논문 18-43-07-12 The Journal of Korean Institute of Communications and Information Sciences '18-07 Vol.43 No.07
https://doi.org/10.7840/kics.2018.43.7.1163

에너지 IoT 환경을 위한 Multicast 방식의 Lightweight 수요반응 프로토콜 제안 및 분석

박 헌 일◆, 박 현 진*, 이 성 환*, 최 진 식°

Design and Analysis of Multicast Based Lightweight Demand Response Protocol for Energy IoT Environment

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

요 약

유틸리티 회사, 수요반응사업자 입장에서 지능화 된 수요반응 서비스를 제공하기 위한 표준 수요반응 프로토콜로 Open Automated Demand Response(OpenADR)이 개발되었다. 주거공간에서도 단순한 절전이 아닌 에너지 소비의 지능화 개념으로 사용자의 만족도를 높이면서 에너지 사용 효율을 늘릴 수 있는 수요반응 기술이 사용되고 있다. 주거영역 수요반응 서비스가 확대되고 있고, 주거공간에서는 다수의 수요자 및 IoT 디바이스들이 점 대 다 점으로 연결되기 때문에 기존 OpenADR2.0b 메시징 프로토콜인 HTTP/XML기반의 데이터 인/디코딩 오버헤드가 큰 서버-클라이언트 구조의 Unicast 메시지 전송은 적합하지 않다. 본 논문에서는 Smart Energy IoT 환경에서의 수요반응 서비스를 위한 Broker 기반 Multicast 메시지 전송 방식의 경량 수요반응 프로토콜을 제안하고 IoT 메시징 프로토콜 중 하나인 Message Queuing Telemetry Transport (MQTT) 기반으로 수요반응 프로토콜 구현 및 성능을 비교 검증 하였다.

Key Words : Demand Response, OpenADR, MQTT, Multicast, Broker

ABSTRACT

Open Automated Demand Response (OpenADR) has been developed as a standard protocol to provide intelligent demand response services for utility companies and demand response operators. In the residential space, demand response technology is being used to increase energy efficiency while increasing user satisfaction with the concept of intelligent energy consumption rather than simple power saving. However, demand response services to residential areas are expanding. In the residential space, many users and IoT devices are connected in a point-to-multipoint manner. Therefore, existing OpenADR 2.0b messaging protocol, HTTP / XML based data modeling and unicast message transmission are not appropriate. In this paper, we propose a lightweight demand response protocol by using Broker which is based on multicast message transmission method in Smart Energy IoT environment, implement multicast based demand response Message Queuing Telemetry Transport (MQTT) protocol and compare the performance to that of OpenADR protocol.

※ 본 연구는 2018년도 산업통상자원부 및 산업기술평가관리원(KEIT)의 지원에 의해 수행되었습니다('10053671').
◆ First Author : Hanyang University Department of Computer Science, heonil8@nate.com, 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, phj3372@hanyang.ac.kr, zmagician@naver.com
논문번호 : 201806-D-123-RN, Received April 17, 2018; Revised June 15, 2018; Accepted June 18, 2018

Ⅰ. 서 론

1.1 배경

다양한 전력기기의 발전으로 인하여 전 세계적으로 전력 사용량이 점차 증가하고 있다. 특히 여름철이나 겨울철 냉난방기기 등의 사용량 급증에 따른 정전사태와 같은 심각한 문제를 야기하고 있다. 전체 전력 사용량 중 주거 공간 전력사용량의 비율이 점차적으로 증가하고 있으며, 주거공간에서의 전력사용은 기후변화 및 환경오염과 밀접한 관계를 보여 전력 사용의 효율화에 대한 공공의 관심과 우려가 커지고 있다[1].

이러한 문제들을 극복하기 위하여 효율적으로 전력망을 관리할 수 있는 다양한 방법이 연구되고 있으며, 그 중에서 고객의 전력 수요가 가용 가능한 전력 공급을 초과하지 않도록 수용가의 전력 사용량을 조절하는 수요반응이라는 기술이 등장하게 되었다. 수요반응은 에너지 생산량에 맞추어 소비량을 미리 설정해서 수요가 공급을 넘지 않도록 조절하는 에너지 소비의 지능화 기술이다. 전력 생산자인 유틸리티와 전력 소비자 간에 정보를 교환하기 위한 프로토콜로 OpenADR Alliance에서 표준화 된 Open Automated Demand Response(OpenADR)[2]가 제안되었다. OpenADR 프로토콜은 현재 많은 Utility 회사나 공공기관, 공장, 빌딩 등에서 수요반응 프로토콜로 사용하고 있다.

공장이나 빌딩 중심의 블랙아웃 방지를 위한 에너지 관리에 관한 연구[3]에서 더 나아가 일반 사용자의 에너지 사용에 대한 관심과 스마트 홈(smart home) 기술로 발전하면서 최근에는 가정이나 소규모 집합건물에서 지능화된 에너지 관리를 위한 수요 반응 연구가 활발히 진행되고 있는 상황이다. IEC SG25 스마트 홈 분과에서는 가정이나 소규모 빌딩, 집합 건물에서 에너지 수요 반응 관리를 위한 Energy Management Agent(EMA) 표준화가 진행되었다[4]. 스마트 홈에서 에너지 관리를 위한 EMA 기술은 사물인터넷(Internet of Things; IoT) 기술과 결합하여 모든 사물이 시간과 장소에 구애 없이 홈 네트워크를 이용하여 각 노드들의 전력 상태 정보를 송수신 하거나 수요를 제어할 수 있는 다중 EMA 기반의 초연결 통신환경을 제공할 것이다. 스마트 홈에서의 초연결 통신 기술은 전력망과 스마트 홈 에너지 최적화 제어에 대한 에너지 수요반응 서

비스를 제공할 수 있는 에너지 IoT[5]로 발전할 것으로 예상된다.

에너지 IoT 환경에서 에너지 수요관리 서비스의 경우 많은 수의 초경량 디바이스들이 연결되기 때문에 기존 점대점 방식 OpenADR 프로토콜보다 점대 다점 방식인 Multicast 방식의 전송이 가능해야 한다. 또한 동일한 수요반응 이벤트를 다수의 수요자에게 전송할 경우 Hyper Text Transfer Protocol(HTTP) /Extensible Markup Language(XML) 기반의 메시지 및 데이터 인/디코딩 오버헤드가 커져 수많은 IoT 단말들에게 원활한 서비스를 제공하는 데 있어서 한계가 드러나게 된다[6].

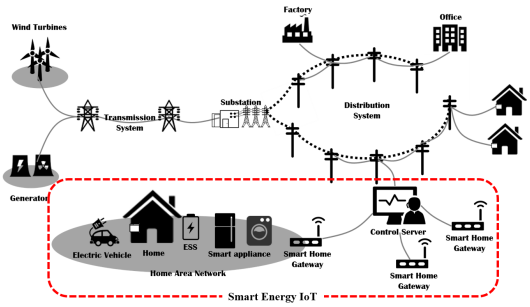


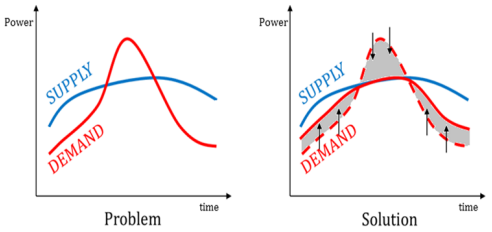
그림 1. 스마트그리드와 에너지 IoT의 구조
Fig. 1. Architecture of Smartgrid and Energy IoT[6]

본 논문에서는 Smart Energy IoT 환경에서의 보다 효율적인 수요반응 서비스를 위한 Multicast 메시지 전송 방식의 경량 수요반응 프로토콜을 제안하고 IoT 메시징 프로토콜 중 하나인 Message Queuing Telemetry Transport(MQTT)를 기반으로 수요반응 프로토콜 구현 및 성능을 비교 검증 하였다. 본 논문의 구성은 I장 서론에서 배경 및 관련기술의 제한점을 제시하고, II장 본론에서는 MQTT 기반 Multicast 방식의 설계, 수요반응에서 Multicast 방식 메시지 전송의 사용 시나리오, 경량 OpenADR2.0b의 JSON 포맷을 설명하고 Ⅲ장의 실험에서는 Multicast 메시지 전송 방식, Lightweight 프로토콜의 필요성 분석 및 검증한다. 마지막으로 Ⅳ장 결론을 기술하였다.

1.2 관련 기술 및 제한점

1.2.1 수요반응(Demand Response)기술

수요반응이란 그림 2처럼 전기 공급과 소비자들의 최대 전기 사용량의 차이를 완화시키고, 수요의 균형을 맞추기 위하여 일시적으로 전기 사용량을 증가 및 축소시키기 위해 소비자의 전력 사용 패턴의 변화를 유도하는 에너지 최적 사용 기술을 말한다[7][8]. 수요반응의 일반적인 흐름은 발전 회사들이 발전소에서 전력을 생산해서 전력거래소에 전력을 판매하고 전력거래소는 전력수급관리를 하면서 수요반응사업자에게



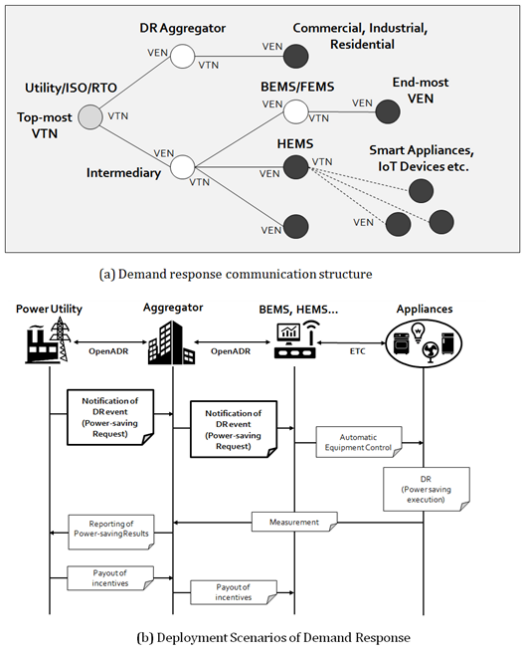
(a) Demand and supply problem (b) Solving through Demand Response

그림 2. 수요반응의 목적
Fig. 2. Objective of Demand Response

전력을 판매한다. 전력거래소는 Virtual Top Node(VTN)로 수요반응인 수요 감축 지시를 통해서 수요자원관리를 관리한다. 대표적인 전력거래소는 한국의 한국거래소(KPX), 미국의 PJM, CAISO 등이 있다. 수요관리사업자(중개업자)는 수요자원을 발굴, 전력을 입찰, 정산, 감축량을 산정하고 전력시장에 수요반응 자원을 등록한다. 수요관리사업자는 Virtual End Node(VEN)로 받은 수요 감축에 대한 감축량을 산정하고 수요반응참여고객에게 수요 감축 지시를 내린다. 수요관리사업자의 한국의 대표적인 기업은 KT, 벽산파워 등이 있다. 수요반응참여고객은 수요관리사업자와 계약을 체결하고 수요 감축 지시에 따라서 감축을 한다. 수요관리사업자는 VTN으로 수요반응참여고객의 VEN에게 수요반응을 내리게 된다. 수요반응참여고객은 수요감축지시에 따라 수요 감축을 하면 수요관리 사업자로부터 인센티브를 받게 된다.

(1) 통신 구조 및 전개시나리오

수요반응 통신 구조는 그림 3의 (a)와 같이 Utility 회사나 전력거래소가 많은 수의 VTN(Server)-VEN (Client)의 계층화 된 구조로 구성이 된다. 스마트 홈 등 주거 공간으로 수요반응이 확대될 경우 다양한 형태로의 수요반응 전개 시나리오가 가능하다. 수요관리사업자는 전력거래소나 Utility 회사에서 전력을 구매하거나 수요자원 계약을 맺어 수요반응 자원을 관리하게 한다. 수요관리사업자는 주거 영역에 대한 수요반응 자원 관리를 한다. 주거 공간의 많은 자원을 관리하기 위해서는 위와 같은 다양한 계층 구조를 효율적으로 구성하여야 하며, 그림 3의 (b)처럼 BEMS(Building Energy Management System)나 HEMS(Home Energy Management System)와 연결되어 있는 Smart Appliances와 IoT Devices들 간에도 수요반응 기술을 확대하기 위한 메시지 전송 프로토콜 연구가 필요하다.



(a) Demand response communication structure

(b) Deployment Scenarios of Demand Response

그림 3. 수요반응 통신 구조 및 전개 시나리오
Fig. 3. Architecture and Deployment Scenarios of Demand Response

(2) 수요반응 프로그램들

수요반응 프로그램은 그림 4의 (a)와 같이 요금 기반 수요반응과 인센티브 기반 수요반응으로 나눠진다. 요금 기반 수요반응은 소비자가 지불하는 전력 가격을 시간별로 차등 적용하여 소비자가 이에 반응하여 전력 사용패턴을 변화시키는 방식이며, 인센티브 기반 수요반응은 전력 공급자 또는 전력 시스템 운영자에 의해 운영되며 소비자의 전력 사용을 줄이게 하여 이에 대한 인센티브를 지급하는 방식이다. 가격 기반 수요반응 프로그램들은 시간별 차등 적용된 가격정보를 일괄적으로 보내며, 인센티브 기반 수요반응 프로그램들은 수요반응사업자와 감축전력량을 사전에 계약하고 이에 대한 수요반응 신호를 중요도 level 값으로 보낸다.

그림 4의 (b)는 OpenADR Alliance에서 제공하는 DR Program Implementation Guide에서 수요반응 프로그램 Event 예시로 보여주고 있는 프로그램 별 Use case이다. 가격 기반 수요반응 프로그램 중 하나인 CPP(Critical Peak Pricing)의 Event와 인센티브 기반 수요반응 프로그램 중 하나인 Emergency DR(or Fast DR Dispatch)이다. 그림 4의 CPP와 Emergency DR 프로그램을 보면 수요반응 신호에 수요반응 중요도를 Level로 지정 하고 Price 정보를 포함을 알 수 있다.

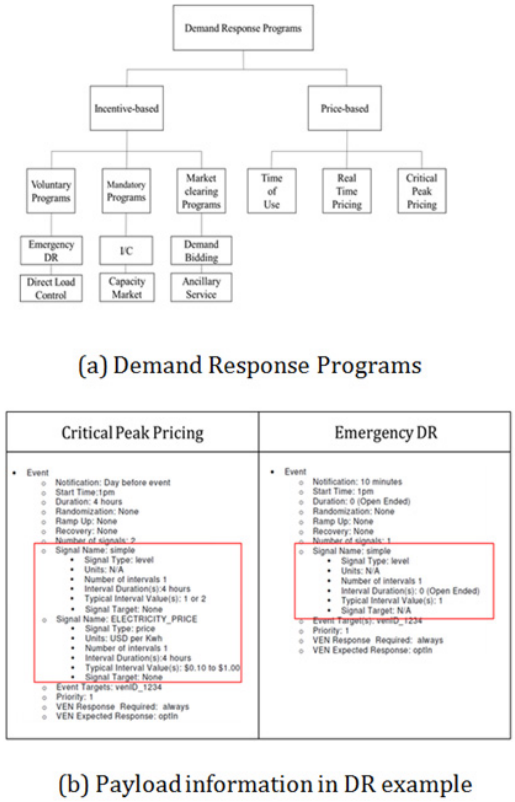


그림 4. 수요반응 프로그램 및 예시
Fig. 4. Demand Response Programs and Examples[9]

수요반응 서비스를 위해서는 에너지 기기 내의 에너지 사용 정보를 얻거나 다양한 에너지 관련 기기나 시설에 대한 고유 인증절차, 에너지 사용을 제어하는 명령을 전달하기 위해 높은 수준의 보안성과 개방성이 요구된다. 따라서 IoT 소형기기 및 다양한 전력기기들이 표준화된 방법으로 연동하고 안전하게 정보를 전달할 수 있는 표준화된 수요반응 프로토콜이 필요하고 CPP와 Emergency DR 프로그램처럼 수요 반응의 중요도가 높고 빠르게 정보를 전달할 수 있는 기존의 Unicast 방식의 통신이 아닌 새로운 통신 방식이 필요하다. 현재 수요반응 이벤트(DR Event)를 다수의 수요자에게 전송할 경우 Unicast 기반의 중복 메시지로 보내기 때문에 Virtual Top Node(VTN)에 불필요한 자원 낭비와 부하가 발생이 된다. IoT 소형기기 및 다양한 전력기기들이 안전하게 전달할 수 있는 프로토콜은 IoT 프로토콜이 적합하며, 수요반응의 중요도가 높고 수요반응 이벤트의 정보를 다수의 수요자에게 빠르게 전달하려면 Multicast 메시지 전송 방식이 필요하다.

1.3 OpenADR Protocol 및 제한점

1.3.1 OpenADR2.0b

OpenADR은 Open Alliance에서 배포한 지능형 수요반응에 적용되는 표준 통신 프로토콜이다. 현재 OpenADR은 1.0버전 이후 2.0버전까지 나왔으며, 기존의 1.0버전이 제한된 벤더의 수, 인증 프로그램의 미비, 비 국제 표준으로 인한 상호 운용의 어려움, 로컬 DR 프로그램에 맞춰진 통신 등의 단점을 가지고 있었던 반면, 2012년 개발 된 2.0 버전은 다수의 벤더 수용, 인증 테스트 도구, Organization for the Advancement of Structured Information Standards (OASIS) 기반 국제 표준, 대다수 DR 프로그램에 적용되는 유연성 등의 장점을 가진다[10].

OpenADR은 전송 및 보안 메커니즘을 포함하는 통신 데이터 모델로서 2개의 노드, 즉 전력 서비스 제공자와 고객 사이의 정보 교환을 제공한다. OpenADR에서 수요반응 통신 참여자는 Virtual Top Node(VTN)과 Virtual End Node(VEN)으로 구성되며 VTN은 OpenADR 통신에서 수요관리 Event를 생성하는 역할을 하며 End Device나 중간 Aggregator 서버들에게 OpenADR Signal을 전송한다. VEN은 일반적으로 OpenADR 통신에서 클라이언트 역할을 하며 수요반응 Event에 응답한다.

그림 5에서 보여주는 바와 같이 OpenADR2.0b 표준의 통신 프로토콜은 TCP/IP기반의 HTTP와 XMPP를 사용하며, XML형태의 데이터 포맷을 이용한다. HTTP와 XMPP 프로토콜은 Server-Client 구조를 갖으며 오직 Unicast 메시지 전송 방식만을 지원한다. 일반적인 Multicast와 Unicast라고 가정하면 Multicast가 데이터중복 전송으로 인한 자원 낭비를 최소화 할 수 있다. Multicast 방식의 경우에는 데이터 복사가 서

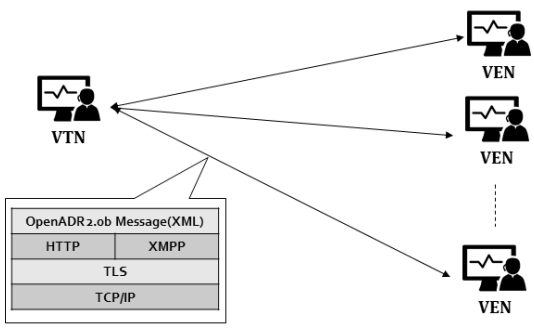


그림 5. Unicast 방식의 OpenADR2.0b 모델
Fig. 5. OpenADR2.0b Model of Unicast method

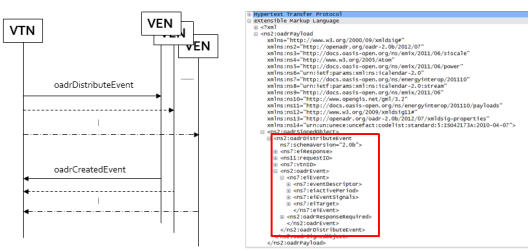
버가 아닌 라우터에서 발생하므로 서버의 부하는 적다. 위와 같은 OpenADR 2.0b 시스템은 전력 사용량에 따라 사용자 수요를 제한하는 에너지 최적화에 한정되어 있다. 따라서 사용자는 에너지 관리에 대한 만족도가 떨어 질 수 있기 때문에 사용자의 에너지 수요에 따른 에너지 분배나 실시간 에너지 사용 제어에 대한 기술이 필요하며 이를 위해 다양한 방법 또한 연구되어 지고 있다[11,12].

본 논문의 목적은 Broker 기반의 Publish/Subscribe 통신 구조에서 Multicast 기반 경량 수요반응 프로토콜을 제안, 구현 및 성능 분석에 있다. 또한, Lightweight 기반 IoT 프로토콜과 JSON 방식의 메시지 포맷 기술을 사용한다.

OpenADR2.0b은 기본적으로 네 가지 서비스(EiRegisterParty, EiReport, EiEvent, EiOpt)를 제공한다. 수요자원의 사용 패턴 변화를 유도하기 위해 Event Signal을 보내는 서비스는 EiEvent이다. 그림 6-(a)에서와 같이 EiEvent서비스를 위한 과정은 VTN에서 Event Signal을 담은 DistributeEvent 메시지를 전송하며, 이에 대한 응답으로 VEN에서 CreatedEvent 메시지를 전송한다. 그림 6-(b)는 DistributeEvent 메시지의 XML Data Format이다. 메시지를 보내는 VTN에 대한 정보들과 Event Signal에 대한 구체적인 값(설명, 기간, 신호정보, 대상)들이 포함되어 있다.

OpenADR2.0b는 그림 7 (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 Event서비스, (d)이벤트 스케줄에 대한 수용가능여부를 정의하는 Opt 서비스가 있다.

본 논문에서는 그림 7처럼 OpenADR2.0b의 (a)의



(a) EiEvent Message Flow (b) XML Data Format

그림 6. OpenADR2.0b EiEvent 서비스의 메시지 플로우 및 XML 데이터 포맷
Fig. 6. Message Flows & XML Data Format of OpenADR2.0b EiEvent Service

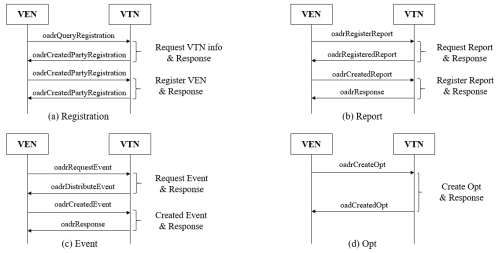


그림 7. OpenADR2.0b 동작 플로우
Fig. 7. OpenADR2.0b Process Flow

등록 과정과 (c)의 Event 과정을 이용하여 경량 IoT 프로토콜과 JSON을 이용해서 경량 수요반응 통신 프로토콜과 Multicast 메시지 전송 방식을 제안한다.

1.3.2 에너지 IoT 환경에서의 OpenADR2.0b 제한점

앞서 기술에 대한 설명과 문제점은 두 가지로 정리할 수 있다. 첫 번째는 텍스트 기반의 heavyweight 메시지 및 인/디코딩에 따른 오버헤드 문제이며, 두 번째는 에너지 IoT 환경에서의 Unicast방식은 동일한 메시지를 보내기 때문에 자원의 낭비 문제가 있다. 이와 같이 주거공간에서의 수요반응 서비스 측면에서는 OpenADR2.0b의 HTTP과 XMPP는 XML데이터를 Payload에 담은 통신은 여러 단점이 존재 한다.

본 논문에서는 에너지 IoT 환경에서 OpenADR 2.0b 제한점을 근거한 Multicast 방식 Lightweight 수요반응 프로토콜의 필요성을 주장한다. 다음은 OpenADR2.0b에 쓰이는 HTTP와 XMPP에 대한 설명이다.

- HTTP(Hypertext Transfer Protocol)

HTTP는 장치 간 Hypertext를 교환하는 프로토콜이다. Binary형식의 프로토콜에 비해 데이터 용량이 큰 프로토콜 헤더를 갖는다. 텍스트 메시지를 인코딩/디코딩하기 위한 파서를 구현해야 되고 파싱과정이 많아 질 경우 프로세싱 오버헤드를 일으킬 수 있다.

HTTP는 Server-Client 구조의 Request-Response 방식으로 데이터 교환이 이루어진다. 기본적으로 PULL방식의 메시지를 전달하기 때문에 많은 수의 Client가 연결된다면 Server쪽에 부하가 커진다. PUSH 방식의 메커니즘 통신을 하기 위해서는 Client 쪽에 별도의 Server를 구현해야 하지만 방화벽 문제가 있기 때문에 통신에 어려움이 발생한다.

- XMPP(eXtensible Massaging and Presence Protocol)

XMPP는 XML형식의 데이터기반 통신을 하는 프로토콜이다. HTTP와 마찬가지로 텍스트 기반 헤더에 따른 프로세싱 오버헤드가 발생 할 수 있다.

XMPP는 HTTP와 다르게 양방향 통신이 가능하여 PULL방식과 PUSH방식의 메커니즘으로 통신이 가능하다. 하지만 Multicast 메시지 전송 방식이 없기 때문에 Server에서 송신하는 동일한 패킷을 Client에게 전달하는 것이 비효율적이다.

1.4 MQTT기반의 수요반응 프로토콜 설계

표 1과 같이 데이터의 전송을 위해서 다양한 통신 프로토콜이 존재한다. OpenADR2.0b 표준에서 정의하고 있는 HTTP와 XMPP는 Multicast를 지원하지 않으며 문자 기반의 메시지 header와 XML 포맷의 payload를 갖기 때문에 소형 장치 같이 제한된 장치에 제약적인 프로토콜이다. 반면 MQTT와 CoAP는 Multicast 메시징 방식이 지원되며 binary header를 갖는 제한된 IoT Devices를 위한 경량 메시지 프로토콜이다. 리소스가 제한된 IoT 장치에 인/디코딩 오버헤드를 줄일 수 있는 수요반응 통신 프로토콜은 MQTT, CoAP이다. AMQP(Advanced Message Queuing Protocol)는 Broker기반 메시지 지향 미들웨어를 위한 개방형 표준 응용 계층 프로토콜이며 DDS는 (Data Distribution Service) 는 실시간 시스템의 실시간성(real-time), 규모가변성(scalable), 안전성(dependable), 고성능 (high performance)를 가능하게 하는 Object Management Group(OMG) 표준 출판/구독 (Publish/Subscribe) 네트워크 커뮤니케이션 미들웨어이다.

표 1.메시징 프로토콜 비교
Table 1. Comparison of Messaging Protocols

	HTTP	XMPP	MQTT	DDS	CoAP	AMQP
Transport	TCP/IP	TCP/IP	TCP/IP	TCP, UDP/IP	UDP/IP	TCP/IP
Network Layer	IP Layer	IP Layer	IP Layer	IP Layer	IP Layer and 6LowPan	IP Layer
Multicast Support	X	X	O	O	O	O
Interaction Model	Server-Client	Server-Client	Publish-Subscribe	Publish-Subscribe, Server-Client	Server-Client	Publish-Subscribe
Security	HTTPS	TLS+ SASL	TLS	TLS, DTLS, DDS security	DTLS	TLS+ SASL
Constrained Devices	No	No	Yes	No	Yes	No
Addressing	URL	JID	Topic	Topic /Key	URL	Topic /Key
QoS	X	X	O	O	O	O

본 논문의 제한점으로는 MQTT 프로토콜을 이용하여 Multicast 메시지 전송 방식의 Lightweight 수요반응 프로토콜을 구현하여 비교분석해 보았다.

MQTT(Message Queue Telemetry Transport)는 TCP/IP 프로토콜 위에서 사용하기 위한 ISO Standard(ISO/IEC PRF 20922) Publish/Subscribe 기반의 경량 메시징 프로토콜이다. 네트워크 대역폭이 제한적인 원격 환경과의 연결을 위해서 설계되었다. 메시지 포맷이 기본적으로 2bytes의 고정헤더가 있으며, 메시지 타입에 따라 가변헤더, 가변길이의 메시지 페이로드가 존재한다. 메시지 크기를 가능한 1bit라도 줄이기 위하여 가변길이를 포함한 메시지 포맷이다.

MQTT Client간 통신은 항상 Broker를 통해 이루어진다. Client는 Broker에게 자신이 원하는 토픽(Topic)을 구독(Subscribe)신청 할 수 있고, 그 이후 해당 토픽으로 발행(Publish)되는 메시지를 모두 받을 수 있게 된다. 반대로 Client가 발행하고 싶은 메시지가 있다면 Broker에게 원하는 토픽으로 메시지를 보내어, 해당 토픽을 구독하고 있는 Client들에게 메시지를 전달해 준다. 즉, Unicast와 Broadcast, Multicast 방식의 메시지 전송은 메시징이 모두 가능하여 1:1뿐만 아니라 Broker을 통해 1:N / N:1 / N:N 등 사물간 다양한 형태의 정보교환을 지원 할 수 있다.

MQTT는 신뢰성 있는 통신을 위하여 3단계(0, 1, 2)의 QoS를 지원한다. Level 0 방식은 오직 한번만 메시지를 보내며 Ack를 기다리지 않는다. Level 1은 메시지를 보내기 전 저장해 놓았다가 Broker에게 Ack를 받았을 경우 저장해 놓은 버퍼를 비우고, Ack를 받지 못하였을 경우엔 다시 한 번 메시지를 보낸다. Level 2는 Broker가 전달 받은 메시지의 ID값을 다시 한 번 확인하는 메시지 PUBREC를 Client에게 보내고, 이것에 대한 Ack(PUBREL)가 왔을 때 Subscriber에게 메시지를 보낸 후 Broker에 저장되어 있는 메시지를 삭제한다. 이 후 Client에게 모든 것이 완료됐다는 PUBCOM를 보내고 Client는 버퍼를 비우게 된다. MQTT 기능인 QoS를 통하여, IoT 환경의 패킷 손실이 있어 품질이 열악한 네트워크에서도 신뢰성을 보장받는 통신을 할 수 있게 된다. MQTT Broker 기반의 Multicast 메시지 전송 방식은 MQTT publish로 Topic으로 그룹화해서 broker에게 보내면 해당 topic을 구독

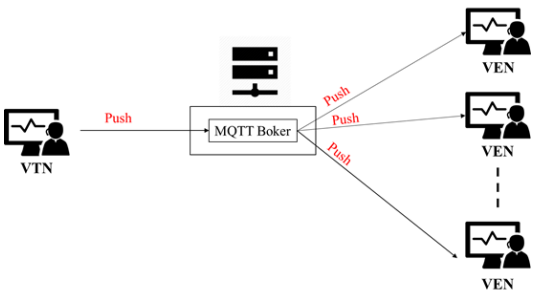


그림 8. MQTT Multicast 기반 메시지 전송
Fig. 8. MQTT Multicast based message transmission

하는 모든 Client가 subscribe로 메시지를 받는다.

앞서 설명한 바와 같이 MQTT는 경량화 된 2bytes Message Header를 가짐으로써 경량의 데이터를 전송한다. 또한 Broker에게 Publish-Subscribe방식의 메시지 교환을 하기 때문에 VEN들이 동일한 그룹토픽을 구독할 경우 그림 8과 같이 Broker를 통한 Multicast 방식 메시지 전송이 가능하다.

Ⅱ. 본 론

2.1 MQTT 기반 Multicast 방식 설계

2.1.1 Unicast, Multicast 방식 비교

수요 반응에서 Multicast 메시지 전송 방식의 목적은 송신 측에서 동일한 메시지를 여러 번 메시지 전송을 하여 불필요한 자원 낭비를 하지 않기 위해서이다. 그림 9는 Unicast 메시지 전송 방식과 Muticast 메시지 전송 방식에 대한 차이이다. Unicast 메시지 전송 방식은 VTN이 N개의 VEN들에게 동일한 메시지를 각각 보내면 Broker에서 순차대로 N개의 VEN들에게 보내는 방식이다. Muticast 메시지 전송 방식은 VTN이 동일한 메시지 한 개만 보내면 Broker가 N개의 VEN들에게 보내는 방식이다. 따라서 송신 측(VTN)에서 N-1개의 메시지에 대한 불필요한 자원 낭비와

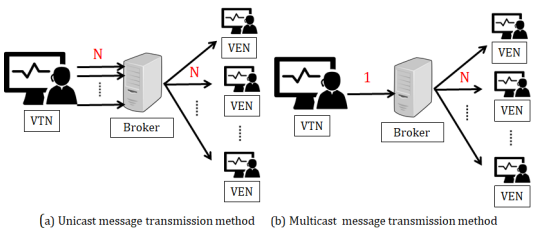


그림 9. Unicast/Muticast 메시지 전송 방식에 대한 차이
Fig. 9. Unicast/Muticast Message transmission method difference

부하가 발생을 예방할 수 있다. 예로 VTN이 Unicast 방식으로 VEN에게 각각 동일한 메시지 100개를 보내면 99개의 메시지에 대한 불필요한 자원 낭비와 데이터 트래픽으로 인한 Broker에 부하가 생길 수 있다.

2.1.2 MQTT의 Topic 기반 그룹화 메시지 전송

그림 10처럼 VTN이 Broker를 통해 그룹화 된 Group 1의 VEN들에게 메시지 전송을 보내면 Group 1에 속한 VEN 모두 수신 하게 된다. 즉, MQTT의 Multicast 메시지 전송 방식은 Topic의 발행(Publish) 메시지를 보내면 Client들이 동일한 글자가 들어간 Topic을 구독(Subscribe)하면 Broker로부터 메시지를 전송 받기 때문에 Topic으로 그룹화가 가능해진다. 예로 Client가 OpenADR#으로 구독을 하게 되면 OpenADR가 붙은 Topic의 메시지는 선택적으로 받을 수 있다.

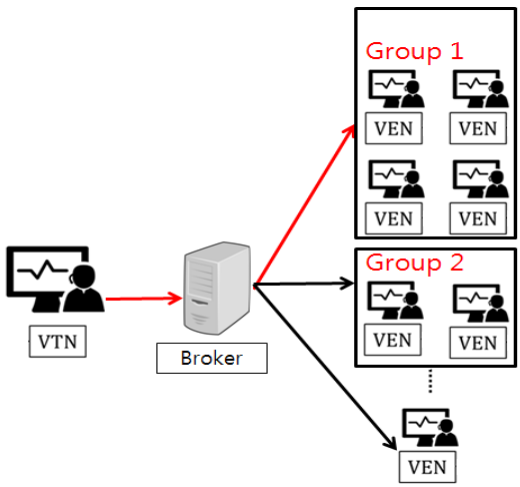
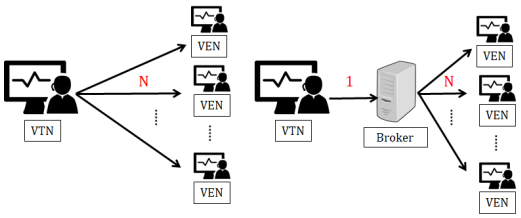


그림 10. Broker 기반 Multicast 방식의 그룹화 메시지 전송
Fig. 10. Group Message Transmission of Broker Multicast

2.1.3 Borker기반 Publisher/Subscriber의 구조

본 논문의 Multicast 방식 MQTT 기반 OpenADR 수요반응 프로토콜의 통신구조는 Broker가 Server가 되고 VTN과 VEN들이 Client가 된다. 따라서 VTN과 VEN은 Publisher/Subscriber가 되며 Broker을 통해서 통신하게 된다. VTN과 VEN의 모든 연결이 Broker에 연결이 되어 Broker 부하는 늘 수 있으나 VTN에 연결되는 세션의 수는 1개 이므로 VTN에서 부하가 적다. 하지만 HTTP/XML 기반 OpenADR2.0b는 VTN이 Server이고 VEN이 Client이다. VTN 1개당 VEN가 N개이면 세션의 수는 N개가 되므로 VTN이



(a) HTTP/XML OpenADR (b) MQTT/JSON OpenADR

그림 11. HTTP/XML 기반 OpenADR 구조와 MQTT/JSON 기반 OpenADR 구조
Fig. 11. OpenADR architecture based on HTTP / XML and OpenADR architecture based on MQTT / JSON

부하가 많다.

수요반응참여고객들에게 Server/Client 방식의 Unicast 메시지 전송 방식으로 VTN이 수요반응 이벤트를 내리면 Server인 VTN의 부하가 많아질 것이다. 그림 11에서 (a)는 HTTP/XML기반 OpenADR는 Unicast 메시지 전송 밖에 없기 때문에 여러 세션을 맺고 있는 클라이언트들에게 순차적으로 같은 메시지를 여러 번 전송하기 때문에 부하로 인해 메시지 전송의 지연 시간이 길다.

그림 11의 (b)도 MQTT/JSON 기반 OpenADR의 Unicast 방식으로 N개의 VEN에 전송하게 되면 VTN에서 Broker로 보내는 메시지가 N개가 되고 Broker가 VTN에게 받는 메시지가 증가한다. Multicast 방식으로 메시지를 전송하면 VTN이 Broker에게 메시지를 한번만 전송하고 Broker가 VEN들에게 그룹화 전송해서 부하가 적다. 1대 대 다수의 수요반응 통신을 했을 경우 메시지의 오류와 동시에 전송이 될 경우 속도 저하가 일어날 수도 있다. 즉, 집합 건물 환경이 아닌 개인 주거 공간 형태(그룹화가 되지 않은 형태)의 수요반응참여자가 수요반응 이벤트를 단일 메시지로만 받아야 할 때 속도 저하가 일어난다.

2.1.4 MQTT Push 메커니즘 기반의 Multicast 수요반응 메시지 전송

그림 12의 (a)처럼 Pull 메커니즘은 주기적인 Poll 메시지의 응답 메시지로 수요반응 이벤트 Event를 송신 했다. 그림 12의 (b)의 Multicast 방식은 주기적인 Poll 메시지가 없는 Push 기반 전송이기 때문에 데이터 트래픽 양이 적고 장치의 리소스 사용에 효율적이다. VTN에서 Event가 생기면 즉시 VEN으로 메시지를 전송하기 때문에 실시간 이벤트 전송에 적합하다. 또한 Multicast 메시지 전송 방식 기반으로 다수의 Client에 메시지를 전송하기 때문에 전송 속도가 빨라진다. Multicast 방식은 Push 메시지 방법으로 구현되

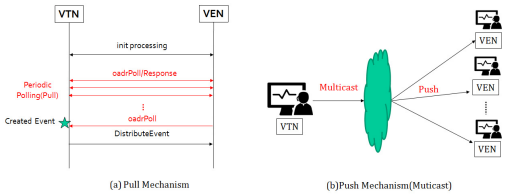


그림 12. Pull & Push 방식의 전송 메커니즘
Fig. 12. Interaction Mechanism of Pull & Push

기 때문에 방화벽 문제로 인해 통신에 어려움이 생긴다. 방화벽 문제를 Broker Server가 있는 Publish/Subscribe방식의 통신 방법으로 해결할 수 있다.

2.2 수요반응에서 Multicast 방식 메시지 전송의 사용 시나리오

수요반응에서 Multicast 메시지 전송 방식의 시나리오는 3가지 예시를 들 수 있다.

첫째, 그림13의 (a)처럼 VTN이 선택적으로 다수의 VEN에게 수요반응 이벤트를 송신할 때 Multicast 메시지 전송 방식을 이용할 수 있다. 전력 값은 변화하기 때문에 특정 시간에 수요반응 이벤트를 정확히 내려 줘야한다. 둘째, 그림 13의 (b)처럼 VEN이 특정 다수의 VTN에게 전력 누적량을 UpdateReport 해주는 것에 쓰일 수 있다. 셋째, 그림 13의 (c)와 (d)처럼 산업설비나 큰 빌딩에서 실시간 적으로 공급보다 전력이 수요가 많을 때 특정 VEN이 수요자원 관리 사업자에게 일정 시간 때 수요반응 취소를 요청하면 수요자원 관리사업자의 VTN가 특정 VEN들에게 수요반응 이벤트를 취소 메시지를 전송할 때 쓰일 수 있

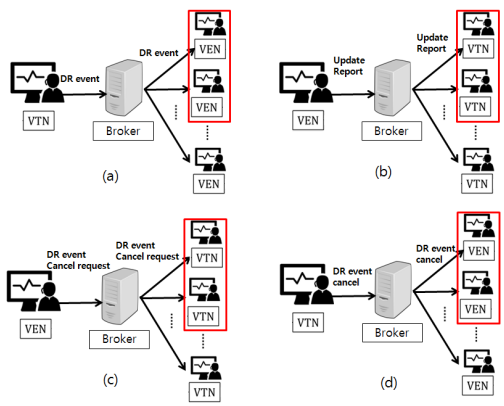


그림 13. 수요반응에서 Multicast 메시지 전송의 사용 시나리오 구조
Fig. 13. Scenarios for the use of Multicast message transmission in demand response

표 2. XML과 JSON 포맷의 Payload
Table 2. Payload of XML and JSON Format.

Format	Payload	Data
XML	<ns1: oadrPayload xmlns:ns1 ="http://openadr.org/oadr-2.0b/2012/07" xmlns:ns2 ="http://docs.oasis-open.org/ns/energyinterop/201110" .... xmlns:ns14 ="urn:un:unece:uncefact:codelist:standard:5: ISO42173A:2010-04-07"> <ns1:oadrSignedObject> <ns1:oadrResponse ns2:schemaVersion="2.0b"> **<ns2:eiResponse> ---------- (a)** **<ns2:responseCode>200</ns2:responseCode> ---- (b)** <ns2:responseDescription>OK </ns2:responseDescription> **<ns3:requestID></ns3:requestID> ---------- (c)** **</ns2:eiResponse> ---------- (a)** **<ns2:venID>cc085337</ns2:venID> ---------- (d)** </ns1:oadrResponse> </ns1:oadrSignedObject> </ns1:oadrPayload>	499 bytes
JSON	{"Service":"eiResponse","responseCode":200,"requestID":null,"venID":"cc085337" } (a) (b) (c) (d)	75 bytes

	Data Mapping	
entity	XML	JSON
Service (a)	<ns2:eiResponse>.... </ns2:eiResponse>	“Service”:“eiResponse”
Response Code (b)	<ns2:responseCode>200 </ns2:responseCode>	“responseCode”:200
Request ID (c)	<ns3:requestID> </ns3:requestID>	“requestID”:null
VEN ID (d)	<ns2:venID>cc085337 </ns2:venID>	“venID”:“cc085337”

다. 본 논문의 제한사항으로 3가지 시나리오 중에 첫 번째 시나리오만 실험하고 분석한다.

2.3 경량 OpenADR2.0b의 JSON 포맷

이전 논문[6]에서는 소형의 네트워크 장비에서도 원활한 통신이 가능한 경량 메시지를 위한 CoAP 이용하여 에너지 IoT 환경에 알맞은 경량의 수요반응 프로토콜을 구현하고 비교 분석하였다. CoAP/JSON 기반 OpenADR2.0b 프로토콜이 HTTP/XML 기반 OpenADR2.0b 보다 데이터 트래픽 양이나 전송 속도에 대한 이점이 있었다. 이전 논문[6]에서 표 2처럼 IoT 경량 데이터 모델 Payload를 비교한 적이 있다. 표 2의 Poll 메시지는 VEN이 주기적으로 수요반응 이벤트가 있는지 VTN에게 질의하는 메시지이며 VTN이 수요반응 이벤트가 있을 때 수요반응 이벤트 메시지를 응답으로 내리게 된다. 본 논문에서도 MQTT/JSON 기반 OpenADR 2.0b도 CoAP/JSON기반 OpenADR2.0b[6]과 같은 데이터 모델링을 따랐다. XML은 확장이 용이하고 쉽게 해석과 작성이 가능하다는 장점이 있으나 불필요한 태그로 인한 용량의 증가와 반복구조 혹은 배열 형식을 나타낼 때 불필요한 데이터가 중복되어 나타나기 때문에 파싱의 속도가 느려진다는 단점을 갖는다. 그에 비해 제안 하는 JSON 기반의 데이터 모델링은 내용을 계층적이지 않고 직렬화 된 구조이다. JSON으로 설계한 데이터 모델링이 직렬화 된 구조이기 때문에 계층화 된 구조 보다 시간 복잡도 측면에서 처리 속도가 빠르다. 또한, 함축적으로 최소한의 값만을 갖기 때문에 상대적으로 용량이 작으며, 작은 용량으로 인해 빠른 파싱 속도를 갖는다는 장점이 있다. 또한 객체와 배열의 구조를 갖고 있기 때문에 데이터 구성이 보다 효율적으로 가능하고 간편한 구현의 장점을 갖는다. 하지만 구조가 함축적이기 때문에 대용량의 메타 데이터 전송에 적합하지 않다는 단점을 갖는다.

Ⅲ. 실 험

3.1 실험환경

본 실험을 위한 전체 스마트 홈 에너지 IoT의 환경 구성은 그림 14와 같다. 스마트 에너지 IoT 환경에서 VTN은 Linux Ubuntu PC에서 실행되며 에너지 사용 현황 모니터링 및 기기의 제어가 가능한 에너지 관리

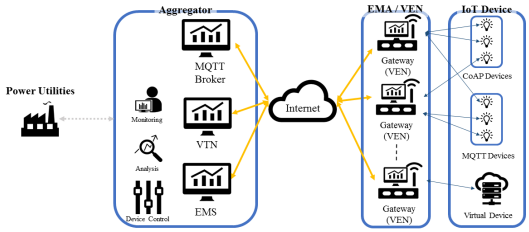


그림 14. 스마트 에너지 IoT 환경 구성도
Fig. 14. Architecture of Smart Energy IoT Environment

시스템의 역할을 수행한다. 각 각의 VEN은 유무선 공유기를 위한 OpenWRT(Open Wireless Router)라는 임베디드 리눅스 운영체제를 설치해 놓은 라우터 위에서 실행된다. 해당 라우터는 에너지 IoT 환경에서 게이트웨이의 역할을 겸하며 개별적으로 IoT 프로토콜을 이용하여 IoT 장비들과 직접적인 통신을 통하여 에너지 관리 에이전트의 역할을 수행한다. 실험 기기의 사양은 다음의 표 3과 같다. 비교를 위하여 HTTP/XML을 사용하는 OpenADR2.0b 프로토콜은 EPRI 사에서 공개하고 있는 오픈소스를 이용하여 실험 하였다.

표 3. 실험 환경
Table 3. Experiment Environment

		HTTP/XML		MQTT/JSON	
		VTN	VEN	VTN	VEN
PC Spec	CPU	Intel Core i5-4200M 2.50GHz	Atheros AR9132 400MHz	Intel Core i5-4200M 2.50GHz	Atheros AR9132 400MHz
	RAM	4GB	64MB	4GB	64MB
Operating System		Ubuntu 14.04 LTS	OpenWRT	Ubuntu 14.04 LTS	OpenWRT
Language		Jruby	C++	Java	C

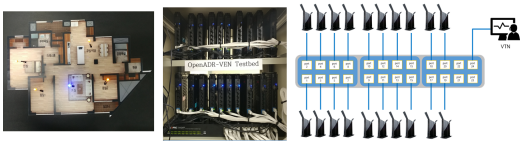


그림 15. 실험 환경
Fig. 15. Test Environment

표 4. 실험 측정 항목 및 측정 방법
Table 4. Measurement list and methods

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

그림 15와 같은 실험 환경에서 HTTP/XML 포맷의 OpenADR2.0b 프로토콜과 MQTT/JSON 포맷의 OpenADR2.0b 프로토콜은 표 4와 같이 Event response time, Data traffic 두 가지 항목에서 비교 및 분석한다.

3.1.1 VEN 개수에 따른 Event Response Time 비교

본 논문에서 HTTP/XML과 MQTT/JSON 방식의 Unicast 방식의 데이터 트래픽과 이벤트 응답 시간을 비교 했다. MQTT Multicast 기반의 Lightweight 수요반응 프로토콜의 검증 및 기존 방식과의 비교를 위해 MQTT Unicast, Multicast 방식을 비교 분석 하였다. 첫 번째 비교 항목은 VEN의 개수에 따른 Event Response Time이다. 본 실험에서 VTN은 VTN에게 연결 된 모든 VEN들에게 DistributeEvent 메시지를 Unicast 방식 또는 Multicast 메시지 전송 방식으로 전송하며, 이에 대한 응답으로 모든 VEN으로부터 CreatedEvent를 받은 시간까지 측정하였다. 그림 16은 Multicast 메시지 전송 방식의 대한 메시지 플로우와 Wireshark이고 Unicast 방식, Multicast 방식 모두 그림 17처럼 DistributeEvent, CreatedEvent 메시지 JSON 데이터 모델링을 따랐다.

그림 18은 두 가지 방식의 Event Response Time에

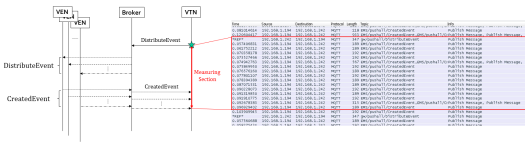


그림 16. Event Response Time 측정 구간
Fig. 16. Measuring Section of Event Response Time

(a)DistributeEvent

```
DistributeEvent{
"Responsedescription":String,
"EndTime":integer,
"RequestID": String,
"StartTime":integer,
"Service":"String,
"EndYMD": integer,
"Response": integer,
"Value":double,
"OptType": String,
"StartYMD": integer,
"EventID": integer,
"ModificationNumber": integer,
"TargetVEN": String
}
```

(b)CreatedEvent

```
CreatedEvent
{
"EMAID": String,
"VENID": String
"Version": integer,
"Response": integer,
"RequestID": String
"OptType": String,
"Value": double,
}
```

그림 17. DistributeEvent, CreatedEvent 메시지의 JSON 데이터 모델링
Fig. 17. Modeling JSON data for DistributeEvent, CreatedEvent message

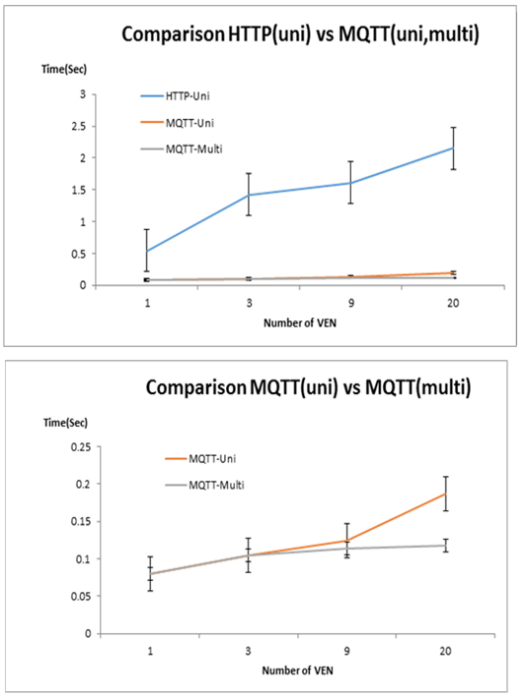


그림 18. HTTP Unicast방식과 MQTT Unicast/Multicast방식에 따른 Event Response Time 비교 그래프
Fig. 18. Comparison Graph of Event Response Time according to HTTP Unicast and MQTT Unicast/Multicast

대한 비교를 보여준다. 두 방식 모두 VEN의 개수에 따라 증가하는 추세를 보이지만 HTTP Unicast 방식이 VEN의 개수에 따라 1개 일 때 0.5435sec, 3개 일 때 1.1526sec, 9개 일 때 1.367sec, 20개 일 때 2.3389sec이며 MQTT Unicast 방식의 VEN 개수에 따른 응답시간이 각각 0.0798sec, 0.1049sec, 0.1243sec, 0.1872sec이다. 반면 MQTT Multicast 방식의 VEN 개수에 따른 응답시간은 각각 0.0795sec, 0.1051sec, 0.1134sec, 0.1170sec으로 측정되었다. MQTT프로토콜을 이용한 방식이 기본적으로 15배 정도 빠른 반응속도를 보이며 Unicast와 Multicast 방식의 차이는 VEN 개수가 적을 때는 거의 차이를 보이지 않다가 VEN 개수가 늘어날수록 급격하게 차이가 벌어지며 VEN 20개일 때, 약 1.6배 정도의 응답속도 차이가 나는 것을 확인 할 수 있다. VEN 개수가 100개, 1000개 등으로 늘어날수록 응답속도는 이보다 더 큰 차이를 보이게 될 것으로 예상 된다.

3.1.2 VEN 개수에 따른 Data Traffic 비교

두 번째는 VEN의 개수에 따른 Data Traffic이다. 그림 19의 구조(a)에서와 같이 VTN으로 들어오는 패

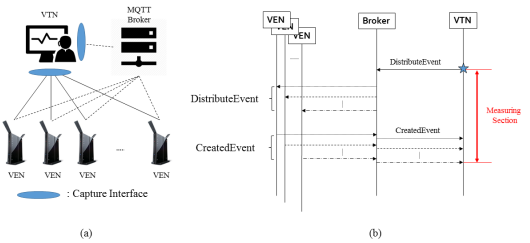


그림 19. Data Traffic 측정 구간
Fig. 19. Measuring Section of Data Traffic

킷을 캡쳐 하였으며, 그림 19-(b)에서 보이는 것과 같이 VTN에 연결되어 있는 모든 VEN에게 이벤트를 전송하고 해당 이벤트에 대한 모든 응답까지의 패킷에 대한 데이터양을 측정하여 비교하였다.

그림 20은 각 각 VEN 개수에 따른 HTTP Unicast 방식과 MQTT Unicast/Multicast 방식의 데이터양을 보여준다. 데이터양은 VEN의 개수와 정비례하여 증가함을 확인 할 수 있다. HTTP/XML 포맷과 MQTT/JSON방식의 Lightweight 포맷의 Data Traffic을 비교하여 보여준다. HTTP Unicast방식의 Event-Response데이터양은 VEN의 개수가 1개일 때

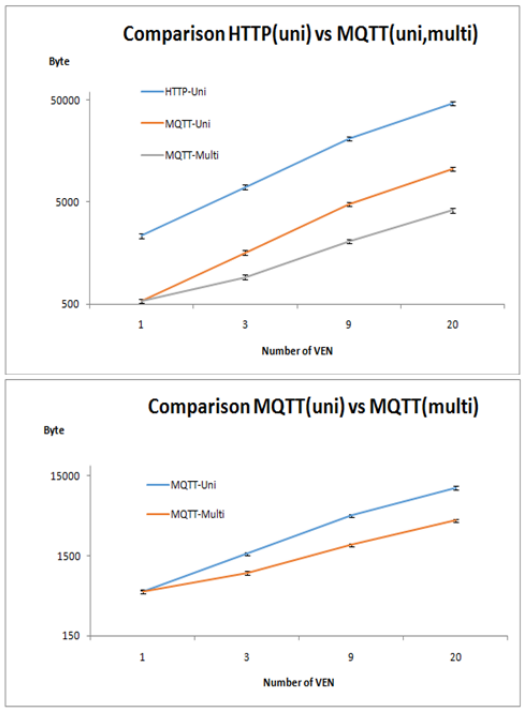


그림 20. HTTP Unicast방식과 MQTT Unicast/Multicast방식에 따른 Data Traffic 비교 그래프
Fig. 20. Comparison Graph of Data Traffic according to HTTP Unicast and MQTT Unicast/Multicast

약 2340bytes, 3개 일 때는 약 7020bytes, 9개 일 때는 약 21060bytes, 20개 일 때는 약 46800bytes이며, MQTT Unicast방식의 Event-Response 데이터양은 VEN의 개수가 1개일 때 약 531bytes, 3개 일 때는 약 1593bytes, 9개 일 때는 약 4779bytes, 20개 일 때는 약 106203bytes이다. 반면 MQTT Multicast 방식은 VEN의 개수가 1개일 때 약 537bytes, 3개 일 때는 약 917bytes, 9개 일 때는 약 2057bytes, 20개 일 때는 약 4147bytes이다. HTTP와 MQTT Unicast를 비교하면 대략 4.5배정도 차이가 나며, MQTT Unicast와 Multicast 방식의 차이는 VEN 3개 일 때 약 1.7배, 9개 일 때는 약 2.3배, 20개 일 때는 약 2.56배로 차이가 점차적으로 증가하는 것을 확인 할 수 있었다. Multicast 방식의 메시지 전송은 송신자(VTN-Broker)인 수요관리사업자나 유틸리티 회사측에서 데이터 트래픽과 송신 시간, 데이터 트래픽이 감소하기 때문에 이점이 있다. 따라서 에너지 IoT 환경에서 기기의 수가 증가하게 될 경우 기존 OpenADR 프로토콜은 데이터의 트래픽이 급격하게 늘어날 수 있는 반면 제안한 MQTT기반 OpenADR 2.0b에 Multicast 메시지 전송 방식을 추가한 수요반응 프로토콜은 트래픽의 증가폭이 낮고 거의 일정하게 유지할 수 있기 때문에 대량의 기기들을 수용할 수 있다.

Ⅳ. 결 론

본 논문에서는 수요반응 서비스를 제공함에 있어서 OpenADR2.0b 프로토콜을 주거 공간의 다수의 수요자들과 소형 네트워크 기기들로 구성 된 스마트 에너지 IoT 통신 환경에 적합한 MQTT 기반의 Lightweight 수요반응 프로토콜에서 Multicast 메시지 전송 방식을 제안 하였다. 제안된 Multicast 메시지 전송 방식은 JSON 데이터 포맷으로 기존의 HTTP/XML 프로토콜 기반의 OpenADR2.0b 프로토콜과 비교하여 전송부하를 줄이면서도 Unicast 전송 방식에 비해 전송 시간을 줄였다. 실험 결과 본 논문에서 전송 메시지 량에서도 MQTT/JSON과 HTTP/XML Unicast 전송 방식을 비교해 봤을 때 MQTT/JSON Unicast 전송 방식이 약 1/4.5배 수준으로 데이터 트래픽이 줄어들었으며, 약 15배 더 빠른 응답시간을 나타내는 것을 확인 할 수 있었다. 또한, MQTT/JSON Unicast와 Multicast 전송 방식을 비교해 봤을 때 Multicast 전송 방식이 1/1.6배 수준으로 데이터 트래픽이 줄었으며, 약 2.56배 더 빠른 응답시간을 나타내는 것을 확인 할 수 있었다. 최종적으로 MQTT Multicast 메시지 전송 방식이 추가된 OpenADR 2.0b이 가장 빠른 응답시간과 데이터 트래픽 양이 낮았다. 따라서 리소스가 제한된 IoT 장치에 인/디코딩 오버헤드를 줄일 수 있는 수요반응 통신 프로토콜은 MQTT/JSON이다. 본 논문에서 제안한 Multicast 기반의 MQTT 프로토콜은 Lightweight 수요반응 프로토콜을 제공하고 주거 공간의 다수의 수요자들과의 통신 및 소형 네트워크 기기를 갖는 에너지 IoT 환경에서 원활한 Multicast 방식 수요반응 자동화 서비스가 제공되기 때문에 널리 활용될 것으로 기대된다.

향후과제로는 VTN 1개와 여러 VEN이 통신하는 1대 n방식에서 앞으로는 Microgrid와 같은 분산 에너지 자원(Distributed Energy Resource: DER) 환경에서의 VEN들이 여러 VTN과 n대 n 수요반응 프로그램을 등록하여 보다 더 효율적인 수요반응 서비스가 가능 할 것으로 보이며, 이러한 상황에 따른 n:n방식의 수요반응 프로토콜에 대한 연구가 필요 할 것으로 보인다.

References

[1] C. A. Craig and S. Feng, "Exploring utility organization electricity generation, residential electricity consumption, and energy efficiency: A climatic approach," *Applied Energy*, vol. 185, pp. 779-790, 2017.

[2] http://www.openadr.org/

[3] 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, Feb. 2014.

[4] 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: CD N1857, 2017

[5] H. Lee, S. Oh, S. Ko, B. Lee, J. Nam, Y. Kim, and S. Lee, "Research on standard of

internet of things for power and energy sectors," *J. KICS*, vol. 33, no. 12, pp. 12-21, Nov. 2016.

[6] H. Park, S. Kim, S. Kang, H. Park, I. Kim and J. Choi, "Implementation and analysis of CoAP-based lightweight OpenADR2.0b protocol for smart energy IoT environment," *J. Korea Inst. Inf. Commun.*, vol. 42, no. 4, pp. 904-914, 2017.

[7] M. H. Albadi and E. F. El-Saadany, "A summary of demand response in electricity markets," *Electric Power Syst. Res.*, vol. 78, no. 11, pp. 1989-1996, 2008.

[8] Paulson Institute, *Demand Response*, 2015.

[9] OpenADR 2.0 Demand Response Program Implementation Guide, OpenADR Alliance.

[10] O. Alliance, *The OpenADR primer*, ed: Technical report, 2012.

[11] S. C. Kang and J. S. Choi, "Design and implementation of realtime demand and response gateway in smart home based on MQTT," in *Proc. Symp. KICS*, pp. 60-61, Jun. 2016.

[12] S. Y. Kim, B. W. Jang, and J. S. Choi, "Design and implementation of real time demand response protocol based on OpenADR," *JCCI*, 2016.

박 헌 일 (Heon Il Park)

2015년 2월 : 한국외국어대학교 정보통신공학과 학사

2018년 8월 : 한양대학교 컴퓨터 소프트웨어 석사

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

박 현 진 (Hyun Jin Park)

2017년 2월 : 한신대학교 컴퓨터 공학부 학사

2017년 3월~현재 : 한양대학교 컴퓨터 소프트웨어 석사과정

<관심분야> IoT, Smart Home, Smart Grid

이 성 환 (Sung Hwan Lee)

2009년 2월 : 단국대학교 전기전자컴퓨터공학부 학사

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