Zero Configuration Networking 을 구현하는 경량화 사물인터넷 플랫폼에 관한 연구

김동현, 엄경호, 박민지, 엄예림, 박윤미, 송재승*
세종대학교

quri159@gmail.com, rudgh9242@gmail.com, iorw0224@gmail.com, djadpfa1002@gmail.com
yoommooya@gmail.com, jssong@sejong.ac.kr

A Study on the Lightweight IoT Platform with Zero Configuration Networking

Kim Dong Hyun, Eom Kyeong Ho, Park Min Ji, Eom Ye Rim, Park Yoon mi, Song JaeSeung
Sejong Univ.

요 약

사물 인터넷 및 5G 와 같은 산업이 성장함에 따라 사물 인터넷 기반의 다양한 서비스가 등장하고 있다. 그 중에는 엣지 컴퓨팅이라는 기술이 있다. 본 연구는 이 기술을 기반으로 한 Edge 에서 중앙 서버로의 요청이 집중되는 것을 막고, Device 의 관리를 더욱 수월하게 하기 위함을 목적으로 한다. Edge 에서 요청을 처리할 서버는 최대한 경량화 시키는 것을 목표로 하였고, 관리 체계에서는 oneM2M 규격을 채택하였다. 또한, Device 관리에 있어서는 Zero configuration Networking 을 제공하는 Avahi 를 사용하여 중앙 서버의 Device 등록 절차를 더욱 간편히 하였다.

Ⅰ. 서 론

주어진 일과 자원을 효율적으로 분배하는 것은 언제나 더 나은 퍼포먼스를 위해 고려되어야 할 주제였다. 이러한 관점에서 4 차 산업 혁명과 함께 발전한 IoT 와 더불어 Edge Computing 이 주목받고 있다. 여기서 Edge 란 디지털화된 네트워크와 사람 또는 단말기가 연결되는 지점을 뜻한다. 기존 클라우드 컴퓨팅에서는 자원의 분배와 일의 처리는 중앙 집중적이었다. 하지만 Edge Computing 에서는 단말기와 가까운 네트워크 Edge 에서 일부 기능을 수행함으로써 기존 클라우드 컴퓨팅보다 효율적인 자원관리 및 분배를 할 수 있어졌다. 이러한 Edge Computing 에서 활용할 수 있는 프로그램은 다양한 점들이 있을 것이다. 그 이점 중에서 실시간으로 가까운 point 에서 빠른 분석을 할 수 있다는 점을 활용하여 어디에나 설치가 가능한 가벼운 프로그램을 설계하는 것, 즉 경량화에 초점을 두어 개발을 시작하였다. 오브젝트의 상호 관계를 가시적으로 표현해주는 Resource Browser 를 제외한 모든 소스 코드는 C 언어 기반의 오픈 소스를 활용하여 작성하였으며 서버의 주요 로직들은 순수 C 로 직접 작성하였다. 이는 기존 Product 들에 비해 상당히 가볍고 빠른 응답을 기대할 수 있다. 이러한 프로그램은 IoT 기기를 관리를 효과적으로 관리하는 데에 사용할 수 있으며 관리 체계에서는 oneM2M 규격을 채택하였다. oneM2M 오브젝트를 효율적으로 관리하기 위한 Resource Tree 는 기존 Product 들에 비해 해당 Tree 가 메모리에 항시 적재되어 관리되기 때문에 속도면에서도 큰 이점을 가진다. 또한, Edge Computing 에서 Edge 를 중심으로 주변의 기기들을 더 효율적으로 관리할 수 있도록 등록 대행과 Zeroconf 환경을 제공하도록 했다[1]. 여기서 Zeroconf 란, DHCP 환경이 없는 네트워크에서 Peer to Peer 연결이나 Wireless 환경에서 수동 설정 없이 자동으로 네트워킹 할 수 있는 환경을 만들어주는, 일종의 기법이다. 기존의 oneM2M 표준에서는 Device 가 oneM2M Server 에 등록이 되기 위해서 Device 가 자체적으로 등록 절차를 밟아야 한다[2]. 이러한 등록 절차를 Edge 에서 대신 수행해 준다면 Device 들을 더 효율적으로 관리할 수 있다. 그리고 대신 등록을 수행해 주는 과정에서 Zeroconf 를 사용하면 Edge 에서 자동으로 Device 들을 Discovery 할 수 있다. 본 연구에서는 Edge 에서 Zeroconf 를 제공하기 위해 Avahi 를 사용했다[3]. Avahi 란 MDNS 와 DNS-SD 를 복합하여 Linux 환경에서 Zeroconf 기능을 제공하는 오픈 소스 기반 프로그램이다[4]. Avahi 를 사용하면 Local Area Network 에서 Device 들을 Discovery 할 수 있고 Device 의 IP 주소와 연결 포트 번호 등 기본적인 정보를 알아낼 수 있다. Avahi 는 이러한 Discovery 기능 외에도 Publication, Resolution 기능을 추가로 제공한다.

Ⅱ. 본론

oneM2M 은 기존 일정한 규격이 존재하지 않은 상황에서 각 IoT 서비스마다 파편화된 플랫폼 구조로 운영되는 상황을 해결하고자 발족한 표준화 단체이다. oneM2M 플랫폼에서 오브젝트를 주된 4 가지 카테고리인 CSE, AE, CNT, CIN 으로 나누어 관리하며 오브젝트 간에 관계는 계층 구조로 이루어져 있다. oneM2M 경량화 서버에서 주된 기능을 구현하고자 필요한 요소는 HTTP 서버, JSON Parser, DB, Avahi 4 가지로 정의하였다. Architecture 는 Edge 에서 사용될 경량화 Server, 그리고 Edge 와 사용자가 서로 정보를 주고받기 위한 Application, 그리고 Local Area Network 내의 oneM2M Device 로 구성된다.

2.1 경량화 Server

HTTP 서버는 경량화를 위해 최소 기능만을 제공하는 C 언어 기반 오픈 소스를 선택하였다. 해당 서버에서 oneM2M Operation 을 처리하기 위한 로직도 순수 C 언어로 작성하였다. 동일하게 C 언어 기반 오픈 소스를 활용하여 기능들을 구현한 JSON Parser 와 DB 는 단순히 서버 소스 코드에 헤더 파일을 포함하는 것으로 호환할 수 있다. 이처럼

[그림-1] oneM2M Operation

3 가지 요소가 C 언어를 기반으로 하였기 때문에 호환성이 뛰어나며 경량화에 강점을 보인다. [그림-1]은 oneM2M 주요 오브젝트 중 하나인 AE 가 클라이언트의 요청으로 어떻게 서버 내에 생성이 되고 저장되며, 다시 읽을 수 있는지 보여준다.

클라이언트는 HTTP POST 요청을 통해 서버로 Create 요청을 보낸다. Body 에는 AE 생성에 필요한 정보가 JSON 형태로 담겨 있다. 서버는 해당 JSON 을 parsing 후 AE 구조체를 생성한다. 그 후 해당 구조체 정보를 DB 에 저장하는 형태이다. 클라이언트는 HTTP GET 요청을 통해 서버로 Retrieve 요청을 보낸다. 서버는 해당 요청을 받으면 DB 에 저장된 값을 통해서 AE 구조체를 생성한다. 그 후 해당 구조체 정보를 통해 JSON 을 생성하고 클라이언트에게 반환한다. 위와 같은 과정을 통해 구현된 operation 은 C 언어의 경량성 덕분에 다른 프레임워크에서 구현한 oneM2M operation 보다 더 빠른 응답을 기대할 수 있다.

2.2 JSON Parser

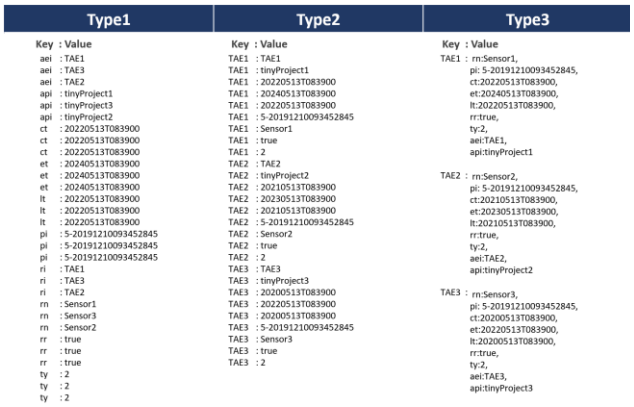
JSON Parser 에서는 경량 라이브러리인 cJSON 오픈 소스를 활용하여 구현하였다. cJSON 은 라이브러리가 단일 파일과 단일 헤더로만 이루어져 있어 importing 이 간단하다는 장점이 있다. 사용자로부터 요청이 온 경우, 서버로부터 전달받은 payload 를 분석하고 그 결과 주어진 객체에 값을 저장한다. 또한 값들을 재조립하여 반환하는 경우, 객체를 oneM2M 표준의 json 포맷으로 변환한다.

2.3 Berkeley DB

본 논문에서 사용한 데이터베이스는 버클리 DB 이다. 빠른 속도와 경량화된 시스템을 추구하는 프로젝트의 취지에 맞게 라이브러리형 데이터베이스를 선택했다. 버클리 DB 는 C 로 작성된 오픈 소스 기반의 작고 빠른 임베디드 데이터베이스이고, 내장형 데이터베이스 엔진으로 C/C++/C#/JAVA 등의 다양한 언어와 다양한 운영체제에서 사용할 수 있다. 일반 관계형 데이터베이스와 달리 버클리 DB 는 SQL 질의문은 지원하지 않는다. 대신 간단한 API 를 이용해 데이터를 저장하거나 조회할 수 있다. 또한 데이터를 key, value 형태로 레코드를 저장하고 관리하는 NoSQL 형식이라서 개발자가 key 와 value 값의 구조를 결정하는 것에 제약이 크지 않다는 것이 장점이다.

본 프로젝트에서는 경량화된 시스템을 위해 oneM2M 의 mandatory data 만 추출하여 DB 에 저장하였다. 다음은 버클리 DB 에서 oneM2M 데이터를 key, value 형식으로 효율적으로 저장하고 관리하기 위한 세 가지 저장 구조를 제안한다.

[그림-2]는 oneM2M 리소스 중 AE 를 세 가지 방식으로 DB 에 저장한 형태이다. Type 1 은 key 에 속성의 이름을 저장하고, value 에 속성값을 저장하는 구조이다. Type 2 는 id 를 key 로 사용하고, value 에 속성값을 저장하는 구조이다. Type 1, Type 2 와 같은 방식으로 저장하기 위해 Cursor 를 이용해 레코드가 들어오는 순서대로 입력받고, set_flag 옵션을 DB_DUP 으로 설정해 같은 key 에 대한 중복 value 값을 허용한다. 구조체 변경 사항이 있을 때 비교적 수정이 수월하다는 장점이 있다. Type 2 의 경우



Type1

Key	: Value
aei	: TAE1
aei	: TAE3
aei	: TAE2
api	: tinyProject1
api	: tinyProject3
api	: tinyProject2
ct	: 20220513T083900
ct	: 20220513T083900
ct	: 20220513T083900
et	: 20240513T083900
et	: 20240513T083900
et	: 20240513T083900
lt	: 20220513T083900
lt	: 20220513T083900
lt	: 20220513T083900
pi	: 5-20191210093452845
pi	: 5-20191210093452845
pi	: 5-20191210093452845
ri	: TAE1
ri	: TAE3
ri	: TAE2
rn	: Sensor1
rn	: Sensor3
rn	: Sensor2
rr	: true
rr	: true
rr	: true
ty	: 2
ty	: 2
ty	: 2

Type2

Key	: Value
TAE1	: TAE1
TAE1	: tinyProject1
TAE1	: 20220513T083900
TAE1	: 20240513T083900
TAE1	: 20220513T083900
TAE1	: 5-20191210093452845
TAE1	: Sensor1
TAE1	: true
TAE1	: 2
TAE2	: TAE2
TAE2	: tinyProject2
TAE2	: 20210513T083900
TAE2	: 20230513T083900
TAE2	: 20210513T083900
TAE2	: 5-20191210093452845
TAE2	: Sensor2
TAE2	: true
TAE2	: 2
TAE3	: TAE3
TAE3	: tinyProject3
TAE3	: 20200513T083900
TAE3	: 20220513T083900
TAE3	: 20200513T083900
TAE3	: 5-20191210093452845
TAE3	: Sensor3
TAE3	: true
TAE3	: 2

Type3

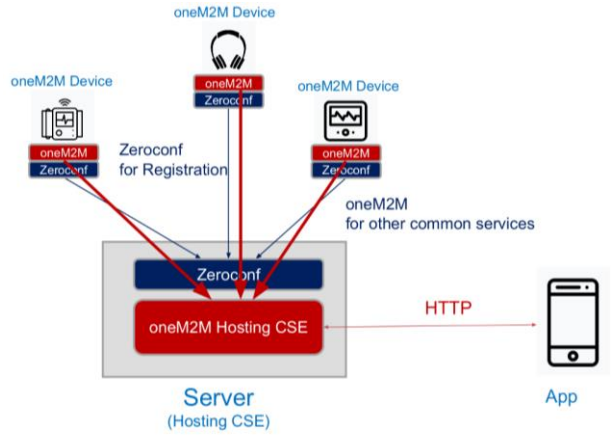
```
Key : Value
TAE1 : rn:Sensor1,
       pi: 5-20191210093452845,
       ct:20220513T083900,
       et:20240513T083900,
       lt:20220513T083900,
       rr:true,
       ty:2,
       aei:TAE1,
       api:tinyProject1

TAE2 : rn:Sensor2,
       pi: 5-20191210093452845,
       ct:20210513T083900,
       et:20230513T083900,
       lt:20210513T083900,
       rr:true,
       ty:2,
       aei:TAE2,
       api:tinyProject2

TAE3 : rn:Sensor3,
       pi: 5-20191210093452845,
       ct:20200513T083900,
       et:20220513T083900,
       lt:20200513T083900,
       rr:true,
       ty:2,
       aei:TAE3,
       api:tinyProject3
```

[그림-2] AE 저장 방식 별 DB 구조

value 값을 저장할 때 별도의 속성 이름을 저장하지 않고 순서로 속성을 구분하는 구조라 속성이 추가되거나 제거되는 등 구조체의 변경 사항이 생기면 코드를 수정하기 힘들다는 단점이 있다. Type 3 는 id 를 key 로 사용하고, value 에 구분자를 이용해 속성값을 나열하는 구조이다. [그림-2]에서는 각 속성을 구분하는 구분자로 콤마를 사용했다. 앞의 두 가지 구조와는 다르게 같은 key 에 중복 value 값이 들어오지 않기 때문에 set_flag 옵션을 DB_DUP 으로 설정하지 않아도 된다. value 값에 속성 이름과 값을 쌍으로 저장하고, 서버에서 Get 요청이 있을 시 문자열을 파싱해 구조체로 반환하는 방식으로 사용된다.

2.4 Zeroconf

Edge Server 에 기존의 oneM2M 표준에서는 존재하지 않는 Zeroconf Discovery 기능을 추가한다. 이 기능은 oneM2M 표준의 filter criteria 에 ZeroconfDiscovery 라는 새로운 옵션을 추가해서 구현된다. Application 에서 Server 로 Zeroconf Discovery 요청을 보내면, Server 에서는 Avahi 를 사용하여 Zeroconf 를 수행한다. 모든 Device 에는 Avahi-Client 가 존재하기 때문에 자신의 IP 주소와 Service Type, port 번호, Host Name, 마지막으로 txt_record 를 통한 txt 메시지를 기본적인 정보를 Local Area Network 내에 BroadCast 한다. 따라서, Server 는 모든 Device 로부터 정보를 얻을 수 있게 되며, 이를 통해 Device 의 정보뿐만 아니라, Local Area Network 내의 Device 들의 List 를 얻을 수 있다. 이 List 를 Response Body 에 넣어서 보내면 Application 은 해당 Edge 주변의 Device 들을 확인할 수 있다. 즉, 자동으로 Device 들을 찾아서 원하는 Device 를 선택할 수 있게 되는 것이다. 추가로 Server 와 Application 사이의 인증 키를 Response Header 에 추가하여 보낸다. Application 은 Server 로부터 받은 Device List 를 보고 등록을 수행할 Device 의 Service Type 과 리소스 네임을 포함한 정보들을 요청 Body 에 넣고, Server 로부터 받은 인증 키를 Header 에 삽입하여 Post 요청을 보낸다. Edge Server 에서는 인증 키를 그대로 가지고 있으므로, 인증 키가 옳은 요청만 Application 에서 온 요청으로 인지한다. 따라서, 잘못된 Client 로부터의 요청을 막는 효과를 기대할 수 있게 되는 것이다. 올바른 인증 키를 보낸 요청에만 AE 리소스와 CNT 리소스를 생성하고 Device 의 등록을 완료한 후, 등록 완료 메시지를 응답한다. 마지막 단계는, Device 가 자신의 측정값을 주기적으로 Server 로 보낼 수 있도록 Server 에서 관련 요청을 보내는 것이다. 이때는 Avahi 를 통해서 얻어낸 Device 의 URL 로 Server 에 생성된 리소스의 정보와 등록 완료 메시지를 담아 요청을 보낸다. Device 는 Server 로부터 받은 리소스의 URL 로 주기적으로 측정값을



[그림-3] Zeroconf Architecture

보내게 되며, Avahi 의 Publication 기능 수행을 차단하여 Broadcast 를 중지한다.

2.5. Resource Browser

TinyIoT 는 웹 기반 통합 관리 도구, Resource Browser 를 제공한다. Resource Browser 를 통하여 TinyIoT 플랫폼 활용의 개발자 편의성을 지원한다. 해당 Browser 가 제공하는 서비스로는 리소스 브라우저 및 디바이스 관리가 있다. 리소스 브라우저 서비스는 TinyIoT 서버에 저장된 리소스를 Tree 구조로 시각화하여 내부 구조 파악에 용이하게 한다. 디바이스 관리 서비스로 URI 를 이용하여 리소스에 접근할 수 있어 리소스 관리가 용이하며 REST API 를 통해 새로운 리소스를 생성하거나, 기존 리소스를 조회 혹은 삭제할 수 있다. [그림-4]은 TinyIoT 의 Resource Browser UI 이다.

Ⅲ. 결론

위와 같이 oneM2M 플랫폼 구성 요소들을 구현한다면 지금까지 공개된 다른 플랫폼들에 비해 경량화에 강점을 보일 수 있다. 또한 본론에서 제시한 기능들 외에도 Avahi 를 더 응용하고 필요한 기능에 대해서 더 생각해본다면, 구현할 수 있는 기능은 더 많이 존재한다. Avahi 의 txt-record 사용하여 Device 가 제공하는 정보를 Txt 형식으로 원하는 대로 Broadcast 할 수 있다. 그리고 Avahi 의 Publication 기능을 조정할 수 있다는 점을 이용하여 등록 취소 또한 구현할 수 있을 것이다. 결과적으로 본 연구는 기존 플랫폼에 비해 경량화에 강점을 보이며 새로운 기능을 추가하였다. Zeroconf 기능을 추가하여 원하는 Device 를 자동으로 찾아서 등록해줌으로써 등록 대행 기능을 제공하게 되었다.

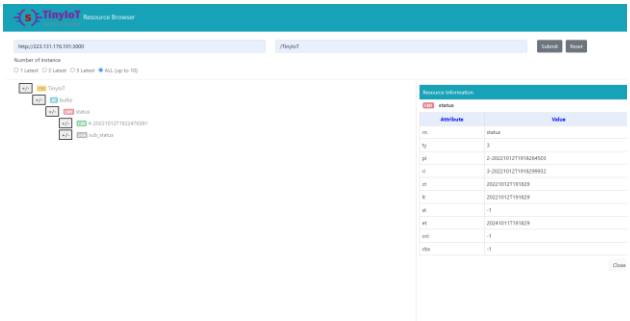
ACKNOWLEDGMENT

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT 연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2022-2021-0-01816). 교신저자 송재승 교수.

참 고 문 헌

[1] Steinberg, Daniel/ Cheshire, Stuart, "Zero Configuration Networking: the Definitive Guide"

[2] J. Swetina, G. Lu, P. Jacobs, F. Ennesser and J. Song, "Toward a standardized common M2M service layer platform: Introduction to oneM2M," in *IEEE Wireless Communications*, vol. 21, no. 3, pp. 20-26, June 2014



[그림-4] TinyIoT Resource Browser

[3] K. Cao, Y. Liu, G. Meng and Q. Sun, "An Overview on Edge Computing Research," in IEEE Access, vol. 8, pp. 85714-85728, 2020

[4] A. J. Jara, P. Martinez-Julia and A. Skarmeta, "Light-Weight Multicast DNS and DNS-SD (lmDNS-SD): IPv6-Based Resource and Service Discovery for the Web of Things," 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2012, pp. 731-738