센서와 미들웨어간의 통신을 위한 아키텍처 설계 및 구현

정종태°, 정국상, 최덕재
전남대학교 전산학과
{northstar°, handeum}@iat.chonnam.ac.kr, dchoi@chonnam.ac.kr

Design and Implementation of the Architecture for Communication between Sensors and Middlewar

Jongtae Jeong° Kugsang Jeong, Deokjai Choi
Department of Computer Science, Chonnam National University

요 약

유비쿼터스 컴퓨팅 환경의 인프라는 센서, 미들웨어, 그리고 응용 프로그램으로 구성된다. 유비쿼터스 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 아키텍처를 제안한다. 기존의 유비쿼터스 컴퓨팅 시스템에서 센서와 미들웨어 사이의 통신 기능을 담당하는 컴포넌트는 컴포넌트 내에서 데이터를 가지고 있고 더불어 이를 처리하는 기능도 함께 존재했다. 그렇기 때문에 데이터를 처리할 때 같은 메커니즘을 가지고 있을지라도 받아들이는 데이터가 다르다면 센서 수와 같은 컴포넌트가 존재해야 한다. 또한 센서와 미들웨어간의 통신 기능을 담당하는 컴포넌트를 만들기 위해서는 미들웨어와 센서에서 제공하는 API를 이용하여 개발자가 직접 코딩을 해야 한다. 이럴 경우 개발자의 시간과 노력이 많이 필요로 한다. 두 문제점을 해결하기 위하여 먼저 데이터와 이 데이터를 처리하는 부분을 분리시킨다. 이러한 메커니즘은 SNMP에서 도입하였다. SNMP를 구성하는 요소 중에서는 자료를 처리하는 부분은 에이전트가 담당하고, 자료를 저장하는 부분은 MIB이 담당한다. 결과적으로 해당 컴포넌트의 재사용이 가능하게 된다. 또한 MIB과 에이전트의 개발 시간을 단축하기 위해서 SNMP를 이용한 툴킷을 이용한다. 이렇게 함으로써 센서와 미들웨어 사이에 통신하는 컴포넌트를 개발하는 시간이 절약되며 개발자의 수고가 덜게 된다.

1. 서론

최근 IT패러다임이 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 위한 인프라는 크게 세 부분으로 구성되어 있다. 사용자들이 흔히 접할 수 있는 사물이나 사람들을 감지하는 많은 종류와 수의 센서 부분, 그 센서에서 발생된 데이터를 받아들이는 미들웨어 부분, 그리고 이러한 데이터를 사용자가 직접 이용할 수 있도록 하는 응용 프로그램이 있다. 여기서 중요한 것은 이 세 요소들은 유기적으로 동작하는 것이다. 즉 센서에서 감지된 데이터는 미들웨어에 잘 전달이 되어야 하고, 미들웨어는 많은 데이터를 사용자가 원하는 정보를 사용자가 사용하는 응용 프로그램에게 전달해야 한다. 결국 중요한 것은 메시지를 전달하는 각 구성요소들(센서-미들웨어, 미들웨어를 구성하는 요소, 미들웨어-응용 프로그램)간의 통신이다. 이 중에서 1차적으로 센서가 감지한 주변 환경 정보를 미들웨어로 전달하는 것이 무엇보다도 중요하다. 따라서 본 논문에서는 센서와 미들웨어간의 통신방식을 기술하려 한다.

현재까지 미들웨어에 대한 연구는 Dey의 Context 미들웨어[2], CALAIS[3], Gaia[4], 그리고Oxygen[5]등의 프로젝트가 진행되어 왔다. 이 미들웨어들은 각각의 센서와 통신하기 위한 독자적인 컴포넌트를 가지고 있다. 하지만 이 미들웨어들도 센서와 통신하기 위한 방법으로 크게 두 가지로 분류가 가능하다. 첫 번째, 센서에서 이벤트가 발생되어 미들웨어에 통보하는 방식과 두 번째, 미들웨어에서 필요한 정보를 센서 측으로부터 얻는 방식이다.

데이터를 생성하는 센서와 데이터를 수집하고 가공하는 미들웨어는 서로 통신할 수 있는 기능을 가진 컴포넌트가 존재해야 한다. 이 대표적인 예로 Dey의 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 API를 제공한다고 할지라도 개발자가 손수 코딩작업을 해야 하기 때문에 개발의 용이성이 떨어진다.

위젯에서 지적한 문제를 해결하기 위해서 기존에 망 관리에서 사용한 SNMP 개념을 적용한다. 왜냐하면, SNMP는 망 관리 분야에서 오랜 시간 사용하여 정립이 잘 되어 있고, 특히 이벤트 처리하는 부분이 센서에서 이벤트를 처리하는 것과 유사하기 때문이다. SNMP를 구성하고 있는 요소는 MIB, 에이전트, 그리고 매니저가 있다. 이 중 에이전트는 네트워크 장비들과 통신하면서 관리자가 필요한 정보를 획득, 특정 값을 설정, 그리고 측정된 값이 일정 범위를 벗어날 경우 통보한다. 또한 에이전트는 획득한 데이터를 가지고 있지 않고, MIB에 저장하게 된다. 이처럼 데이터를 저장하는

* 본 연구보고서는 정보통신부 정보통신연구진흥원에서 지원하고 있는 정보통신기초연구지원사업의 연구결과입니다.

부분과 계산하는 부분이 분리가 가능하다. 결과적으로 컴포넌트 내에 센서간의 호환성 문제를 극복할 수 있다.

SNMP 는 1988년에 RFC 1157으로 표준이 발표되었다. 그리고 프로토콜이 간단하기 때문에 망 관리에서는 많이 사용되고 있다. 그렇게 때문에 정립이 잘 되어 있다. 그리고 망 관리에서 SNMP를 많이 사용하기 때문에 이를 지원하는 많은 툴들이 현존하고 있다. 따라서 이런 툴을 이용하면SNMP의 에이전트를 만드는 것이 쉽다는 것이다. 다시 말하면 위젯은 미들웨어가 제공하는 API와 센서에서 제공하는 API를 바탕으로 개발자가 손수 코딩작업을 해야 했다. 이 때문에 컴포넌트를 만드는 일에 많은 시간과 노력이 필요했다. 하지만 SNMP관련 툴을 이용함으로써 센서와 미들웨어 사이를 담당하는 통신 컴포넌트, 즉 에이전트를 개발하는 시간과 노력을 절약하게 된다.

본 논문의 구성은2장에서는 제안한 시스템의 구조에 대해 기술하며, 3장에서는 메시지 중점으로 본 시스템의 동작 과정을 기술한다. 마지막으로 결론에 대해 기술한다.

2. 시스템 구조

본 논문에서 제안한 시스템은 망 관리에서 사용되는 SNMP에 기초하여 센서와 미들웨어간의 통신을 지원하는 것을 목적으로 한다. 이 시스템은 SNMP의 구성요소인 MIB, 에이전트, 그리고 매니저를 포함하고 있다. 이 시스템은 기본적으로 MIB과 에이전트가 독립적으로 존재한다. 따라서 자연스럽게 데이터를 저장하는 부분과 데이터를 처리하는 부분이 분리가 된다. 즉, 센서에서 감지한 데이터는 에이전트를 거쳐서 MIB으로 저장하고, 에이전트는 매니저가 요청한 데이터에 대해서는 MIB에 접근하여 데이터를 전달한다.

2.1 구성요소

그림 1은 본 아키텍처를 도식화 하였다. 본 시스템은 크게 3계층, 6개의 컴포넌트로 구성되어 있다. 다음은 6개의 컴포넌트의 역할을 살펴본다.

- Physical Entity: 주위 환경에 존재하는 센서와 컴퓨팅 장치이다.
- Agent: 하드웨어 센서 혹은 소프트웨어 센서에서 상황 정보를 얻는 컴포넌트이다. 이 에이전트는 두 가지의 연산자를 가지고 있다. 첫째, 센서의 값을 가지고 올 수 있도록 하는 Get 연산자와 둘째, 매니저가 일정한 값을 지정한 후, 센싱된 값이 그 범위를 벗어날 때 알려주는 Trap 연산자가 있다.
- Manager: 매니저는 Get연산자와 Set 연산자를 가진다. Get연산자는 원하는 데이터를 에이전트에 요청할 때, Set연산자는 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.

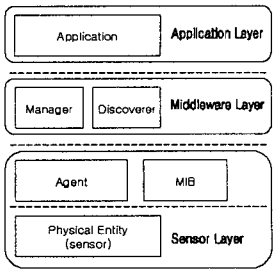

그림 1. Functional 아키텍처

- Discoverer: 응용 프로그램에게 응용 프로그램이 관심 있는 매니저의 위치를 알려준다.
- MIB: Physical Entity에 대한 정보를 객체로 표현하고 이러한 객체들이 구조화된 모임이다. MIB는 객체 자신의 정보나 센서가 감지한 정보 즉 상황 정보를 저장한다.
- Application: 사용자가 이용하는 프로그램이다.

3. 동작과정

이 섹션에서는 메시지가 이동하는 것을 중심으로 본 아키텍처 동작 과정을 설명하고자 한다. 본 아키텍처에서 사용하는 응용 프로그램은 실내 온도를 측정할 수 있는 시스템이다. 이 응용 프로그램에서는 가정에서 실내 온도를 측정하여 사용자에게 알려주거나 특정 가전기기를 제어할 수 있다. 메시지 이동 방식은 Trap 방식과 Get 방식이 있다.

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

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

3.1. Trap 메커니즘

Trap 방식은 그림 2이다. 이는 응용 프로그램이 방안의 온도를 측정하여 사용자가 설정한 온도(예, 15℃)보다 높은 경우에 정보를 받을 수 있도록 하는 과정을 설명한다.

(1) 응용 프로그램은 위치 에이전트를 관리하는 매니저가 어디 있는지를 Discoverer에 의해서 위치를 안다.
(2) Discoverer는 응용 프로그램이 요구한 매니저의 위치를 알려준다.
(3) 응용 프로그램은 매니저에게 방안의 온도에 대한 정보를 받을 수 있도록 등록한다.
(4) 매니저는 응용 프로그램이 등록한 부분을 에이전트에게 전달한다.
(5) 센서는 방안의 온도를 감지하여 에이전트에게 보낸다.
(6) 에이전트는 매니저가 설정한 조건에 부합하는지 검사한다.
(7) 만약 매니저에서 지정한 조건이 일치한다면 에이전트는 매니저에게 해당되는 정보를 전달한다.
(8) 매니저는 에이전트에서 받은 정보를 응용 프로그램에게 전달

한다.

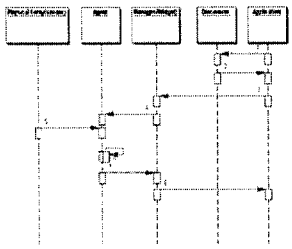

그림 2. Trap 메커니즘

3.2. Get 메커니즘

Get 방식은 그림3과 같다. 이는 응용 프로그램이 방안의 온도가 얼마나 되는지를 알 수 있도록 있는 과정을 설명한다.

(1) 응용 프로그램은 위치 에이전트를 관리하는 매니저가 어디 있는지를 Discoverer에게 질의한다.

(2) Discoverer는 응용 프로그램이 요구한 매니저의 위치를 알려 준다.

(3) 응용 프로그램은 매니저에게 방안의 온도에 대한 정보를 받을 수 있도록 등록한다.

(4) 매니저는 응용 프로그램이 등록한 부분을 에이전트에게 전달한다.

(5) 센서는 방안의 온도를 측정하여 에이전트에게 보낸다.

(6) 에이전트는 매니저가 설정한 조건에 맞는 값을 매니저에게 전달한다.

(7) 매니저는 에이전트에서 받은 정보를 응용 프로그램에게 전달한다.

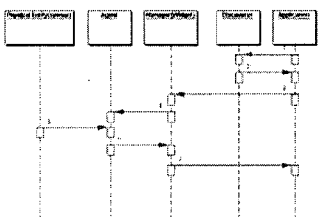

그림 3. Get 메커니즘

4. 구현

본 시스템에서 사용되는 응용 프로그램은 주거 환경 시스템 중에서 온도에 대한 정보를 알려주는 것이다. 이때 사용되는 센서는 그림 6과 같이 소프트웨어로 구현된 온도 센서를 이용하였다. 그리고 에이전트 개발에 있어서 효율성을 극대화하기 위해서 AdventNet사의 Agent Toolkit Java Edition 5를 활용하였다.

응용 프로그램은 사용자가 원하는 값을 갖기 위해서 그림 6과 같은 센서에서 그림 4로 값이 전달된다. 그리고 임계값을 특정 온도값(15℃)으로 정했을 경우에 상황 정보가 임계값 이상이면 그림 5와 같이 에이전트가 매니저에게 Trap 메지지를 통해 알려준다.

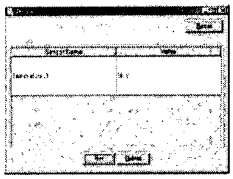

그림 4. 응용 프로그램

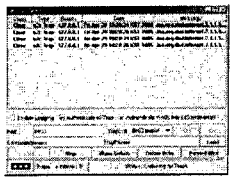

그림 5. Trap 메시지

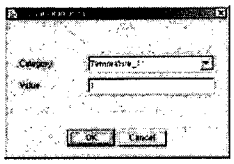

그림 6. 소프트웨어로 구현된 온도 센서

5. 결론

센서와 미들웨어간의 통신에서 기존의 시스템들은 데이터와 계산 기능을 한 컴포넌트에 둠으로써 해당 컴포넌트의 재사용을 불가능하게 했다. 또한 개발자는 센서와 미들웨어에서 제공하는 API에 기초하여 컴포넌트를 만들었다. 이 두 가지 문제를 해결하기 위해서 본 논문에서는 SNMP메커니즘을 도입하였다.

SNMP를 구성하는 요소 중에 데이터를 저장하는 부분과 데이터를 처리하는 부분이 분리가 되어 있다. 즉 SNMP는 MIB과 에이전트가 존재하여 두 부분이 분리가 가능하다. 또한 SNMP는 오랜 세월 동안 정립이 잘되어 있고 많이 사용하기 때문에 SNMP를 지원하는 라이브러리나 툴킷을 제공하는 벤더들이 많고, 이러한 라이브러리나 툴킷을 사용할 수 있는 장점이 있다. 이렇듯 SNMP를 이용함으로써 컴포넌트의 재사용이 가능하게 되고, 컴포넌트를 개발하는 개발자는 좀더 빠른 시간에 통신을 담당하는 컴포넌트를 개발할 수 있다.

참고문헌

[1] Hiroyuki NAKAMURA, "A Development Scenario for Ubiquitous Networks: Viewed from the Evolution of IT Paradigms", NRI Papers,Nomura Research Institute, No. 26, May 1, 2001

[2] Anind K. Dey, "Providing Architectural Support for Building context-Aware Application", Georgia Institute of Technology, pp.44, Nov. 2000

[3] Giles J. Nelson, "Context-Aware and Location System", Clare College,

[4] Gaia, http://gaia.cs.uiuc.edu/index.html

[5] Oxygen project, http://oxygen.lcs.mit.edu

[6] William Stallings, *The Practical Guide to Network-Management Standards*, ADDISON-WESLEY PUBLISHING COMPANY, pp.73, Oct. 1993

[7] 조완수저, *UML 객체지향 분석설계*, 홍릉과학출판사, pp.110, Apr. 15. 2000