논문 18-43-03-08 The Journal of Korean Institute of Communications and Information Sciences '18-03 Vol.43 No.03
https://doi.org/10.7840/kics.2018.43.3.529

에너지 IoT 환경에서 실시간 수요반응 서비스를 위한 CoAP Observe 기반 Push Mechanism 설계 및 분석

박 현 진◆, 박 헌 일*, 이 성 환*, 최 진 식°

Design and Analysis of Push Mechanism Based on CoAP Observe for Demand Response in Energy IoT Environment

Hyun Jin Park◆, Heon Il Park*, Sung Hwan Lee*, Jin Seek Choi°

요 약

본 논문은 스마트 홈 및 가전 중심의 Energy Internet of Things(EIoT)분야에서 에너지 공급자와 소비자 입장에서 만족도를 높이며 에너지 사용 효율을 높일 수 있는 실시간 수요반응(Demand Response; DR) 기술의 연구에 있다. 본 논문의 목적은 EIoT 환경에서 다수의 스마트 홈 기기들에 대하여 실시간 에너지 관리를 위한 Constrained Application Protocol(CoAP) Observe 기반 실시간 수요반응 프로토콜의 설계 및 분석에 있다. 우선 실시간 수요반응 서비스를 위해 Open Automated Demand Response(OpenADR) mechanism을 분석하고 실시간 상호작용 기법을 제시한다. 다음은 CoAP/JSON 기반 Observe 기능을 활용해 OpenADR Push mechanism을 설계한다. 끝으로 구현을 통한 성능 분석 및 Pull mechanism과 성능 비교를 통해 Push mechanism이 가질 수 있는 성능과 실시간 서비스 측면에서 가질 수 있는 장점을 제시 한다.

Key Words : Demand Response, OpenADR 2.0b, Energy IoT, Push mechanism, Pull mechanism

ABSTRACT

This paper studies on realtime Demand Response(DR) mechanism for improving energy efficiency in smart home and home appliance-based Energy Internet of Things(EIoT) environment while preserving quality of experience for energy producer and consumer. The objective of this paper is the design and implementation of real-time DR protocol using Constrained Application Protocol(CoAP) / JSON based Observe capability in order to provide real-time energy management of smart home and appliances in EIoT. At first, this paper studies on Open Automated Demand Response(OpenADR) mechanism and shows the necessary of real-time interactive demand response and energy management services in EIoT environment. Next, this paper designs OpenADR push mechanism using CoAP Observe capability. Finally, this paper implements and compares the push and pull mechanisms to verify the advantages of the proposed push mechanism in terms of performance and efficiency in real-time environment.

※ 본 연구는 2017년도 산업통상자원부 및 산업기술평가관리원(KEIT)의 지원에 의해 수행되었습니다('10053671').
◆ First Author : Hanyang University Department of Computer Science, phj3372@hanyang.ac.kr, 학생회원
° Corresponding Author : (ORCID:0000-0003-1554-3879)Hanyanng University Department of Computer Science, jinseek@hanyang.ac.kr, 종신회원
* Hanyang University Department of Computer Science, heonil8@nate.com, zmagician@naver.com
논문번호 : KICS2017-09-283, Received September 30, 2017; Revised December 20, 2017; Accepted December 20, 2017

Ⅰ. 서 론

다양한 전력기기의 발전으로 전 세계적 전력 사용량은 점차 증가하고 있다. 특히 여름철이나 겨울철 냉난방기기 등의 사용량 급증에 따른 발전과 수요의 불균형은 추가적인 발전소 건립, 전기요금 폭탄, 정전사태와 같은 심각한 문제를 야기할 수 있다. 뿐만 아니라 날로 치솟는 유가와 함께 지속적인 온실가스의 증가에 따른 지구 온난화 문제는 심각한 수준에 도달하면서 효율적인 에너지 사용에 대한 관심이 날로 커지고 있다.

에너지 생산자 입장에서 에너지를 효율적으로 사용하고 안정적으로 관리하기 위해 전력망의 '수요반응'이라는 개념이 등장하였다[1]. 수요반응은 에너지 생산량에 맞추어 소비량을 미리 설정해 수요가 공급을 넘지 않도록 하는 에너지 소비의 지능화 기술이다. 에너지 생산자와 에너지 서비스 공급자 혹은 공장, 빌딩에게 전기 판매 및 전반적인 에너지 관련 정보를 교환하기 위한 프로토콜로 OpenADR Alliance[2]에서 표준화된 Open Automated Demand Reponse(OpenADR)가 제안되었다[3]. OpenADR 프로토콜은 현재 국내외에서 많은 유틸리티, 공급자, 소비자(e.g. Building, Factory) 등이 수요반응 표준 프로토콜로 사용하고 있다.

최근 스마트 가전을 통한 에너지 절약 기술이 발전하면서 소비자 입장의 에너지 절약과 요금 부담을 줄이기 위한 에너지 관리 및 수요 반응 연구가 활발히 진행되고 있다. 특히 스마트 홈 분야에서 모든 사물이 장소에 구애 없이 인터넷을 통하여 정보를 주고받으면서 제어할 수 있는 Internet of Things(IoT) 통신환경이 제공 되면서 스마트 홈에서 수요반응 기술이 급격히 발전하고 있다. 스마트 홈에서 수요반응 기술은 IoT 통신 기술과 스마트 가전 기술이 결합되면서 효율적 에너지 관리를 위한 Energy IoT(EIoT) 기술로 발전되고 있다.

그러나 기존 Utility와 Provider 및 Provider와 Customer 간 수요반응 표준 프로토콜로 사용되는 OpenADR 2.0 b은 HyperText Transport Protocol /eXtensible Markup Language(HTTP/XML) 기반의 통신방식이다. OpenADR2.0b HTTP/XML 기반 통신 방식은 많은 수의 초경량 디바이스들이 연결된 EIoT 환경에서 사용될 경우 대용량 트래픽의 발생으로 에너지 소모 등 큰 오버헤드를 유발할 수 있다. 또한 정전 시와 같은 긴급 상황에서도 OpenADR의 전송 mechanism은 Pull mechanism으로 수요반응 이벤트(DR Event) 메시지를 이용하여 주기적으로 변경된 정보를 확인하는 방식으로 되어 있어 변경된 정보를 확인하는데 시간이 많이 소요될 수 있다. 이에 따라 스마트 가전 기기 들의 실시간 에너지 사용 정보를 활용하여 효율적이면서도 EIoT 환경과 초경량 기기에 적합한 실시간 수요반응 전송 mechanism이 필요하게 되었다.

본 논문의 목적은 EIoT 환경에 맞는 경량화 된 CoAP Observe기능을 활용한 실시간 수요반응 프로토콜에서 Push mechanism의 설계 및 분석에 있다. 특히 CoAP 실시간 수요반응 프로토콜을 설계하고 구현을 통해 검증한다. 제안된 프로토콜은 스마트 홈 및 가전기기들에 대한 실시간 수요반응 서비스를 위해 HTTP/XML 기반의 OpenADR mechanism을 경량화한 CoAP/JSON 기반 수요반응 mechanism을 제공한다. CoAP/JSON 기반 Observe 기능을 활용해 OpenADR Push mechanism을 제공함으로써 기존의 Pull mechanism이 갖는 대용량 트래픽의 발생과 실시간 서비스의 한계를 극복 한다. 본 논문에서는 실험을 통해 CoAP/JSON 방법과 HTTP/XML 기반의 OpenADR 방식을 비교하여 트래픽 발생량이 현저하게 줄어들고 Push mechanism이 Pull mechanism에 비해서 보다 빠른 응답 속도를 가진다는 것을 증명한다. 끝으로 CoAP 기반 Push mechanism은 빠른 데이터 처리속도로 인해 스마트 에너지 홈과 EIoT에서 수요반응 mechanism에 적합하고 실시간 전송 mechanism을 통해 에너지 관리 효율성 측면에서 Push mechanism이 가질 수 있는 성능과 이점을 제시한다.

본 논문의 구성은 다음과 같다. Ⅱ 장에서는 수요반응, OpenADR, Push과 Pull mechanism 특징, CoAP, EMS로 EIoT 관련 기술 및 연구에 대해 소개한다. Ⅲ 장에서는 CoAP를 기반으로 한 Push mechanism 설계와 JavaScript Object Notation(JSON) 포맷을 이용하여 연구한 OpenADR2.0b 프로토콜의 포맷에 대하여 소개하고 해당 프로토콜과 Push, Pull mechanism 비교 분석 및 검증을 위한 실험을 소개한다. 마지막으로 Ⅳ 장에서 결론과 Ⅴ 장에서 향후 과제를 기술한다.

Ⅱ. 본 론

2.1 Energy IoT Environment에서 Demand Response 관련 기술

2.1.1 Energy IoT

스마트그리드는 기존의 전력 시스템에 정보통신기

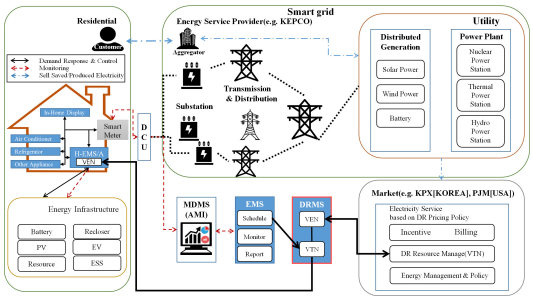

그림 1. Smart Grid와 Energy IoT 구조
Fig. 1. Structure of Smart Grid and Energy IoT

술(Information and Communication Technology; ICT)을 접목하여 기존의 단방향 전력망에서 에너지 공급자와 소비자가 양방향으로 전력 생산 및 사용에 대한 정보를 교환함으로써 전력망 공급망의 효율적인 생산 및 분배가 가능하도록 하는 차세대 지능형 전력망을 의미한다. 그림 1과 같이 EIoT 기술은 스마트그리드 환경과 연결된 스마트 홈 IoT 환경에서 에너지 소비자들의 최대 전기 사용량의 차이를 완화시키고, 전기 공급과 수요의 균형을 맞추기 위하여 일시적으로 전기 사용량을 축소시키기 위해 소비자의 전력 사용 패턴의 변화를 유도하는 에너지 사용 최적화 기술을 말한다. 즉, EIoT 기술은 가정이나 사용자 입장에서 수많은 종류의 전력 기기(에어콘, 냉장고, 에너지저장장치 등)들이 에너지 관리 시스템(Energy Management System; EMS)과 수요반응 시스템에 연동되어 에너지를 효율적으로 모니터링하고 거래 및 관리하기 위한 연결 기술이다.

2.1.2 에너지 관리 시스템(Energy Management System; EMS)

EIoT 환경에서 수많은 종류의 기기들의 에너지의 소비량, 공급량 등을 효율적으로 관리하기 위해서는 EMS가 필요하다. 일반적인 EMS는 정보통신기술을 이용하여 에너지 소비자가 에너지방침, 에너지 효율 목표 설정 및 목표를 달성하기 위한 프로세스 및 절차 수립을 위한 시스템을 의미한다. EMS는 수요처 및 용도에 따라 다양한 형태가 존재 한다. 가정용 EMS(Home EMS; HEMS)은 단독 주택 또는 공동 주거 공간에서의 개별 가구 내 소비자 관점에서 에너지 관리 서비스를 제공한다. 건물 EMS(Building EMS; BEMS)은 건물 관리자가 사용자의 쾌적하고 기능적인 업무환경을 효율적으로 유지/관리하기 위해 정보통신 기술을 이용하여 합리적인 건물 에너지 사용이 가능하도록 건물에너지 제어/관리/경영 통합 시스템이다. 공장 EMS(Facility EMS; FEMS)은 공장 내 에너지 소비량과 시스템 운전 상태 등을 모니터링 한 후, 소비패턴 분석 및 시뮬레이션을 통해 비효율적으로 운전 중인 설비 및 시스템을 파악하여 최적의 EMS을 구축함으로서 공장 내의 에너지 손실을 제거 및 개선하여 에너지 비용 절감과 온실 가스 감축을 도모하는 에너지 절약 시스템이다. 가정이나 빌딩 등에서 에너지 사용 효율을 높이기 위해 에너지 관리 서비스를 위해 제3 자 공급 업체(또는 "제3 자 공급 업체")에서 에너지 관리 기능을 제공하는 에너지 관리 에이전트(Energy Management Agent; EMA) 기술이 있다. EMA란 EMS의 에이전트 역할로 ISO / IEC 15067-3에서 정의된 개념이다.[5] 에너지 제공자(Utility)가 에너지 관리를 위해 가정 내 기기들을 통해 에너지 효율을 높이기 위해 EMA를 통한 간접 부하 제어 방법을 사용할 수 있다. 또한 인센티브 기반 및 간접적인 방법을 사용하는 수요 관리 방법을 수행할 수 있다. 지정된 시간의 사용 시간 가격 또는 실시간 가격이나 보류중인 공급 제한에 대한 이벤트 통지가 포함된다. 즉, 가정 내 다양한 EIoT 기술을 사용하는 EMA는 수요반응을 통해 에너지 수급 상황 변화에 대응 하는 유연한 구조를 가질 수 있도록 한다.

2.1.3 수요반응(Demand Response; DR) 시스템

수요관리는 소비자의 전기사용 패턴을 합리적인 방향으로 유도하기 위한 전력회사의 제반활동으로 정의된다. 전력공급 설비의 확충에 중점을 두어 온 종전의 공급 측 관리와는 대응되는 개념이다. 수요반응은 수요관리의 하위개념이며, 전력수요 피크 등의 이유로 인해 수급위기 발생 시 전기 요금의 조정이나 부하감축 지시에 의한 전력절감 등을 통해 수급의 균형을 유지하기 위한 활동이다. 수요반응의 목적은 크게 최대수요의 억제, 최대부하의 이전, 기저부하의 증대로 구분될 수 있다.

수요반응의 종류는 가격 기반 수요반응과 인센티브 기반 수요반응으로 나눌 수 있다. 가격기반 수요반응은 소비자가 지불하는 전력가격을 시간대 별로 차등하여 적용하고, 이에 소비자가 반응하여 전력의 사용

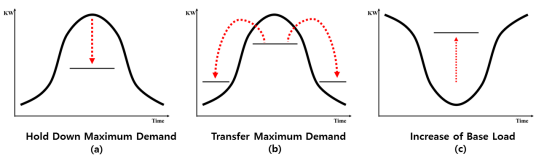

그림 2. 수요반응 목적
Fig. 2. The purposes of Demand Response

표 1. 수요반응 종류
Table 1. Sort of Demand Response

Sort	Subclass
Price - Based DR	TOU, Time-of Use Pricing
	CPP, Critical Peak Pricing
	RTP, Real-time Pricing
Incentive - Based DR	Direct Load Control
	Interruptible/Curtailable Service
	Demand Bidding
	Emergency Demand Response
	Capacity Market
	Ancillary Service

패턴을 변화시키는 방식의 수요반응 프로그램이다. 인센티브 기반 수요 반응은 전력공급자 또는 전력시스템 운영자에 의해 운용되는 제도로 수요반응 프로그램에 참여할 소비자를 미리 모집하여 필요할 경우 참여자로 하여금 전력 사용을 줄이게 하고 인센티브를 지급하는 시스템이다.

수요반응 시스템(DR Management System: DRMS)의 역할은 전력 생산량을 초과하는 최대 수요의 발생을 억제하는데 있다. 그림 2처럼 수요반응 시스템은 피크 시간대의 부하를 보다 부하가 낮은 경 부하시간대로 이전하며 경 부하 시간대의 수요를 증대하도록 유도한다. 이를 위해 다양한 전력 기기 내의 에너지 사용 정보를 얻거나 다양한 에너지 관련 기기나 시설에 대한 고유 인증절차를 수행하고, 에너지 사용을 제어하는 명령을 전달하기 위해 높은 수준의 보안성과 개방성이 요구된다. 따라서 다양한 전력 기기들이 쉽게 연동하고 안전하게 정보를 전달할 수 있는 표준화된 수요반응 프로토콜이 반드시 필요하다. 또한 모든 전력 기기를 수요반응 기반의 플랫폼화 하여 공공 시설물이나 전력 기기에 부착하여 상호 연동할 수 있어야 한다.

2.1.4 OpenADR 2.0b

OpenADR은 Open Alliance에서 개발한 지능형 수요반응(Demand Response, DR) 서비스를 위한 HTTP/XML 기반의 표준 프로토콜이다. 2.0버전은 다수의 벤더 수용, 인증 테스트 도구, Organization for the Advancement of Structured Information Standards (OASIS) 기반 국제 표준으로 대다수 수요 반응 프로그램에 적용되는 장점을 가지고 있다. OpenADR에서 수요반응 통신 참여자는 Virtual Top Node(VTN)와 Virtual End Node(VEN)로 구분된다. VTN은 수요반응의 이벤트를 제공하는 역할을 하고 End Device 혹은 중간 서버에게 OpenADR Signal을 제공한다.

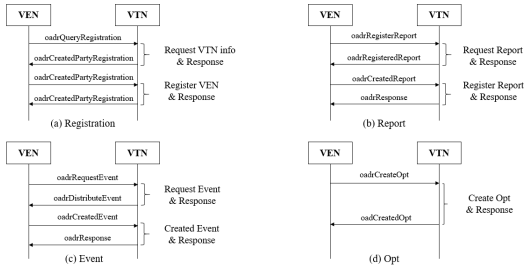

그림 3. OpenADR2.0b 동작과정
Fig. 3. OpenADR2.0b Process Movement

OpenADR2.0b는 그림 3 (a)-(d)와 같이 총 네 가지의 서비스를 지원한다. (a) VTN과 VEN이 서로의 정보를 교환하고 ID값을 발급하여 연결을 수립하는 Registration 서비스, (b) VTN과 VEN의 보고능력을 알려주고 구독을 신청하는 Report 서비스, (c)수요자원의 사용 패턴 변화를 유도하기 위한 신호 메시지를 보내는 이벤트 서비스, (d)이벤트 스케줄에 대한 수용가능여부를 정의하는 Opt 서비스가 있다. 그림 3는 각 서비스의 동작 과정을 보여준다. 단, 본 논문에서는 그림 3. (a) Registration, 그림3. (b)의 Report 등록 과정과 그림 3. (c)의 이벤트 서비스를 확장하고, 확장 기능을 이용한 프로토콜 성능을 측정한다.

2.1.5 OpenADR 2.0b Push and Pull Mehcanism

표 2는 OpenADR2.0b 프로토콜에서 Push와 Pull 방식에 대한 비교이다. Pull 방식은 메시지 송수신에 지연이 있으나 수동적으로 설정 할 수 없는 많은 수신자가 있을 때 적절한 방식이다. 반면 Request 메시지를 받았을 경우에만 메시지를 전송할 수 있기 때문에 실시간 통신 방식에는 적합하지 않다. Push 방식은 수신자가 네트워크 방화벽 뒤에 있는 경우 기술적인 어려움이 있으나 실시간 통신 방식에 적합하고 데이터 트래픽 양을 줄이며 Pull request의 전송 지연 시간을

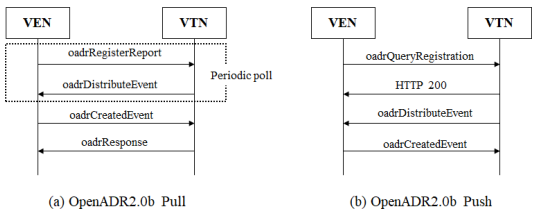

그림 4. OpenADR2.0b Pull, Push 메시지 플로우
Fig. 4. OpenADR2.0b Pull, Push Message Flow

표 2. Push and Pull 특징[4]
Table 2. Push and Pull Features[4]

	Push	Pull
features	Delivery is immediate(except when delivery is being rate-limited to avoid overwhelming the endpoint). There is no added latency from pull requests.	Delays will occur between message publication and delivery.
	Delivers one message per request and limits maximum number of outstanding messages.	Achieves high throughput at low CPU and bandwidth by allowing batched delivery and acknowledgments as well as massively parallel consumption. May be inefficient if aggressive polling is used to minimize message delivery time.
Recommended for	Subscribers with low traffic	Large numbers of subscribers which are created dynamically and cannot be manually configured
	Subscribers that need closer to real-time performance	Requiring configuration with SSL certificates and a web server for push subscription

줄일 수 있는 장점이 있다.

2.1.5.1 OpenADR 2.0b Push mehcanism

Push mechanism은 이벤트가 발생 될 경우 Server(VTN)에서 즉각적으로 Client(VEN)으로 메시지를 전송하기 때문에 실시간 이벤트 전송에 적합하다. 즉각적인 전력 소비 관리 및 요금 실시간 모니터링을 가능케 하며 효과적인 에너지 관리 및 실시간 스케줄링이 가능하다. 또한 수요반응 중 하나인 실시간 차등적 요금제 관리를 가능하게 한다. 나아가 Pull mechanism의 주기적인 Polling 메시지가 없기 때문에 부하가 적어진다.

Push mechanism의 단점으로는 Request-Response 방식의 통신을 하는 구조에서는 VTN과 VEN 양쪽에 Server와 Client를 모두 구현해야 하며, VEN이 방화벽 뒤에 있다면 통신에 어려움이 생긴다. 이를 해결하기 위해서는 Publish-Subscribe 방식의 통신을 하고 TCP 세션을 항상 유지하여 Server에서 Client로 메시지가 전달되도록 해야 한다. 하지만 이 경우에는 각 장치별 자원 사용량이 늘어나게 되는 단점이 발생한다.

2.1.5.2 OpenADR 2.0b Pull mehcanism

OpenADR2.0b에서 Pull mechanism은 주기적으로 VEN에서 이벤트에 대한 Request를 보내면 VTN이 Response를 보내주는 방식이다. VTN와 VEN 사이에 TCP세션을 유지할 필요 없어 각 장치의 리소스 사용에 효율적인 통신을 할 수 있다. 또한 Pull mechanism은 VEN에 방화벽이 있어도 통신이 가능하다. 일반적으로 Pull mechanism은 실시간으로 데이터를 제공하지 않으며 Polling 메시지 간 간격이 존재하여 정보를 수신하기 위해서는 Polling 주기만큼의 대기 시간이 필요하며, 다수의 기기들과 통신할 경우 Request-Response 지연시간 문제의 가능성이 내제 되어있다. VEN으로부터 Request메시지를 받았을 경우에만 VTN이 이벤트를 전달해 줄 수 있으므로 실시간 통신 방식에 적합하지 않으며 VEN의 수가 늘어날 경우 각 VEN의 주기적인 Polling(Request)메시지에 의하여 VTN의 부하가 급격하게 커진다. 따라서 EIoT 환경에서 수요관리 사업자가 다양한 요구조건의 수요자를 관리하기에는 적합하지 않은 방식이다.

2.1.6 Constrained Application Protocol (CoAP)

CoAP는 Application 계층에서 사용되는 자원이 제한된 기기와 자원이 제한된 센서 등 IoT 네트워크에서 효율적인 메시지 교환을 위해 개발된 프로토콜이다. 기본적으로 User Datagram Protocol(UDP)의 비동기식 전송 방식을 취하여 Transmission Control Protocol(TCP)를 이용하는 다른 IoT 프로토콜에 비해 더 가볍고 부하가 적다. 반면 TCP에 비해 신뢰성이 낮지만 재전송 및 타이머 관리를 옵션으로 포함하고 있고 보안을 위해 Datagram Transport Layer Security(DTLS) 계층을 사용할 수 있다. CoAP는 신뢰성 있는 통신을 위하여 메시지에 확인형(Confirmable, CON), 비 확인형(non-confirmable, NON), 승인(acknowledgement, ACK), 리셋(reset, RESET)과 같이 네 가지의 메시지 타입을 정의한다. 신뢰성 있는 전달을 위해서 CON 메시지를 전송하며, CON 메시지에 포함 된 메시지 ID는 ACK 메시지에 동일하게 들어가게 된다. 수신 노드가 CON 메시지에 대한 처리를 할 수 없을 경우에는 ACK 메시지 대신 RESET 메시지를 보낸다. 더불어 CoAP는 기본으로 일대일(1:1) 방식을 취하지만, IPv6 환경에서는 일대다(1:N) 혹은 다대다(N:N) 멀티캐스트를 지원할 수 있다. 또한 HTTP와 유사하게 Request-Response 패러다임을 제공하고 Restful API와 유사하게 이용하기 때문에 기존의 HTTP 웹 프로토콜과 상호 운용이 용이하다. CoAP에서 지원하는 Observe 기능은 클라이언트가 서버에게 자원의 상태를 알려줄 것을 요청하는 옵션이다. 클라이언트는 서버의 관심 있는 자원을 등록할 수 있고, 이를 통해 자원의 변화가 생길 때 별도의 요청 메시지 없이 Server측에서 변경 된 내용을

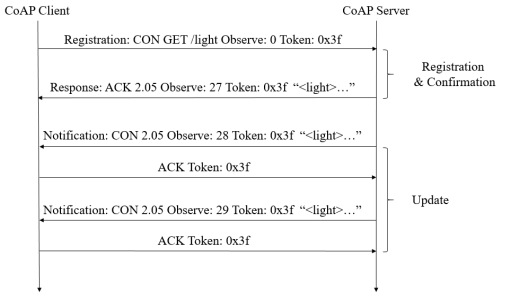

그림 5. CoAP Observe 동작 과정
Fig. 5. Process Movement of CoAP Observe

알림(Notification)으로 전송해 줄 수 있다[5]. 그림 5는 온도에 대한 정보를 주고받는 Observe Process 예시이다. Client에서 서버에게 GET형태의 메시지로 Observe를 요청하여 Registration을 수행한다. 이후 온도에 대한 정보가 update 되었을 때 변경 된 정보를 추가적인 요청 없이 Server에서 Client에게 알림을 준다.

2.2 실시간 수요반응 서비스를 위한 CoAP Observe 기반 Push mechanism 설계

2.2.1 CoAP 기반의 실시간 수요반응 프로토콜의 개요

본 논문은 CoAP 기반의 수요반응 프로토콜[1]을 실시간 수요반응이 가능하도록 확장한 프로토콜이다. CoAP 기반의 실시간 수요반응 프로토콜은 실시간 서비스를 위해 기존의 Pull mechanism[1] 외에 Push mechanism을 추가 확장한다. Push mechanism은 기존의 수요반응 registration 과정에 Observe 등록과정이 추가된다. Observe registration 과정에서 클라이언트가 서버의 관심 있는 자원을 등록한다. 기존 Pull mechanism에서 등록된 자원이 갱신될 때 주기적 polling 메시지를 통해 자원의 상태를 확인하던 것을 Push mechanism에서는 Observe update 과정을 활용해 서버가 클라이언트에게 등록된 자원의 상태 변화를 실시간으로 알려준다. Observe 등록과정과 Observe update 과정은 다음 3.2.1과 3.2.2 절에 자세히 기술한다.

2.2.2 Observe 기능을 이용한 Push mechanism 설계

2.2.2.1 CoAP Observe 등록과정

Observe 등록과정은 그림 6 같다. 그림 6의 (a)와 같이 일반적인 OpenADR2.0b VTN과 VEN의 등록과정을 마친 후 CoAP Observe 옵션을 사용하기 위해서

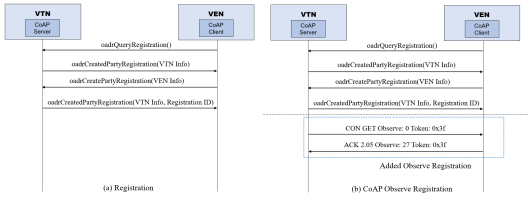

그림 6. 기존 Registration 과정과 Observe을 활성화하기 위해 추가된 Registration 과정
Fig. 6. Original Registration Process and Added Registration Process for activating Observe

그림 6의 (b)와 같이 기존의 등록과정에 추가된 등록과정을 거친다. 추가 등록과정은 먼저 VEN(CoAP Client)이 VTN(CoAP Server 측)에게 CON 형태의 메시지를 이용하여 수요반응 상태에 대해 전달받을 이벤트 자원를 등록하고, VTN은 VEN에게 ACK 메시지 형태로 응답 메시지를 보낸다.

2.2.2.2 CoAP Observe Update 과정

Observe update 과정은 그림 7 같다. 그림 7의 (a)와 같이 일반적인 OpenADR2.0b VTN과 VEN의 Pull mechanism의 이벤트 생성 및 처리를 위한 polling update 과정이다. 이 방식에서 VEN는 주기적인 polling 메시지를 활용하여 자신과 관련된 이벤트가 있는지 확인한다. VTN은 이벤트가 있을 때 polling 메시지에 대한 응답으로 Distribute Event 메시지를 전송한다. 이벤트가 없을 경우는 ACK 만을 보낸다. 그림 7의 (b)는 Observe update 과정으로 등록된 이벤트가 발생할 경우 VTN이 Distribute Event를 해당 VEN에게 즉시 보낸다. 해당 VEN은 Created Event로 응답한다.

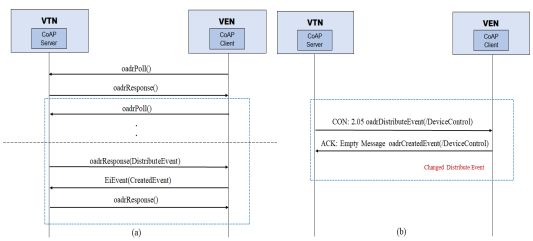

그림 7. Pull 과 Push mechanism의 메시지 플로우 비교
Fig. 7. Comparison between Pull and Push Mechanism message flow

2.2.3 CoAP 기반의 실시간 수요반응 프로토콜의 포맷

본 논문에서 사용하는 CoAP 프로토콜은 기존 프로토콜의 XML의 대용량의 데이터의 모델링을 이용하지 않고 제한된 네트워크에서 가볍고 빠른 통신을

표 3. XML과 JSON 포맷의 Distribute Event Payload
Table 3. Payload of Distribute Event XML and JSON format

Format	Payload	Data
XML	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <nsl:oadrPayload xmlns:nsl="http://openadr.org/oadr-2.0b/2012/07" xmlns:ns2="http://docs.oasis-open.org/ns/energyinterop/201110" xmlns:ns3="http://docs.oasis-open.org/ns/energyinterop/201110/payloads" xmlns:ns4="urn:ietf:params:xml:ns:icalendar-2.0"> <nsl:oadrSignedObject> <nsl:oadrDistributeEvent ns2:schemaVersion="2.0b"> <ns2:eiResponse> <ns2:responseCode>200</ns2:responseCode> <ns2:responseDescription>OK</ns2:responseDescription> <ns3:requestID>2e4327de62e02151b0f2</ns3:requestID> </ns2:eiResponse> <ns3:requestID>2b3dddfd214b55be2b4e</ns3:requestID> <ns2:vtnID>EPRI_VTN</ns2:vtnID> <nsl:oadrEvent><ns2:eiEvent> <ns2:eventDescriptor> <ns2:eventID>cadb39168ed73352b0f8</ns2:eventID> <ns2:modificationNumber>0</ns2:modificationNumber> <ns2:modificationReason> </ns2:modificationReason> <ns2:priority>0</ns2:priority> <ns2:eiMarketContext> <ns5:marketContext> </ns2:eiMarketContext> <ns2:createdDateTime>2017-02-09T06:41:17.000Z</ns2:createdDateTime> <ns2:eventStatus>far</ns2:eventStatus> <ns2:testEvent>false</ns2:testEvent> <ns2:vtnComment> </ns2:vtnComment></ns2:eventDescriptor> <ns2:eiActivePeriod> <ns4:properties> <ns4:components></ns2:eiActivePeriod> <ns2:eiEventSignals> <ns2:eiTarget> <ns2:venID>cc0853370e25e04d58e3</ns2:venID> </ns2:eiTarget></ns2:eiEvent> <nsl:oadrResponseRequired> </nsl:oadrEvent> </nsl:oadrDistributeEvent> </nsl:oadrSignedObject> </nsl:oadrPayload>	995 bytes
JSON	{"Responsedescription":"MIR","EndTime":"1620","RequestID":1,"StartTime":"1619","Service":"oadrDistributeEvent","EndYMD":"20170516","Response":100,"Value":100,"OptType":"optIn","StartYMD":"20170516","EventID":1,"ModificationNumber":0,"TargetVEN":MIR_VEN1"}	313bytes

목적하는 JSON 방식으로 데이터 모델링하여 설계하였으며 예시는 표 3과 같다. 표 3는 이벤트가 발생할 경우 서버가 보내는 Distribute Event 메시지의 예이다. 메시지 포맷의 비교를 위해 OpenADR2.0b 기반의 HTTP/XML 메시지와 본 논문에서 제안한 CoAP/JSON 방식의 메시지를 나타냈다. 본 논문에서 활용된 이벤트 시작 시간, 끝 시간, 목표 VEN, 수요반응 레벨 등은 데이터 모델링은 CoAP/JSON 기반 OpenADR2.0b[1]과 같은 Undefined Modeling Language(UML) 데이터 모델링을 따랐다.

Ⅲ. 실 험

3.1 실험 환경

본 논문의 실험을 위한 전체 EIoT 구조는 그림 8과 같다. EIoT 구조에서 에너지 공급자(Power Utility)는 EMS을 통해 에너지를 효율적으로 관리 모니터링 하면서 VTN을 통해 직/간접으로 사용자의 에너지를 제어한다. 가정 내 기기들은 EMA를 통해 에너지 효율을 높이기 위해 제어 및 관리 된다. 다수의 EMA 들이 인터넷을 통해 상위 VTN에 연결된다.

VTN은 Linux Ubuntu PC에서 실행되며 에너지 사용 현황 모니터링 및 기기의 제어가 가능한 EMS의 역할을 수행한다. 각 EMA/VEN은 유무선 공유기를 위한 Open Wireless Router(OpenWRT)라는 임베디드 리눅스 운영체제를 설치해 놓은 라우터 위에서 실행된다. 해당 라우터는 EIoT 환경에서 Gateway 역할을 겸하며 개별적으로 EMA 역할을 수행하고 IoT 기기들과 IoT 프로토콜을 이용하여 직접적인 통신을 담당한다. 실험 장비의 사양은 다음의 표4와 같으며 실제 실험 환경은 그림 9와 같다. 단, 가정 내 IoT 기기들은 아두이노와 라즈베리 파이를 이용해 구현한다. HTTP/XML을 사용하는 OpenADR2.0b 프로토콜은 EPRI 사에서 공개하고 있는 오픈소스를 이용하였고 CoAP/JSON을 사용하는 OpenADR2.0b 프로토콜을 직접 구현하여 비교 실험하였다. 본 실험의 제한사항

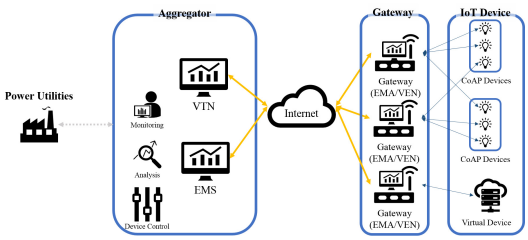

그림 8. Energy IoT 구조
Fig. 8. Energy IoT Architecture

표 4. 실험 환경
Table 4. Experiment Environment

		HTTP/XML		CoAP/JSON	
		VTN	VEN	VTN	VEN
PC Spec	CPU	Intel Core i7-6700HQ	Atheros AR9132 400MHz	Intel Core i7-6700HQ	Atheros AR9132 400MHz
	RAM	4GB	64MB	4GB	64MB
Operating System		Ubuntu 14.04 RTS	OpenWRT	Ubuntu 16.04 RTS	OpenWRT
Language		JRuby	C++	Java	C

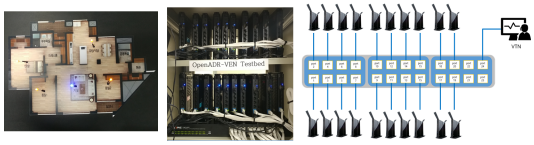

그림 9. 실험 환경
Fig. 9. Experiment Environment

으로는 HTTP/XML과 CoAP/JSON Pull mechanism을 비교하고 이를 결과를 바탕으로 CoAP/JSON Pull mechanism과 CoAP/JSON Push mechanism을 비교 분석하였다.

여기서는 소단원에 관한 내용을 간단히 살펴본다. 여기서는 소단원에 관한 내용을 간단히 살펴본다.

3.2 실험 시나리오 및 결과

본 연구의 검증을 한 실험의 시나리오는 다음과 같다. 각 프로토콜에 따른 VEN은 해당 프로토콜을 지원하는 VTN에 연결되며 VTN과 VEN의 연결은 일대다의 연결을 갖는다. 실험은 VTN에 속하는 VEN 의 개수를 1개, 3개, 9개, 20개로 늘리며 반복한다. 각 VEN은 VTN에게 접속 한 뒤 초기 등록 과정 후 일종의 KEEP ALIVE 메시지로 Poll 메시지를 주기적으로 전송한다. VTN이 자신에게 연결 된 VEN들과 Polling 메시지를 주고받는 도중 하나의 VEN에 대하여 사용량에 대한 이벤트를 발생하고 Distribute Event 메시지를 보낸다. 이 때 이벤트는 Pull 방식과 Push 방식으로 이루어진다. 연구의 분석을 통해 HTTP/XML 포맷의 OpenADR2.0b 프로토콜과 CoAP/JSON 포맷의 OpenADR2.0b 프로토콜은 표 5와 같이 데이터 트래픽, 이벤트 응답시간 두 가지 항목에서 비교 분석하였다. 이를 바탕으로 CoAP/JSON 기반의 Pull mechanism과 Push mechanism을 위 실험과 동일하게 데이터 트래픽, 이벤트 응답시간 두 가지 항목에서 5회에 걸쳐 측정하고 평균값을 비교 분석하였다.

표 5. 실험 측정 항목 및 측정 방법
Table 5. List of measurement and the methods of measuring

List of measurement	Methods of measuring
Event Response Time based on the number of VEN	Analyze and Compare the Event response time that send Event to All VEN which are connected to VTN
Data Traffic based on the number of VEN	Analyze and Compare the quantity of Data til the Event response that send Event to All VEN which are connected to VTN

3.2.1 Pull mechanism 기반의 VEN 개수에 따른 Data Traffic 비교

본 연구에서 구현한 CoAP 기반의 JSON 형식의 OpenADR2.0b의 검증 및 기존 방식과의 비교를 위해 세 가지의 데이터를 비교 분석 하였다. 첫 번째는 VEN의 개수에 따른 데이터 트래픽이다. 그림 10과 같이 첫 번째 VEN이 VTN에 연결되어 Init Processing을 시작하는 순간부터 일정 시간 동안 패킷을 캡쳐 한 후 각 포맷에 따른 초당 데이터양을 측정하여 비교하였다.

그림 11은 VEN 개수에 따른 HTTP/XML Pull mechanism과 CoAP/JSON Pull mechanism의 시간당 데이터 양을 보여준다. 데이터 양은 VEN의 개수와 정비례하여 증가함을 확인할 수 있다. HTTP/XML와 CoAP/JSON Pull mechanism 모두 비슷한 증가 양상을 보였지만 HTTP/XML 방식의 데이터양이 VEN의 개수가 1개일 때는 평균 8340.197 byte/sec, 3개 일 때는 평균 21130.510 byte/s, 9개 일 때는 평균 50062.597 byte/s, 20개 일 때는 105051.262 byte/s로 CoAP/JSON 방식이 VEN의 개수가 1개일 때는 평균 877.392 byte/sec, 3개 일 때는 평균 2592.325 byte/s, 9개 일 때는 평균 7368.251 byte/s, 20개 일 때는 16052.44 byte/s과 비교하여 약 7배 더 큰 것을 볼 수 있다. 또한 VEN 개수에 따른 증가폭은 HTTP/XML 방식이 VEN의 개수가 3개에서 20개로 증가함에 있어서 약 83920.751 byte/s, CoAP/JSON 방식에서 1340.113 byte/s로 HTTP/XML 방식이 CoAP/JSON 방식과 비교하였을 때 약 7배 더 큰 것을 확인 할 수 있었다. 따라서 EIoT 환경에서 기기의 수가 증가하게 될 경우 기존 HTTP/XML 기반의 OpenADR 프로토콜은 데이터 트래픽이 크게 늘어날 수 있는 반면 제안한 프로토콜은 그 증가폭이 낮고 비교적 일정하게 유지할 수 있기 때문에 대량의 기기들의 수용을 가능케 한다.

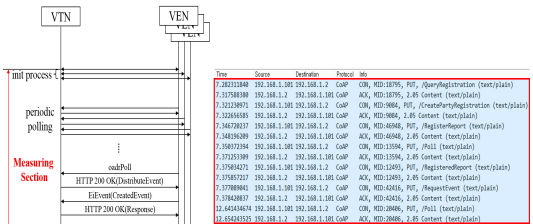

그림 10. 데이터 트래픽 측정구간
Fig. 10. Data Traffic measuring Section

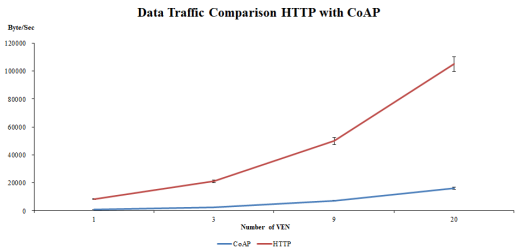

그림 11. HTTP/XML 포맷과 CoAP/JSON 포맷 데이터 트래픽 비교
Fig. 11. Data Traffic Comparison between HTTP/XML Format and CoAP/JSON

3.2.2 Pull mechanism 기반의 VEN 개수에 따른 Event Response Time 비교

두 번째 비교 항목은 VEN의 개수에 따른 이벤트 응답시간 비교이다. 본 실험에서 VTN은 VTN에게 연결 된 VEN들이 주기적으로 Polling 메시지를 보내던 중간에 하나의 VEN에게 이벤트 메시지를 내려준다. 이벤트는 VEN의 Polling 메시지에 대한 ACK 메시지에 담겨서 VEN에게 전송된다. 그림 12과 같이 이벤트를 내려주기 위한 Polling 메시지부터 이벤트가 정상적으로 생성되었다는 CreatedEvent 메시지에 대한 응답메시지까지 5회에 걸쳐 시간을 측정하고 평균값을 비교한다.

그림 13은 두 포맷의 이벤트 응답시간에 대한 비교를 보여준다. 두 방식 모두 VEN의 개수에 따라 증가

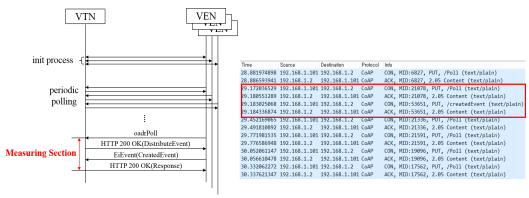

그림 12. 이벤트 응답시간 측정 구간
Fig. 12. Event Response Time measuring section

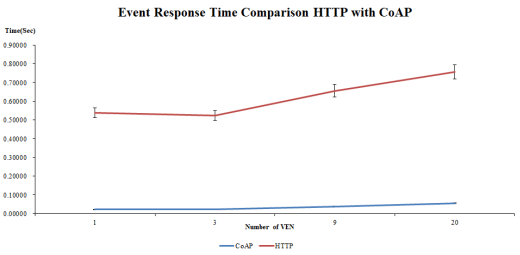

그림 13. HTTP/XML 포맷과 CoAP/JSON 포맷의 이벤트 응답시간 비교
Fig. 13. Event Response Time Comparison between HTTP/XML and CoAP/JSON

하는 추세를 보이지만 HTTP/XML 방식이 VEN의 개수에 따라 1개 일 때 0.5381 sec, 3개 일 때 0.5223 sec, 9개 일 때 0.6556 sec, 20개 일 때 0.7569 sec로 CoAP/JSON 방식의 VEN 개수에 따른 응답시간이 각각 0.0219 sec, 0.0239 sec, 0.0370 sec, 0.0553 sec에 비해 기본적인 응답시간은 약 19배로 큰 차이가 나타나는 것을 확인 할 수 있으며 VEN의 개수가 3개에서 20개로의 증가를 볼 때 그 증가폭이 CoAP/JSON 방식은 0.0313 sec, HTTP/XML 방식은 0.235 sec로 약 7.5배의 차이를 보임을 확인 할 수 있다. 따라서 EIoT 환경에서 기기의 수가 증가하게 될 경우 기존 HTTP/XML 기반의 OpenADR 프로토콜은 이벤트에 대한 반응 시간이 급격히 늘어날 수 있는 반면 제안한 프로토콜은 그 증가폭이 낮고 일정하게 유지할 수 있기 때문에 대량의 기기들을 수용할 수 있다.

본 논문에서는 EIoT 환경에서 에너지의 효율적인 사용 및 에너지 최적화를 위한 경량화 된 수요반응 프로토콜의 개발 및 성능분석에 있다. 특히 에너지 수요반응 서비스를 제공함에 있어서 보편적으로 이용되는 OpenADR 2.0b 프로토콜을 EIoT 네트워크 환경에서 원활한 통신이 가능하도록 IoT 용 CoAP 프로토콜에 JSON 포맷으로 구현하고 기존의 HTTP/XML 기반의 OpenADR 프로토콜과 비교 분석하였다. 분석 결과 본 연구에서 제안한 CoAP/JSON 방식의 OpenADR2.0b 프로토콜이 기존의 HTTP/XML 방식의 OpenADR2.0b 프로토콜에 비해 1/7 수준의 더 적은 데이터 트래픽을 갖고 약 7배 더 빠른 응답시간을 갖는 것을 확인 할 수 있었다. 에너지 환경의 소형 네트워크 기기에서 데이터 트래픽과 이벤트 응답시간은 다양한 제어 서비스로 확장되어가고 있는 EIoT 서비스에서 중요한 요소가 된다[1].

3.2.3 CoAP Push와 Pull Mechanism Data Traffic 비교

구현 한 CoAP 기반의 JSON 형식의 OpenADR2.0b의 Push와 Pull mechanism 검증 및 비교를 위해 두 가지의 데이터를 비교 분석 하였다. 그림 14와 같이 첫 번째 VEN이 VTN에 연결되어 Init Processing을 시작하는 순간부터 30초 간 패킷을 캡쳐 한 후 각 포맷에 따른 초당 데이터양을 측정하여 비교하였다.

그림 15는 각 각 VEN 개수에 따른 CoAP/JSON Push 와 Pull mechanism의 초당 데이터양을 보여준다. Pull mechanism에서 데이터양은 눈에 띄게 VEN의 개수와 정비례하여 증가함을 확인 할 수 있다. 반면 Push mechanism은 VEN의 개수가 증가함과 비례

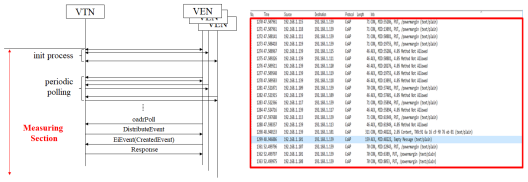

그림 14. 데이터 트래픽 측정 구간
Fig. 14. Data Traffic measuring section

하여 Data양이 증가하지만 큰 차이를 보이지는 않는다. CoAP/JSON Pull mechanism이 VEN의 개수가 1개일 때는 평균 877.392 byte/sec, 3개 일 때는 평균 2592.325 byte/s, 9개 일 때는 평균 7368.251 byte/s, 20개 일 때는 160522.44 byte/s 인 반면 CoAP/JSON Push mechanism은 VEN의 개수가 1개일 때 평균 151 byte/s, 3개 일 때는 평균 284 byte/s, 9개 일 때는 평균 532 byte/s, 20개 일 때는 평균 648 byte/s로 20개일 때 약247배 차이가 나는 것을 확인 할 수 있다. 이를 바탕으로 VEN의 개수가 늘어날수록 데이터양은 더 차이를 가짐을 유추할 수 있다. EIoT 환경에서 기기의 수가 증가하게 될 경우 기존 Pull mechanism 방식은 데이터양이 급격하게 늘어날 수 있는 반면 제안한 Push mechanism 방식은 그 증가폭이 낮고 일정하게 유지할 수 있기 때문에 대량의 기기들을 수용할 수 있다.

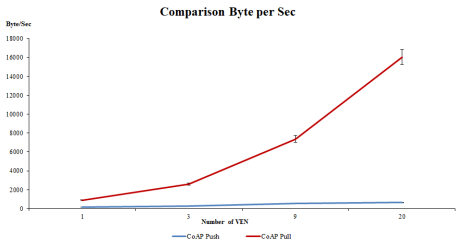

그림 15. CoAP/JSON Push & Pull mechanism 데이터 트래픽 비교
Fig. 15. Data Traffic Comparison between CoAP/JSON Push and Pull

3.2.4 CoAP Push와 Pull Mechanism 이벤트 응답시간 비교

마지막 비교 항목은 VEN의 개수에 따른 이벤트 응답시간이다. 본 실험은 VTN에게 연결 된 VEN들이 주기적으로 Polling 메시지를 보내던 중간에 하나의 VEN에게 이벤트 메시지를 내리는 것과 Polling 메시지 없이 VTN에서 VEN으로 바로 Push 메시지를 보내는 것으로 Pull mechanism과 Push mechanism의 이벤트 응답시간을 비교하는 실험이다. Pull mechanism에서 이벤트는 VEN의 Polling 메시지에

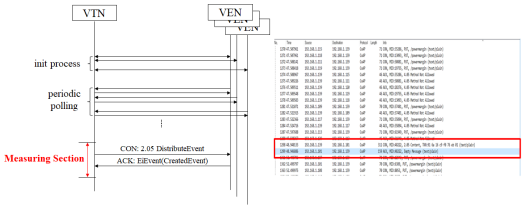

그림 16. 이벤트 응답시간 측정 구간
Fig. 16. Event-Response measuring section

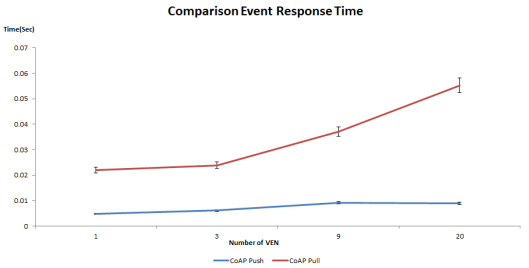

그림 17. CoAP/JSON Push & Pull mechanism 이벤트 응답시간 비교
Fig. 17. Event Response Time Comparison between CoAP/JSON Push and Pull

대한 ACK 메시지에 담겨서 VEN에게 전송되고, Push mechanism에서 이벤트는 VTN이 이벤트를 내리고 VEN은 이에 대한 ACK로 CreatedEvent를 보낸다. 그림 16과 같이 이벤트를 내려주기 위한 메시지부터 이벤트가 정상적으로 생성되었다는 CreatedEvent에 대한 응답메시지까지의 시간을 측정하여 비교한다.

그림 17는 제시한 두 가지 mechanism의 이벤트 응답시간에 대한 비교를 보여준다. 두 가지 mechanism 모두 VEN의 개수에 따라 증가하는 추세를 보이지만 Pull mechanism이 VEN의 개수에 따라 1개 일 때 0.021958 sec, 3개 일 때 0.023936 sec, 9개 일 때 0.037026 sec, 20개 일 때 0.055313 sec로 Push mechanism의 VEN 개수에 따른 응답시간이 각각 0.004883 sec, 0.00615 sec, 0.009172 sec, 0.009037 sec에 비해 VEN이 20대일 때 응답시간은 약 6배로 VEN의 개수가 증가할수록 응답시간의 차이는 더 커질 것으로 유추할 수 있다. 더욱이 Poll 주기 간격이 늘어날수록 Pull mechanism과 Push mechanism의 응답시간차이는 더욱 커질 것이다. 따라서 EIoT 환경에서 기기의 수가 증가하게 될 경우 기존 Pull mechanism을 기반으로 한 OpenADR 프로토콜은 수요반응 반응 속도가 급격히 늘어날 수 있는 반면 제안한 Push mechanism은 그 증가폭이 낮고 거의 일정하게 유지할 수 있기 때문에 대량의 기기들을 수용할 수

있다.

Ⅳ. 결 론

본 논문에서는 EIoT 환경에서 스마트 홈 기기의 효율적인 에너지 사용과 에너지 최적화를 위한 경량화된 CoAP/JSON 기반 실시간 수요반응 mechanism을 제안했다. 제안한 CoAP/JSON 방식은 에너지 수요반응 서비스를 제공함에 있어서 널리 이용되는 OpenADR2.0b 프로토콜에서 EIoT 기기를 위해 경량화한 프로토콜로 기존의 HTTP/XML 기반의 OpenADR 프로토콜과 비교 분석하였다. 또한 EIoT 네트워크 환경에서 실시간 수요 반응에 적합한 CoAP/JSON 기반 Push mechanism을 제시하고 기존의 Pull mechanism과 비교 분석하였다. 분석 결과 본 연구에서 제안한 CoAP/JSON 기반 OpenADR2.0b 프로토콜이 기존의 EPRI에서 제시하는 표준 HTTP/XML 기반의 OpenADR2.0b 프로토콜에 비해 1/7 수준의 더 적은 데이터 트래픽을 갖고 약 7배 더 빠른 응답시간을 갖는 것을 확인 할 수 있었다. 또한 CoAP/JSON 기반 Push mechanism은 앞서 비교한 CoAP/JSON 기반 Pull mechanism에 비해 VEN 20대 일 때 약 1/247배 더 적은 데이터 트래픽 량이 발생하고 약 6배 더 빠른 응답시간을 갖는 것을 검증 하였다. 본 연구 결과 전력 및 에너지 서비스를 위한 소형 네트워크 기기에서 데이터 트래픽 량과 응답시간의 감소는 실시간 에너지 서비스의 개발 측면에서 중요한 요소가 된다. 더욱이 제시한 Push mechanism은 EIoT환경에서 기존 Pull mechanism이 가지고 있는 VEN의 Polling 메시지에 의한 자원 낭비와 VTN의 복잡도 문제를 해결할 수 있으며, 가격기반 실시간 수요반응과 위급 상황 시 실시간 수요반응을 가능케 한다. 따라서 본 논문에서 제안한 경량 프로토콜과 Push mechanism을 이용하면 소형 네트워크 기기를 갖는 EIoT 환경에서 보다 많은 수요자를 관리하고 원활한 수요반응 자동화 서비스 및 실시간 상호작용이 수요반응에 적용될 수 있어 효과적인 에너지 관리에 널리 활용될 것으로 기대된다.

Ⅴ. 향후 과제

향후 EIoT 환경에서 VTN-VEN 수요관리 기능을 Cloud 환경을 적용하여 Cloud 기반으로 에너지 소비 형태를 분석하여 이를 바탕으로 에너지 소비를 효율적으로 관리를 하는 기능 연구가 필요하다. 나아가 OpenADR2.0b에서 제시하고 있는 가격기반, 인센티브 기반 수요반응 정책에 Push mechanism을 적용하여 실시간 스케줄 및 에너지 관리에 대한 효과 분석이 필요하다. 끝으로 소비자 측면에서 가격기반 수요반응을 바탕으로 현재 전력 사용량과 전력 사용량에 따른 가격 모니터링을 통해 자발적인 소비자 전력 절약을 위한 기능 정의 연구가 필요할 것으로 보인다.

References

[1] H.-I. Park, S.-Y. Kim, S.-C. Kang, H.-J. Park, I.-Y. Kim, and J. S. Choi, “Implementation and analysis of CoAP-Based lightweight OpenADR2.0b protocol for smart energy IoT environment,” *J. KICS*, vol. 42, no. 04, Apr. 2017

[2] Retrieved Mar. 13, 2017. from http://www.openadr.org/

[3] O. Alliance, *The OpenADR primer*, Technical report, 2012.

[4] Google push and Pull mechanism comparison, “https://cloud.google.com/pubsub/docs/subscriber”

[5] S. R. Jan, F. Khan, F. Ullah, N. Azim, and M. Tahir, “Using coap protocol for resource observation in IoT,” *IJETCSE*, vol. 21, no. 2, Apr. 2016.

[6] Interconnection of Information Technology Equipment Home Electronic System, ISO/IEC JTC 1/SC 25/WG 1, Mar. 06, 2017

[7] J. H. Park, Y. M. Hwang, J. Y. Kim, and J. J. Lee, “A study on the implementation of demand response system in smart grid,” *The J. Korea Soc. Commun. and Space Technol.*, vol. 10, no. 1, pp. 44-48, 2015.

[8] S. C. Kang and J. S. Choi, “Design and implementation of realtime demand and response gateway in smart home based on MQTT,” in *KICS Int. Conf. Commun.*, pp. 60-61, Jeju lsland Korea, Jun. 2016.

[9] S. Y. Kim, B. W. Jang, and J. S. Choi, “Design and implementation of real time demand response protocol based on OpenADR,” *JCCI*, Sokcho, Korea, Apr. 2016.

[10] Y. T. Yoon, “Real time demand response policy and system,” *Seoul National University*,

vol. 60, no. 9, 2011.

[11] S. E. Lee, *Major country's demand response operation case for expanding new regeneration energy*, KEPCO.

박 현 진 (Hyun Jin Park)

2017년 2월 : 한신대학교 컴퓨터 공학부 졸업

2017년 3월~현재 : 한양대학교 컴퓨터 소프트웨어 석사과정

<관심분야> IoT, Smart Home, Smart Grid

박 헌 일 (Heon Il Park)

2015년 2월 : 한국외국어대학교 정보통신공학과 졸업

2016년 9월~현재 : 한양대학교 컴퓨터 소프트웨어 석사과정

<관심분야> IoT, Smart Home, Smart Grid, Server

이 성 환 (Sung Hwan Lee)

2009년 2월 : 단국대학교 전기전자컴퓨터공학부 졸업

2017년 3월~현재 : 한양대학교 컴퓨터소프트웨어 석박통합과정

<관심분야> Smart Grid, Energy Efficiency, ICT Convergence

최 진 식 (Jin Seek Choi)

1985년 2월 : 서강대학교 전자공학과 학사

1987년 2월 : 한국과학기술원 네트워크 석사

1995년 2월 : 한국과학기술원 네트워크 박사

2004년~현재 : 한양대학교 컴퓨터소프트웨어학부 교수

<관심분야> Network control and management framework, energy management framework for smart-Grid, software-defined networking, mobile IP, carrier Ethernet, switching and Routing