EPCglobal 리더 프로토콜을 통한 센서장치 접근을 위한 리더 에뮬레이션 시스템

(Reader Emulation System for Accessing Sensor Device Through EPCglobal Reader Protocol)

최 승 혁 † 김 태 용 † 권 오 흠 †† 송 하 주 ††
(Seung-Hyuk Choi) (Tae-Yong Kim) (Oh-Heum Kwon) (Ha-Joo Song)

요 약 최근의 RFID 응용은 태그를 이용한 사물의 식별 이외에 빛, 온도, 습도와 같은 센서 장치를 함께 활용하는 것이 일반적인 추세이다. RFID 태그의 정보는 리더를 통해 이벤트 처리 방식으로 사용되는 반면, 센서 장치는 장치 의존적인 함수호출 방식으로 접근된다. 따라서 RFID 응용개발자의 입장에서는 두 가지 데이터 처리 방식을 혼용하여 프로그램을 개발해야 하고 센서장치에 의존적인 함수 인터페이스들을 사용해야 한다. 이에 본 논문에서는 센서 장치를 센서 태그 형식으로 사용할 수 있게 하는 센서리더 에뮬레이터를 제안한다. 센서리더 에뮬레이터를 사용하면 단일한 방식으로 RFID 태그와 센서 장치를 접근할 수 있어 RFID 응용프로그램을 효과적으로 개발할 수 있다. 또한 태그의 위치가 고정된 응용일 경우에는 고가의 센서태그와 센서리더 대신 사용될 수 있으므로 저비용으로 동일한 기능을 제공할 수 있다.

키워드 : 무선인식, 리더프로토콜, 센서리더, 센서태그, 리더에뮬레이션

Abstract RFID applications use tags to identify objects, but recent applications tend to include diverse sensor devices such as light, temperature, and humidity sensors as well. RFID tag information is usually processed via the event driven model. However sensor devices are usually accessed via the functional call model. Therefore application developers have to deal with mixed data access models and device dependent interface functions. In this paper, we propose a sensor reader emulator that provides a consistent access interface to sensor devices regardless of the types of devices. SRE provides a more efficient way of developing RFID applications by providing a single application programmer's view to RFID tags and sensor devices. In applications where tags are fixed to a place, SRE can replace expensive sensor tags and sensor readers with inexpensive sensor devices reducing the total cost while providing the same functionality.

Key words : RFID, Reader Protocol, Sensor Reader, Sensor Tag, Reader Emulation

1. 서 론

RFID 기술의 응용범위가 초기에는 무선식별이라는 응용분야에 집중되었으나 최근에는 온도, 습도, 압력 등과 같은 각종 센싱(sensing) 정보를 혼용한 응용이 확대되고 있다[1-3]. 이러한 복합적인 응용에서는 센서 RFID 리더(이하 '리더'로 표기)를 통해 접근되는 식별위주의 태그와 센싱 기능을 갖춘 센서 태그가 함께 사용된다. 또한 일반 센서 장치(sensor device)를 함께 적용하는 경우도 흔히 발생한다.

그림 1은 이러한 복합적인 RFID 응용에서 태그로부터 리더, 그리고 미들웨어에 이르기까지의 시스템 구조를 나타낸 것이다[4-6]. 여기서 RFID 응용 시스템의 핵심이라 할 수 있는 RFID 미들웨어[7]는 RFID 리더를

· 이 논문은 2009년 교육과학기술부로부터 지원받아 수행된연구임(지역거점연구단육성사업/차세대물류IT기술연구사업단)

† 학생회원 : 부경대학교 정보공학과
port30@hanmail.net
ktyuzz@gmail.com
†† 정 회 원 : 부경대학교 IT융합학과 교수
ohkwn@pknu.ac.kr
hajoosong@pknu.ac.kr

논문접수 : 2010년 1월 12일
심사완료 : 2010년 5월 14일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제16권 제8호(2010.8)

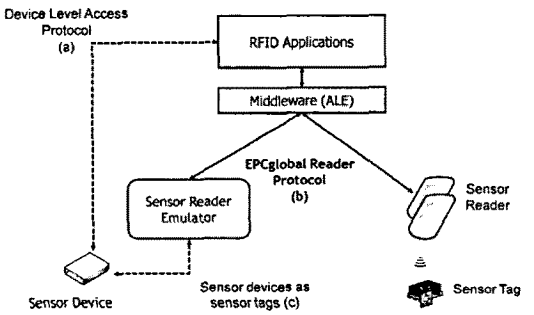

그림 1 RFID 태그와 센서 장치를 혼용한 응용의 예

통해 보내는 각종 센싱 정보를 이벤트(Event) 형식으로 입력 받아 내부적으로 처리한 후 리더프로토콜(Reader Protocol)을 통해 상위의 응용프로그램으로 전달한다(그림 1의 (b) 경로). 반면 일반 센서 장치는 응용 프로그램의 수준에서 디바이스 드라이버에 대한 함수호출 형식으로 접근하게 된다(그림 1의 (a)). 따라서 개발자의 입장에서는 전달 경로와 전달 방식이 다른 센싱 정보를 동시에 처리해야 하는 부담이 따른다.

본 논문에서는 이러한 비효율을 없애기 위해서 그림 1의 (c)에서 보인 것처럼 센서 장치를 센서태그처럼 접근할 수 있게 하는 센서리더 에뮬레이터(Sensor Reader Emulator, SRE)를 제안한다. 그림 2는 SRE의 기능을 센서리더와 비교하여 개념적으로 나타낸 것이다. 센서리더의 경우에는 무선인식 기술을 이용하여 센서태그들로부터 태그식별자인 EPC(Electronic Product Code)와 센싱값을 전달받고, 이것을 이벤트 형식으로 변환한 다음 호스트 컴퓨터에서 실행되는 미들웨어에 전달한다. 전달되는 메시지 포맷은 제품별로 다를 수 있으나 EPCglobal의 표준을 지원하는 리더의 경우에는 EPCglobal

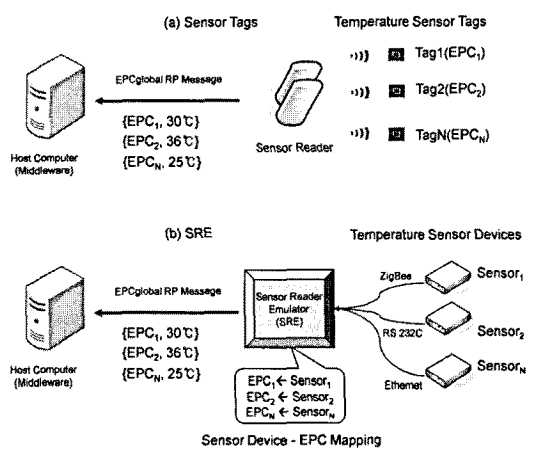

그림 2 SRE 개념도

Reader Protocol 또는 LLRP 프로토콜에 따라 메시지를 전송한다.

SRE의 경우에는 센서장치가 지원하는 통신 프로토콜(ZigBee, RS232C, Ethernet, TCP/IP 등)을 통해 센싱 값만을 전달 받는다. 그리고 센서 장치 별로 미리 부여된 EPC를 검색하여 센서태그와 동일한 형식의 이벤트를 만든 다음 호스트 측으로 전달한다. 따라서 호스트 측에서는 센서리더와 SRE에서 전달되는 메시지를 구분하지 않고 RP 메시지 형식으로 동일하게 사용할 수 있다. SRE를 사용하여 센서장치를 이용하면 다음과 같은 이점이 있다.

- 개발자 입장에서는 RFID 리더와 센서 장치를 구분하지 않고 리더형식으로 모든 장치를 접근할 수 있다.
- 기존의 센서 장치를 센서태그로 대체하거나 장치에 변경을 가하지 않고서도 RFID 미들웨어와 연동할 수 있다.
- 각 센서 장치에 대해 EPC를 부여하여 전역적으로 식별할 수(globally identifiable) 있다.
- 응용 프로그램이 센서의 인터페이스를 직접 호출하여 사용하지 않는다. 따라서 센서 장치와 응용 프로그램 간의 의존성을 감소시킬 수 있다. 센서 장치 측에서 발생할 수 있는 여러 가지 변경 사항(예를 들어 센서 장치의 추가, 장치의 설정 변경 등)에 맞추어 응용 프로그램을 변경하는 일을 줄여 준다.
- 무선인식을 사용하기 어렵거나 불필요한 환경이라면 첫째, 값 비싼 센서태그 대신 저렴한 센서 장치로 동일한 효과를 누릴 수 있다. 둘째, 무선인식을 사용하는 센서리더 보다 안정적으로 센싱 데이터를 수집할 수 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 RFID 리더 관련 프로토콜 및 관련 연구에 대해 설명한다. 3장에서는 본 논문에서 제안하는 SRE 구조와 동작 방식에 대해 설명하고 4장에서는 실제 센서장치를 적용한 예를 보인다. 5장에서는 결론을 짓도록 한다.

2. 관련 연구

2.1 RFID 리더 관련 표준

RFID 기술에 관한 국제 표준화 단체인 EPCglobal에서는 RFID 리더와 미들웨어(또는 호스트)와의 연동을 위한 통신프로토콜로서 Reader Protocol Standard 1.1 [4](이하 RP)과 Low Level Reader Protocol 1.0.1[8](이하 LLRP)를 제안하고 있다. RP와 LLRP가 제안되기 이전에는 RFID 리더의 제공 업체 또는 제품별로 상이한 통신 프로토콜을 사용하였고 그로 인해 미들웨어 측에서는 각각의 리더마다 별도의 통신 모듈을 갖추고 있어야 했다.

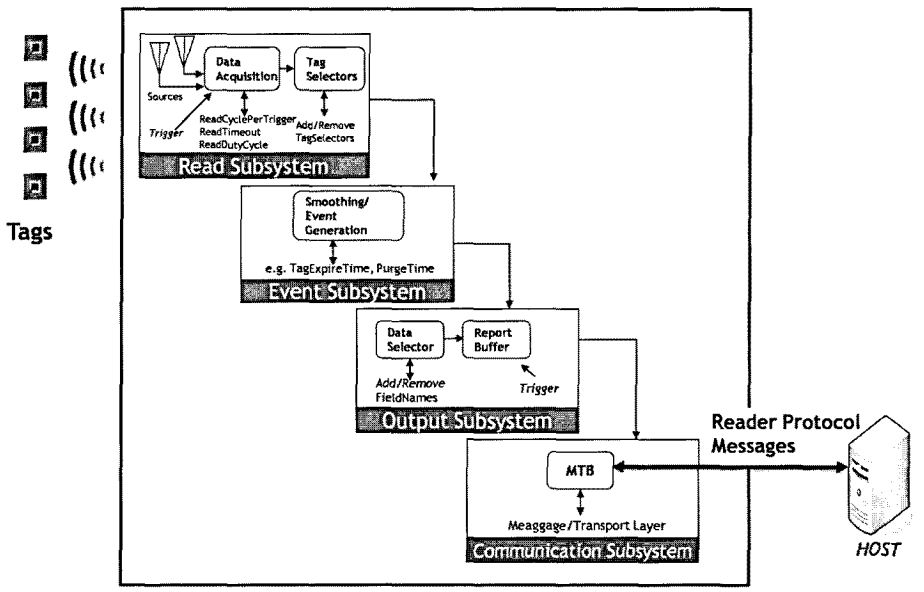

그림 3 EPCglobal Reader Protocol에서 제안하고 있는 리더의 구조

그림 3은 RP 표준에서 제안하는 리더의 구조를 나타낸 것이다. RP를 준수하는 리더는 네 개의 서브 시스템(Sub-system)으로 구성된다. Read 서브시스템은 태그 데이터를 RF 신호를 이용하여 읽어 들이되 원하는 패턴의 EPC[1] 만을 가진 태그를 읽을 수 있다. Event 서브시스템은 불규칙하게 읽혀진 태그들을 정제하여 이벤트 형식으로 가공한다. Output 서브시스템은 이전 단계에서 전달된 각종 태그 관련 정보들 중에서 호스트 측에서 요청된 데이터들만을 선별하여 메시지 큐에 보관하는 기능을 수행한다. Communication 서브시스템은 지정된 통신 프로토콜 및 포맷(Message Transport Binding, MTB)으로 메시지를 호스트(미들웨어) 측으로 전달하는 기능을 수행한다.

EPCglobal에서 제시하고 있는 또 하나의 리더 인터페이스인 LLRP는 RP에 비해 리더의 기능을 더욱 단순화하고 있다. RP의 Event 서브시스템과 같은 고수준의 기능을 제외시키되 RF 동작을 제어하기 위한 저 수준의 인터페이스를 확장시켰다.

2.2 센서장치와 RFID 리더를 통합하기 위한 연구

센서장치를 포함한 센서 네트워크 기술과 RFID 기술은 센서와 통신 기술을 이용하여 실 세계의 이벤트를 자동으로 감지하기 위한 기술이라는 공통의 목적을 가지고 있다. [9-11]는 서로 독립되어 발전되어온 두 기술을 통합하기 위한 기법들을 제안하고 있다. [9]은 일반 가정에서 유비쿼터스 의료를 위한 RFID와 센서네트워크의 통합 모델을 제안하고 있다. 여기에서는 RFID 리더를 직렬연결을 통해 센서노드에 연결하는 접근 방법을 사용하였다. 즉, RFID 태그 정보가 센서노드에 전달되고 이것이 다시 싱크노드(Sink Node)를 거쳐 호스트에 연계되도록 한 것이다. 즉, 제안하는 시스템과는 반대의 접근방법을 사용하였다. 이 방식은 일반적으로 센서노드에 비해 많은 정보를 제공하는 RFID 리더 정보를 충분히 활용하지 못한다. 무선연결을 사용하며 제한된 기능만을 제공하는 센서노드가 유선연결 중심에 상대적으로 강력한 기능을 갖는 RFID리더의 대리자가 되는 것이므로 기능적으로도 비대칭적이라 할 수 있다.

[10]은 두 기술을 통합하기 위한 세가지 방안을 제시하고 있다. 첫 번째 방안은 RFID 리더 직렬연결을 통해 스마트스테이션(Smart Station)에 연결하고 이것이 센서네트워에 연결되는 것이다. 두 번째 안은 축소된 RFID 리더기능을 자체적으로 내장한 스마트노드를 사용하는 것이다. 이러한 두 가지 방안 모두 센서네트워 위주의 통합이므로 [9]에서와 동일한 문제점을 갖고 있다. 특히 제한된 용량의 배터리를 사용하는 것이 일반적인 센서노드에서 RFID 리더를 동작시키기는 어렵다. 세 번째 방안은 센서네트워의 센서노드 기능을 능동태그에 내장하는 방법, 즉, 태그의 기능을 확장하여 센서네트워크를 RFID 영역으로 포함시키는 방법이다. 이것은 센서노드들을 RFID 기술에서 제안하고 있는 클래스 3 및 클래스4 태그와 같이 구성하는 것이다. 따라서 이미

1) Tag Identifier를 나타내며 Electronic Product Code(EPC)를 사용한다.

RFID 기술에서 포함하고 있는 방안이며 장기적으로 적합한 모델이나 현재 이미 사용중인 센서장치 및 센서네트워크에는 적용할 수 없는 방안이다. [11]은 센서네트웍의 각 노드에 EPC를 부여하여 리더를 통해 RFID 태그와 동일한 방식으로 제공하는 EPC 센서네트워크(EPC Sensor Network) 개념을 제안하였다. 센서장치 또는 센서노드를 센서태그 형식으로 제공한다는 의미에서 개념적으로 본 논문의 제안시스템과 동일하다 볼 수 있다. 다만, 해당 논문에서는 제안된 개념을 구현하기 위한 구체적인 구현 방법을 제시하고 있지 않다. 또한 이전 연구들과 마찬가지로 기존의 센서장치 및 센서네트워크를 통합하기 위한 방안을 포함하고 있지 않다.

2.3 센서장치를 포함한 오픈 소스 및 상용 RFID 플랫폼

현존하는 대다수의 오픈 소스 혹은 상용 RFID 미들웨어에서, RFID 리더는 물론 센서장치 및 바코드 리더 장치들을 연결하여 사용할 수 있다. 오픈 소스 미들웨어인 Rifidi[13]의 Edge Server는 센서장치와의 연결을 위해 Sensor Abstraction Layer를 제공한다. 이것은 다양한 리더와 상호작용을 하기 위한 인터페이스로써 플러그인(Plug-in) 방식으로 리더 및 센서장치를 추가할 수 있게 한다[2]. 상용 RFID 미들웨어는 대표적으로 IBM사의 'WebSphere Premises Server'(이하 WebSphere) [3]가 있다. WebSphere는 내부적으로 EPCglobal에서 제시하는 ALE와 EPCIS표준을 준수하고 있으며, 센서장치의 관리 및 데이터 수집을 개방형 플랫폼인 Eclipse Equinox OSGi Framework을 이용한다[14]. WebSphere는 센싱을 할 수 있는 주체가 되는 모든 디바이스(device)에 대해 센서라고 통칭하고 있다. 센서장치를 비롯한 능동형, 수동형 태그 리더들에 의해 수집된 센싱이벤트를 일반화시킨 Generic 이벤트를 통해 상위 레벨로 전송하고, 이를 다시 Business Logic에 맞게 처리하는 레벨에서 의미 있는 데이터를 도출하여 이를 전사적 응용 또는 비즈니스 응용으로 전송하게 되는 것이다. Device Kit Markup Language(DKML)이란 장치 구성에 관련된 설정을 담당하는 XML 마크 업 언어로써, 어댑터 내부에 센서를 위한 공통적인 서비스 지향 인터페이스를 자동으로 스텁(Stub) 형식의 코드로 생성하는 기능이 제공된다[3]. 오라클사의 'Oracle Sensor Edge Server'(이하 오라클서버)는 RFID리더와 프린터, 그리고 기온, 압력 등을 측정하는 일반적인 센서를 통합하는 유연한 드라이버 아키텍처를 제공한다. Device Abstraction Layer에서 각종 센서장치 및 리더들을 위한 인터페이스로 드라이버(Driver)를 제공하고 있으며, 추가적인 센서장치가 발생할 시, 그 장치의 드라이버만 제공되면 장치와 상호작용이 가능해진다. 특히 오라클서버와 플러그인할 수 있는 컴포넌트와 표준 드라이버 인터페이스와 같은 경우 사용자가 직접 구현할 수 있다[15]. 또한 Sensor Edge Server Configuration 인터페이스를 통해 센서 하드웨어에 의존적인 기능을 미들웨어에서 쓸 수 있게 해준다. 센서에서 수집되어 올라오는 정보는 불필요하거나 중복된 정보를 포함하는데, 이를 센서 그룹 단위 즉, 개별적인 센서 혹은 한 개의 논리적인 개체로 인식할 수 있는 기능이 제공된다[16]. 이상에서 살펴본 상용 및 오픈 소스 RFID 미들웨어들은 제안하는 시스템과 유사하게 어댑터 또는 컴포넌트 형식으로 각종 센서 장치들을 추가할 수 있는 확장성을 제공하고 있다. 반면 이들 시스템에서는 각종 이벤트의 처리 및 응용 수준에서의 처리에 있어 RFID 태그와는 별개의 장치로 취급된다. 즉, 각 센서 장치들에 대해 RFID 태그와 같은 방식으로 다루는 것은 불가능하다, 더욱이 각 장치에 대해 EPC를 부여하지 않으므로 장치에 대한 접근 및 정보 공유는 EPCglobal Network 수준으로까지 확장될 수는 없다. 제안하는 시스템에서는 각 센서장치에 고유 EPC를 부여하는 기능을 추가함으로써 응용 수준에서의 공통된 뷰를 제공하고 센서가 제공하는 정보를 EPCglobal Network 수준에서 연계가 가능하도록 하고자 한다.

3. SRE의 설계와 구현

3.1 SRE의 구조

그림 4는 센서 리더 기능을 제공하기 위한 SRE의 구조를 보인 것이다. SRE서버는 다수의 SRE 인스턴스들을 관리하고 실행하는 컨테이너(Container)이다. Java Runtime이 제공하는 관리프레임웍인 JMX를 이용하여 표준화된 관리 인터페이스를 제공할 수 있도록 하였다. 여기에는 새로운 SRE의 추가 및 기존 SRE의 삭제와 SRE의 실행 및 중지 등의 인터페이스를 제공한다. SRE 서버를 실행중인 상태에서 새로운 클래스의 SRE 인스턴스(Instance)의 추가 및 삭제가 가능하도록 하기 위해 M-Let 클래스로더(Class Loader)의 동적 클래스 로딩기능을 사용하였다. SRE서버의 관리는 자바에서 제공하는 JConsole을 이용할 수 있으며 JMX 프레임웍을 지원하는 관리도구는 모두 사용 가능하다. 관리도구는 SRE 인스턴스의 추가 및 삭제 이외에도 EPCglobal RP 표준에서 정의하고 있는 각종 리더 설정관련 명령을 GUI를 포함한다.

연동되는 센서 또는 센서네트웍의 관리는 해당 장치의 고유한 관리도구를 통해 이루어지며 제안하는 시스템의 영역에 포함되지 않는다. 센서네트웍의 경우에는 각 센서노드별로 상이한 EPC를 부여할 수 있으며 그 부여 방법 및 어댑터를 통한 인터페이스는 어댑터 구현

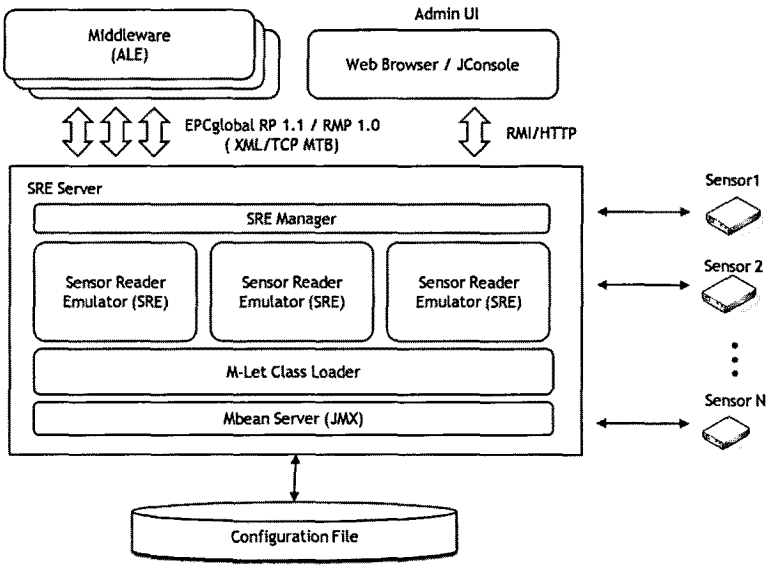

그림 4 SRE 서버 구조

자의 자율에 맡긴다. 센서네트워크를 사용하는 경우에는 단일한 디바이스에 대해 다수의 EPC를 부여할 수 있다.

그림 5는 제안하는 SRE 인스턴스의 구조를 나타낸 것이다. SRE는 개념적으로 RP의 스펙을 구현하기 위한 RP변환부와 센서어댑터관리부라는 두 개의 모듈로 구성된다. 센서어댑터관리부에서 측정된 센싱값은 해당 센서장치에 부여된 TAG ID와 함께 RP 메시지 변환부로 전달된다. RP변환부는 전달 받은 TAG ID와 센싱값을 4단계의 서브시스템을 통해 RP의 이벤트형 메시지로 변환하여 호스트로 전달한다.

센서어댑터관리부는 센서와의 연결을 담당하는 센서어댑터들을 생성, 삭제 및 관리하는 모듈이다. RP변환부의 각 서브시스템은 RP 스펙에서 정의하고 있는 기능을 수행하도록 구현하였다. 이들의 기능 및 내부 구조는 RP 스펙과 동일하므로 상세한 내역은 생략한다.

센서어댑터 관리부는 센서장치들과의 인터페이스를 수행하는 센서어댑터들을 관리하는 부분이다. 센서어댑터는 SRE의 구동 시 자바의 리플렉션(reflection) 기능을 이용하여 동적으로 클래스를 적재(loading)하며, 센서장치 별로 하나의 인스턴스를 할당한다. 센서어댑터와 센서장치는 크게 2가지 방식으로 연결된다. 첫째는 센서어댑터가 운영체제의 디바이스 드라이버를 통해 센서장

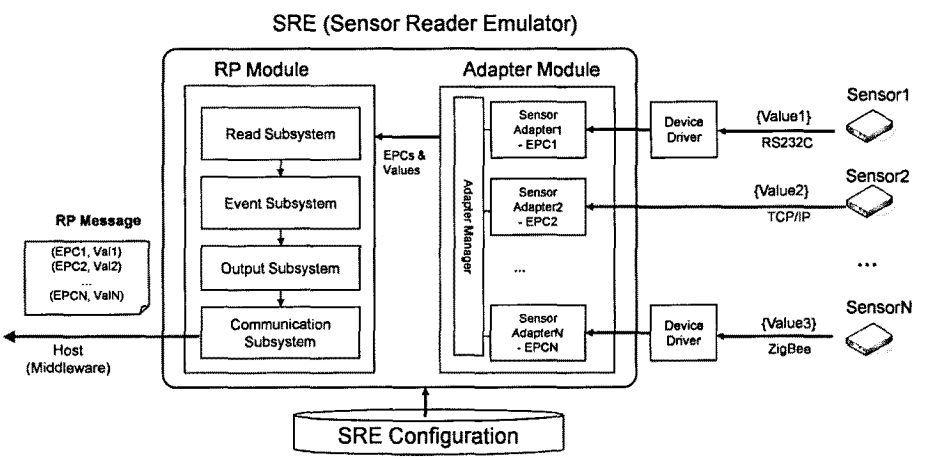

그림 5 SRE 인스턴스의 구조

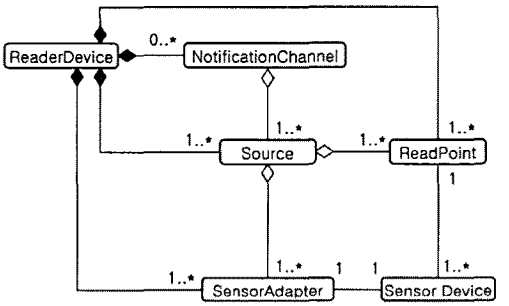

그림 6 주요 클래스들 간의 관계

치와 연결되는 경우이다. 그림 5의 센서1 또는 센서N과 같이 디바이스 드라이버와 센서간에는 자체적인 통신 수단(RS232C를 이용한 직렬통신 또는 ZigBee 등)을 통해 연결되고 센서어댑터는 디바이스 드라이버의 래퍼(wrapper)와 같은 역할을 수행한다. 이 경우에는 센서어댑터는 주로 Java Native Interface(이하 JNI)를 이용하여 작성된다. 두 번째 방법은 디바이스 드라이버를 사용하지 않고 센서어댑터와 센서간에 직접연결 방법이다. 그림 5의 센서2의 경우에는 TCP/IP를 이용하여 센서장치와 연결한 것을 나타낸 것이다. 이 경우에는 센서어댑터를 순수하게 자바의 소켓(socket) 라이브러리 클래스를 이용하여 작성할 수 있으므로 센서어댑터는 자바만으로도 작성될 수 있다.

그림 6은 센서장치, 센서어댑터, 그리고 일부 중요한 RP 객체들간의 연관 관계를 그림으로 표현한 것이다. NotificationChannel, Source, ReadPoint는 RP 표준에서 제시되고 있는 리더 내부 객체들이다. RFID로부터 태그를 읽어 들이고 이벤트를 생성하는 역할을 담당하는 Source 객체에 대해서 센서어댑터(SensorAdapter) 인스턴스는 여러 개 사용될 수 있다. 그러나 하나의 SensorAdapter 인스턴스가 여러 개의 Source에 공유되지는 않는다. ReadPoint는 일반 RFID리더의 경우에는 안테나에 해당하고 태그에 대한 물리적인 위치 정보를 제공하게 된다. SRE에서는 다수의 센서장치들을 묶어(Grouping) 하나의 논리적인 ReadPoint를 할당할 수 있도록 하였다. 즉, 물리적으로 인접한 RFID 태그들이 동일한 안테나에 의해 읽혀지는 것처럼, 인접하게 설치된 센서 장치들에 대해 동일한 ReadPoint를 할당함으로써 물리적 인접성에 대한 정보를 제공하도록 할 수 있다.

3.3 센서어댑터부의 구성

센서어댑터는 센서장치를 센서태그처럼 사용할 수 있게 하는 핵심적인 부분이다. 센서어댑터관리부는 각 센서장치와의 연결을 담당하는 센서어댑터와 센서어댑터들을 관리하기 위한 센서어댑터관리자로 구성된다. 센서어댑터관리자는 센서어댑터들의 리스트를 유지하고 센서어댑터의 추가 및 삭제 기능을 담당한다.

센서어댑터들은 센서 장치들과 인터페이스를 담당하는 부분이다. 이를 위해 센서어댑터는 센서 장치를 직접 접근하거나(그림 4의 센서2) 또는 운영체제에서 제공되는 디바이스 드라이버를 거칠 수도 있다(그림 4의 센서1과 센서N). 센서들과의 물리적인 연결은 RS232C, TCP/IP, Ethernet, ZigBee, Bluetooth, USB 등 다양한 인터페이스를 사용할 수 있는데 이것은 센서 장치의 고유한 특징에 따른 것이며 제안하는 시스템과는 무관하다. 센서장치 별로 별도의 센서어댑터 클래스를 작성해야 하는데 모든 센서어댑터들이 공통적으로 제공해야 하는 기능을 CustomAdapter라는 이름의 추상클래스로 제공한다. 표 1과 표 2는 CustomAdapter 클래스의 데이터필드와 인터페이스를 간략하게 정리한 것이다. 실제 적용되는 센서어댑터는 공통 기능 이외에 필요한 기능을 상속하는 과정에서 추가할 수 있다.

표 1에서 cacheValue는 센서장치에서 읽어 들인 센싱값을 의미하고, readCycle은 센서장치에서 값을 읽는 주기를 나타낸 것이다. 장치가 제공하는 센싱값을 읽어 들이기 위해 소요되는 최소의 시간과 RP 변환부에서 읽어가는 주기가 일치하지 않을 수 있기 때문이다. 예를 들어 장치에서 센싱값을 읽기 위한 최소 시간이 5초 소

표 1 센서어댑터 객체의 데이터 필드들

데이터 필드	용도
sensorName	대응되는 센서의 이름
sensorEPC	대응되는 센서에 부여된 EPC, Tag URI 포맷을 사용한다. 다음은 SGTIN-64비트를 사용한 Tag URI 포맷의 예이다. urn:epc:tag:sgtin-64:3.0652642.800031.400
venderFieldName	센싱값이 전달되는 필드 이름. EPC RP 표준의 벤더필드(vendor field) 영역을 사용한다.
cacheValue	캐쉬된 센싱값
readCycle	센서 장치로부터 센싱값을 읽는 주기(밀리초 단위)
readPoint	센서 장치에 부여될 Read Point 이름

표 2 CustomAdapter 클래스의 인터페이스

메소드	의미
initialize	센서 장치를 초기화한다
Finalize	센서 장치를 종료한다
Start	장치로부터 센싱값을 주기적으로 읽기 시작한다
Stop	주기적으로 읽기를 멈춘다
getEPC	적용되는 센서장치에 부여된 EPC를 알려준다.
setReadCycle	readCycle을 지정한다. 0을 지정하면 캐쉬를 사용하지 않음
getSensorValue	센서장치의 센싱값을 알려준다

모되는데 반해 RP 변환부에서 센싱값을 요청하는 간격이 2초로 설정될 수도 있기 때문이다. 따라서 이 경우에는 RP 변환부로 전달되는 값이 요청 간격보다 이전의 값이 된다. 이 기능은 센싱값의 변동이 급격하지 않은 응용에서 장치의 응답특성이 서로 다른 센서 장치를 혼용해서 사용하는 경우에 유용하게 사용할 수 있다. 만약 센싱값의 변동이 큰 경우라면 캐쉬 기능을 사용하지 않고(readCyle = 0으로 지정) RP 변환부에서 요청될 때에만 값을 읽는다.

3.2 데이터 필드값의 생성

EPCglobal RP 스펙에서는 태그에 대한 읽기 과정에서 태그ID 이외에 다양한 데이터를 추가적으로 제공한다. 표 3은 RP 표준에서 제안하고 있는 태그의 데이터 필드들이며 이것을 SRE에서 어떻게 지원하는지를 나타낸 것이다. 데이터 필드들은 센서어댑터 관리부와 RP 변환부를 거치면서 각각의 태그ID에 대응하는 값이 추가된다.

표준 스펙에서와 같이 호스트 쪽에서 RP 메시지를 이용하여 원하는 데이터필드만을 선택적으로 보고하도록 설정할 수 있다. SRE에서는 센싱값의 전달을 위해 표 3에서 나타내어진 벤더 필드를 사용한다. 필드 명은 SRE 관리 프로그램에 의해 주어지며 RP 표준에 따르기 위해 관리자(사용자)가 부여한 이름에 'lit:'접두어를 추가하여 사용된다. 예를 들어 온도 센서장치를 사용하여 측정된 온도 값을 전달하기 위해서는 SRE 관리프로그램에서 해당 센서어댑터를 등록하면서 센싱값이 전달될 필드 명을 'temp'로 지정하면 실제로 RP 메시지에서는 'lit:temp'라는 이름으로 전달된다.

3.3 센싱값의 전달

그림 7은 RP 변환부와 센서어댑터관리부간의 센싱정보 전달과정을 나타낸 것이다. RP 변환부에서 TAG ID를 읽어 오는 기능은 Source 객체가 담당하며, 그 시작은 RP 변환부의 Read Trigger에 의해 구동된다. 다음은 이 과정을 상세히 나타낸 것이다.

① ReadTrigger는 Source 객체에 Read를 수행할 것을 요구한다.

② Source 객체는 센서 어댑터 관리자에게 등록되어 있는 모든 센서어댑터에 값을 가져올 것을 요구한다(getActivateTagList).

③ 센서 어댑터 관리자는 현재 센서가 활성(enabled) 상태에 있는지 확인한다.

④ 활성상태에 있다면 센서 어댑터에 값을 요청하는 메시지를 비동기적으로 전송하고, 그 값이 전달될 객체(Future)를 센서어댑터별로 저장한다(update).

⑤ 센서어댑터는 센서장치에 센싱값를 요구하기 전에, 센서 API 의존적인 정보를 가져온다.

⑥ 센서어댑터는 자신의 내부에 캐쉬되어(caching, 그림 7의 6-1과 7-1) 있는 센싱값을 전달하거나, 센서장치에 직접 요청하여 센싱값을 가져 온다(그림 7의 6~7).

표 3 RP 표준에서 제공하는 태그의 데이터 필드 명들

필드 이름	의미	SRE에서의 지원
eventTriggers	태그 읽기를 유발한 트리거들의 이름	좌동
eventType	이벤트 이름	좌동
eventTimeTick	이벤트가 발생한 시각(틱단위)	좌동
eventTimeUTC	이벤트가 발생한 시각(UTC)	좌동
readerEPC	리더의 EPC	SRE 관리 프로그램을 통해 설정
readerHandle	리더 Handle 이름	좌동
readerName	리더의 이름	좌동
readerRole	리더 역할 이름	좌동
readerNowTick	Notification을 보내는 시점의 리더의 시간(틱단위)	좌동
readerNowUTC	Notification을 보내는 시점의 리더의 시각(UTC)	좌동
tagType	태그의 종류. 예) ISO, Gen2	'Gen2'로 설정
tagID	바이너리 포맷의 Tag ID	센서장치에 부여된 EPC의 바이너리
tagIDasPureURI	Pure Identifier 포맷의 Tag ID	센서장치에 부여된 EPC의 Pure Identifier 포맷
tagIDasTagURI	Tag URI 포맷의 Tag ID	센서장치에 부여된 EPC의 Tag URI 포맷
sourceName	태그가 읽힌 Source 객체의 이름	좌동
sourceFrequency	Source의 읽기 주기	좌동
sourceProtocol	Source 객체의 Air RF 프로토콜	'SRE'
notifyChannelName	Notification 채널 명	Notification 채널 명
notifyTriggerName	Notification을 유발한 Trigger의 이름	좌동
<VenderName>:<Any>	리더 밴더에서 사용 가능한 필드 명	SRE 관리 프로그램을 통해 센서장치 별로 설정된 필드 이름을 사용

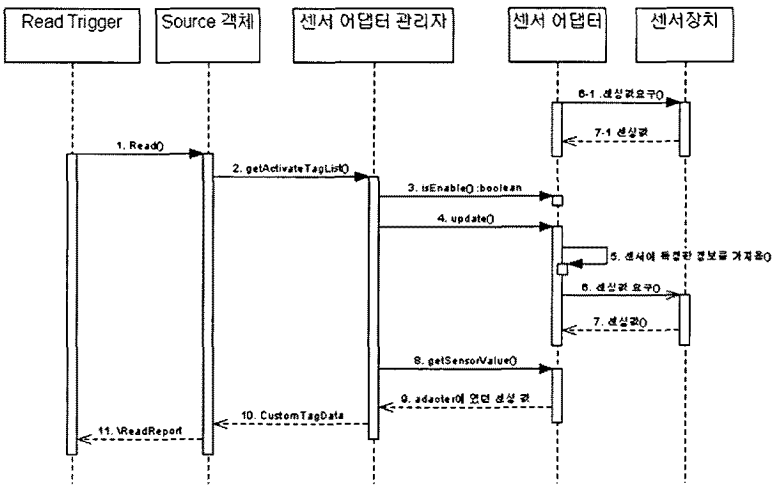

그림 7 센서어댑터에서 RP변환부로 센싱정보가 전달되는 과정

⑦ 이후, 센서 어댑터 관리자에서 각 센서어댑터별로 getSensorValue를 호출하며 Future 객체를 통해 센싱값을 전달 받는다. 이때, 센서어댑터에 미리 설정된 EPC와 ReadPoint가 함께 전달된다.

3.4 이벤트 생성

일반적으로 RFID 리더의 읽기는 다소 불안정하다. 즉 리더의 읽기가능 영역 내에 존재하더라도 태그가 읽혀지지 않는 경우가 있는 반면, 의도하지 않던 영역에 존재하는 태그가 읽히는 경우도 허다하다. EPCglobal RP 표준에서는 이러한 불안정성을 극복하고 동일한 태그가 지나치게 자주 계속해서 보고되는 것은 막기 위해 이벤트 서브시스템(Event Subsystem)을 도입하였다. 이것은 새로 읽혀진 태그와 사라진 태그들은 즉각 보고하되 계속해서 읽혀지는 태그들은 간헐적으로 보고하도록 하는 기법이다.

SRE에서는 센서와의 접속을 통해 센싱값을 성공적으로 읽어 올 수 있는 경우에는 태그가 읽힌 것으로 그렇지 않은 경우에는 태그를 읽지 못한 것으로 모델링한다. 따라서 통신 에러, 센서 장치의 이상 등으로 인해 센서 어댑터가 readCycleTime 내에 센싱값을 전달받지 못하면 태그가 읽히지 않은 것으로 간주된다. 센서장치들의 경우 Ethernet, TCP/IP, RS232C, Bluetooth, ZigBee 등과 같은 일반적인 통신 프로토콜을 통해 센서어댑터 연결되므로 RFID 태그 읽기에 비해 안정적인 통신이 이루어진다. 따라서 RFID 태그에 비해 이벤트는 더 적게 발생된다.

4. SRE의 적용 사례

본 논문에서 제안하는 센서 리더 에뮬레이터의 실용성을

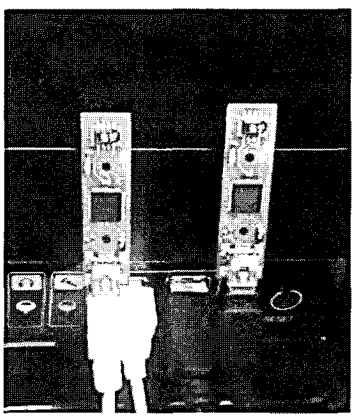

그림 8 Oak Sensor Devices

확인하기 위해 Oak Sensor Device를 적용해 보았다(그림 8 참조). 이 장치는 온도와 습도를 측정할 수 있다.

표 4는 Oak Senor를 위한 센서어댑터 클래스에서 주요한 메소드를 정리한 것이다. Oak Sensor Adapter는 디바이스 드라이버와의 인터페이스를 위해 Java Native Interface(JNI)를 이용하였다. 다수의 센서장치 중에서 접근하는 센서를 식별하기 위해서는 센서장치별로 제조사에 의해 부여된 아이디를 사용하였다.

OakSensor의 디바이스 드라이버는 접근하는 포인터가 있어야 하는데, 자바에 경우 따로 포인터를 저장하는 데이터 타입이 존재하지 않아, long 타입의 변수에 그 값을 저장한다. 그리고 센싱값을 측정할 때, 이 변수를 디바이스 드라이버에 접근할 수 있는 포인터로 JNI에서 캐스팅(casting)하여 디바이스 드라이버를 식별할 수 있도록 하였다. 디바이스 드라이버에서 전달되는 데이터는

표 4 Oak Sensor Adapter의 함수

함 수 원 형	설명
Shutdown	OakSensor와 연결된 디바이스 드라이버를 메모리상에서 지움.
createSensorAdapter	OakSensor와 연결하는 디바이스 드라이버를 메모리상에 만듦..
getChannel	OakSensor의 경우 온도와 습도를 물리적으로 한 장치에서 읽는데, 이것을 구분하는 정수를 반환함.
getId()	OakSensor와 실제 연결할 때 필요한 아이디를 반환함.
getSensorValue()	OakSensor가 센싱한 값을 가져옴.

double형의 Wrapper인 Double형을 JNI 코드 내에서 생성하여 반환한다. 이는 JNI 코드 내부에서 문제가 발생한 경우 NULL 참조변수를 통해 내부에서 에러가 발생하였다는 것을 알려줄 수 있도록 한다. 이러한 경우에는 장치를 비활성화시켜 더 이상 작동하지 않도록 하였다

SREReaderDevice에서 현재 활성화된 센서어댑터 중 캐싱하지 않는 어댑터에게 센싱 값을 요구할 수 있으며, 이 때 각각의 센서 장치를 순차적으로 읽게 한다면 총 센서의 측정 시간은 각 센서들의 지연시간의 합이 된다. 따라서 먼저 읽혀지는 센서 장치의 경우에는 비교적 오래된 값을 보고하게 될 것이다. 이와 같은 문제를 방지하기 위해 각 SensorAdapter에 하나의 쓰레드를 각각 할당하여, 병렬적으로 센싱 값을 읽도록 하였다. 그리고, 이 과정에서 쓰레드(thread)를 생성하고 제거하는 작업이 빈번하게 발생하게 된다. 따라서 SREReaderDevice에서 쓰레드풀(thread pool)을 두어 여기에서 사용할 쓰레드를 가져와 센서어댑터마다 병렬적으로 센싱작업을 하고 난 다음 쓰레드풀에 이를 다시 반환하도록 구현하여 다수의 쓰레드가 생성하고 제거되는 것에 따른 오버헤드를 줄였다.

그림 9는 Oak 센서장치를 위해 사용된 센서리더에뮬레이터의 클래스 다이어그램이다. 이 SensorReaderDevice는 여러 개의 센서 어댑터들을 관리하는 센서어댑터 관리자로서, ReaderProtocol상의 Source와 ReaderDevice가 연관되어, 실제 Reader Protocol 1.1을 지원하는 리더처럼 보이게 한다. CustomTagInfo는 실제 리더에 읽힌 태그와 그 부가적인 정보를 담기 위한 클래스이며, 이 CustomTagInfo는 센서 리더 에뮬레이터에 경우, 센서에 사용자가 설정한 EPC와 Readpoint, 그리고 센싱 값 등을 담아 CustomTagData에 유지한다. CustomTagData는 TagSelector, DataSelector, EventSub System을 거쳐 ReadReport로 바뀌게 되고, MTB 단계를 거치면 상위에서 전달 받을 수 있는 메시지형태가 된다. 현재 SRE에서는 XML 포맷을 지원하며, 그림 10은 Sensor Reader Emulator에서 센싱된 값을 보낼 때 생성되는 XML 메시지의 예이다.

그림 10은 실제 적용되어 호스트로 보내질 RP 1.1을 준수하는 SRE의 메시지이다. 이 메시지에서 tagID는 센서 장치에 부여된 EPC의 바이너리이며, 그 외의 tagIDAsPureURI 및 tagIDAsTagURI는 센서 장치에 부여된 EPC의 바이너리를 해당 포맷에 맞게 변환한 것

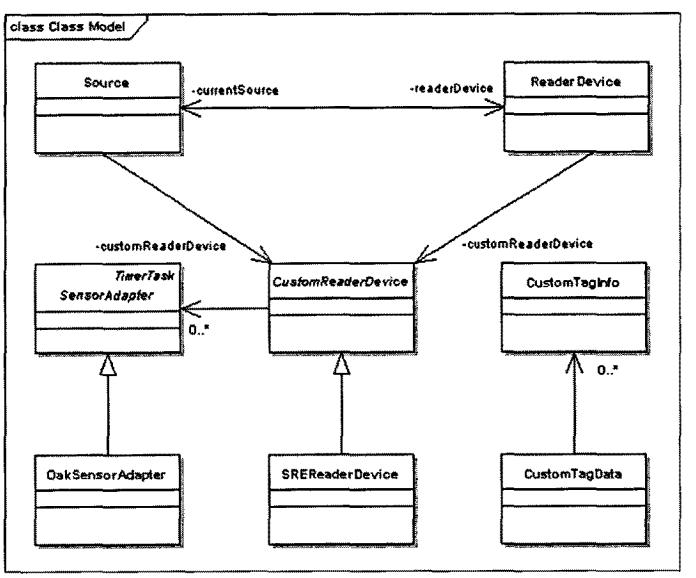

그림 9 SensorReaderEmulator의 구현된 클래스다이어그램

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<notification xmlns=" urn:epcglobal:xsd:1"  xmlns:lit=" http://www.rclit.com" >
  <id>1</id>
  <reader>
    <readerEPC>3500000000AABBCCDDEEFF5</readerEPC>
    <readerName>SRE</readerName>
    <readerRole/>
  </reader>
  <notifyTriggerName/>
  <notifyChangeName/>
  <readReport>
    <sourceReport>
      <sourceInfo>
        <sourceName>s0</sourceName>
      </sourceInfo>
      <tag>
        <tagID>3500B2E0200029D00081D39F</tagID>
        <tagIDAsPureURI>urn:epc:id:gid:732674.669.8508319</tagIDAsPureURI>
        <tagIDAsTagURI>urn:epc:tag:gid-96:732674.669.8508319</tagIDAsTagURI>
        <tagEvent>
          <eventType>evObservered</eventType>
          <time>
            <eventTimeUTC>2009-12-24T05:04:06.328KST</eventTimeUTC>
          </time>
        </tagEvent>
        <other>
          <lit:temp>300.35999328643084</lit:temp>
          <lit:readPoint>ANT2</lit:readPoint>
        </other>
      </tag>
    </sourceReport>
  </readReport>
</notification>
```

그림 10 SRE를 통해 생성된 XML 메시지

이다. RP1.1에선 other 태그 내에 벤더필드를 포함하도록 규정하고 있다. Oak 센서 어댑터의 경우에는 lit:temp란 벤더필드에 센싱값을 넣어 전달하도록 하였다. 데이터 셀렉터에 위의 벤더필드가 포함되어 있지 않으면, 센싱값이 제외된다. 이 데이터 셀렉터에서 센싱값이 제외되지 않게 SRE관리 프로그램에서 벤더필드를 Oak 센서어댑터에서 추가한 같은 이름으로 추가해야 한다. 그리고 노티피케이션 메시지로만 센서에 부여된 Read-Point를 알려고 할 경우엔, SRE관리 프로그램에서 lit:readPoint를 추가한다. 그렇게 함으로써, 센싱값을 뜻하는 lit:temp와 lit:readPoint는 other태그 밑의 자식태그가 되어 호스트 측으로 전달된다.

SRE Server는 J2SE 5.0부터 자바 기반의 어플리케이션을 모니터링 하고 관리하는 기능을 제공하기 위하여 JRE에 포함된 JMX를 기반으로 개발하였고, SRE 내부의 주요 구성요소들은 JMX의 MBean으로 구성하였다. 따라서 JMX를 지원하는 각종 관리도구를 그대로 사용할 수 있다. 그림 11은 J2SDK에서 기본으로 제공되는 관리도구인 JConsole을 사용하여 SRE 서버의 동작 상태를 모니터링하는 화면으로써 메모리 사용량, CPU 사용비율, 쓰레드 사용량 등을 포함한 각종 동작 상태를 감시할 수 있다.

그림 12는 JConsole에서 MBean객체로 표시되는 RP 객체들을 관리하는 모습을 보여준다. RPManager 객체

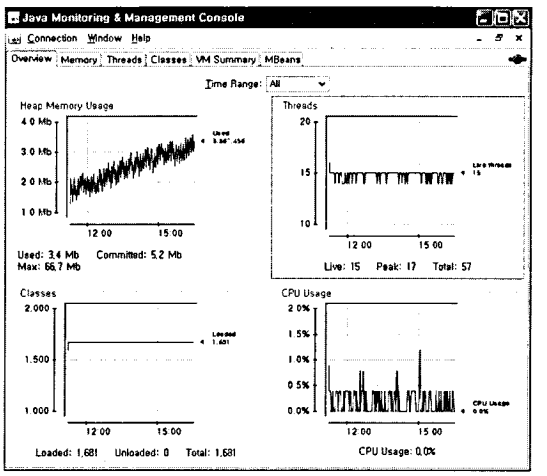

그림 11 JConsole을 이용한 SRE 서버 모니터링

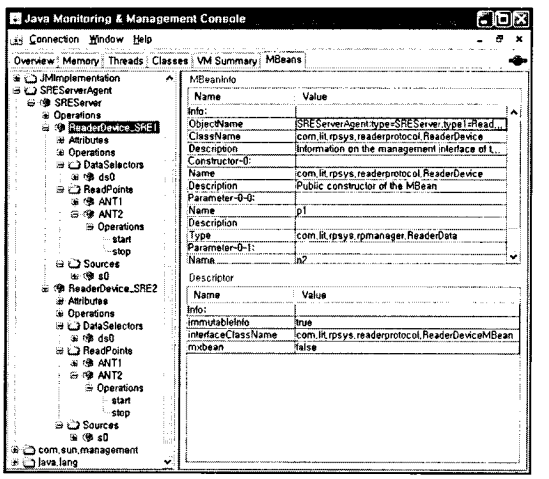

그림 12 JConsole을 사용한 RP 객체들의 관리

에 다수의 SRE Source, Trigger, Antenna 등 각종 RP 객체들의 상태를 확인하고 필요에 따라 객체들에 대해 적용할 수 있는 각종 오퍼레이션을 수행할 수 있다. 여기에서는 두 개의 SRE 인스턴스를 사용하고 있는 모습을 보여준다. 화면의 객체들이 보여주듯이 RP 객체들이 모두 소프트웨어적으로 지원되므로 일반 리더를 사용하는 것과 동일하게 센서장치를 센서태그와 같은 방식으로 접근할 수 있게 된다.

5. 결 론

Sensor Reader Emulator는 센서장치에 대한 새로운 접근 방식이다. RFID 리더와 센서장치를 동시에 사용하는 경우 응용프로그램에서는 리더에 접근하기 위해서 Reader Protocol을 사용하고 센서장치에 접근방식은 다

바이스 드라이버를 이용한 함수 호출 방식을 사용한다. 개발자의 입장에서 이것은 복잡도를 증가시키는 요인으로 볼 수 있다. 이에 본 논문에서는 센서장치를 센서태그 형식으로 사용할 수 있게 하는 SRE 시스템을 제안한다. 이 프로그램을 사용하면 RFID 리더와 센서 장치들을 단일한 인터페이스를 사용하여 접근할 수 있음으로써 여러 가지 장점을 제공하게 된다. 현재는 EPCglobal RP 1.1 표준에 따라서 센서 장치를 접근할 수 있으며 추후에는 LLRP 1.01 프로토콜도 추가적으로 지원할 예정이다. 또한 태그 필터링에 있어서도 EPC에 기반한 필터링 이외에 센싱값에 기반한 필터링 기능의 추가를 위한 연구가 진행될 것이다.

참 고 문 헌

[1] Christian Floerkemeier, Matthias Lampe, Christof Roduner, "Facilitating RFID Development with the Accada Prototyping Platform," PerCom Workshops, pp.495-500, 2007.

[2] Rifidi Edge Server, "Edge Server Architecture," http://wiki.rifidi.org/index.php/Edge_Server_Architecture#Esper

[3] IBM Corporation, "WebSphere Sensor Events," http://www-01.ibm.com/software/integration/sensor-events/index.html

[4] EPCglobal, Reader Protocol Standard Version 1.1 Ratified Standard, 2006.

[5] EPCglobal. The Application Level Events (ALE) Specification, Version 1.1.1, 2009.

[6] Joseph E. Hoag, Craig W. Thompson, "Architecting RFID Middleware," IEEE Internet Computing vol.10, No.5, pp.88-92, 2006.

[7] Sanjay E. Sarma, "Integrating RFID," ACM Queue vol.2, No.7, pp.50-57, 2004.

[8] EPCglobal, Low Level Reader Protocol (LLRP), Version 1.0.1, 2007.

[9] Loc Ho, Melody Moh, Zachary Walker, Takeo Hamada, Ching-Fong Su, "A prototype on RFID and sensor networks for elder healthcare: progress report," Proceedings of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis, August 2005.

[10] Lei Zhang, Zhi Wang, "Integration of RFID into Wireless Sensor Networks: Architectures," Opportunities and Challenging Problems, Grid and Cooperative Computing Workshops. GCCW '06. Fifth International Conference, 2006.

[11] Jongwoo Sung, Tomas L. Sanchez, Daeyoung Kim, "EPC Sensor Network for RFID and USN Integrated Infrastructure," to appear in Fifth Annual IEEE International Conference on Pervasive Computing and Communications(Percom), 2007.

[12] Claudio E. Palazzi, Alessandro Ceriali, Marco Dal Monte, "RFID Emulation in Rifidi Environment," Dipartimento di Matematica Purae Applicata Università degli Studi di Padova, pp.4-13, 2009.

[13] Rifidi Edge Server, "Edge Server Architecture," http://wiki.rifidi.org/index.php/Edge_Server_Architecture#Esper

[14] Simeon Furrer, Wolfgang Schott, Hong Linh Truong, and Beat Weiss, "The IBM Wireless Sensor Networking Testbed," IBM Research GmbH Zurich Research Laboratory, pp.3-4, 2005.

[15] Sebastian Frischbier, "Existing RFID Infrastructures Comparison And Evaluation," Databases and Distributed Systems Group Dept. of Computer Science, pp.25-28, 2006.

[16] Oracle Corporation, "Oracle Sensor Edge Server," http://www.oracle.com/technology/products/sensor_edge_server/collateral/Oracle_SES_1013_Technical_Presentation.pdf

최 승 혁

2008년 부경대학교 컴퓨터멀티미디어전공 졸업(공학사). 2008년~현재 부경대학교 정보공학과 석사과정 재학중. 관심분야는 객체지향 프로그래밍, 유비쿼터스 센서네트워크

김 태 용

2009년 부경대학교 컴퓨터멀티미디어전공 졸업(공학사). 2009년~현재 부경대학교 정보공학과 석사과정 재학중. 관심분야는 소프트웨어 공학, 데이터베이스, 유비쿼터스 센서네트워크

권 오 흠

1988년 서울대학교 컴퓨터공학과 졸업(공학사). 1991년 KAIST 전산학과(공학석사). 1996년 KAIST 전산학과(공학박사). 현재 부경대학교 IT융합응용공학과 교수. 관심분야는 알고리즘 설계 및 분석, 분산 컴퓨팅, 유비쿼터스 센서네트워크

송 하 주

1993년 서울대학교 컴퓨터공학과 졸업(공학사). 1995년 서울대학교 컴퓨터공학과 졸업(공학석사). 2001년 서울대학교 컴퓨터공학과 졸업(공학박사). 2003년 ㈜아이티포웹 부장. 현재 부경대학교 IT융합응용공학과 부교수. 관심분야는 데이터베이스, 유비쿼터스 센서네트워크