팅 디바이스이다.

• Agent: 하드웨어 센서 혹은 소프트웨어 센서에서 상황정보를 얻는 컴포넌트이다. 에이전트는 두 가지의 연산자를 가지고 있다. 첫째, 센서의 값을 가지고 올 수 있도록 하는 Get 연산자와 둘째, 매니저가 일정한 값을 지정한 후에 센서에서 그 값의 이상일 경우 알려주는 Trap 연산자가 있다.

• Manager: 매니저는 Get 연산자와 Set 연산자를 가진다. Get 연산자는 원하는 데이터를 에이전트에 요청할 때, Set 연산자는 에이전트에 특정 값을 설정할 때

사용한다. 그리고 매니저의 역할은 크게 두 부분으로 나눌 수 있다. 첫째 자신의 하위에 있는 에이전트를 관리하는 역할을 가진다. 매니저는 에이전트에서 발생된 문제를 알아야 하고 문제가 있는 에이전트가 있다면 적절한 조치를 취해야 한다. 둘째 에이전트에서 받은 데이터를 분석하여 원하는 응용 프로그램에게 전달하는 역할을 한다.

• Discoverer: 응용 프로그램이 관심 있는 매니저나 자원의 위치를 알 수 있도록 한다.

• MIB: Physical Entity 에 대한 정보를 객체로 표현하고 이러한 객체들이 구조화된 모임을 MIB 이라고 한다. MIB 는 객체 자신의 정보나 객체가 감지한 정보 즉 상황정보를 저장한다

• Application: 사용자가 이용하는 프로그램이다.

3. 동작 과정

이 섹션에서는 메시지가 이동하는 것을 중심으로 본 아키텍쳐을 동작 과정을 설명하고자 한다. 본 아키텍쳐에서 사용되는 응용 프로그램은 In/Out Board[2]이다. 이 프로그램은 일정 공간에 사람의 출입 여부를 사용자에게 통보한다. 메시지 이동 방식은 Trap 방식과 Get 방식이 있다.

Trap 방식은 사용자가 지정된 값이 센서에서 감지되었을 경우 에이전트가 매니저에게 알려주는 방식이고 Get 방식은 사용자가 알고 싶은 값을 매니저에게 질의를 하면 매니저가 에이전트를 통해서 얻어 오는 방식이다[6].

그림 2, 3 은 순차 다이어그램[7]으로 컴포넌트 사이에 메시지 전달 과정을 간단한 예를 통해서 보여주고 있다.

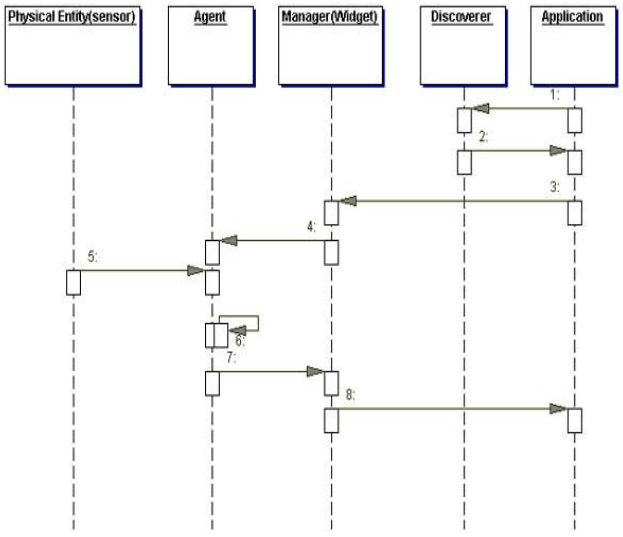

그림 2. Trap 메커니즘

3.1. Trap 메커니즘

Trap 방식은 그림 2 이다. 응용 프로그램에서 철수가 사무실의 출입 여부 정보만을 받을 수 있도록 하는 과정을 설명한다.

(1) 응용 프로그램은 위치 에이전트를 관리하는 매니저가 어디 있는지를 Discoverer 에 의해서 위치를 안다.
(2) 응용 프로그램은 매니저에게 철수가 들어왔는지 나갔는지의 정보를 받을 수 있도록 등록한다.
(3) 매니저는 응용 프로그램이 등록한 부분을 에이전트에게 전달한다.
(4) 센서는 사람들의 출입을 감지하여 에이전트에게 보낸다.
(5) 센서가 보낸 상황정보는 MIB에 저장된다.
(6) 에이전트는 매니저가 지정한 값과 동일한지를 비교한다.
(7) 만약 매니저에서 지정한 값이 일치한다면 에이전트는 매니저에게 해당되는 정보를 전달한다.
(8) 매니저는 에이전트에서 받은 정보를 응용 프로그램에게 전달한다.

3.2. Get 메커니즘

Get 방식은 그림 3 과 같다. 응용 프로그램에서 철수가 사무실에 있는지 없는지를 알 수 있는 과정을 설명한다.

(1) 응용 프로그램은 위치 에이전트를 관리하는 매니저가 어디 있는지를 Discoverer 에 의해서 위치를 안다.
(2) 응용 프로그램은 매니저에게 철수가 사무실에 있는지에 대한 정보를 얻고자 요청한다.
(3) 매니저는 응용 프로그램이 요청한 정보를 에이전트에게 전달한다.
(4) 센서는 사람들의 출입을 감지하여 에이전트에게 보낸다.
(5) 센서가 보낸 상황정보는 MIB에 저장된다.
(6) 에이전트는 매니저가 요청한 정보에 대해서 최근에 발생된 정보를 매니저에게 값을 전해준다.
(7) 매니저는 에이전트에서 받은 정보를 응용 프로그램에게 전달한다.

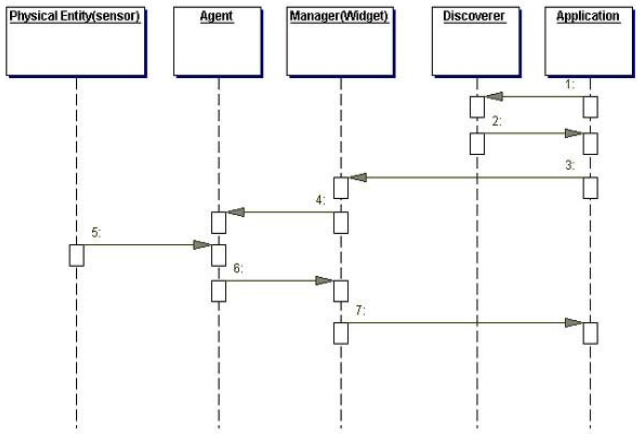

그림 3. Get 메커니즘

4. 결론

센서와 미들웨어간의 통신에서 기존의 시스템들은 데이터와 계산 기능을 한 컴포넌트에 둠으로써 해당 컴포넌트의 재사용을 불가능하게 했다. 또한 개발자는 센서와 미들웨어에서 제공하는 API 에 기초하여 컴포넌트를 만들었다. 이 두 가지 문제를 해결하기 위해서 본 논문에서는 SNMP 메커니즘을 도입하였다. SNMP 를 구성하는 요소 중에 데이터를 저장하는 부분과 데이터를 처리하는 부분이 분리가 되어 있다. 즉 SNMP 는 MIB 과 에이전트가 존재하여 두 부분이 분리가 가능하다. 또한 SNMP 는 오랜 세월 동안 정립이 잘 되어 있고 많이 사용하기 때문에 SNMP 를 지원하는 라이브러리나 툴킷을 제공하는 벤더들이 많고, 이러한 라이브러리나 툴킷을 사용할 수 있는 장점이 있다. 이렇듯 SNMP 를 이용함으로써 컴포넌트의 재사용이 가능하게 되고, 컴포넌트를 개발하는 개발자는 좀더 빠른 시간에 통신을 담당하는 컴포넌트를 개발할 수 있다.

참고문헌

[1] “A Development Scenario for Ubiquitous Networks: Viewed from the Evolution of IT Paradigms”, Hiroyuki NAKAMURA, Nomura Research Institute
[2] “Providing Architectural Support for Building context-Aware Application”, Anind K. Dey, Georgia Institute of Technology
[3] “Context-Aware and Location System”, Giles J. Nelson, Clare College
[4] Gaia, http://gaia.cs.uiuc.edu/index.html
[5] Oxygen project, http://oxygen.lcs.mit.edu
[6] *The Practical Guide to Network-Management Standards*, William Stallings, ADDISON-WESLEY PUBLISHING COMPANY
[7] *UML 객체지향 분석·설계*, 조완수저, 홍릉과학출판사