ETRI

2013
Electronics and
Telecommunications
Trends

스마트 유무선 네트워크 특집

IETF CoAP 기반 센서 접속 프로토콜 기술 동향

Trends of IETF CoAP Based Sensor Connection Protocol Technology

고석갑 (S.K.Ko) 관제디바이스연구실 선임연구원
박일균 (I.K. Park) 관제디바이스연구실 선임연구원
손승철 (S.C. Soon) 관제디바이스연구실 선임연구원
이병탁 (B.T. Lee) 관제디바이스연구실 실장

기존의 센서 접속 기술은 그 분야에 따라 다양한 표준과 규격이 존재하였다. 빌딩 자동제어 및 제어 네트워크용으로는 BACnet이 있으며, 공장제어 및 계측제어를 위한 Modbus 등이 다양한 프로토콜이 존재한다. 최근 M2M(Machine-to-Machine) 및 IoT(Internet of Things), WoT(Web of Things) 시대로 접어 들음에 따라 센서 및 각종 디바이스 간에 상호 통신할 수 있는 공통 프로토콜이 필요하게 되었다. IoT 프로토콜로 IETF(Internet Engineering Task Force) CoAP(Constrained Application Protocol) 프로토콜이 핵심 역할을 하고 있으며, 차세대 센서 접속 프로토콜로 IETF CoAP이 많이 사용될 것으로 예상된다. 본고에서는 IETF CoAP 표준화 동향 및 규격에 대해 살펴본다.

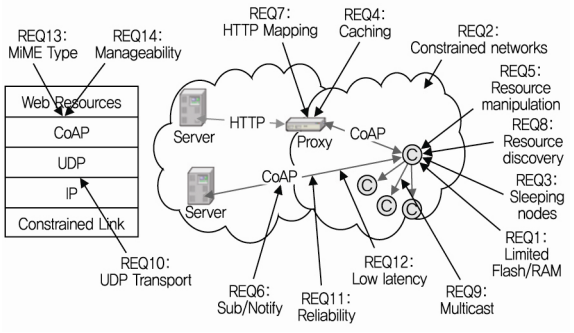
Ⅰ. 서론

그 동안의 인터넷이 컴퓨터가 사람에게 서비스를 제공하는 것이었다면, 최근에는 기계와 기계 간의 통신을 통해 다양한 서비스를 제공하는 것으로 바뀌어 가고 있다. 인터넷에는 슈퍼 컴퓨터로부터 1cm 크기의 소형 센서에 이르기까지 다양한 기기가 연결 및 통합되어 가고 있으며, 이러한 사물들의 인터넷이라는 의미에서 IoT (Internet of Things)라는 용어가 만들어졌다. 유사한 용어로, M2M(Machine-to-Machine) 통신, Smart Object, WoT(Web of Things), MTC(Machine Type Communication), D2D(Device-to-Device) 등이 있다.[1]

인터넷 표준 단체인 IETF(Internet Engineering Task Force)[2]에서는, 다양한 기기가 인터넷에 연결될 것을 예상하여 IPv6, 6LowPAN 등 표준활동을 진행해 왔으며, 최근에는 6LoWPAN, CORE, ROLL, LWIG 워킹그룹 등이 저전력, 소형 장치에 들어가는 표준을 개발하고 있다. 그 중 CORE(Constrained RESTful Environments) 워킹그룹에서는, 메모리, 에너지, 성능 등에 제약이 있는 M2M 환경을 위한 웹 기반 프로토콜인 CoAP(Constrained Application Protocol) 이라는 프로토콜을 만들고 있다[3]. 본고에서는 이러한 IETF CoAP 프로토콜에 대해 상세히 살펴본다

Ⅱ. 센서 접속 기술동향

기존의 센서 접속 기술은 그 분야에 따라 다양한 표준과 규격이 존재하였다. 빌딩 자동제어 및 제어 네트워크 용으로는 BACnet이 있으며, 공장제어 및 계측제어를 위한 Modbus 등이 다양한 프로토콜이 존재한다. 최근 M2M 및 IoT 시대로 접어 들음에 따라 센서 및 각종 디바이스 간에 상호 통신할 수 있는 공통 프로토콜이 필요하게 되었다. IoT 관련 표준화로는 ITU(International Telecommunication Union) 및 ETSI(European Telecommunication Standards Institute)가 주로 관련 단체 조율 등 큰 그림을 그리고 있으며, IETF 에서 IP 기반 IoT 표준화를 주도하고 있다. 또한 IPSO(Internet Protocol for Smart Objects Alliance)에서는 IETF 표준이 빨리 자리잡을 수 있도록 호환성 테스트, 사례 연구, 화이트 페이퍼 발간 등 지원을 하고 있다. 따라서, 차세대 센서 접속 프로토콜로 IETF CoAP이 많이 사용될 것으로 보인다.



(그림 1) IETF CoRE WG 표준화 범위

IETF CoRE 워킹그룹은 M2M 노드 간에 통신할 수 있는 CoAP 프로토콜을 중심으로 프로토콜을 개발하고 있다. (그림 1)은 IETF 79차 회의 슬라이드에서 발췌한 것으로, CoRE 워킹그룹의 표준화 범위를 나타내고 있다. CoRE 노드의 제약사항, 즉 RAM, ROM 등 메모리 크기를 비롯하여, 손실 및 전송속도, 지연 같은 네트워크의 제약사항, 노드의 전력 및 슬리핑 모드에 대한 고려, 리소스 등록 및 탐색, 신뢰성 있는 전달, 이벤트 통지 방법, 멀티캐스트, Proxy에서의 캐싱 및 HTTP 매핑, 데이터 포맷, 보안 고려 등이 표준화 범위에 해당한다.

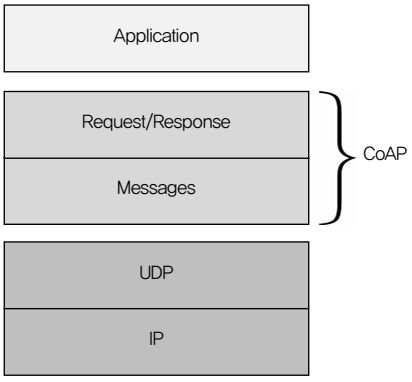
현재 CoAP 표준화가 마무리 되어감에 따라, 구현 업체 간 상호 연동성 시험을 위한 Plugtest가 ETSI, ProbeIT, IPSO 공조로 진행되어 왔고, 2013년 11월에 제 3차 시험이 완료 되었으며, ETRI도 이 시험에 참여하였다.

Ⅲ. CoAP 기본 프로토콜

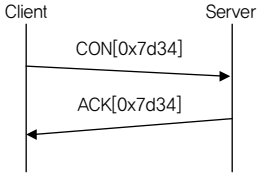
1. CoAP Specification

IETF CoRE 워킹그룹의 핵심 프로토콜인 CoAP이 워킹그룹 드래프트 18에서 RFC 표준으로 채택되었다[3]. CoAP 프로토콜은 저전력, 고손실 네트워크 및 소용량, 소형 노드에 사용될 수 있는 특수한 웹 전송 프로토콜이다. RESTful 사상을 따르고 있기 때문에, 기존의 HTTP 웹 프로토콜과도 쉽게 변환 및 연동이 될 수 있다.

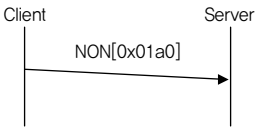
(그림 2)은 추상적 계층상의 CoAP의 위치를 나타낸다. CoAP은 기본적으로 UDP와 같은 데이터그램 방식의 트랜스포트 계층 위에서 비동기적으로 전송되는 것을 다룬다. 따라서 신뢰성 있는 전달을 위한 재전송 및 타이머 관리를 옵션으로 포함하고 있다. 보안을 위해서 UDP 계층과 CoAP 계층 사이에 DTLS(Datagram Transport Layer Security) 계층이 사용될 수 있다. CoAP은 확인형(confirmable), 비확인형(non-confirmable), 승인(acknowledgement), 리셋(reset)의 4가지 메시지 타입을 정의하고 있다. CoAP 메시지는 요청(request)과 응답(response)의 상호작용으로 전달된다.



(그림 2) CoAP의 추상적 계층



(그림 3) 신뢰성 있는 메시지 전송



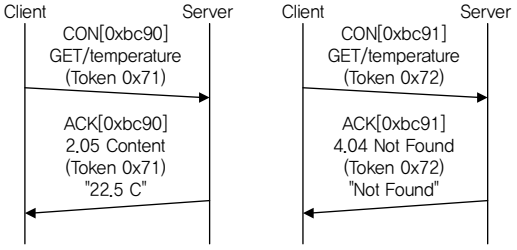
(그림 4) 비신뢰성 메시지 전송

(그림 3)는 신뢰성 있는 전달을 위한 메시지 전달을 보인다. 신뢰성 있는 전달을 위해서는, 확인형(CON) 메시지를 전송하며, 여기에 포함된 메시지 ID는 승인 메시지에 동일하게 들어가게 된다. 만약 수신자가 확인형 메시지를 처리할 수 없을 때는, 승인(ACK) 메시지 대신 리셋 메시지를 보낸다.

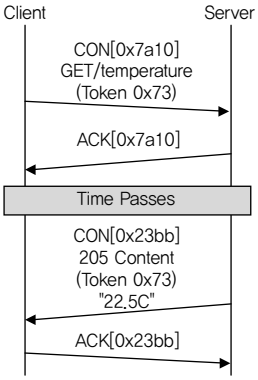
계속적인 센서 데이터 값을 얻어오는 것처럼 신뢰성 있는 전송이 요구되지 않을 때는, (그림 4)와 같이, 비확인성 메시지(NON)을 전송할 수 있다. 메시지 ID는 메시지 중복 검출용으로 사용된다. 만약 수신자가 비확인성 메시지를 처리할 수 없는 경우에는 리셋 메시지로 응답할 수 있다.

CoAP 요청에 따른 응답은 토큰이라는 필드를 통해 서로 짝을 이룬다. 반면 메시지 ID는 하나의 트랜잭션을 구분하는데 사용된다. (그림 5)는 GET 요청에 대해 piggy-back 형태로 응답하는 모습을 보인다.

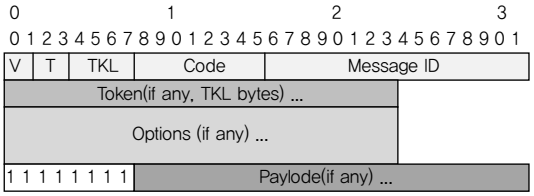
서버가 요청 메시지에 대해 즉시 응답하기 어려운 경우, 뒤늦게 응답을 할 수 있다. 이 경우 일단, 승인(ACK) 메시지만 보낸 후, 준비가 되면 데이터를 담아 응답할 수 있다. (그림 6)은 이러한 과정을 보이며, 요청



(그림 5) piggy-back 형태의 응답



(그림 6) 지연된 별도의 응답

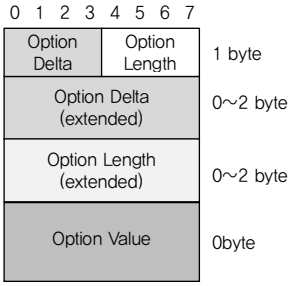


(그림 7) CoAP 메시지 포맷

메시지의 토큰이 유지되는 것을 볼 수 있다.

CoAP메시지는 (그림 7)과 같이 간단한 바이너리 포맷으로 인코딩 된다. 메시지는 4바이트 고정 헤더를 포함한다. 이어서 0에서 8바이트 길이의 토큰이 위치한다. 그 다음으로는 옵션이 온다. 페이로드가 있는 경우, 그 다음부터 데이터그램 끝까지 배치된다.

- 첫 2비트 V는 버전을 의미한다. CoAP 버전은 현재는 이진수 01이어야 한다.
- 두번째 2비트 T는 메시지 타입을 의미한다. 0은 확인형, 1은 비확인형, 2는 승인, 3은 리셋을 나타낸다.
- TKL은 토큰 필드의 길이를 나타낸다. 단위는 바이트이며 0에서 8까지의 값을 사용할 수 있다.
- Code는 3비트는 클래스(class)를, 5비트는 자세한 내용(detail)을 의미한다. CoAP 문서에서는 c.dd와 같은 형태로 표현한다. 클래스는 0은 요청, 2는 성공적인 응답, 4는 클라이언트 에러 응답, 5는 서버 에러 응답이다. 요청의 경우, 0.01은 GET, 0.02는 POST, 0.03은 PUT, 0.04는 DELETE를 나타낸다. 코드 0.00은 빈 메시지이다.
- 메시지 ID는, 중복 확인 및 확인성/비확인성 메시지에 대한 짝으로서 승인/리셋 메시지에 사용된다.
- TKL에 따라, 메시지 헤더 다음에 토큰 값이 온다.
- 토큰에 이어서 옵션들이 올 수 있는데, 메시지 끝에 도달하였거나, 페이로드 마커로 더 이상 옵션이 없음을 알 수 있다.
- 페이로드 마커는 0xFF 이며, 이후 페이로드는 데이터그램의 끝까지 위치한다.

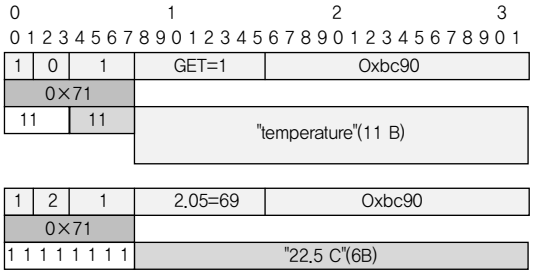


(그림 8) CoAP 옵션 포맷

CoAP 메시지의 옵션(Option)은 (그림 8)과 같은 TLV 형태를 띤다.

- 옵션 델타(Option Delta)는 0에서 12 사이의 값은 옵션 번호를 나타내는데, 바로 앞에 있는 옵션의 옵션 번호와의 차이로 표현한다. 13의 경우, 1바이트 옵션 델타 확장 필드를 사용하며, 옵션 델타 값은 옵션 델타 확장 필드의 값에서 13을 더한 값이다. 14의 경우, 2바이트 옵션 델타 확장 필드를 사용하며, 옵션 델타 값은 옵션 델타 확장 필드 값에서 269를 더한 값이다. 15의 경우 페이로드 마커를 나타낸다.
- 옵션 길이(Option Length)는 옵션 값의 길이를 나타내며, 0에서 12 사이의 값을 가진다. 13의 경우, 옵션 길이 확장 필드를 사용하며, 옵션 길이 확장 필드에 그 값에서 13을 더한 값이다. 14의 경우, 옵션 길이는 옵션 길이 확장 필드의 값에 269를 더한다. 15는 예약되어 있다.

현재 정의된 옵션은 Content-Format, ETag, Loca-



(그림 9) CoAP 메시지 인코딩 예제

tion-Path, Location-Query, Max-Age, Proxy-Uri, Proxy-Scheme, Uri-Host, Uri-Path, Uri-Port, Accept, If-Match, If-Non-Match, Size1가 있다.

(그림 9)는 (그림 5)의 예제 메시지를 인코딩한 것을 나타낸다.

그 외에 CoAP 표준에서는 페이로드 포맷, 캐쉬, 프록시 동작, CoAP URI 스킴, 서비스 및 리소스 탐색 방법, 멀티캐스트 CoAP, DTLS를 이용한 보안, CoAP과 HTTP 연계 등에 대해 서술하고 있다.

2. CoRE Link Format

IETF CORE 워킹그룹에서는 가장 먼저 RFC-6690, 즉, Core Link Format 표준을 완성하였다[4]. Link Format 표준은 M2M 노드가 다른 노드를 찾거나 연계 동작을 할 수 있도록, 노드가 가지고 있는 자원이나 속성을 제공하는 포맷을 정의한다. 이는 기존의 HTTP 웹의 Web Linking[5] 표준을 M2M 환경에 맞게 확장하였다. 링크들은 HTTP Link 헤더 필드 대신에 메시지 페이로드로서 전달되며, 리소스를 탐색할 수 있는 디폴트 인터페이스인 "./well-known/core"가 정의되었다. 이 표준은, 'hosts'라는 새로운 관계 타입을 정의하였는데, 이는 요청을 받은 리소스를 가지고 있는 서버를 의미한다.

Link Format 표준을 이용한 사용 예로써 '탐색' 기능이 있다. 만약 CoAP 서버의 IP 주소를 아는 경우, 유니캐스트로, 디폴트 인터페이스로 GET 요청을 보내어, 서버가 가지고 있는 리소스에 대해 확인할 수 있다. 여기에는 리소스 타입, 인터페이스 서술, 미디어 타입 등이 포함된다. 질의 문자열(query string)에 속성을 주어 필요한 내용만 골라서 확인할 수도 있다. 유니캐스트 이외에, 멀티캐스트 리소스 검색을 통하여, 필요한 리소스를 검색할 수 있다.

M2M 환경에서 리소스를 쉽게 탐색하고 관리할 수 있도록 디렉토리 서비스를 사용할 수 있는데, M2M 서버가 POST 메시지로 리소스 디렉토리의 "./well-known/core"에 리소스를 등록할 수 있다.

CoRE 리소스 탐색 인터페이스는 다음과 같은 상호작용을 지원한다.

- 디폴트 포트에 대해 GET /.well-known/core를 수행하면, 서버에서 가능한 링크의 집합을 리턴한다. 이 링크는 그 서버가 가지고 있는 리소스뿐만 아니라, 다른 서버가 가지고 있는 리소스도 포함될 수 있다.

질의 문자열에 따라, 그에 맞는 응답을 수행한다. 예를 들어, GET /.well-known/core?rt=temperature-c는 리소스 타입이 temperature-c인 리소스만 필터링하라는 것을 의미한다.

- 프록시와 같은 좀 더 좋은 서버의 경우, POST 메시지를 통해 리소스를 등록 받는 기능을 수행한다.

다음 예는 같은 인터페이스 서술을 공유하는 두 개의 센서에 대한 링크를 포함한다. 복수 개의 링크는 쉼표(,)로 구분되며, 링크 파라미터 앞에는 세미콜론(;)을, 넣으며, if="sensor"라는 것에서 "if"는 인터페이스 서술을 의미한다.

```
REQ: GET /.well-known/core

RES: 2.05 Content
</sensors/temp>;if="sensor",</sensors/light>;if="sensor
```

다음 예는, 계층적인 링크 서술을 보인다. "rt="는 리소스 타입을 의미하며, 어떤 리소스인지, 즉 어떤 센서인지를 나타낸다.

```
REQ: GET /.well-known/core

RES: 2.05 Content
</sensors>;ct=40

REQ: GET /sensors

RES: 2.05 Content
</sensors/temp>;rt="temperature-c";if="sensor",
</sensors/light>;rt="light-lux";if="sensor"
```

클라이언트는 질의 문자열을 이용하여, 원하는 정보만 필터링하여 얻을 수 있다. 다음 예는 리소스 타입이 light-lux인 리소스에 대한 링크만 얻는 과정이다. 만약 복수 개의 속성값이 있는 경우, 스페이스로 구분하여 표시한다. 이 예제에서 /sensors/light 링크는 light-lux와 core.sen-light 라는 두 개의 리소스 타입 속성 값을 가지고 있다.

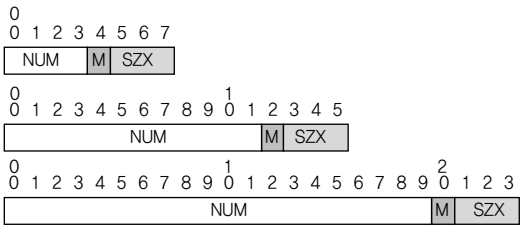
```
REQ: GET /.well-known/core?rt=light-lux

RES: 2.05 Content
</sensors/light>;rt="light-lux core.sen-light";if="sensor"
```

Ⅳ. CoAP 확장 프로토콜

IETF CoRE 워킹그룹은 앞에서 설명한 프로토콜외에 여러가지 프로토콜을 추가로 개발하고 있다. 본 고에서는 큰 데이터를 전송하기 위해 블록단위로 전송하는 방법에 관한 Blockwise Transfer in CoAP[6]과 지속적인 결과를 받을 수 있는 Observing Resources in CoAP[7] 문서에 대해 소개한다.

1. Blockwise Transfer

기본 CoAP 메시지는 온도센서, 전등 스위치, 빌딩 자동화 장치등 작은 페이로드를 다룰 수 있게 되어 있다. 그러나, 펌웨어 업데이트와 같은 큰 페이로드를 전송할 필요가 있다. 이를 위해서 Blockwise Transfer 문서에서는 Block Option 1 Block Option 2 등을 정의하여 블록 단위로 페이로드를 전송할 수 있게 한다. 블록

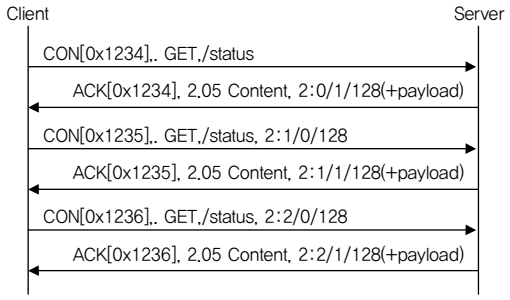


(그림 10) 블록 옵션 값 포맷

옵션 1은 요청 메시지 에페이로드를 보내는 경우에 사용되며, 블록 옵션 2는 응답 메시지에 페이로드를 포함하는 경우에 사용된다. (그림 10)은 블록 옵션 값의 포맷을 나타낸다. NUM은 블록 번호를 의미하며, M는 앞으로 남은 블록이 있는 지 여부, SZX는 블록의 사이즈를 나타낸다. 좀 더 정확히는 SZX는 블록 사이즈의 지수를 나타내며, 블록 사이즈는 2^(SZX+4)로 계산된다.

(그림 11)은 블록 옵션 2의 간단한 사용 예제이다. 콜론(:)의 앞의 숫자는 블록옵션 1인지 블록옵션 2인지를 나타내며, 콜론(:) 다음의 숫자는 블록 번호, 슬래쉬(/) 다음은 M(more) 비트, 두 번째 슬래쉬 다음의 숫자는 블록 사이즈를 나타낸다. 이 예제에서 서버는 3개의 블록을 전송하고 있으며, 처음 두 개의 페이로드는 128바이트이고, 마지막 ACK의 페이로드는 1~128바이트 크기이다.

블록 전송 문서에서는 요청 시 블록의 크기를 제한하는 경우, 전송 중 블록 크기를 변경하는 경우, 블록 1을 사용하여 PUT과 같은 요청 메시지에 블록 단위로 페이로드를 전송하는 방법, 블록 1과 블록 2를 혼합하여 사용하는 방법, Observe와 블록 2를 혼합하여 사용하는



(그림 11) 블록 단위 전송 GET 예제

방법 등에 대해 다루고 있으며, 전체 페이로드의 크기를 나타내는 사이즈 1, 사이트 2옵션을 추가로 정의하였다.

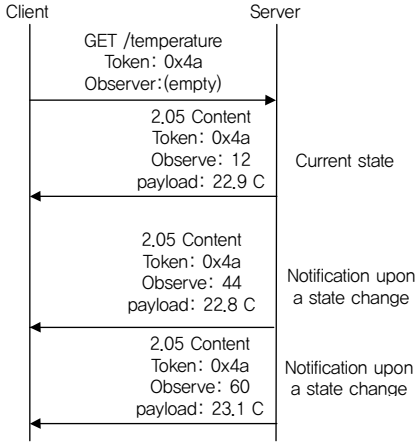
2. Observe

CoAP 서버의 리소스는 시간에 따라 변화할 수 있다. Observe 문서는, 서버의 리소스가 변화할 때마다 클라이언트에게 메시지를 전송할 수 있는 방법을 정의한다. 이를 위해 Observe 옵션을 정의하고 있으며, 요청 메시지에 포함함으로써 리소스가 변화할 때마다 통보를 해달라는 등록 기능을 수행하고, 응답 메시지에서는 통보할 때마다 증가하는 시퀀스 번호 역할을 한다.

(그림 12)는 Observe를 사용하여 온도가 변할 때마다 통보 받는 예제를 보인다. 처음 요청을 전송할 때 질의 문자열을 이용하면, 원하는 조건을 만족할 때만 통보를 받을 수도 있다.

3. 그외 CORE WG 표준 문서

IETF CORE 워킹그룹에서는 앞에서 설명한 표준 문서 이외에 다양한 추가 표준을 개발 중이다. 빌딩 자동화 시나리오의 경우 멀티캐스트를 이용하여 전등을 일시적으로 끈다던가 하는 기능이 필요한데 Group Communication 문서[8]에서는 이러한 IP 멀티캐스트 위에서 CoAP을 사용하는 가이드라인에 대해 설명하고 있다.



(그림 12) Observe 예제

CoAP은 웹 전송 프로토콜이지만, M2M 서비스를 위해서는 리소스의 인터페이스를 정의할 필요가 있다. Core Interfaces[9] 문서에서 이러한 인터페이스를 정의하는 형식과 예제 프로파일을 다루고 있다. 예제 프로파일에서는 기능 세트의 리스트, 디바이스 명세 기능 세트, 센서 기능 세트, 액츄에이터 기능 세트를 보이고 있다.

CoAP은 HTTP를 참조하여 만들었지만, CoAP과 HTTP 간 메시지 변환 및 연동을 위해서는 여러가지 고려사항이 있다. HTTP-CoAP Mapping 문서[10]에서는 URI 매핑에 대한 방법들, 적용 옵션, 프로토콜 변환에 관련된 고려사항 등을 자세하게 다루고 있다.

Links in JSON 문서[11]에서는 CoRE Link Format을 인터넷에서 사용할 때 편리하도록 JSON 으로 나타내는 포맷을 정의한다.

CoRE WG에서는 UDP 기반의 CoAP 표준화가 완료됨에 따라 IETF 87차 회의에서는 Alternative Transport 섹션을 할당하여 UDP 이외의 다른 전송 계층, 예를 들면 SMS, TCP, WebSocket, ASCII 전용 통신 매체 같은 다양한 전송계층 위에서의 CoAP 사용에 관한 논의를 하였다.

Ⅴ. 결론

그 동안 센서 접속 방법은 센서 노드 업체 자체의 프로토콜을 사용하거나, 다양하고 복잡한 산업표준들을 이용하여 왔다. 센서 노드 제조업체는 각각의 시스템에 맞도록 많은 규격과 프로토콜을 구현하여야만 했고, 이로 인해 많은 비용과 시간이 소요되어, 큰 수익을 얻을 수 없었다. 그러나, 이제, 다양한 기기가 서로 연결되어 동작하는 M2M 및IoT 시대가 도래하고 IETF CoAP과 같은 표준화된 프로토콜을 사용할 수 있게 됨에 따라,

쉽게 다양한 시스템과 서비스에 센서 노드가 설치 및 연동될 수 있게 되었다. 이러한 M2M 및 IoT 기술과 서비스는, 인터넷 기술이 우리 생활에 파고들어 온 것처럼, 점차 폭넓고 깊게 확장될 것이다. 이러한 세계적인 흐름에 뒤쳐지지 않기 위해서는, 관련 프로토콜에 대한 활발한 연구개발 및 표준화 참여, 특허 확보가 필요할 것으로 보인다.

용어해설

RESTful REST는 Representational State Transfer(표현 가능한 상태 전송)의 줄임말 이며, 웹과 같은 분산 시스템을 위한 소프트웨어 아키텍처의 한 종류임. HTTP 위에 별도의 계층 없이, HTTP의 GET, PUT, POST, DELETE의 메소드를 사용하여 데이터를 교환함. 데이터의 식별, 즉 리소스의 구조는 URL을 통해 표현됨. 단순하면서도 다양한 기능을 제공하여 손쉽게 서비스를 개발할 수 있음.

약어정리

CoAP	Constrained Application Protocol
CORE	Constrained RESTful Environments
D2D	Device-to-Device
DTLS	Datagram Transport Layer Security
ETSI	European Telecommunication Standards Institute
IETF	Internet Engineering Task Force
IoT	Internet of Things
IPSO	Internet Protocol for Smart Objects Alliance
ITU	International Telecommunication Union
M2M	Machine-to-Machine
IPSO	Internet Protocol for Smart Objects Alliance
MTC	Machine Type Communication
WoT	Web of Things

참고문헌

[1] 고정길 외, “스마트 디바이스와 사물인터넷 (IoT) 융합 기술 동향,” 전자통신동향분석, 2013.

[2] The Internet Engineering Task Force, http://www.ietf.org

[3] Z. Shelby, K. Hartke, and C. Borman, “Constrained Application Protocol (CoAP),” draft-ietf-core-coap-18 (work in progress), IETF, June 2013.

[4] Z. Shelby, “Constrained RESTful Environment (CoRE) Link Format,” RFC6690, IETF, Aug. 2012.

[5] M. Nottingham, “Web Linking,” RFC 5988, IETF, Oct. 2010.

[6] C. Bormann and Z. Shelby, “Blockwise transfers in CoAP,” draft-ietf-core-block-12 (work in progress) IETF, June 2013.

[7] K. Hartke, “Observing Resources in CoAP,” draft-ietf-core-observe-10 (work in progress), IETF, Sept. 2013.

[8] A. Rahman, Ed, E. Dijk, Ed. “Group Communication for CoAP,” draft-ietf-core-groupcomm-16 (work in progress), IETF, Oct. 2013.

[9] Z. Shelby and M.V. Vival, “CoRE Interfaces,” draft-ietf-core-interfaces-00 (work in progress), IETF, June 2013.

[10] A. Castellani et al., “Best Practices for HTTP-CoAP Mapping Implementation,” draft-ietf-core-http-mapping-01 (work in progress), IETF, July 2013.

[11] C. Bormann, “Representing CoRE Link Collections in JSON,” draft-ietf-core-links-json-00 (work in progress), IETF, June 2013.