무선 센서 네트워크에서 객체 타입을 이용한 서비스 디스커버리

김승구*, 이중수*, 곽동필*, 이영희*
*한국정보통신대학교 공학부
e-mail : {kskkck, jslee, dpk94, yhlee}@icu.ac.kr

Object type-based service discovery in wireless sensor network

Seung-Ku Kim*, JoongSoo Lee*, Dong-Pil Kwak*, Young-Hee Lee*
*Dept. of Engineering, Information Communication University

요 약

최근 유비쿼터스 환경 실현을 위해서 무선 센서 네트워크에 관한 많은 연구가 진행되고 있다. IEEE 802.15.4 에 기반한 ZigBee 에 관한 연구는 이미 많은 진행이 되어 있다. 최근에는 IPv6 기술이 이슈화 되면서 무선 센서 네트워크 환경에도 IPv6 주소를 사용 할 수 있을 것으로 기대되고 있다. 본 논문에서는 IPv6 주소를 갖는 센서가 서비스를 제공하기만 하는 기존의 패러다임에서 벗어나 센서가 서비스를 찾는 모델을 제시한다. 그래서 IPv6 주소를 갖는 센서 네트워크의 보다 효율적인 서비스 디스커버리를 위해 객체 타입을 IPv6 주소의 일부분으로 정의해 준다. 이러한 객체 타입을 이용한 IPv6 기반의 센서 네트워크의 효율성을 본 논문에서 검증한다.

1. 서론

최근 유비쿼터스 환경 실현을 위해 여러 분야에서 다양한 연구가 진행 중이다. 이러한 유비쿼터스 환경 실현을 위한 한 방안인 무선 센서 네트워크에 관한 기술은 최근에 이슈화 된 새로운 분야이다. 느린 전송 속도, 저전력 환경, 그리고 낮은 가격 등의 특징을 갖는 IEEE 802.15.4 표준 아래 ZigBee 기술을 이용한 많은 연구가 이미 진행 되어왔다[1][7].

그러나 보다 나은 유비쿼터스 환경을 제공하기 위해 기존의 인프라와 무선 센서 네트워크의 통합이 필수 적이다. 현재 ZigBee, 6LoWPAN 등의 여러 단체에서는 센서 네트워크와 인프라의 통합을 위해 많은 노력을 하고 있다. 최근에는 IPv6 기술이 이슈화 되면서 무선 센서 네트워크 환경에서도 IPv6 주소를 사용 할 수 있을 것으로 기대되고 있다. 현재 6LoWPAN 이라는 워킹 그룹에서 이 연구를 진행 중인데 IPv6 주소를 갖는 센서 노드를 통해 기존의 인프라와의 보다 나은 연결성, 프로토콜의 재사용, 기존 인프라를 통한 높은 신뢰성 등을 얻을 수 있다[3][6]. 하지만 현재 6LoWPAN 에서 진행중인 IP 기반의 인프라와 무선 센서 네트워크간의 결합에 관한 연구는 아직 미흡하다.

객체가 기존 인프라를 통한 원활한 서비스 제공을 받기 위해 서비스 디스커버리에 대한 연구는 필수적이다. 기존의 연구들에서는 센서 노드가 서비스를 제공하는 시나리오가 대부분이었다. 하지만 센서 노드가 서비스를 찾아야 하는 시나리오도 고려해야 할 것이다. 예를 들면 급성 심장병 환자의 심장 박동수 측정을 하는 센서의 경우를 생각해 볼 수 있다. 갑자기 환자의 심장 박동수가 높아 지게 될 경우 즉시 병원이나 119 서비스를 요청해야 할 것이다. 이러한 시나리오에서는 기존의 센서 노드가 서비스를 제공하는 방식으로는 구현되기 어렵다.

그래서 본 논문에서는 센서 노드가 서비스를 찾는 모델을 제안한다. 이러한 모델을 만족하기 위해서는 센서 노드가 LoWPAN 환경 특성에 따라 서비스를 찾고, 제어하고, 유지해 주는 서비스 디스커버리 프로토콜이 필요하다. 그리고 본 논문에서 제안하는 객체 타입에 따라 IPv6 주소의 일부분을 정의 해주는 방법은 보다 효율적인 서비스 디스커버리에 사용 될 것이다.

본 논문에서는 인프라와 무선 센서 네트워크의 통합에 필수 요소인 객체 타입을 이용한 효율적인 서비스 디스커버리를 제안한다. 2 장에서는 서비스 디스커버리를 위한 기능적 구조를 설명하고, 3 장에서는 IPv6 인터페이스 ID 를 이용한 객체 타입의 할당 방법을 말한다. 그리고 4 장에서는 효율적인 서비스 디스커버리를 위한 객체의 타입 구성에 대해 설명하고, 5 장에서는 계층적 구조를 이용한 의미 기반의 서비스 디스커버리에 대해 제안한다. 그리고 6 장에서는 구현 방법을 제시하고 마지막 7 장에서는 본 논문에 대한 결론을 짓는다.

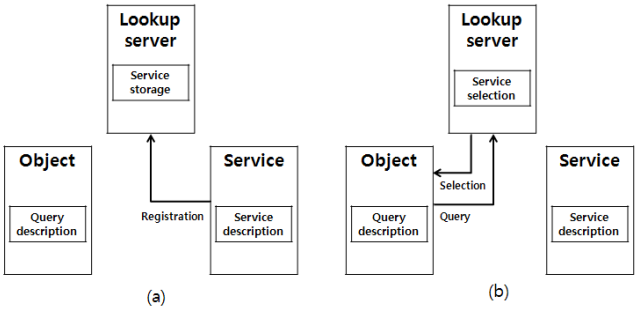
2. 서비스 디스커버리를 위한 기능적 구조

객체 타입의 적절한 할당을 통한 효율적인 서비스 디스커버리를 위해서는 서비스 디스커버리를 위한 기능적 구조를 고려해야 한다. 우선 서비스 디스커버리를 위한 기본 기능은 <표 1>과 같이 나타난다.

<표 1> 서비스 디스커버리를 위한 기본 기능

기능	설명
Service description	서비스의 정보를 나타낸다.
Query description	객체가 서비스를 찾기 위해 요청하는 정보를 나타낸다.
Service storage	Service description 을 저장한다.
Service selection	많은 서비스들 중 가장 적절한 서비스를 찾는다.
Security	보안 문제를 해결 한다.

Service description 은 서비스에서 제공하는 이름, 종류, 위치, 주소 등과 같은 정보를 나타낸다. 이러한 Service descriptions 는 어떤 곳에서 저장 혹은 관리 되어야 하는데 그 역할을 하는 것이 Service storage 이다. Service storage 방법에는 자신이 직접 서비스를 저장하는 방법과 Lookup server 와 같은 다른 시스템을 통해 서비스를 저장하는 방법이 있다.



(그림 1) 서비스 디스커버리를 위한 기능적 구조

하지만 자신이 직접 서비스를 저장하는 방법은 그 서비스를 찾기 위해 브로드 캐스트 혹은 멀티 캐스트를 이용하기 때문에 센서 네트워크 환경에는 적합하지 않다. 따라서 본 논문에서는 (그림 1)의 (a)와 같은 Lookup server 를 통해 서비스를 저장하는 모델을 제안한다.

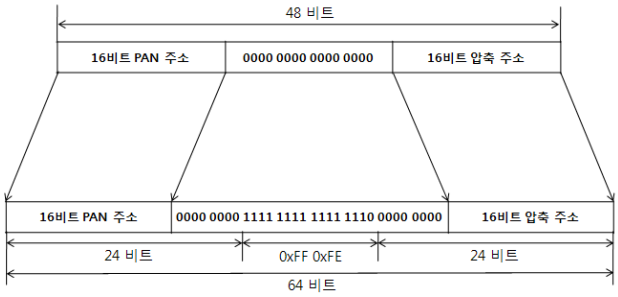
Query description 은 객체가 서비스를 찾기 위해 요청하는 이름, 종류, 위치, 주소 등과 같은 정보를 나타낸다. 본 논문에서는 객체 타입을 이용해 Query description 을 제공한다.

Service selection 은 Query description 과 Service description 을 이용하여 많은 이종의 서비스들 중에서 가장 적절한 서비스를 찾는 역할을 한다. Service selection 에는 서비스를 찾는 객체가 최적화된 알고리즘에 의해 직접 선택하는 방법과 위에서 언급한 특정 Lookup server 와 같은 다른 시스템을 통해 선택하는 방법이 있다. 본 논문에서는 객체의 제한적인 기능으로 인해 (그림 1)의 (b)와 같은 다른 시스템을 통해 서비스를 선택하는 모델을 제안한다. 그리고 Service selection 할 때 Service missing (적절한 서비스를 찾지 못한 경우)과 Service duplication (같은 서비스가 여러 개 있는 경우)과 같은 문제가 발생 할 수 있다. 이러한 문제점들을 해결하는 방안도 고려해야 한다[9][10].

3. IPv6 인터페이스 ID 를 이용한 객체 타입의 할당

현재 IPv6 의 주소는 64 비트의 프리픽스 부분과 64 비트의 IPv6 인터페이스 ID 로 구성되어 있다. 그런데 IEEE 802.15.4 는 IEEE 64 비트의 확장 주소 또는 16 비트의 압축 주소를 사용한다. 확장 주소는 IEEE EUI-64 표준에 따라 할당되고 압축 주소는 PAN coordinator 에 의해 동적으로 할당 된다.

본 논문에서는 PAN coordinator 에 의해 동적으로 할당되는 압축 주소를 사용한다. 압축 주소를 사용하게 되면 확장 주소에 비해 헤더 크기도 줄일 수 있고 IPv6 인터페이스 ID 를 만들 때 객체 타입을 할당할 공간도 만들 수 있다.



(그림 2) 16 비트 압축 주소의 IPv6 인터페이스 ID 로 변환

하지만 16 비트 압축 주소를 이용한 EUI-64 만드는 방식은 현재 없다. 그래서 16 비트 압축 주소로 IPv6 인터페이스 ID 를 만들기 위해서는 “pseudo 48-bit address” 방식이 필요하다. 첫 번째로 가장 왼쪽 32 비트에는 16 비트 PAN 주소와 16 비트의 ‘0’으로 형성된다. 그리고 나머지 오른쪽 16 비트에는 16 비트 압축 주소가 사용된다. 이렇게 만들어진 “pseudo 48-bit address”로 EUI-64 주소를 만들기 위해 왼쪽 24 비트와 오른쪽 24 비트 사이에 FF FE 를 삽입해 준다[4][5].

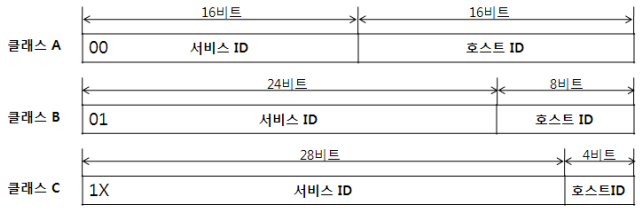
하지만 이렇게 만들어진 주소는 전세계적으로 유일하지 않다. 그래서 이렇게 만들어진 EUI-64 주소의 “Universal/Local”을 나타내는 7 번째 비트를 Local 주소임을 나타내는 ‘0’으로 설정한다. 위에서 설명한 16 비트 압축 주소를 이용해 IPv6 인터페이스 ID 를 만드는 방식은 (그림 2)에 나타난다[2].

이러한 방식의 IPv6 인터페이스 ID 에 객체 타입 정보를 제공하기 위해서는 16 비트의 PAN 주소와 16 비트 압축 주소 사이에 32 비트의 사용하지 않는 공간을 객체 타입을 표현하는데 사용한다. IPv6 인터페이스 ID 에 32 비트의 객체 타입 정보를 제공 함으로서 객체 정보를 표현하기 위한 공간의 낭비를 줄여준다. 그리고 IP 계층에서 타입 정보를 할당하기 때문에 표준화가 쉽게 이루어 진다.

4. 효율적인 서비스 디스커버리를 위한 객체 타입의 구성

Query description 은 32 비트 객체 타입을 이용해 자신이 어떤 객체인지, 그리고 어떤 서비스를 찾고 있는지를 나타낸다. 32 비트에 이를 표현하기 위해서는 서비스와 호스트 종류에 따라 객체 타입을 나누어야 한다. 서비스와 호스트 종류에 따라 객체 타입을 나누는 이유는 같은 호스트라도 다른 서비스를 찾을 수도 있고, 다른 호스트라도 같은 서비스를 찾을 수도

있기 때문이다. 예를 들면 같은 심장병 환자 객체라도 급성 심장병 환자의 경우에는 119 서비스를 요구할 것이고 만성 심장병 환자의 경우에는 내과 서비스를 요구 할 수도 있다. 그리고 혈압 환자 역시 심장병 환자가 요구하는 내과 서비스가 필요 한 경우가 있다. 그렇기 때문에 객체 타입을 서비스 ID 와 호스트 ID 로 나누어 주었다.



(그림 3) 32 비트 인터페이스 ID 에 객체 타입 할당 방법

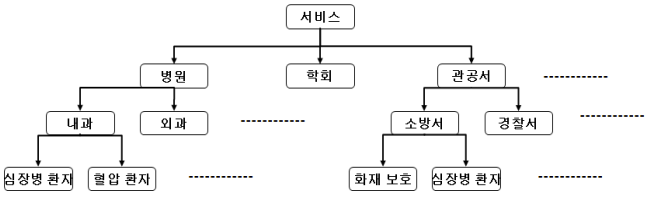
32 비트 공간으로 서비스와 객체 정보를 효율적으로 표현하기 위해 (그림 3)과 같이 세가지 클래스로 나누어 준다. 첫 번째로 클래스 A 는 하나의 서비스가 많은 종류의 객체에게 서비스를 제공 할 때 사용된다. 클래스 A 는 제일 왼쪽의 식별용 비트가 '00'로 시작된다. 그리고 16 비트의 서비스 ID 와 16 비트의 호스트 ID 로 나누어 진다.

클래스 B 는 하나의 서비스가 클래스 A 에 비해 다소 적은 수의 객체에게 서비스를 제공 할 때 사용된다. 클래스 B 의 식별용 비트는 '01'이 되고 24 비트의 서비스 ID 와 8 비트의 호스트 ID 로 나누어 진다.

마지막으로 클래스 C 는 하나의 서비스가 작은 종류의 객체에게 서비스를 제공 할 때 사용된다. 클래스 C 의 식별용 비트는 '1'이고 28 비트의 서비스 ID 와 4 비트의 호스트 ID 로 나누어 진다.

5. 계층적 구조를 이용한 의미 기반의 서비스 디스커버리

Lookup server 에서 Service selection 기능을 수행 시 적절한 서비스를 찾지 못할 경우 Service missing 이 발생한다. Service missing 이 발생하여 필요한 서비스를 제공 받지 못하면 서비스 디스커버리의 신뢰성이 떨어진다.



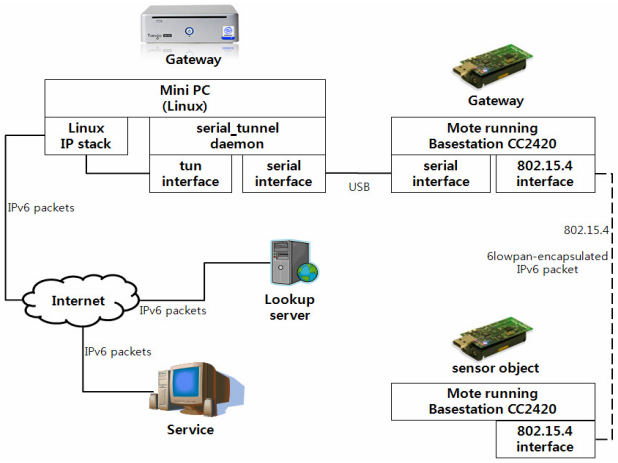
(그림 4) 객체 타입의 계층적 구조

이러한 문제를 해결하기 위해 (그림 4)와 같이 유사한 기능에 따라 서비스들을 연결하여 계층적 구조를 만들었다. 이 계층적 구조는 의미 기반의 서비스 디스커버리에 사용된다.

Lookup server 는 객체로부터 서비스 요청이 온 경우 트리의 가장 아래 노드부터 검색한다. 만약 객체가 요구하는 서비스와 일치하는 서비스가 있는 경우에는 사용자가 요청한 정확한 서비스를 제공 해 줄 수 있다. 하지만 가장 아래 노드에 일치하는 서비스가 없는 경우 같은 부모 서비스를 갖는 유사한 서비스를 찾아 제공한다. 이 서비스는 사용자가 요청한 정확한 서비스는 아니지만 객체에게 도움이 되는 서비스를 제공 함으로서 Service missing 이 되는 것을 막아준다.

6. 구현

본 논문의 내용을 구현하기 위해 센서 노드에 IPv6 를 올린 기존의 연구인 "Connecting Wireless Sensor Networks to the Internet - a 6lowpan Implementation for TinyOS 2.0"에 기반하여 진행되었다[11].



(그림 5) 테스트베드 구축 환경

테스트 베드 구축을 위해 (그림 5)와 같이 센서 객체로 사용 될 TelosB 와 게이트웨이로 사용 될 리눅스 기반의 미니 PC 를 사용한다. 기존 연구의 헤더 압축 기술, 패킷 단편화, 메시 네트워크 지원, IPv6 stateless 자동 주소 할당 방식을 사용한다. 여기서 IPv6 stateless 자동 주소 할당 시 압축 주소를 부여하여 3 번에서 설명한 것과 같이 32 비트의 적절한 타입을 센서 객체에 할당한다. 그리고 임의의 Lookup server 를 이용하여 적절한 서비스를 찾아주도록 한다. 게이트웨이를 여러 곳에 배치하여 이동성에 관해서도 실험 할 예정이다.

7. 결론

본 논문에서는 IP 기반의 인프라와 무선 센서 네트워크 통합 시 센서 노드가 인프라 내의 서비스를 찾아 주는 모델을 제안한다. 이때 센서 노드는 객체 타입을 서비스와 객체 종류에 따라 나누어 IPv6 주소 공간에 할당 함으로서 패킷 헤더 크기를 줄여주고 찾고자 하는 서비스를 나타내준다. 그리고 서비스를 계층적 구조로 나누어 의미 기반의 서비스 디스커버리를 제공하여 검색의 신뢰성을 높여준다.

향후 의미를 갖는 서비스 디스커버리에 더해 컨텍

스트에 기반한 서비스 디스커버리를 이용하여 service duplication 이 발생하지 않게 사용자의 요구에 따라 우선순위를 정해주고, 좀 더 효율적이고 신뢰성 있는 모델을 제안 할 예정이다.

참고문헌

[1] IEEE 802.15.4-2006 standard
[2] GUIDELINES FOR 64-BIT GLOBAL IDENTIFIER (EUI-64) REGISTRATION AUTHORITY
[3] N. Kushalnagar, G. Montenegro, C. Schumacher, “6LoWPAN: Overview, Assumptions, Problem Statement and Goals”, Internet-Draft, Aug 2007
[4] R.Hinden, and S.Deering, “IPv6 Addressing Architecture”, RFC 4291, Feb 2006
[5] G.Montenegro, N.Kushalnagar, and J.Hui, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks”, RFC4944, Sep 2007
[6] K.Mayer, and W.Fritsche, “IP-enabled Wireless Sensor Networks and their integration into the Internet”, InterSense ’06, May 2006
[7] 권훈 외 5 명, “무선 센서 네트워크와 IPv6 기반 인터넷 간의 연동 모델”, Journal of Korea Multimedia Society, Nov 2006
[8] C.Frank, V.Handziski, H.Karl, “Service Discovery in Wireless Sensor Networks”, Berlin, Mar 2004
[9] S.Helal, “Standards for Service Discovery and Delivery”, Pervasive computing, Sep 2002
[10] R.Marin-Perianu, P.Hartel, and H.Scholten, “A Classification of Service Discovery Protocols”, Jun 2005
[11] Matus Harvan, “Connecting Wireless Sensor Networks to the Internet - a 6lowpan Implementation for TinyOS 2.0”, Uni Bremen, May 2007