UPnP기반의 센서 네트워크 관리 구조

송형주° 김대영
실시간 임베디드 시스템 연구실
한국정보통신대학교 공학부
{iamhjoo°, kimd}@icu.ac.kr

UPnP based Sensor Network Management Architecture

Hyungjoo Song° Daeyoung Kim
Real-time and Embedded Systems Lab.
Information and Communications University

요 약

본 논문은 현재 서비스 발견(Service Discovery) 기술로서 각광을 받고 있는 UPnP(Universal Plug and Play)기반의 서비스 활용 및 관리 기능을 적용하기 위한 센서 네트워크 구조를 제안한다. 이를 통해 센서 네트워크 사용자는 수동적인 설정 없이 센서 노드들을 바로 설치 할 수 있으며, 또한 각 센서들이 제공하는 서비스 및 센서 네트워크 전반에 사용될 수 있는 서비스(가령, 컨텍스트 정의 및 활용, 센서 위치 추적, 동기화, 전원 관리 등)를 쉽게 이용할 수 있고, 또한 이하 센서 네트워크 관리도 원활히 할 수 있게 된다. 그러나 TCP/IP 기반에서 동작하는 UPnP는 초소형의 센서 노드에는 적합지 않다. 이에 본 논문에서는 UPnP에 적합하지 않는 초소형의 센서 노드를 위한 브리지 역할 뿐만 아니라, 센서 네트워크 전반의 관리 서비스를 제공하는 BOSS(Brdige Of SensorS)라는 구조를 제안한다.

1. 서 론

지난 몇 년간 센서 네트워크는 미국을 중심으로 주로 군사·과학 분야에서 이용되어 왔다. 그러나 최근 들어 센서 네트워크는 유비쿼터스 컴퓨팅의 핵심 기술로서 평가 받으며, 그 활용도를 스마트 홈, 빌딩, 그리고 물류 등 더 많은 곳으로 응용 범위를 넓혀가고 있다. 하지만 이러한 센서 네트워크의 활용 범위가 넓어질수록, 사용자는 더 많은 센서 노드들을 설정 하고, 관리해야 하는 문제에 직면하게 된다. 또한 센서 노드들의 표준화가 이뤄지지 않았기 때문에, 각 센서 노드들의 서비스들을 이용하는 프로그램을 작성하는 개발자들은 해당 센서 노드 플랫폼에 종속적인 프로그램을 작성할 수밖에 없다.

UPnP는 이러한 문제 해결에 좋은 대안이 될 수 있다. UPnP는 마이크로소프트사(MS)에서 주도하고 있는 서비스 발견 미들웨어로서, UPnP를 지원하는 장치는 사용자의 특별한 설정 없이, 네트워크에 연결된 후, 사용자로 하여금 해당 장치로부터 제공되는 서비스를 바로 이용할 수 있게 해준다.[1] 이러한 UPnP를 센서 네트워크에 적용하면, 센서 네트워크 이용자는 수많은 센서들을 수동적인 설치 없이 바로 이용 및 관리 할 수 있으며, 개발자 또한 플랫폼 독립적인 서비스 발견 기술을 사용함으로써, 좀 더 능률적인 프로그래밍이 가능하게 된다. 그러나 UPnP 프로토콜은 TCP/IP기반의 HTTP 프로토콜 기반 위에서 동작하기 때문에, 낮은 성능의 CPU와 제한된 메모리를 사용하는 초소형의 센서 노드에는 적합하지 않다.

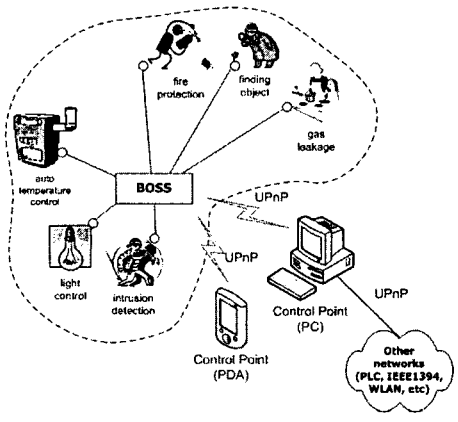

그림 1 UPnP 기반의 센서 네트워크 관리 구조

본 연구에서는 UPnP에 적합하지 않은 센서 노드를 위해 BOSS(Bridge Of the SensorS)라는 브리지 역할의 센서 네트워크 베이스 노드를 제안한다. BOSS는 센서 네트워크 환경에 UPnP를 적용하기 위한 핵심 구조로서, UPnP가 적합하지 않은 센서 노드들을 대신하여, UPnP 프로토콜을 처리해줌과 동시에, 센서 네트워크 환경에서 사용될 수 있는 다양한 UPnP 기반의 서비스 및 관리 환경을 제공한다.

2. BOSS를 이용한 UPnP기반의 센서네트워크 구조

그림 1은 UPnP 기반의 센서 네트워크 구조를 나타내는 것으로서, 컨트롤 포인트(Control Point), BOSS, non-UPnP 센서 노드들로 구성되어 있다.

컨트롤 포인트는 센서 네트워크의 서비스를 이용하고, 센서 네트워크를 관리할 수 있는 장치로서, PDA나 노트북 등이 사용된다. BOSS는 컨트롤 포인트와 non-UPnP 센서 노드들을 연결해 주는 브리지 역할 및 센서 네트워크에 관련된 서비스들을 제공해 주는 역할을 한다. 그림 1과 같이 컨트롤 포인트와 BOSS와의 통신은 UPnP 프로토콜을 사용한 반면에 BOSS와 non-UPnP 센서 노드들은 센서 네트워크 프로토콜을 사용하여 통신한다.

3. BOSS 구조

BOSS는 그림 2에서와 같이 서비스 매니저, 컨트롤 매니저, 이벤트 매니저, 서비스 테이블, 센서 네트워크 관리 서비스 등 5개의 컴포넌트로 구성되어있다. UPnP의 핵심 기능은 장치 및 서비스 발견, 디바이스 컨트롤 그리고 이벤트 처리다. 따라서 UPnP를 사용하기 위해서는 각 기능별 UPnP 프로토콜을 구현해야 한다. 하지만, non-UPnP 센서 노드는 UPnP 프로토콜을 처리할 수 없기 때문에, BOSS는 non-UPnP 센서 노드를 대신해서 UPnP 메시지를 처리하고, 전달해주는 매니저 구조를 적용한다. 서비스 테이블은 BOSS가 관리하는 센서 노드들의 서비스들을 저장하고 있는 테이블이다. 마지막으로 센서 네트워크 매니저 서비스는 BOSS가 제공하는 센서 네트워크 관리 및 BOSS가 포함된 센서 네트워크 전반에 관련된 서비스로서, 사용자는 컨트롤 포인트를 통해 이러한 서비스를 이용하여, 센서 네트워크를 관리 및 활용할 수 있다.

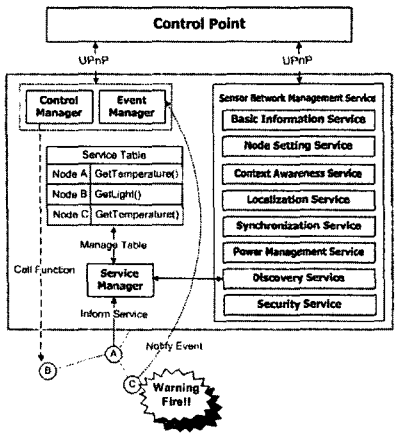

그림 2 BOSS 구조 및 기능

3.1 서비스 매니저

UPnP 장치가 네트워크에 참여할 때, 그 장치는 자신이 가지고 있는 서비스를 SSDP(Simple Service Discovery Protocol)라는 UPnP 서비스 발견 프로토콜을 이용해, 컨트롤 포인트에게 알려준다. 그러나 TCP/IP의 기반에서 동작하지 않는 초소형의 센서 노드는 이런 UPnP 발견 메시지를 이용할 수 없다. BOSS의 서비스 매니저와 Discovery Service는 non-UPnP 센서 노드를 대신해서 이런 역할을 해준다.

그림 3은 이러한 서비스 매니저의 내부 구조 및 동작 순서를 나타낸다. 센서 노드가 센서 네트워크에 추가되면, 자신의 서비스를 서비스 테이블 매니저에게 보내고, 이 때 서비스 테이블 매니저는 받은 서비스를 서비스 테이블에 추가하게 된다. 이후 컨트롤 포인트는 UPnP 이벤팅을 통해, 센서가 추가된 것을 알게 되고, BOSS의 Discovery Service중 GetSensorDescription()을 통해, Sensor 노드들에 대한 장치 설명 문서를 요구하면, 장치/서비스 기술자(Device/Service Descriptor)는 XML기반의 장치/서비스 문서를 작성하고, 그것을 컨트롤 포인트에게 보내준다.

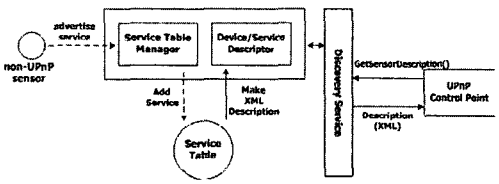

그림 3 서비스 매니저 내부 구조와 동작 순서

3.2 컨트롤 매니저

컨트롤 포인트가 UPnP 장치의 서비스를 발견하게 됐다면, 그 때부터는 서비스가 제공하는 액션(action)을 호출할 수 있다. UPnP에서는 이것을 컨트롤(Control)이라고 한다. 이것은 원격 프로시저 호출(RPC)의 일종으로, UPnP에서는 컨트롤 포인트와 UPnP 장치 사이의 SOAP(Simple Object Access Protocol)을 사용하여, 이러한 컨트롤을 처리한다.

그림 4는 컨트롤 매니저의 내부 구조와 동작 순서를 나타낸다. 컨트롤 포인트가 관심 있는 서비스의 액션을 호출할 때, 컨트롤 포인트는 액션 호출에 대한 SOAP 메시지를 컨트롤 메시지의 액션 핸들러(Action Handler)에게 보낸다. 이때 액션 핸들러는 그 정보를 이용하여 관련된 센서 노드에 서비스를 호출한다. 그런 다음, 센서 노드는 서비스 결과 값을 다시 액션 핸들러에게 보내고, 액션 핸들러는 결과 메시지를 다시 SOAP의 형태로 만든 다음, 컨트롤 포인트에게 보낸다.

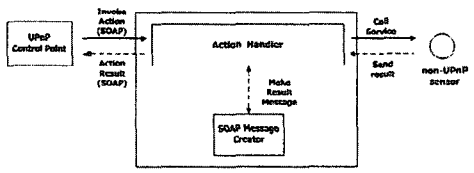

그림 4 컨트롤 매니저 내부 구조와 동작 순서

3.3 이벤트 매니저

컨트롤 포인트가 UPnP 장치의 장치/서비스를 발견한 후에는, UPnP 이벤팅(eventing)을 이용할 수 있다. 컨트롤 포인트는 이벤팅을 이용해 UPnP 장치의 상태 변화에 대한 정보를 얻을 수 있다. UPnP 이벤팅은 Publisher/Subscriber 모델을 이용한다. 즉 컨트롤 포인트는 이벤팅의 Subscriber가 되고, 반면에 UPnP 장치는 Publisher가 된다. 따라서 컨트롤 포인트는 이벤팅을 이용하기 전에, 관심 있는 장치의 이벤팅을 등록해야 한다. 이후 상태 변수가 변할 때, UPnP 장치는 GENA(General Event Notification Architecture)프로토콜을 이용하여 컨트롤 포인트에게 이벤트를 통지한다.

그림 5는 이벤트 매니저의 내부 구조와 동작 순서를 나타낸다. 위에서 언급한 것처럼, 컨트롤 포인트는 이벤트를 등록하기 위해 컨트롤 매니저의 이벤트 핸들러에게 GENA 메시지를 보낸다. 그런 다음, non-UPnP 센서 노드에서 이벤트가 발생했을 때, 이를 통지 받은 이벤트 핸들러는 GENA 메시지를 통해 다시 컨트롤 포인트에게 이벤트 사실을 통지한다.

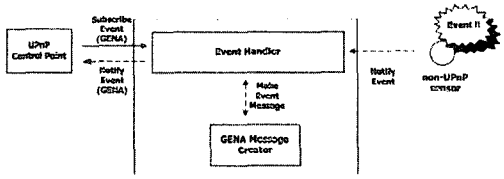

그림 5 이벤트 매니저 내부 구조와 동작 순서

3.4 센서 네트워크 관리 서비스

BOSS도 UPnP를 지원하는 센서 네트워크 베이스 노드 역할을 하는 장치로서, UPnP 서비스를 제공한다. 센서 네트워크 관리 서비스는 BOSS가 제공하는 서비스로서, 센서 네트워크 전반에 걸쳐 적용되며, 표.1과 같은 다양한 서비스로 이뤄져있다.

서비스	설명
Basic Information	이하 센서 노드들의 기본적인 정보를 제공. (예)센서 노드 장치 정보, 센서 노드 개수, 지원하는 서비스 목록, 센서 노드 토폴로지 등.
Node Setting	해당 센서 노드의 설정 값을 얻거나, 조절하는 기능을 제공. (예)이벤트 임계값 조절
Context Awareness	UPnP의 단순한 이벤트 기능을 보완하기 위한 컨텍스트 인지 기능 제공. 컨텍스트는 사람, 장소, 사물과 같은 개체들의 상황을 서술한 정보로서[2], 사용자는 이 서비스를 이용하여, 이러한 컨텍스트를 정의할 수 있다. 예를 들면 "사용자가 방에 있고, 방안의 온도가 25℃ 이상이면, 에어컨을 켜라"라는 컨텍스트를 사용자가 정의하면, BOSS의 컨텍스

	트 매니저는 각 센서들의 센싱 된 값의 조합이 위의 컨텍스트를 만족할 때 해당하는 동작을 실행하게 된다.
Localization	각 센서의 위치 정보를 제공한다. 이러한 서비스는 스마트 홈 애플리케이션에서 열쇠처럼 자주 잃어버리는 물건에 센서를 부착해서 이용할 수 있다.
Synchronization	센서 네트워크에서는 장치간의 클럭 동기화가 매우 중요하다. 센서 네트워크를 처음 설치하거나, 새로운 센서 노드를 추가할 때, 이 서비스를 이용해 노드간의 동기화를 할 수 있다.
Security	허가받은 사용자만이 센서 네트워크를 사용 하게끔 인증 기능 제공
Power Management	센서 노드들의 전원을 관리해주는 서비스. (예) 현재 남아 있는 배터리 체크

표 1 센서 네트워크 매니저 서비스

4. 장치/서비스 설명(Description) 설계

UPnP 장치는 그 장치에 대한 설명과 서비스를 설명하기 위해, XML 기반의 문서를 이용한다. 장치 설명은 벤더 이름, 장치 정보, 제공하는 서비스 및 컨트롤과 이벤트 처리를 위한 URL 등의 정보를 포함한다. 서비스 설명은 장치 설명에 포함된 서비스 정보 이외에도 그들의 인자들과 그것들의 데이터 타입, 또한 상태 변수들의 정보와 이벤트 특성 등을 정의한다.[3]

4.1 장치 설명 구현

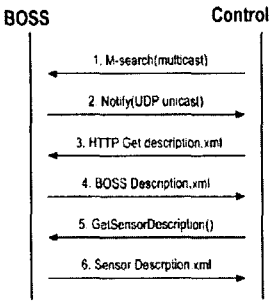

그림 6 BOSS와 컨트롤 포인트 사이의 발견(Discovery)과정

우리가 제안한 구조에서는 일반 UPnP장치의 장치/서비스 설명처럼, 동일한 기능을 가진 여러 개의 센서 노드마다 일일이 장치 및 서비스 설명 문서를 생성하는 것은, 많은 센서들로 이뤄진 센서 네트워크 환경에서는 비효율적이다. 이러한 문제를 해결하기 위해, 우리는 BOSS와 그 이하의 센서 노드들의 설명 문서를 두 부분으로 나눠서 처리한다. BOSS에 대한 장치 설명은 UPnP 포럼에서는 아직 센서 노드에 맞는 장치 설명 표준을 제공하고 있지 않기 때문에, UPnP 포럼에서 제공하는 Basic Device 1.0 표준을 통해 구현하고, 이와 다르게 각 센서들의 장치 설명을 가진 XML 문서는 서비스 매니저에 의해 동적으로 만들어져, 이를 BOSS의 Discovery Service인 GetSensorDescription()라는 액션으로 제공한다.

그림 6은 이런 과정을 나타내는 그림이다. 컨트롤 포인트가 UPnP장치를 검색하면, UPnP장치인 BOSS는 응답 메시지를 보내고, 이후 컨트롤 포인트가 설명 문서를 다시 요청하게 된다. 그러면 BOSS의 설명 문서가 컨트롤 포인트로 보내지고, 이때부터는 컨트롤 포인트는 BOSS의 서비스 중 하나인 GetNodeDescription()을 이용해서, BOSS가 관리하는 센서 노드들의 장치 정보를 얻을 수 있다. 아래 코드는 그림 2의 센서 노드 A, C에 관한 장치 설명 부분이다.

```
<sensorList>
  <sensor>
    <sensorType>urn:schemas-resl-icu-ac-kr:device:TemperatureSensor:1
    </sensortype>
    <sensorID>nodeA:nodeC</nodeID>
    <manufacturer>RESL</manufacturer>
    <serviceType>urn:schemas-resl-icu-ac-kr:GetTemperature:1
    <argument>..</argument>
    </serviceType> ...</sensor></sensorList>
```

위 코드의 대부분은 <sensorID>를 제외한 UPnP의 설명 문서와 유사하다. <sensor>필드는 똑같은 종류의 노드가 다중으로 포함되어 있는 센서 노드를 설명하기 위한 필드이다. 노드A와 노드C는 같은 서비스를 제공하는 같은 회사의 제품이다. 따라서 <sensorID>를 통해서 기능이 같은 장치는 설명의 중복 없이 정의할 수가 있다.

4.2 서비스 설명 구현

BOSS의 서비스 설명은 그림 2처럼 두 종류로 나뉜다. 하나는 각 센서들이 제공하는 서비스로, 이는 서비스 매니저에 의해 서비스 테이블로 정의되는 것이고, 다른 하나는 표1에서 설명한 BOSS가 자체적으로 제공하는 센서 네트워크 관리자 서비스이다. 다음은 센서 네트워크의 기본 정보(basic information)서비스 중의 하나인 센서 노드 개수를 구하는 *GetTotalSensors*라는 액션을 구현한 것이다.

```
...
<action>
  <name>GetTotalSensors</name>
    <argumentList>
      <argument>
        <name>GetTotalSensors</name>
        <direction>out</direction>
        <relatedStateVariable>totalSensors</relatedStateVariable>
      </argument>
    </argumentList>
</action>
...
<serviceStateTable>
    <stateVariable sendEvents="no">
      <name>totalSensors</name>
      <dataType>ui2</dataType>
    </stateVariable>
<serviceStateTable>
...
```

5. 결론

본 논문에서는 UPnP 기반의 서비스 및 관리 기능을 활용하기 위한 센서 네트워크 구조를 제안했다. 이 구조를 사용하면, UPnP에 적합하지 않은 센서 노드들로 이뤄진 센서 네트워크 환경에서도 BOSS를 통해 UPnP를 적용할 수 있게 되고, 이에 따라 센서 네트워크 사용자는 센서 노드를 쉽게 설치할 수 있으며, PDA같은 컨트롤 포인트 장치를 통해 쉽게 센서 네트워크의 서비스 활용 및 관리를 할 수 있다.

또한 최근 들어 UPnP는 홈 네트워크 환경의 핵심 미들웨어가 될 것이라는 평가를 받으며, LAN, IEEE1394, PLC 등 다양한 네트워크 환경에서의 장치들 상호간의 서비스를 이용할 수 있게끔 쓰이고 있다. 따라서 우리가 제안한 구조를 사용한다면 스마트 홈 환경에서도 UPnP를 통한 센서 네트워크와 다른 네트워크와의 연결이 쉬워지게 되고, 이에 따라 센서 네트워크의 활용이 극대화 될 수 있으며, 이는 좀 더 유비쿼터스 환경에 적합한 애플리케이션의 개발에 도움이 될 것이다.

6. 참고 문헌

[1]UPnP Forum, http://www.upnp.org

[2]A.K.Dey, G.D.Abowd, "Toward a better understanding of context and context-awareness.", GVU Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology

[3]UPnP Device Architecture Reference Specification Version 1.0, Microsoft Corporation, June 2000, http:/www.upnp.org