논문 20-45-06-21 The Journal of Korean Institute of Communications and Information Sciences '20-06 Vol.45 No.06
https://doi.org/10.7840/kics.2020.45.6.1110

에너지 IoT 환경을 위한 MQTT 기반의 Lightweight 수요 반응 프로토콜 구현 및 분석

서 동 성◆, 이 성 환*, 최 진 식°

Implementation and Analysis of Lightweight Demand Response Protocol Based on MQTT for Energy IoT Environment

Dong-Sung Seo◆, Sung-Hwan Lee*, Jin-Seek Choi°

요 약

OpenADR 수요 반응 프로토콜은 스마트 그리드 환경에서 에너지의 효율적 사용과 지능화 된 수요자원관리 서비스를 제공하기 위하여 개발되었다. HTTP/XML기반의 OpenADR 수요 반응 프로토콜은 데이터 트래픽의 오버헤드가 크기 때문에 가정, 빌딩 내의 제약된 환경을 가진 경량 디바이스에서 사용하기 어렵다. 최근 스마트 홈 에너지 IoT 환경에서는 가정 내의 초경량 디바이스들을 위한 경량 에너지 수요 반응 프로토콜이 요구되고 있다. 본 논문에서는 IoT 환경에서 많이 사용되는 MQTT기반의 경량 에너지 수요 반응 OpenADR 2.0b 프로토콜을 제안하고 평가한다. 우선 제안된 프로토콜을 구현하고 테스트베드를 통해 HTTP/XML 기반의 OpenADR 프로토콜과의 호환성을 검증한다. 다음은 테스트베드 실험을 통해 구현된 MQTT의 특징인 Broker 기반의 게시(Publish)/구독(Subscribe) 기능을 이용하여 Push 기반의 실시간 이벤트 전달 메커니즘의 성능을 분석한다. 실험 결과를 통해 HTTP/XML기반의 OpenADR과의 성능을 비교하여 제안된 MQTT/JSON기반의 OpenADR 프로토콜이 데이터 트래픽 측면에서 1/3로 줄어들고 실시간 이벤트 전달측면에서 VEN 개수나 Polling주기와 무관하게 0.01초 이내 빠른 응답속도를 제공함을 증명한다.

Key Words : OpenADR(Open Automated Demand Response), IoT(Internet of Thing), MQTT(Message Queuing Telemetry Transport), Demand Response

ABSTRACT

OpenADR protocol was developed to provide efficient use of energy and intelligent demand resource management services in a smart grid environment. The OpenADR demand response protocol based on HTTP/XML is difficult to use in lightweight devices with limited environments in homes and buildings because of the high overhead of data traffic. Recently, in the smart home energy IoT environment, a lightweight energy demand response protocol for ultra-lightweight devices is required. In this paper, we propose and evaluate a lightweight OpenADR 2.0b protocol based MQTT. First, we implement the proposed protocol, and validate the compatibility with the existing HTTP/XML based OpenADR protocol through the testbed. Next, we analyze a push-based real-time event delivery mechanism using the publish/subscribe feature of MQTT through the experimental

※ 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2019R1F1A1061825).
◆ First Author : Hanyang University Department of Computer Science, sds1zzang@naver.com, 학생(석사), 학생회원
° Corresponding Author : Hanyang University Department of Computer Science, jinseek@hanyang.ac.kr, 정교수, 종신회원
* Hanyang University Department of Computer Science, zmagician@naver.com, 학생(박사), 학생회원
논문번호 : 202004-073-D-RE, Received March 31, 2020; Revised April 28, 2020; Accepted April 28, 2020

evaluation. From experiment results, we validate that the proposed MQTT based OpenADR protocol not only reduced by 1/3 in terms of data traffic, but also supported a response time within 0.01 seconds regardless of the number of VENs and polling intervals in real-time event delivery.

Ⅰ. 서 론

급격한 에너지의 수요와 공급의 불균형 상태가 일어나는 문제들을 극복하기 위하여 가용 가능한 에너지 공급을 초과하지 않도록 전력망의 에너지 수요를 관리하는 방법으로 수요 반응(Demand Response) 기술이 제안되었다.[1] 수요 반응 기술은 에너지 생산량에 맞추어 소비량을 미리 설정하고 제한해서 에너지 수요가 공급을 넘지 않도록 조절하는 에너지 소비 제어의 지능화 기술이다.[2] 스마트 그리드 환경에서 전력 생산자인 유틸리티와 소비자 간에 정보를 교환하기 위한 수요 반응 표준 프로토콜로 OpenADR(Open Automated Demand Response) Alliance에서 OpenADR 2.0 프로토콜[3]이 제안되어 현재 많은 공공 기관, 공장, 빌딩 등에서 전력 에너지 수요 반응 프로토콜로 사용하고 있다. 최근에는 공장이나 빌딩 중심의 에너지 관리에 관한 연구[4]에서 더 나아가 가정이나 소규모 집합건물까지도 지능화된 에너지 관리가 가능한 수요 반응 프로토콜 연구가 활발히 진행되고 있다. 특히 스마트 홈(Smart Home) 기술의 발전으로 에너지 사용에 대한 소비자의 관심이 늘어나면서 에너지 관리를 위한 수요 반응 프로토콜이 요구되고 있다.

가정이나 소규모 빌딩, 집합 건물에서 에너지 관리를 위해 ISO(International Organization for Standardization)/IEC(International Electrotechnical Commission) JTC(Joint Technical Committee) 1/SC(Subcommittee) 25 국제표준화 위원회에서는 에너지 관리 에이전트(Energy Management Agent; EMA)의 표준화를 진행하였다.[5] 에너지 관리 에이전트 표준은 지능적인 에너지 관리 서비스를 제공하는 동시에 스마트 그리드에 대한 수요 반응(Demand Response) 신호에 대한 상호 운용성을 지원한다.[6] 또한 에너지 관리 에이전트는 실시간 통신을 통해 소비 전력 모니터링이 가능하다.[7] 에너지 관리 에이전트는 스마트 홈 환경에서 스마트 가전을 포함한 에너지 기기 전체의 에너지 수요 반응 서비스를 제공하는 에너지 사물인터넷 (Energy Internet of Thing; EIoT) 기술로 발전하고 있다.[8] 에너지 사물인터넷 EIoT 기술은 IoT(Internet of Thing) 통신 프로토콜을 이용하여 에너지 사물들 간에 초연결 양방향 통신과 지능화 된 에너지 관리 서비스를 제공하는 시스템이다.

그러나 기존의 대형 빌딩 등에서 사용하는 HTTP(Hyper Text Transfer Protocol)/XML (Extensible Markup Language) 기반의 표준화된 OpenADR 수요 반응 프로토콜은 스마트 홈 에너지 관리 에이전트가 제약된 환경을 가진 수많은 IoT에게 원활한 실시간 서비스를 제공하는 데 있어서 한계가 존재한다.[8] 최근 이러한 요구에 부응하기 위해 EIoT 환경에서 초연결 서비스 제공을 위한 초경량 수요 반응 프로토콜이 요구되어 여러 연구[1]가 진행되었으나 대부분 IoT 프로토콜 중 하나인 CoAP (constrained application protocol)을 기반으로 한 수요 반응 프로토콜들이 제안되었다.[1,9-11] CoAP는 UDP 프로토콜의 초경량 특성을 갖지만 스마트 홈 환경과 같이 방화벽(firewall)이 존재하는 환경에 적용하기가 어렵고, 신뢰성이 떨어지는 문제가 있다. 또한 최근 MQTT (Message Queuing Telemetry Transport) 기반의 수요 반응 프로토콜을 제안했으나 멀티캐스트 기능 위주로 검증하여 기존 OpenADR 2.0b와의 호환성과 수요 반응 프로토콜 구현 측면에서는 다루지 않았다.[12]

본 논문에서는 스마트 홈과 같은 방화벽이 존재하는 환경에서 초연결, 고신뢰성 및 실시간 수요 반응 서비스 제공을 위한 ISO 표준(ISO/IEC PRF20922) 프로토콜인 MQTT 기반의 수요 반응 프로토콜을 제안하고 평가하고자 한다. 제안된 프로토콜은 구현을 통해 테스트베드를 구축하고 기존 OpenADR 2.0b 프로토콜과의 호환성 실험과 가정과 같은 제한된 환경에서의 수요 반응 이벤트 처리 및 속도에 대한 성능을 분석하고자 한다. 끝으로 실험을 통해 기존 HTTP/XML기반의 OpenADR 프로토콜에 비해 제안된 MQTT 기반의 OpenADR프로토콜은 빠른 응답속도를 제공하여 실시간 서비스를 제공할 수 있음을 보이고자 한다. 또한 데이터 트래픽을 비교하여 제안된 프로토콜이 제한적인 리소스 환경을 가진 에너지 사물인터넷에 적절한 초경량 수요 반응 프로토콜임을 보이고자 한다.

본 논문의 구성은 다음과 같다. II 장의 본론에서는 MQTT기반 JSON방식의 OpenADR2.0b 프로토콜의 구현과 MQTT의 게시(Publish)/구독(Subscribe)을 이용한 Push 이벤트 구현 및 절차에 대해 제시를 하고

실험환경에 대한 부분은 III 장에서 기술하며 IV에서는 실험에 대한 결과, 마지막으로 V장에서 결론을 기술한다.

Ⅱ. 본 론

2.1 MQTT기반 JSON 방식의 OpenADR2.0b구현

IEC PC(program committee) 118에서 PAS(Publicly Available Specification)로 국제 표준화된 수요 반응 프로토콜인 OpenADR 2.0b의 표준을 완성했다. 이 프로토콜은 수요자원관리를 수행하는 서버인 Virtual Top Node(VTN)과 수요 반응 서비스 대상 노드인 Virtual End Node(VEN) 사이[12]에서 수요 반응 데이터를 전달하기 위한 프로토콜로 크게 EiRegisterparty, EiReport, EiEvent, 그리고 EiOpt의 4가지 서비스로 구성[3]된다. 본 논문에서 제안하는 MQTT 기반의 OpenADR 2.0b 프로토콜도 동일한 4가지 서비스를 중심으로 구현하였다. 특히 메시지 Payload에 대한 호환성을 가지기 위해서 기존 XML의 모델을 JSON(Javascript Object Notation)으로 매핑한 경량 데이터 모델의 메시지 Payload를 제시한다. XML은 확장이 용이하고 쉽게 해석과 작성이 가능하다는 장점이 있으나 태그의 반복으로 인한 트래픽의 증가라는 문제점이 있고, 이로 인해 메시지 파싱의 속도가 느린 단점이 있다.[1] 본 논문에서는 XML에 비해 트래픽을 적게 소모하고 파싱 속도가 빠른 JSON 포맷을 이용하여 필수적인 값만을 가지도록 구현하였다. XML을 JSON으로 변경하면 필수 데이터 포맷은 유지되면서 데이터의 크기 또한 기존 510byte에서 170byte로 1/3 줄어든 것을 표 1에서 확인할 수 있다. 또한, IoT 프로토콜인 MQTT를 이용함으로써 양방향 송수신이 자유로운 특징을 통해 HTTP에서 지원이 어려운 Push 방식을 쉽게 지원할 수 있다. 방화벽이 존재하는 스마트 홈 환경에서 같은 IoT 프로토콜인 CoAP를 이용하게 되면 방화벽에 의해 외부에서 들어오는 UDP 포트 차단 문제로 인해 Push 방식의 이용이 불가능하나[8], MQTT는 클라이언트들이 Broker를 중심으로 TCP 연결 및 QoS(Quality of Service)를 이용하여 고 신뢰성을 가진 Push 방식의 이용이 문제없이 가능하다.

표 1. XML과 JSON 포맷의 비교
Table 1. Comparison of XML and JSON format

Data Format	Payload	Data Length
XML	<?xml version="1.0" encoding="UTF-8" standalone="no"?> <p1:oadrPayload xmlns:p1="http://openadr.org/oadr-2.0b/2012/07"> <p1:oadrSignedObject> <p1:oadrQueryRegistration xmlns:p3="http://docs.oasis-open.org/ns/energyinterop/201110"p3:schemaVersion="2.0b"xmlns:p2="http://docs.oasis-open.org/ns/energyinterop/201110/payloads"> <p2:requestID>CDC2413C0A</p2:requestID> </p1:oadrQueryRegistration> </p1:oadrSignedObject> </p1:oadrPayload>	510 byte

JSON	oadrRegisteredReport JSON {
	"venID":"VEN1",
	"requestID":"CDC2413C0A",
	"service":"oadrQueryRegistration"
	}
		170 byte

다음은 OpenADR에서 제시한 4가지 서비스에 대한 프로토콜 연결 절차 및 메시지 payload에 관련된 구현 내용이다.

2.1.1 EiRegisterparty

그림 1은 HTT/XML기반의 OpenADR 첫 번째 등록 서비스인 EiRegisterParty 과정으로, VTN에 VEN을 등록[3]하는 세부 서비스 과정과 메시지 플로우, 그리고 이 플로우를 와이어샤크로 캡처한 것이다. 세부 서비스 과정은 다음과 같다. ① VEN이 등록 전에 필요한 정보인 프로파일 및 프로토콜 정보를 oadrQueryRegistration 메시지를 이용하여 VTN에게 요청한다. ② 이에 대한 응답으로 VTN은 지원 가능한 모든 프로파일과 전송 프로토콜 정보를 oadrCreatedPartyRegistration 메시지에 포함하여 VEN에게 전달한다. ③ VEN은 이용할 프로파일과 전

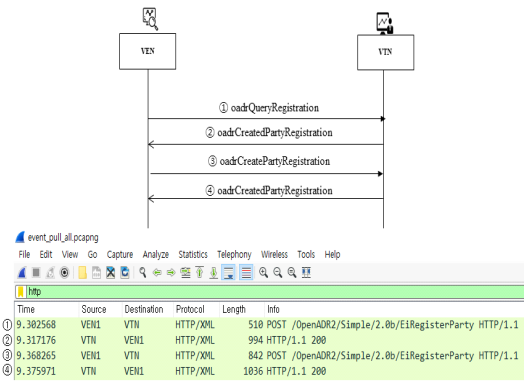

그림 1. HTTP / XML OpenADR EiRegisterparty 과정
Fig. 1. HTTP/XML OpenADR EiRegisterparty Process

송 프로토콜을 oadrCreatePartyRegistration 메시지에 넣어 VTN에 전달하고 ④ VTN은 이에 대한 응답으로 oadrCreatedPartyRegistration 메시지를 보내 VEN이 등록되었음을 전달한다.

그림 2에서 확인 가능한 MQTT/JSON기반의 EiRegisterparty는 HTTP/XML과 마찬가지로 4개의 메시지 전송 과정을 거친 후에 VEN이 VTN에 등록된다. 다만 HTTP와 달리 VTN과 VEN 간의 MQTT 통신은 중간의 Broker를 통해 이루어지고, 메시지의 송신과 수신을 위해서는 Topic에 대한 추가적인 설정이 필요하다. MQTT Client는 메시지 수신을 위해 Broker에게 자신이 수신하기 원하는 Topic에 대해 구독 요청을 할 수 있으며, 구독 요청이 완료된 이후 Broker에 해당 Topic으로 게시된 메시지가 존재하면 해당 Topic을 구독하고 있는 Client에게 메시지를 전달한다.[12] 제안한 MQTT/JSON기반의 OpenADR 2.0b 프로토콜은 OpenADR 2.0b HTTP/XML의 URL에 대응된 Topic을 만들어 이용한다. 다만 수신자를 특정하기 위해 Topic에 수신자의 VTN ID 또는 VEN ID를 추가하여 이용한다.

제안된 MQTT/JSON기반 프로토콜의 단계별 처리 과정은 다음과 같다. ① VTN이 Broker에게 연결 요청을 시도하고 ② Broker로부터 VTN은 연결에 대한 ACK를 수신한다. 연결 수락 후 ③ VTN은 Broker에게 /OpenADR/VTN/2.0b/# Topic의 구독을 요청한다. # 기호의 의미는 /OpenADR/VTN/2.0b/로 시작되는 모든 Topic을 구독한다는 것을 의미한다. ④ VTN의 구독 요청에 대한 응답을 Broker로부터 받은 후에 ⑤ VEN이 Broker에게 연결 요청을 하고 ⑥ Broker에게 oadrQueryRegistration 메시지를 게시하면서 VTN의 응답을 수신하기 위해 /OpenADR/VEN1/2.0b/# Topic의 구독을 요청한다. ⑦ Broker로부터 VEN이 연결 요청에 대한 ACK를 받고 ⑧ Broker가 VEN으로부터 받은 oadrQueryRegistration을 VTN에게 전달한다. ⑨ VEN의 구독 요청에 대한 응답 메시지를 Broker에게서 수신한 후에 ⑩ VTN은 전달받은 oadrQueryRegistration 메시지에 대한 응답으로 oadrCreatedPartyRegistration 메시지를 게시한다. ⑪ 게시된 메시지는 Broker를 거쳐 VEN에게 전달된다. ⑫ VEN이 oadrCreatePartyRegistration 메시지를 Broker에게 게시하고 ⑬ Broker는 이 메시지를 VTN에게 전달한다. ⑭ 마지막으로, VTN은 응답 메시지인

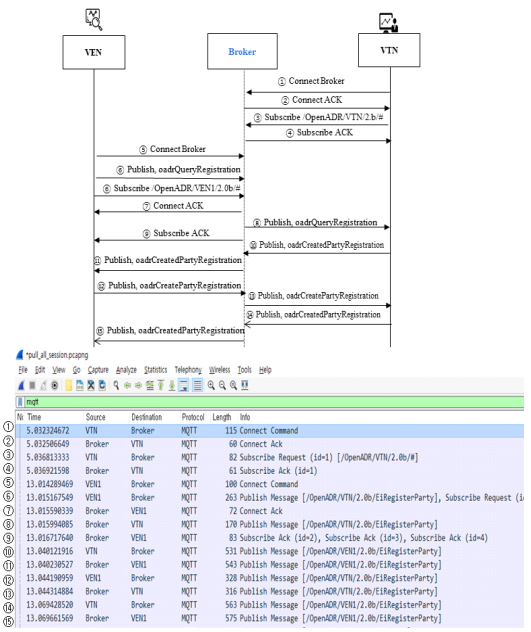

그림 2. MQTT/JSON OpenADR EiRegisterparty 과정
Fig. 2. MQTT/JSON OpenADR EiRegisterparty Process

표 2. MQTT EiRegisterparty의 JSON 형식
Table 2. JSON format in MQTT EiRegisterparty

oadrQueryRegistration JSON

```
{ "venID":"VEN1", #requested VENID
  "requestID":"0FBC38D7817B", #request identifier
  "service":"oadrQueryRegistration" #Message Type
}
```

oadrCreatedPartyRegistration JSON

```
{
  "vtnID":"VTN", #requested VTNID
  "venID":"VEN1", # responsed VEN ID
  "responseCode":"200", #response code
  "responseDescription":"OK",#description of response code
  "requestID":"0FBC38D7817B", #request identifier
  "duration":"PT2S", #requested polling frequency
  "service":"oadrCreatedPartyRegistration", #Message type
  "registrationID":"", #request identifier
  "oadrProfile":{"oadrProfileName":"OpenADR2.0b",
    #type of profile
  "oadrTransports":[{"oadrTransportName":"MQTT"}]
    #type of transport protocol
  }
}
```

oadrCreatePartyRegistration JSON

```
{
  "requestID":"0FBC38D7817B", #request identifier
  "oadrProfileName":"OpenADR2.0b", #profile name used by VEN
  "oadrTransportName":"MQTT", #transport name used by VEN
  "oadrReportOnly":false, #VEN type (report only or full functional)
  "oadrXmlSignature":false, #xml 사용여부 true/false
  "oadrVenName":"VEN1", #VEN name
  "oadrHttpPullModel":true,#communication mode used by VEN
(pull or push)
  "service":"oadrCreatePartyRegistration" #Message Type
}
```

oadrCreatedPartyRegistration를 게시하고 ⑮ Broker는 이를 VEN에게 전달함으로써 등록 과정이 완료된다.

EiRegisterparty에 들어가는 메시지 Type에 대한 설명은 표 2와 같다. 기존 OpenADR/XML의 데이터 형식을 JSON으로 표현했으며 추가적으로 MQTT는 게시/구독 기반이기 때문에 HTTP의 요청(Request)/응답(Response) 구조와는 통신 절차에서 다르다. HTTP는 클라이언트의 요청에 따라 서버가 응답을 직접 전달하는 구조이지만, MQTT는 기본적으로 클라이언트와 클라이언트사이에 Broker를 통해 통신이 이루어지기 때문에 응답 메시지의 수신을 위해서는 자신의 ID가 포함된 Topic 생성이 필요하여 요청 메시지 전송 시 자신의 ID를 포함하여야 한다. 예를 들어 oadrQueryRegistration 메시지의 경우, venID를 추가하여 VTN에서 응답 메시지 전송 시 이용할 수 있도록 하였다. 또한, XML에서 태그 이름을 이용하여 각 메시지의 종류를 구분하였는데, JSON에서는 태그가 존재하지 않기 때문에 service라는 별도의 key를 추가하여 이에 대한 값으로 XML의 태그 값을 입력할 수 있도록 하였다.

2.1.2 EiReport

그림 3은 HTTP/XML기반 OpenADR의 EiReport에 대해 VTN이 VEN에게 주기적인 Report를 요청한 이후 VEN이 주기적으로 Report를 생성하여 전달하는 서비스 과정[3]과 메시지 플로우, 그리고 이 플로우를 와이어샤크로 캡처한 것이다. VTN은 VEN에게 oadrCreateReport 메시지를 이용하여 주기적으로 자원의 상태에 대한 Report를 생성하여 전달하기를 요청한다. 이 후 ① VEN이 자신과 자원의 상태 정보를

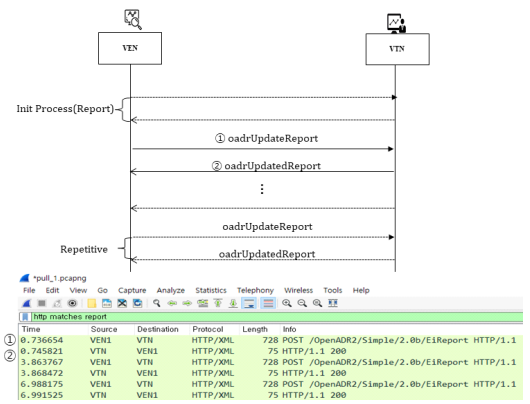

그림 3. HTTP/XML OpenADR EiReport 과정
Fig. 3. HTTP/XML OpenADR EiReport Process

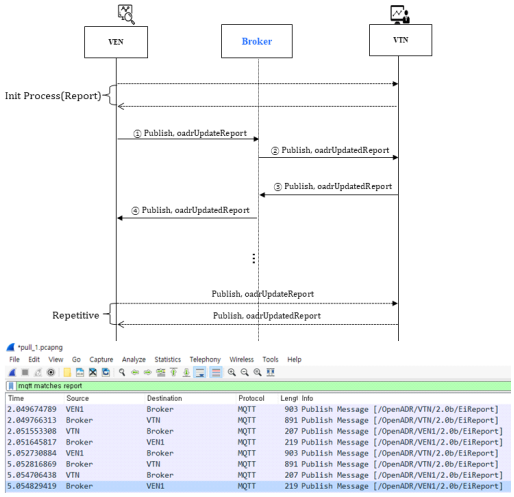

그림 4. MQTT/JSON OpenADR EiReport 과정
Fig. 4. HTTP/XML OpenADR EiReport Process

표 3. MQTT EiReport의 JSON 형식
Table 3. JSON format in MQTT EiReport

oadrUpdateReport JSON

```
{
  "venID":"VEN1", #requested VEN ID
  "requestID":"0fbc38d7817b",  #request identifier
  "service":"oadrUpdateReport", #Message type
  "oadrReport":[{"duration":"PT120M", #report duration
  "reportRequestID":"AD4C12C14D", #report request identifier
  "reportSpecifierID":"E76F46EE91", #report specific id
  "reportName":"TELEMETRY_USAGE", #report name
  "createdDateTime":"2020-03-11 11:32:15", #created time of this
report
  "oadrReportDescription":[{"rID":"VEN1",
  "resourceID":"VEN1", #resource identifier
  "oadrMinPeriod":"PT2S", #Energy usage minimum period
  "oadrMaxPeriod":"PT20S", #Energy usage maximum period
  "powerAttributes":[{"hertz":0.000000, #pulse frequency of power
  "voltage":13.000000, #voltage of power
  "ac":false}] #Is this AC power? (True or False)
    }]
  }]
}
```

oadrUpdatedReport JSON

```
{
  "venID":"VEN1", //requested VEN ID
  "responseDescription":"OK",#description of response code
  "requestID":"0fbc38d7817b",//request identifier
  "service":"oadrUpdatedReport", #Message type
  "responseCode":200 #response code
}
```

oadrUpdateReport 메시지에 담아 VTN에게 보낸다. ② VTN은 이에 대한 응답으로 oadrUpdatedReport 메시지를 보낸다.

MQTT/JSON기반의 OpenADR EiReport 서비스

는 그림 4와 같이 동작한다. HTTP/XML기반의 EiReport 서비스 순서와 동일하며 중간에 Broker를 거쳐 메시지를 전송하는 차이가 존재한다. 추가적으로 각 메시지 Type에는 EiRegisterparty 서비스와 동일하게 “service” key와 값을 추가하여 메시지 종류를 구분하였다.

VEN이 VTN에게 주기적으로 전달하는 ①② oadrUpdateReport 메시지와 이에 대한 응답인 ③④ oadrUpdatedReport 메시지에 대한 설명은 표 3과 같다.

2.1.3 EiEvent

그림 5는 VEN이 이벤트의 존재 유무를 물어보고 VTN이 수요자원의 사용패턴 변화를 위해 이벤트 신호를 내려주는 HTTP/XML기반 OpenADR의 EiEvent 서비스 과정[3]과 메시지 플로우에 대하여 와이어샤크로 캡쳐한 것이다. 이벤트 신호에는 이벤트 시간, 대상, 내용(level, price, power 등)에 대한 정보가 포함되어 수요 반응을 이끌어 낸다.

MQTT/JSON기반의 OpenADR EiEvent 서비스 절차는 그림 6과 같다. HTTP/XML기반 EiEvent 서비스의 절차와 동일하며 중간에 Broker를 거쳐 메시지를 전송하는 차이가 존재한다. 추가적으로 각 메시지 Type에는 EiRegisterparty 서비스와 동일하게 “service” key와 값을 추가하여 메시지 종류를 구분하였다.

EiEvent 서비스에서 이용되는 메시지에 대한 설명은 표 4와 같다. 여기서 ooadrPoll 메시지는 주기적으로 VEN이 VTN에게 전달할 이벤트가 있는지를 물어보는 메시지이다. oadrDistributeEvent 메시지는 VTN이 수요 반응 이벤트가 있을 때 VEN에게 내려주는 메시지로 “eventSignals“ 값에 이벤트에 대한 세부 항목을 입력하고 내려준다. 만약 VTN이 전달할 이벤트

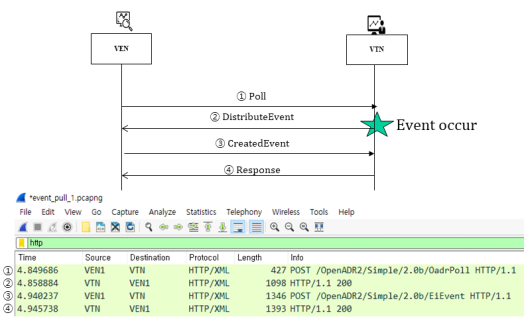

그림 5. HTTP/XML OpenADR EiEvent 과정
Fig. 5. HTTP/XML OpenADR EiEvent Process

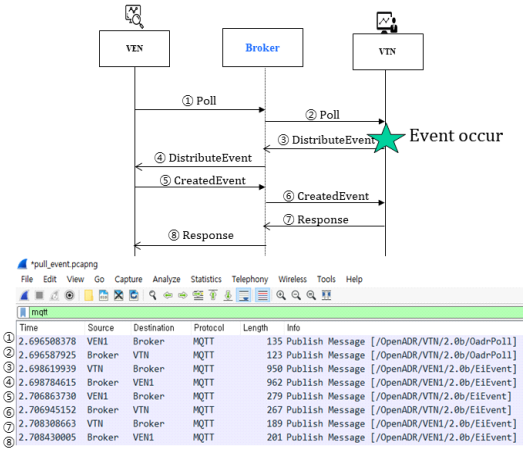

그림 6. MQTT/JSON OpenADR EiEvent 과정
Fig. 6. MQTT/JSON OpenADR EiEvent Process

가 없을 때는 oadrPoll 메시지에 대한 응답으로 oadrResponse 메시지를 전달한다. oadrCreatedEvent 메시지에는 OptType 값에 Opt-IN 또는 Opt-OUT을 넣어 수요 반응 이벤트의 참여 여부를 전달한다. oadrResponse 메시지는 oadrPoll 메시지에 대한 응답 또는 oadrCreatedEvent 메시지에 응답으로 전달된다.

표 4. MQTT EiEvent의 JSON 형식
Table 4. JSON format in MQTT EiEvent

oadrPoll JSON

```
{
  "venID":"VEN1", #requested VEN ID
  "service":"oadrPoll", #Message type
}
```

oadrDistributeEvent JSON

```
{
  "vtnID":"VTN",
  "requestID":"0fbc38d7817b", #requested VEN ID
  "response":[
{
 "responseCode":"200", #response code
 "responseDescription":"OK", #description of response code
 "requestID":"0fbc38d7817b" #event identifier
}
],
  "event":[
{
"eventSignals":[
{
  "intervals":[
   {
  "duration":"PT2S", #event signal interval duration
  "uid":"uid", #event user id
  "value":10 #event value
   }
  ],
```

"signalName":"signalName", #event signal name
"signalType":"Control Event", #event signal type (bi direct, level)
"signalID":"signalID", #event signal ID
"currentValue":"100.0" #current usage value
}
],
"eventID":"eventID", #event identifier
"modificationNumber":"1", #modification Number(count)
"modificationReason":"modificationReason", #modification reason(event reason)
"priority":"-1",
"eiMarketContext":"mirLab", #market address(market reference)
"createdDateTime":"Wed Mar 11 11:29:11 KST 2020", #event create date & time
"eventStatus":"eventStatus", #event status
"testEvent":"false", #if event test or not
"vtnComment":"Event",
"properties":"properties",
"components":"components",
"venID":"VEN1",
"dtStart":"Wed Mar 11 11:38:52 KST 2020", #event start time
"duration":"PT1M", #event duration
"tolerance":"tolerance",
"notification":"notification", #notification duration
"rampUp":"rampUp", #ramp up duration
"recovery":"recovery"
}
],
"service":"oadrDistributeEvent", #Message type
"oadrResponseRequired":"Always" #response mandatory or not
}

oadrCreatedEvent JSON

{
"venID":"VEN1", #responsed VEN ID
"vtnID":"VTN", #requested VTN ID
"responseCode":200, #response code
"responseDescription":"OK", #response code
"requestID":"0fbc38d7817b", #request identifier
"eventID":" eventID ", #request identifier
"modificationNumber":0, #modification number(count)
"optType":"Opt-IN", #if paticipate event or not
"service":"oadrCreatedEvent" #Message type
}

oadrResponse JSON

{
"venID":"VEN1", #requested VEN ID
"service": oadrRegisterReport", #Message type
"responseDescription":"OK", #description of response code
"responseCode":200 #response code
"requestID":" 0fbc38d7817b" #request identifier
}

2.1.4 EiOpt

그림 7은 VEN이 이벤트 참여 가능한 시간을 VTN에 설정하기 위한 HTTP/XML기반 OpenADR EiOpt 서비스 과정[3]과 메시지 플로우에 대하여 와이어샤크로 캡쳐한 것이다.

VEN은 이벤트 참여가 가능한 시간에는 Opt-In, 참여가 불가능 할 때는 Opt-Out 값을 설정하여 VTN에

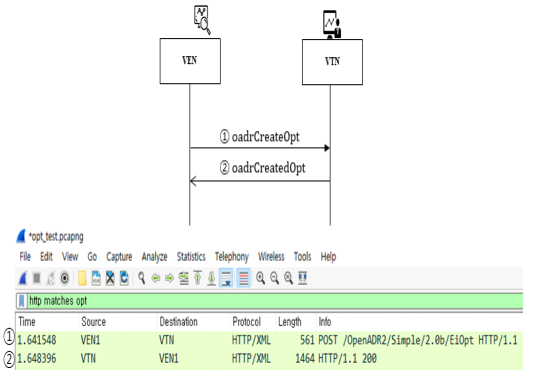

그림 7. HTTP/XML OpenADR EiEvent 과정
Fig. 7. HTTP/XML OpenADR EiReport Process

게 전달한다. ① oadrCreateOpt 메시지는 VEN이 VTN에게 이벤트 참여 가능 시간을 설정하기 위해 전달되며, available 값에 참여 가능한 시간을 입력한다. ② oadrCreatedOpt메시지는 oadrCreateOpt 메시지에 대한 응답으로 VTN이 VEN에게 전달하는 메시지다.

MQTT/JSON기반의 OpenADR EiOpt 서비스 절차는 그림 8과 같다. HTTP/XML기반의 EiOpt 절차와 동일하며 중간에 Broker를 거쳐 메시지를 전송하는 차이가 존재한다. 추가적으로 각 메시지 Type에는 EiRegisterparty 서비스와 동일하게 "service" key와 값을 추가하여 메시지 종류를 구분하였으며 응답 메시지의 수신을 위해서 자신의 ID가 포함된 venID를 key로 하여 value값을 추가로 명시하였다. EiOpt 서비스에 이용되는 메시지에 대한 설명은 표 5와 같다.

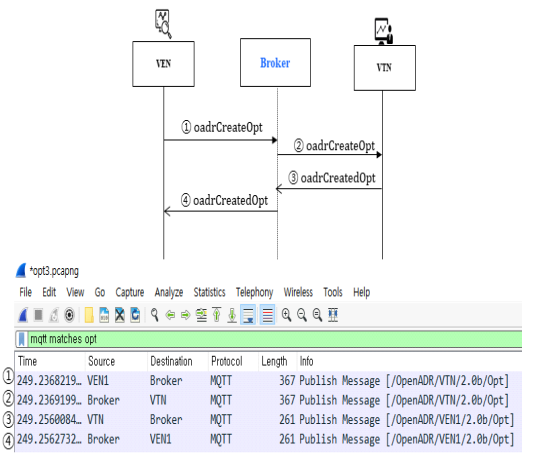

그림 8. MQTT/JSON OpenADR EiOpt 과정
Fig. 8. MQTT/JSON OpenADR EiOpt Process

표 5. MQTT EiOpt의 JSON 형식
Table 5. JSON format in MQTT EiOpt

oadrCreateOpt JSON

```
{
  "venID":"VEN1", #requested VEN ID
  "optID":"E44469C3301B", #opt identifier
  "optType":"Opt-In", #type of opt
  "optReason":"x-schedule", #opt reason(e.g. x-schedule)
  "requestID":"0fbc38d7817b", #request identifier
  "marketContext":"marketContext", #refer market address
  "createdDateTime":"2020-03-12 22:41:07", #created time of this
Message
  "available":[ {"dtStart":"2020-03-14 22:41:07", #opt start time
  "duration":"PT120M" }],
  "service":"oadrCreateOpt" #Message type
}
```

oadrCreatedOpt

```
{
  "optStatus":"Accept",
  "vtnID":"VTN",
  "responseDescription":200, #description of response code
  "requestID":"0fbc38d7817b", #request identifier
  "service":"CreatedOpt", #Message type
  "venID":"VEN1",
  "optID":"E44469C3301B", #opt identifier
  "responseCode":200 #response code
}
```

2.2 MQTT의 게시/구독을 이용한 Push 이벤트의 구현 및 절차

기본적으로 HTTP/XML기반의 OpenADR은 요청/응답 구조의 Pull방식으로 VEN이 먼저 oadrPoll 메시지를 이용해서 요청을 해야 그에 응답으로 VTN이 수요 반응 oadrDistributeEvent 메시지를 보내줄 수 있다. Polling 주기가 길면 이벤트 발생 시점과 메시지 전달 시점의 차이가 커져 긴급절전 등 실시간 수요 반응 서비스를 이용하기 어렵다.[8] 하지만 MQTT/JSON 기반의 OpenADR은 게시/구독 구조이기 때문에 VTN이 oadrPoll 메시지를 기다릴 필요 없이 VEN에게 oadrDistributeEvent 메시지를 전달하는 Push 방식을 이용할 수 있다. Push방법을 이용하게 되면 이벤트 발생 시점과 메시지 전달 시점의 차이가 줄어들어 긴급절전 등의 실시간 수요 반응 서비스 이용이 가능해진다. 그림 9는 Push 방식을 이용하는 이벤트 전달 절차이다. 이용되는 메시지 형식은 기존 EiEvent 서비스의 oadrDistributeEvent, oadrCreatedEvent, oadrResponse 메시지와 동일하다.

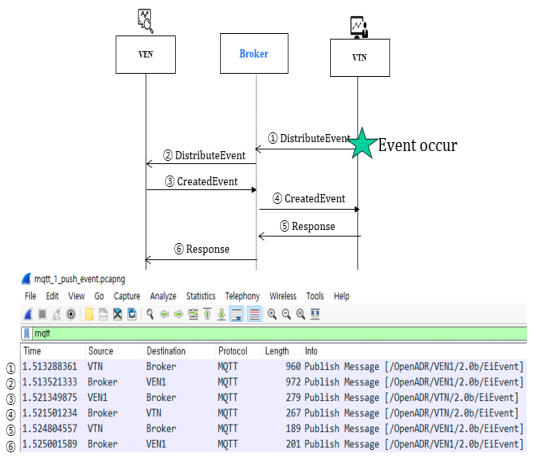

그림 9. MQTT/JSON OpenADR EiEvent PUSH 과정
Fig. 9. MQTT/JSON OpenADR EiEvent PUSH Process

Ⅲ. 실 험

실험을 위한 전체 스마트 홈 에너지 IoT의 테스트베드 스펙 및 환경 구성은 표 6과 같다. 스마트 에너지 IoT 환경에서 VTN은 Ubuntu 16.04가 설치된 PC에서 실행되며 에너지 사용 현황 모니터링 및 기기의 제어가 가능한 에너지 관리 시스템의 역할을 수행한다. 20대의 VEN은 그림 10과 같이 Ubuntu 16.04가 설치된 Raspberry Pi에서 실행된다. 각 Raspberry Pi는 에너지 IoT 환경에서 개별 가정의 에너지 관리 에이전트 역할을 수행한다. 비교를 위하여 HTTP/XML을 사용하는 OpenADR2.0b 프로토콜은 EPRI 사에서 제공한 오픈 소스를 이용하여 실험하였으며 MQTT/JSON의 VTN은 JAVA, VEN은 C로 구현하고 Broker는 Mosquitto version 3.1 Broker를 이용하여 실험하였다.

실험 측정항목은 표 7과 같이 데이터 트래픽과 응답 시간을 측정하였다. 이벤트 확인을 위해 주기적으로 요청하는 oadrPoll 메시지와 이에 대한 응답인 oadrResponse 메시지, 그리고 상태정보를 전달하는 oadrUpdateReport 메시지와 응답인 UpdatedReport

표 6. 실험환경 사양
Table 6. Experiment environment specification

Protocol		HTTP/XML		MQTT/JSON		
		VTN	VEN	VTN	Broker	VEN
PC Spec	CPU	Intel Core i7-7700HQ 2.80GHz	ARM Cortex-A7 쿼드코어 900 Mhz	Intel Core i7-7700 HQ 2.80GHz	Intel Core i5-4200M 2.50GHz	ARM Cortex-A7 쿼드코어 900 Mhz
	RAM	32GB	1GB	32GB	4GB	1GB
Operating System		Ubuntu 16.04				
Language		JAVA	C++	Java	C++	C

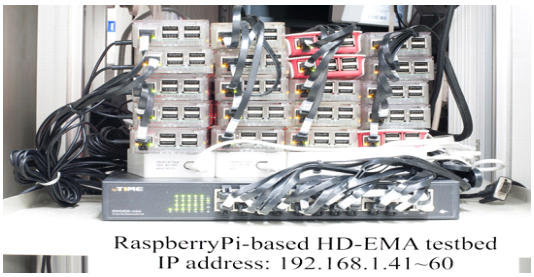

그림 10. 라즈베리파이를 이용한 VEN 실험환경
Fig. 10. VEN experiment environments using Raspberry Pi

표 7. 실험 측정 항목
Table 7. Experimental metrics

실험 항목	내용		조건
데이터 트래픽 측정	HTTP/XML (Pull)		VTN과 VEN이 주기적으로 주고받는 oadrPoll, oadrResponse, oadrUpdateReport, oadrUpdatedReport 메시지의 트래픽을 1분간 측정 (Report, Polling 주기 3초)
	MQTT/JSON (Pull)		
	MQTT/JSON (Push)		VEN이 VTN에게 주기적으로 자신의 상태 정보를 전달하는 oadrUpdateReport, oadrUpdatedReport 메시지를 1분간 측정 (Report 주기 3초)
이벤트 응답시간 측정	Pull	HTTP /XML	VTN에서의 이벤트 발생부터 VEN1가 이벤트를 참여하여 응답을 수신하기까지의 시간을 측정 (Pull: Report, Poll 주기 3초, Push: Report주기 3초)
		MQTT /JSON	
	Push	MQTT /JSON	

메시지에 대해 발생하는 데이터 트래픽을 측정하고, VTN에서 이벤트가 생성되어 VEN에 전달되고, oadrCreatedEvent 메시지가 VTN에 전달된 다음 마지막으로 oadrResponse 메시지를 수신하기까지의 시간을 측정하였다.

그림 11은 실험 환경을 보여주는 그림으로 20대의 VEN이 VTN에 EiRegisiterparty서비스 과정을 거쳐 연결된 이후에 VTN에서 보여지는 Topology그림이다.

그림 12는 20대의 VEN이 주기적인 Report 요청을 받은 후에 일정 시간마다 자신의 상태정보를 oadrUpdateReport 메시지에 담아 VTN에게 전달하였을 때 VTN이 보여주는 전체 에너지 사용량이다.

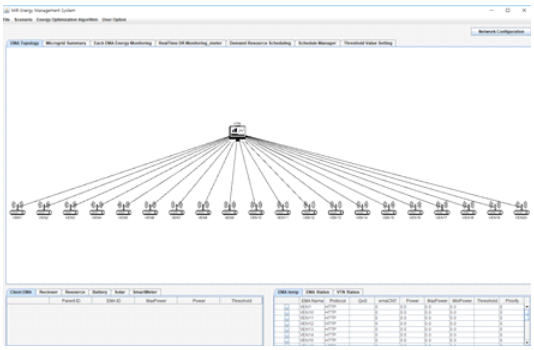

그림 11. VTN-VEN20대 연결 Topology
Fig. 11. Topology for VTN-VEN20 connection

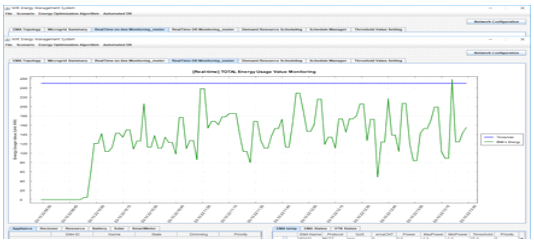

그림 12. VEN1~20 Report에 따른 전체 에너지 사용량
Fig. 12. Total energy consumption according to VEN1~20 Report

Ⅳ. 실험 결과

첫 번째 비교항목은 VEN의 개수에 따른 HTTP/XML OpenADR과 MQTT/JSON OpenADR의 데이터 payload 트래픽에 대한 차이를 비교하였다. 그림 13의 (a) HTTP/XML OpenADR에서는 VEN이 모두 연결되는 VTN에서 들어오는 패킷을 측정하고, (b) MQTT/JSON OpenADR의 경우는 Broker에서 패킷을 측정하였다. 데이터 트래픽의 경우 VEN이 VTN에 연결되어 EiRegisterparty 절차 및 Report 생성 절

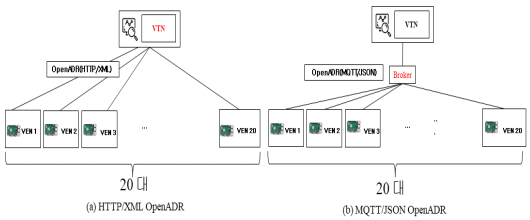

그림 13. VTN-VEN 실험 구성도
Fig. 13. VTN-VEN Experiment Configuration

차를 제외하고 3초 주기의 반복과정인 oadrPoll 메시지와 oadrResponse 메시지, oadrUpdateReport 메시지와 oadrUpdatedReport 메시지 패킷을 1분 동안 캡쳐한 후 평균 데이터양을 측정하여 비교하였다.

표 8과 그림 14에서 볼 수 있듯이 비교 결과 HTTP/XML기반 OpenADR에 비해 MQTT/JSON기반 OpenADR의 데이터 트래픽이 최대 50% 이상 줄어드는 것을 확인할 수 있다. 또한 Push 메커니즘을 사용하게 되면 VEN이 VTN에게 이벤트를 요청하는 oadrPoll 메시지를 제외할 수 있어 HTTP/XML와 대비하여 데이터 트래픽이 약 1/3으로 줄어드는 것이 확인되었다. 따라서 EIoT 환경에서 기기의 수가 증가하게 될 경우 기존 HTTP/XML기반의 OpenADR 프로토콜은 데이터의 트래픽이 급격하게 증가날 수 있는 반면 제안한 프로토콜은 데이터 트래픽이 적어 다수의 IoT기기들을 수용할 수 있다.

두 번째 비교 항목은 VEN의 개수에 따른 이벤트 응답 시간이다. 본 실험에서 VEN들이 3초 주기로 VTN에게 oadrPoll 메시지를 보내며, VTN은 특정 시점에 이벤트를 발생시켜 하나의 VEN에게 이벤트를 내려준다. 이벤트는 VEN의 oadrPoll 메시지에 대한 응답으로 oadrDistributeEvent 메시지에 담겨 VEN에게 전송된다. 이벤트를 요청하기 위한 oadrPoll 메시지부터 이벤트에 정상적으로 참여되었다는 응답인 oadrReponse 메시지 수신까지의 시간을 측정하여 비교한다. 표 9 및 그림 15, 16에서 확인할 수 있듯이 비교결과 Pull 방식일 때 HTTP/XML기반 OpenADR과 MQTT/JSON기반 OpenADR의 응답시간이 Polling 주기의 영향이 커 거의 동일한 것을 확인 할 수 있다. 하지만 Push 방식을 사용하여 이벤트를 내릴 때에는 VTN에서 oadrPoll 메시지를 기다리지 않고 바로 이벤트를 전달할 수 있기 때문에 Pull방식 HTTP/XML과 비교하여 VEN 개수나 Polling 주기에 무관하게 0.01초 이내로 확인 되었다. 이 결과는 Push 방식인

표 8. 데이터 트래픽 (단위 byte)
Table 8. Data traffic (Unit byte)

VEN 개수	1	3	9	20
HTTP (Pull)	108357.6	329499.6	955143.2	2144528
MQTT (Pull)	47428.8	142280.8	431749.6	950178
MQTT (Push)	38995.2	116586.8	351335.6	766688.8

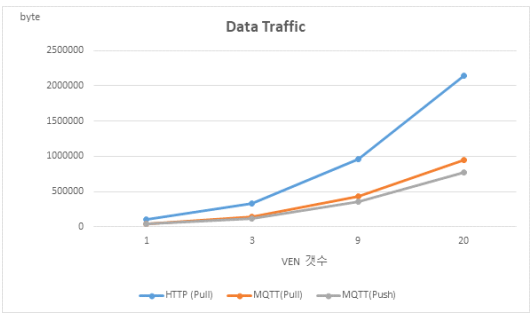

그림 14. HTTP/XML, MQTT/JSON OpenADR Data Traffic 비교 그래프
Fig. 14. HTTP/XML, MQTT/JSON OpenADR Data Traffic Comparison Graph

표 9. 수요 반응에 관련된 이벤트 응답 속도 비교 (단위 초)
Table 9. Comparison of event response rates related to demand response (Unit second)

VEN 개수	1	3	9	20
HTTP (Pull)	1.562254	1.568314	1.570674	1.584682
MQTT (Pull)	1.513766	1.512142	1.513109	1.51433
MQTT (Push)	0.00736	0.007444	0.008652	0.009501

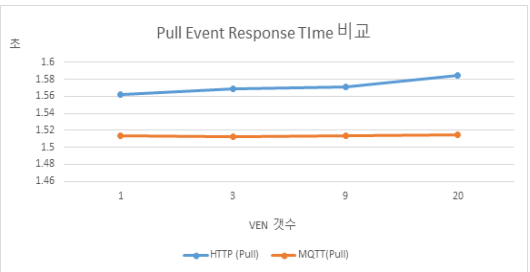

그림 15. Pull방법의 HTTP, MQTT의 이벤트 응답시간 그래프
Fig. 15. Event response time Graph of HTTP and MQTT using pull method

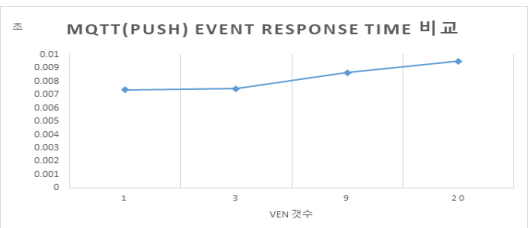

그림 16. Push방법의 MQTT의 이벤트 응답시간 그래프
Fig. 16. Event response time graph of MQTT by Push method

MQTT/JSON 포맷이 긴급한 상황일 때 실시간 수요반응 이벤트를 내리는 데 보다 효과적이라는 것을 나타낸다. 또한 기존 CoAP 기반의 IoT 프로토콜에 비해 스마트 홈과 같은 방화벽이 있는 환경에서 제한을 받지 않고 실시간 고신뢰성 이벤트 메커니즘을 제공할 수 있음을 확인할 수 있었다.

V. 결 론

본 논문에서는 경량 IoT 프로토콜인 MQTT/JSON 기반의 OpenADR 프로토콜을 제안하고 EiRegisterparty, EiReport, EiEvent, 그리고 EiOpt 기능을 구현하여 소규모 빌딩이나 스마트 홈과 같은 에너지 IoT 환경에 맞는 테스트베드를 구축하였다. 구현된 각 기능은 와이어샤크를 사용해 OpenADR 2.0b 표준 프로토콜과 메시지의 포맷과 절차를 비교하여 호환성을 검증하였다. 테스트베드에서 DR이벤트 실험을 실시하여 이벤트 확인을 위해 주기적으로 요청하는 데이터 트래픽이 기존 HTTP/XML방식에 비해 약 1/3로 줄어듦을 확인할 수 있었다. 또한 Push기반의 DR이벤트 응답속도도 VEN 개수나 polling 주기에 무관하게 0.01초 내에서 유지되는 것을 확인하였다. 실험 결과 데이터 트래픽의 감소는 수많은 에너지 IoT 기기들의 연결이 가능하게 하고 이벤트 처리시간을 일정하게 유지할 수 있어 실시간 수요관리 서비스 제공이 가능하다는 것을 확인하였다. 본 논문에서 연구되지 않은 QoS 레벨에 따른 실험과 다른 경량 프로토콜인 CoAP와의 성능 비교는 추후 연구로 남겨둔다. 향후 제안된 프로토콜의 국내/국제 표준화를 추진하여 에너지 IoT 환경에서 차세대 국민 DR프로토콜로 사용될 수 있을 것으로 기대한다.

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, pp. 904-914, Apr. 2017.

[2] M. H. Albadi and E. F. El-Saadany, "Demand response in electricity markets: An overview," *2007 IEEE Power Eng. Soc. General Meeting*, pp. 1-5, Tampa, FL, USA, Jul. 2007.

[3] OpenADR ALLIANCE, "OpenADR 2.0 b profile specification," *Raportti. OpenADR Alliance*, 2013.

[4] D. Delphine, B. W. Jang, Y. S. Shin, S. T. Kang, and J. S. Choi, "Design and implementation of building energy management system with quality of experience power scheduling model to prevent the blackout in smart grid network," *16th Int. Conf. Advanced Commun. Technol.*, pp. 208-211, Pyeongchang, Korea, Feb. 2014.

[5] Information technology - "*Interconnection of information technology equipment - Home Electronic System - Application models - Part 3-3: Model of a system of interacting Energy Management Agents (EMAs) for demand response energy management*," ISO/IEC 15067-3-3, p. 32, Oct. 2019.

[6] J. S. Choi, "A hierarchical distributed energy management agent framework for smart homes, grids, and cities," in *IEEE Commun. Mag.*, vol. 57, no. 7, pp. 113-119, Jul. 2019.

[7] R. Gupta, D. K. Jha, and S. Nagar, "Agent based smart home energy management system," *CSIT 4*, pp. 103-110, Dec. 2016.

[8] H.-J. Park, H.-I. Park, S.-H. Lee, and J. S. Choi, "Design and analysis of push mechanism based on CoAP observe for demand response in energy IoT environment," *J. KICS*, vol. 43, no. 03, pp. 529-540, Mar. 2018.

[9] A. Fachechi, et al., "A new vehicle-to-grid system for battery charging exploiting IoT protocols," *2015 IEEE ICIT*, pp. 2154-2159, Seville, Spain, Jun. 2015.

[10] M. Wei, S. H. Hong, and M. Alam, "An IoT-based energy-management platform for industrial facilities." *Applied Energy*, vol. 164, pp. 607-619, Feb. 2016.

[11] R. Kyusakov, J. Eliasson, J. van Deventer, J. Delsing, and R. Cragie, "Emerging energy management standards and technologies – Challenges and application prospects," in *Proc. 2012 IEEE 17th ETFA 2012*, pp. 1-8, Krakow, Poland, Sep. 2012.

[12] H.-J. Park, H.-I. Park, S.-H. LEE, and J. S.

Choi, “Design and analysis of multicast based lightweight demand response protocol for energy IoT environment,” *J. KICS*, vol. 43, no. 07, pp. 1163-1175 Jul. 2018.

서 동 성 (Dong-Sung Seo)

2018년 2월 : 을지대학교 의료IT학과 졸업
2018년 3월~현재 : 한양대학교 컴퓨터 소프트웨어 석사과정
<관심분야> IoT, Smart Home, Smart Grid, AI, Big Data

이 성 환 (Sung-Hwan Lee)

2009년 2월 : 단국대학교 전기전자컴퓨터공학부 졸업
2017년 3월~현재 : 한양대학교 컴퓨터 소프트웨어 석박통합과정
<관심분야> Smart Grid, 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
[ORCID:0000-0003-1554-3879]