논문 19-44-08-12 The Journal of Korean Institute of Communications and Information Sciences '19-08 Vol.44 No.08
https://doi.org/10.7840/kics.2019.44.8.1527

사물인터넷 기기의 웹 서비스를 위한 CoAP 통신 성능 평가

하 용 준◆, 추 영 열°

Performance Evaluation of CoAP Communication for Web Service of Internet-of-Things Devices

Yong-jun Ha◆, Young-yeol Choo°

요 약

다양한 사물인터넷 응용에서 무선 센서 네트워크와 같이 낮은 계산 성능과 제한된 메모리 용량을 갖는 장비들의 활용이 증대되고 있다. IETF에서는 이와 같이 제한된 성능을 가지는 장비들의 인터넷 접속을 위해 경량화된 통신 표준으로 CoAP를 제안하였다. 그러나 CoAP은 전송계층으로 UDP를 사용하고 있어 TCP와 같은 전송의 신뢰성을 제공하지 못한다. 이를 보완하기 위해 확인형 메시지 형식을 제시하였고 이 형식을 사용하는 통신에서는 수신측에서 응답을 보내도록 규정하고 있다. 또한, 메시지 송신후 일정 시간 내 응답이 없으면 재전송과정을 수행한다. 재전송 타임아웃 값은 RTO 매개변수에 설정된다. 그럼에도 불구하고 WSN에서는 그 제한된 성능으로 인해 통신 지연이 평균 통신지연의 10배 이상인 경우가 발생한다. 통신 오류시 재전송 과정으로 인해 발생하는 이러한 문제는 실시간 사물인터넷 응용에서는 심각한 문제가 된다.

Key Words : Internet-of-Things, Wireless Sensor Networks, CoAP, RESTful, 6LoWPAN

ABSTRACT

Employment of devices such as WSN nodes, which have low computing power and limited memory size, has been increased more in various IoT applications. In order to get those devices with limited capabilities connected into Internet, IETF proposed light-weighted standard communication protocol, CoAP. However, reliable communication is unlikely to be achieved with CoAP because it adopts UDP instead of TCP as the Transport Layer protocol. To cope with this flaw, IETF proposed confirmable(CON) message type, where a recipient of CON message type has to reply with response message. In addition, if there is no response from the recipient within predefined time, the sender goes through re-transmission process. The timeout value is set into RTO parameter of CoAP. Nevertheless, communication delay in the worst case is more than 10 times average delay time. It happens during re-transmission process to deal with communication error, which can cause a serious problem in real-time IoT applications.

※ 본 논문은 중소벤처기업부에서 지원하는 2018년도 산학연협력 기술개발사업(No.S2651983)의 연구수행으로 인한 결과물임을 밝힙니다.
◆ First Author : Tongmyong University Department of Computer Engineering, yongjun86@tu.ac.kr, 학생회원
° Corresponding Author : Tongmyong University Department of Computer Engineering, yychoo@tu.ac.kr, 정회원
논문번호 : 201904-046-D-RU, Received April 12, 2019; Revised May 27, 2019; Accepted June 10, 2019

Ⅰ. 서 론

유비쿼터스 컴퓨팅 개념[1]은 이후 모든 전자적 기능을 갖는 물리적 개체들을 포함하는 사물 인터넷(Internet of Things, IoT)으로 보다 구체화, 확장되었다[2]. 사물 인터넷은 모든 인터넷 통신 기술을 포괄하지만, 그중에서도 무선 통신기술이 주요한 역할을 담당하고 있다. 특히 홈 네트워크나 공장자동화, 재난감시 시스템 등 낮은 비용과 컴퓨팅 사양이 요구되는 분야에서는 WSN(Wireless Sensor Network)은 IoT의 핵심 기술이 될 것이다[3].

인터넷 관련 표준화 기구인 IETF(Internet Engineering Task Force)에서는 WSN 과 같은 낮은 성능을 갖는 기기들의 인터넷 접속을 위해 6LoWPAN(IPv6 over Low power Wireless Personal Area Networks), CoRE(Constrained RESTful Environment) 등의 워킹그룹을 결성하여 표준화를 진행해왔다. CoRE 워킹그룹은 저사양 노드들이 손실이 높고 전송률이 낮은 네트워크 환경에서 경량화된 방식으로 응용계층과 통신할 수 있는 CoAP(Constrained Application Protocol) 표준화를 제안하였다. CoAP의 최대 강점은 RESTful 서비스를 지원한다는 점이다. CoAP 메시지의 메소드는 HTTP와 같이 GET, PUT, POST, DELETE로 구성된다. 응답코드 또한 HTTP와 유사하게 설계되어 있어서 HTTP 클라이언트와 CoAP서버사이 HTTP-COAP 메시지변환을 통해 쉽게 연동할 수 있다. 하지만 HTTP와 CoAP을 연동하였을 때 통신 지연의 주요 요인이 그림 1.의 HTTP를 기반으로 하는 인터넷 망에서 발생하는지, 혹은 CoAP이 구현된 WSN 구간에서 발생하는지 실제 측정을 통해 분석해 보는 것도 필요하다[4-8].

본 논문에서는 HTTP클라이언트(인터넷 환경)와 CoAP서버(WSN의 제약된 환경)를 연결하는 통신시스템을 개발하고, 메시지교환 테스트를 통해 성능측정 및 분석을 수행한다. 성능측정 및 분석을 통해 HTTP-CoAP 간 효율적인 통신방안을 모색하고자 한다. 시스템은 CoAP서버, Border-router ,HTTP_Proxy, Client로 구성되어 있다. Client가 HTTP 요청메시지를 보내면 HTTP_Proxy, Border-router를 통해 CoAP서버로 데이터를 요청하고, 결과 값을 HTTP 응답메시지로 받을 때 까지 소요되는 RTT(Round Trip Time)을 측정하여 성능평가를 실시하였다.

본 논문의 구성은 다음과 같다. 2장에서 이론적배경과 관련연구에 대해 설명한다. 3장에서 CoAP기반 IoT 통신 시스템의 설계 및 성능 측정방법에 대해 기술한다. 4장에서 성능측정 결과 및 분석에 대해 기술하고 5장에서 결론을 맺는다.

Ⅱ. 이론적 배경 및 관련연구

2.1 CoAP 계층구조 및 헤더 압축 기술

CoAP는 제한된 성능 및 전력을 갖는 센서노드와 전송 지연 및 패킷 손실률이 높은 네트워크 환경에서 REST(Representational State Transfer)아키텍쳐 기반의 Resource Discovery, 멀티캐스트 지원, 비동기 트랜잭션 요청 및 응답을 지원하기 위한 경량 프로토콜이다.

그림. 1[8].는 HTTP와 CoAP가 연동되어 동작하는 IoT 통신시스템의 구조를 나타낸다. (a)와 같이 제한적인 환경에서는 클라이언트와 서버가 CoAP를 사용하여 RESTful 방식으로 통신을 하게 된다. 그리고 프록시를 통해 기존의 인터넷 환경과 연동될 수 있다. (b)와 같이 CoAP스택은 HTTP스택과 유사한 구조를 가지고 있어 상호 변환이 용이하다. CoAP는 WSN 환경의 저사양 센서노드들에게 RESTful 방식의 통신을 지원하여 기존의 인터넷과 상호 연동이 가능하도록 한다[7,8].

그림. 2.는 CoAP의 스택 구조를 나타낸다. 하위 계층으로 IEEE802.15.4표준이 주로 사용되나, Wi-Fi나 이더넷 등 여러 표준을 지원한다. 네트워크계층으로는 IPv6를 사용한다. IPv6는 기존의 인터넷 통신 구조를 그대로 활용하면서 대규모 주소 할당이 가능하다는 장점이 있다. 하지만 IPv6는 저사양의 센서노드들이 사용하기에는 많은 전력과 높은 메모리 성능 등을 요구하므로 적합하지 않다. 이에 따라 IETF는

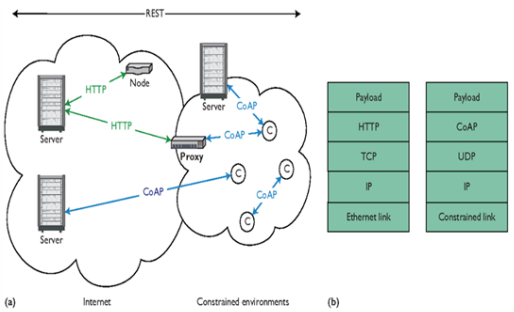


그림 1. (a)HTTP와 CoAP 함께 동작하는 웹 구조 (b)CoAP 스택은 HTTP스택
Fig. 1. (a)Web architecture working with HTTP and CoAP together (b) CoAP stack and the HTTP stack

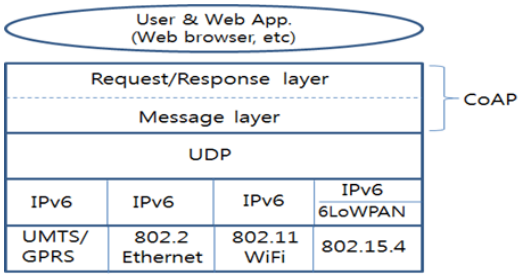


그림 2. CoAP 프로토콜 스택
Fig. 2. CoAP protocol stack

6LoWPAN(IPv6 over Low-Power Wireless Personal Area Networks)워킹그룹을 결성하여 IEEE802.15.4를 기반으로 하는 센서네트워크에서 IPv6를 지원하기 위한 표준을 제안하였다. 그림. 3.은 IEEE802.15.4의 PHY/MAC 구조를 나타낸다. IEEE802.15.4의 PHY 페이로드는 127byte이며, 이는 MAC프레임의 크기와 같다. MAC 프레임의 헤더는 25바이트로 이를 제외하면 102byte의 페이로드를 가지며, 102byte 페이로드 안에는 상위 계층인 IPv6/UDP/CoAP가 포함되어야 한다.

그림. 4. 와 같이 MAC 페이로드 102byte에서 보안 관련 최대 프레임 21bytes를 제외하면 81bytes가 사용가능하여, IPv6헤더 40bytes와 UDP헤더 8bytes를 제외하면 33bytes만 남게 되어 33bytes 이상의 데이터를 처리하려면 단편화 및 재조립 과정이 필요하게 된다.

그림. 5.는 6LoWPAN의 헤더압축을 나타낸다. 6LoWPAN 헤더는 6LoWPAN패킷임을 표시하는 Dispatch코드와 IPv6 정보들을 압축한 LOWPAN_IPHC, IPv6 확장헤더를 위한 LOWPAN_NHC, UDP체크썸으로 구성되며, 48바이

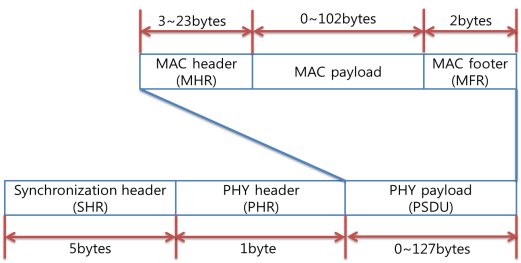


그림 3. IEEE802.15.4의 PHY/MAC
Fig. 3. PHY/MAC of IEEE802.15.4

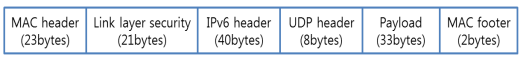


그림 4. IEEE802.15.4의 IPv6 헤더문제
Fig. 4. IPv6 header issues of IEEE802.15.4

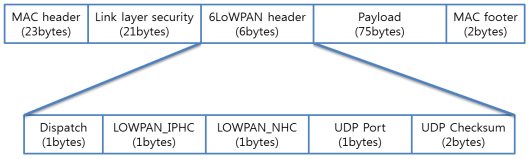


그림 5. 6LoWPAN의 헤더압축
Fig. 5. Header compression of 6LoWPAN

트인 IPv6/UDP헤더를 6byte로 압축하여 75byte의 페이로드를 사용가능하도록 한다[4-6].

전송계층으로는 UDP를 사용한다. UDP 통신은 종단간의 신뢰성 있는 전송이나 중복메시지의 처리를 보장해 주지 않는데, CoAP는 메시지의 신뢰성을 보장하기 위해 확인형(CON), 비확인형(NCON), 승인(ACK), 리셋(RST) 4가지의 메시지 타입을 정의한다. 응용계층과 전송계층 사이에 Request/Response 계층, Message 계층을 구분해서 명시되어 있으며, 서버-클라이언트 구조로 통신을 수행한다.

2.2 CoAP 관련 연구 동향

IoT환경을 구축하기 위한 표준들이 여러 워킹그룹에 의해 진행되고 있다. 그 중에서 대표적인 것으로 애플리케이션 층에서는 CoAP, MQTT가 있고, 인프라구조에서는 IEEE802.15.4, 6LoWPAN, LoRaWAN 등이 있다[9]. MQTT는 ISO표준의 TCP/IP기반 가벼운 메시징프로토콜로 CoAP이 클라이언트-서버 통신 구조를 갖는데 비해 발행-구독 (publish-subscribe) 구조를 가지고 있다[10]. [11]에서는 CoAP와 MQTT(Message Queue Telemetry Transpor)의 성능을 비교/분석하였다. 20%미만의 패킷손실률에서는 MQTT가 CoAP보다 낮은 지연을 보였고, 높은 패킷 손실률에서는 CoAP가 MQTT보다 나은 성능을 보인다는 것을 실험을 통해 확인하고, CoAP와 MQTT는 각각 UDP, TCP 통신프로토콜을 사용하기 때문에 패킷손실에 따른 재전송이 성능차이에 영향을 미치는 것으로 분석하였다[11]. [12]의 연구에서는 CoAP Observe옵션을 사용하여 CoAP를 발행-구독구조를 구현하고, 분산 발행-구조를 IoT환경에 적용하여 성능측정을 하였다. 성능측정 결과, CoAP에 비해 약 45%의 지연을 감소시킨 것을 확인하였다[12]. [13]의 연구에서는 CoAP의 혼잡제어를 위한 방안을 제시하였다. CoAP는 UDP 기반이지만 확인형 메시지를 이용하여 신뢰성을 확보할 수 있다. 확인형 메시지는 RTO(Retransmission Time-Out)안에 ACK를 받지 못하면 메시지를 재전송하도록 하는 옵션이다. CoAP의 센서노드간의 RTT를 측정하여 가변적으로 RTO를

변경하여 CoAP 통신의 혼잡을 제어함으로써 네트워크의 성능향상을 이루었다[13].

한편, 최근에는 Wi-Fi와 같이 비교적 자원이 충분한 장비에 대해서는 종단간 통신의 신뢰성 확보를 위해 TCP 기반의 사물인터넷 통신, CoAP 통신 등의 연구도 제안되고 있다[14-18]. 본 논문에서는 HTTP-CoAP를 연동하여 통신 성능을 측정하고, 성능향상을 위한 방안을 제시한다.

Ⅲ. 성능 측정 시스템 설계

성능측정을 위해 그림. 6.과 같이 웹페이지에서 데이터를 요청하면 HTTP-CoAP 변환을 통해 무선센서노드의 자원을 요청하여 응답하는 통신시스템을 구축하였다.

라즈베리파이3B+ 내부에 node.js를 이용하여 HTTP_Proxy를 개발하였다. HTTP_Proxy는 그림. 7.과 같이 HTTP요청을 수신하면 HTTP-URL를 파싱하여 CoAP-URL로 변환하여 센서네트워크의 CoAP서버에게 데이터를 요청한다. CoAP서버는 HTTP_Proxy로부터 요청을 받으면 해당하는 자원을 센싱하여 HTTP_Proxy에게 응답을 보내고, HTTP_Proxy는 응답을 웹페이지로 반환한다.

CoAP는 er-coap-engine(CoAP-18 draft기준)을 이용하여 CoAP서버를 구현하고 Cooja 시뮬레이터[19]를

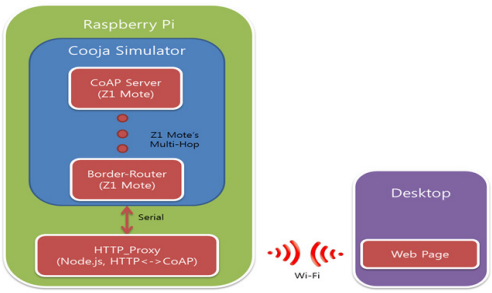


그림 6. CoAP 통신의 설계
Fig. 6. Design of CoAP Communication

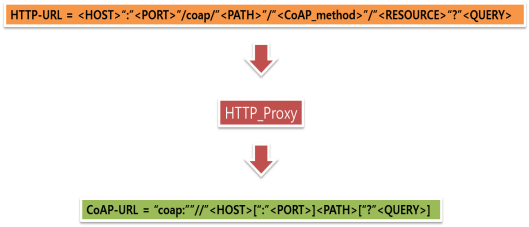


그림 7. HTTP-CoAP 메시지 변환
Fig. 7. HTTP-CoAP message conversion

통해 멀티홉으로 환경을 구축하였다. Cooja 시뮬레이터는 Border-Router를 통해 HTTP_Proxy와 시리얼로 연결되어 통신을 한다. 그림. 8은 HTTP-CoAP 연동 네트워크에서 통신절차를 나타낸다.

성능평가 인자로는 RTT를 사용 하였고 CoAP서버의 단일 홉, 두 홉, 세 홉 환경에서 아래와 같이 성능을 측정하였다.

1) 클라이언트의 웹페이지에서 센서노드의 자원을 HTTP URL로 형태로 요청
2) HTTP_Proxy에서 HTTP URL를 파싱하여 CoAP 메세지로 변환하고 CoAP서버의 자원을 요청
3) CoAP서버에서 요청받은 자원에 대한 응답을 HTTP_Proxy로 전송
4) HTTP_Proxy에서 클라이언트로 수신된 응답을 웹페이지에 반환
5) 1)~5)의 과정을 10,000회 반복
6) Wireshark를 활용하여 통신 패킷을 캡쳐
7) Wireshark의 Timestamp를 확인하여 RTT 계산

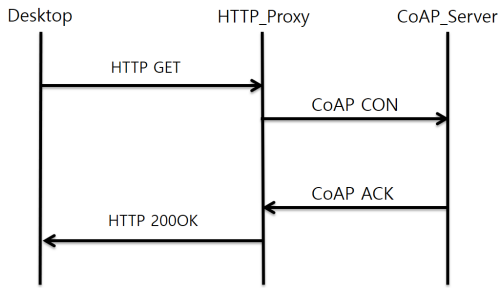


그림 8. HTTP-CoAP 통신 절차
Fig. 8. HTTP-CoAP Communication Procedure

Ⅳ. 성능 측정 결과

4.1 단일 홉의 성능 측정 결과

단일 홉 환경에서 10,000번 RTT를 측정하였다. CoAP의 최대 RTT는 8252.55ms, 최소 RTT는 170.08ms, 평균 RTT는 203.79ms이며, HTTP의 최대 RTT는 8256.17ms, 최소 RTT는 173.61ms, 평균 RTT는 208.10ms이다. 평균 RTT와 편차 가 큰 RTT가 간헐적으로 발생하였으며 최대 RTT는 평균 RTT와 약 40배 이상의 차이가 나는 것을 확인하였다.

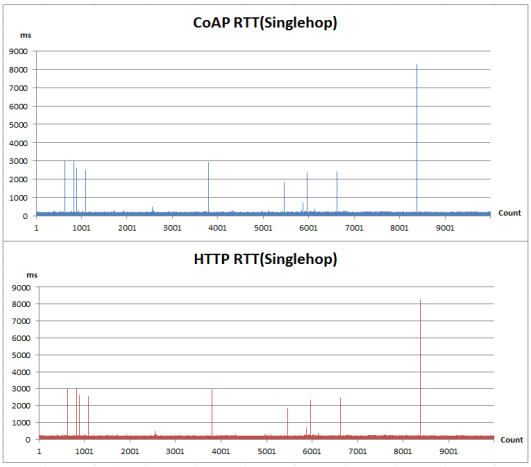


그림 9. 단일 홉 환경의 RTT 측정결과
Fig. 9. RTT measurement result of singlehop

4.2 두 홉의 성능 측정 결과

두 홉의 환경에서 10,000번 RTT를 측정하였다. CoAP의 최대 RTT는 3176.81ms, 최소 RTT는 263.08ms, 평균 RTT는 300.23ms이며, HTTP의 최대 RTT는 3180.66ms, 최소 RTT는 267.49ms, 평균 RTT는 304.66ms이다. 평균 RTT와 편차가 큰 RTT가 간헐적으로 발생하였으며 최대 RTT는 평균 RTT와 약 10배이상 차이가 나는 것을 확인하였다.

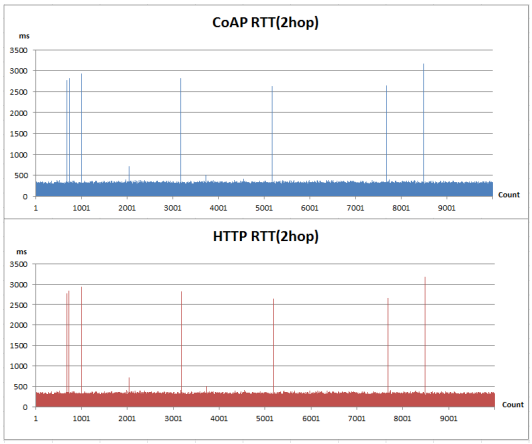


그림 10.두 홉 환경의 RTT 측정결과
Fig. 10. RTT measurement result of 2hop

4.3 세 홉의 성능 측정 결과

세 홉의 환경에서 10,000번 RTT를 측정하였다. CoAP의 최대 RTT는 6742.62ms, 최소 RTT는 328.17ms, 평균 RTT는 378.65ms이며, HTTP의 최대 RTT는 6745.57ms, 최소 RTT는 331.39ms, 평균

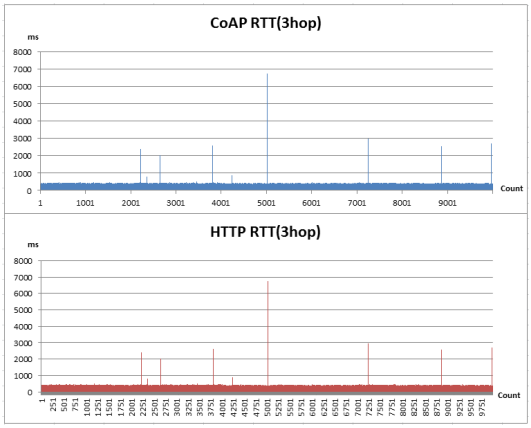


그림 11. 세 홉 환경의 RTT 측정결과
Fig. 11. RTT measurement result of 3hop

RTT는 383.11ms이다. 평균 RTT와 편차가 큰 RTT가 간헐적으로 발생하였으며 최대 RTT는 평균 RTT와 약 17배 이상의 차이가 나는 것을 확인하였다.

4.4 성능 측정 결과 분석

평균 RTT와 편차가 큰 RTT의 발생 원인을 확인하기 위하여 Wireshark를 통해 캡쳐한 패킷들을 분석하였다. 그림. 12.는 단일 홉의 최대 RTT의 패킷을 캡쳐한 내용이고, 그림. 13.은 패킷의 timestamp를 활용하여 각 패킷별로 소요되는 시간을 분석한 것이다. 최대 RTT에서 CoAP메시지의 재전송이 2번 발생하였으며, 통신지연의 대부분이 CoAP의 재전송 타이머인 RTO로 인해 발생하는 응답대기상태인 것을 알 수 있다.

No.	Time	Source	Destination	Protocol	Length	Info
198083	1960.635172	192.168.0.1	192.168.0.60	HTTP	130	GET /coap/Sensor_2/get/test/hello HTTP/1.1
198085	1960.638071	fd00::1	fd00::c30c:0:0:2	CoAP	83	CON, MID:51422, GET, TKN:73 f7 79 a5, /test/he
198092	1960.836624	fd00::c30c:0:0:2	fd00::1	CoAP	88	ACK, MID:51422, 2.05 Content, TKN:73 f7 79 a5,
198093	1960.837803	192.168.0.60	192.168.0.1	HTTP	225	HTTP/1.1 200 OK (text/html)
198094	1960.851891	192.168.0.1	192.168.0.60	HTTP	130	GET /coap/Sensor_2/get/test/hello HTTP/1.1
198096	1960.854066	fd00::1	fd00::c30c:0:0:2	CoAP	83	CON, MID:1019, GET, TKN:2c d7 aa 88, /test/he
198100	1963.536767	fd00::1	fd00::c30c:0:0:2	CoAP	83	CON, MID:1019, GET, TKN:2c d7 aa 88, /test/he
198103	1968.9057	fd00::1	fd00::c30c:0:0:2	CoAP	83	CON, MID:1019, GET, TKN:2c d7 aa 88, /test/he
198110	1969.106619	fd00::c30c:0:0:2	fd00::1	CoAP	88	ACK, MID:1019, 2.05 Content, TKN:2c d7 aa 88,
198111	1969.108058	192.168.0.60	192.168.0.1	HTTP	225	HTTP/1.1 200 OK (text/html)

그림 12. 단일 홉 환경의 최대 RTT 패킷분석 - 1
Fig. 12. Max RTT packet analysis of singlehop - 1

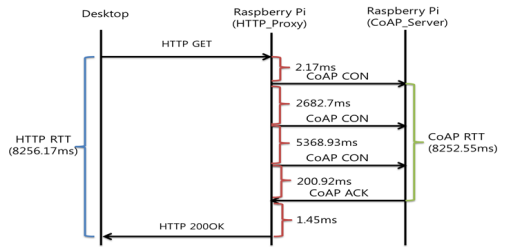
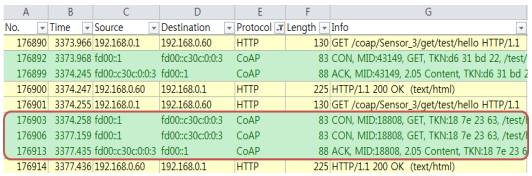


그림 13. 단일 홉 환경의 최대 RTT 패킷분석 - 2
Fig. 13. Max RTT packet analysis of singlehop - 2

그림. 14.는 두 홉의 최대 RTT의 패킷을 캡쳐한 내용이고, 그림. 15는 패킷의 timestamp를 활용하여 각 패킷별로 소요되는 시간을 분석한 것이다. 최대 RTT에서 CoAP메시지의 재전송이 1번 발생하였으며, 통신지연의 대부분이 CoAP의 재전송 타이머인 RTO로 인해 발생하는 응답대기상태인 것을 알 수 있다.

그림. 16.은 세 홉의 최대 RTT의 패킷을 캡쳐한 내용이고, 그림. 17은 패킷의 timestamp를 활용하여 각 패킷별로 소요되는 시간을 분석한 것이다. 최대 RTT에서 CoAP메시지의 재전송이 2번 발생하였으며, 통



No.	Time	Source	Destination	Protocol	Length	Info
176890	3373.966	192.168.0.1	192.168.0.60	HTTP	130	GET /coap/Sensor_3/get/test/hello HTTP/1.1
176892	3373.968	fd00::1	fd00:c30c:0:0:3	CoAP	83	CON, MID:43149, GET, TKN:d6 31 bd 22, /test/
176899	3374.245	fd00:c30c:0:0:3	fd00::1	CoAP	88	ACK, MID:43149, 2.05 Content, TKN:d6 31 bd 2
176900	3374.247	192.168.0.60	192.168.0.1	HTTP	225	HTTP/1.1 200 OK (text/html)
176901	3374.255	192.168.0.1	192.168.0.60	HTTP	130	GET /coap/Sensor_3/get/test/hello HTTP/1.1
176903	3374.258	fd00::1	fd00:c30c:0:0:3	CoAP	83	CON, MID:18808, GET, TKN:18 7e 23 63, /test/
176906	3377.159	fd00::1	fd00:c30c:0:0:3	CoAP	83	CON, MID:18808, GET, TKN:18 7e 23 63, /test/
176913	3377.435	fd00:c30c:0:0:3	fd00::1	CoAP	88	ACK, MID:18808, 2.05 Content, TKN:18 7e 23 6
176914	3377.436	192.168.0.60	192.168.0.1	HTTP	225	HTTP/1.1 200 OK (text/html)

그림 14. 두 홉 환경의 최대 RTT 패킷분석 - 1
Fig. 14. Max RTT packet analysis of 2hop - 1

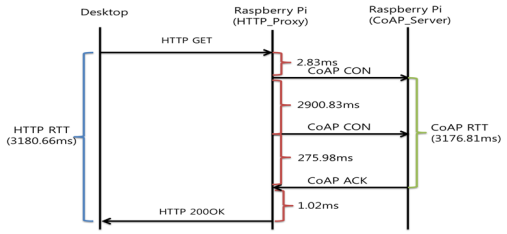
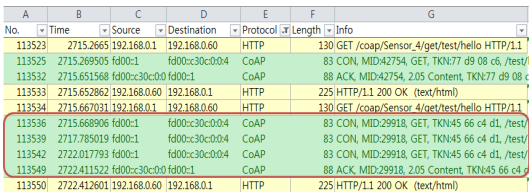


그림 15. 두 홉 환경의 최대 RTT 패킷분석 - 2
Fig. 15. Max RTT packet analysis of 2hop - 2



No.	Time	Source	Destination	Protocol	Length	Info
113523	2715.2665	192.168.0.1	192.168.0.60	HTTP	130	GET /coap/Sensor_4/get/test/hello HTTP/1.1
113525	2715.269505	fd00::1	fd00:c30c:0:0:4	CoAP	83	CON, MID:42754, GET, TKN:77 d9 08 c6, /test/
113532	2715.651568	fd00:c30c:0:0	fd00::1	CoAP	88	ACK, MID:42754, 2.05 Content, TKN:77 d9 08 c
113533	2715.652862	192.168.0.60	192.168.0.1	HTTP	225	HTTP/1.1 200 OK (text/html)
113534	2715.667031	192.168.0.1	192.168.0.60	HTTP	130	GET /coap/Sensor_4/get/test/hello HTTP/1.1
113536	2715.668906	fd00::1	fd00:c30c:0:0:4	CoAP	83	CON, MID:29918, GET, TKN:45 66 c4 d1, /test/
113539	2717.785019	fd00::1	fd00:c30c:0:0:4	CoAP	83	CON, MID:29918, GET, TKN:45 66 c4 d1, /test/
113542	2722.017793	fd00::1	fd00:c30c:0:0:4	CoAP	83	CON, MID:29918, GET, TKN:45 66 c4 d1, /test/
113549	2722.411522	fd00:c30c:0:0	fd00::1	CoAP	88	ACK, MID:29918, 2.05 Content, TKN:45 66 c4 d
113550	2722.412601	192.168.0.60	192.168.0.1	HTTP	225	HTTP/1.1 200 OK (text/html)

그림 16. 세 홉 환경의 최대 RTT 패킷분석 - 1
Fig. 16. Max RTT packet analysis of 3hop - 1

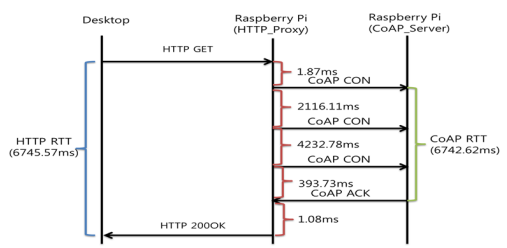


그림 17. 세 홉 환경의 최대 RTT 패킷분석 - 2
Fig. 17. Max RTT packet analysis of 3hop - 2

신지연의 대부분이 CoAP의 재전송 타이머인 RTO로 인해 발생하는 응답대기상태인 것을 알 수 있다.

Table 1.은 단일 홉, 두 홉, 세홉에서 측정된 최대 RTT 패킷을 분석한 내용이다. 최대 RTT에서 대부분의 시간이 CoAP에서 소요되고 있으며, CoAP 재전송 과정에서 RTO로 인해 발생하는 응답대기시간이 대부분의 통신지연을 차지하고 있다. CoAP의 RTO는 ACK_TIMEOUT과 ACK_RANDOMEFACTOR라는 파라미터에 의해 결정된다. CoAP에서 명시하고 있는 기본값은 ACK_TIMEOUT이 2초, ACK_RANDOMFACTOR가 1.5이다. 패킷 전송시 각 패킷마다 RTO는 ACK_TIMEOUT(2초)에서 ACK_TIMEOUT * ACK_RANDOMFACTOR(3초) 사이의 임의의 값으로 설정된다. CoAP의 평균 RTT는 단일 홉의 경우 203.79ms, 두 홉의 경우 300.23ms, 세홉의 경우 378.65ms로 측정되었다. 각 실험환경에서 측정된 평균 RTT와 CoAP표준에서 명시하고 있는 기본 RTO값은 차이가 많이 나고 있다. 따라서 평균 RTT와 편차가 큰 RTT는 CoAP 재전송과정에서 필요 이상의 시간을 응답대기시간으로 소요하고 있다는 것을 알 수 있다.

표 1. 최대 RTT 패킷 분석결과
Table 1. Result of Max RTT packet analysis

	MaxRTT (ms.)	HTTP (ms.)	CoAP (ms.)	number of Re-transmission
single hop	8256.17	3.62	8252.55	2
two hop	3180.66	3.85	3176.81	1
three hop	6745.57	2.95	6742.62	2

V. 결 론

본 논문에서는 WSN과 인터넷을 연결하는 CoAP 기반의 IoT 통신 시스템을 설계 및 개발하고, 테스트 환경을 구축하여 성능측정을 하였다. 6LoWPAN을 이용하여 센서 노드에 IPv6주소를 할당하고, CoAP를 통해 RESTful 서비스를 제공하여 웹페이지에서 센서 노드의 자원을 요청하여 확인할 수 있도록 구현하였다. 단일 홉, 두 홉, 세 홉의 환경에서 각각 10,000번의 RTT를 측정하여 성능측정을 하였다. 성능측정결과, 모든 테스트 환경에서 평균 RTT와 편차가 큰 RTT가 간헐적으로 발생하는 것을 확인하였다. 평균

RTT와 편차가 큰 RTT의 발생 원인을 분석하기 위해 Wireshark를 통해 패킷을 캡처하여 분석하였다. 그 결과, CoAP 재전송과정에서 RTO로 인해 발생하는 응답대기시간이 통신지연이 대부분을 차지하고 있음을 확인하였다.

CoAP표준에서 명시하고 있는 기본 RTO 값은 2~3초사이의 임의의 값이며, 이는 실험환경에서 측정한 평균 RTT와 큰 차이가 있다. 실험 결과 평균 RTT의 몇 배에 해당하는 시간을 응답대기상태로 소요하고 있으며, 이로 인해 평균 RTT와 편차가 큰 RTT가 간헐적으로 발생하고 있음을 확인하였다. 네트워크의 무선 환경에 따라 통ㅅ니 오류로 인행 CoAP 재전송은 불가피하다. 그러나 긴 통신 지연이 재전송과정에서 발생하므로 재전송 응답대기시간을 조절함으로써 전체 통신 지연을 줄일 수 있을 것으로 판단된다. 추후 본 연구 결과를 바탕으로 이를 최소화하기 위한 연구를 진행할 예정이다.

References

[1] M. Weiser, “The Computer for the 21st Century,” *Scientific Am.*, vol. 265, no. 3, pp. 94-105, 1991.

[2] K. Ashton, “That ‘internet of things,” *RFID J.*, vol. 22, no. 7, pp. 97-114, 2009.

[3] S. Yinbiao, et al., “*Internet of things: Wireless Sensor Netowkrs; IEC White Paper*,” IEC, Geneva, Switzerland, 2014.

[4] G. Montenegro, et al., “*RFC4944-Transmission of IPv6 Packet over IEEE 802.15.4 Networks*,” IETF, 2007.

[5] J. Hui and P. Thubert, “*Compression format for IPv6 datagrams over IEEE 802.15.4-based networks*,” No. RFC 6282, 2011.

[6] K. Seong, “IPv6 based internet of things technology trends,” *KISA*, Aug. 2014.

[7] Z. Shelby, “*RFC7252 - The Constrained Application Protocol(CoAP)*,” IETF, Jun. 2014.

[8] C. Bormann, A. P. Castellani, and Z. Shelby, “Coap: An application protocol for billions of tiny internet nodes,” *IEEE Internet Computing,* vol. 16, no. 2, pp. 62-67, 2012.

[9] I. Florea, et al., “Survey of standardized protocols for the internet of things,” *IEEE 21st Int. Conf. CSCS*, pp. 190-196, 2017.

[10] A. Banks and R. Gupta, “*MQTT Version 3.1.1.*” OASIS standard, 29: 89, 2014.

[11] D. Thangavel, et al., “Performance evaluation of MQTT and CoAP via a common middleware,” *2014 IEEE Ninth Int. Conf. ISSNIP*, pp. 1-6, 2014.

[12] J.-H. Jung, D.-K. Choi, and S.-J. Koh, “Distributed pub/sub model in CoAP-based Internet-of Things networks,” *2018 ICOIN,* 2018.

[13] R. Bhalerao, et al., “An analysis and improvement congestion control in the CoAP Internet-of-Things protocol,” *13th IEEE Annu. CCNC*, Las Vegas, NV, USA, 2016.

[14] Y. Y. Choo, et al., “Performance evaluation of CoAP-based Internet-of-Things system,” *J. Korea Multimedia Soc.,* vol. 19, no. 12, pp. 2014-2023, Dec. 2016.

[15] C. Bormann, et al., “RFC8323-CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets,” *IETF*, Feb. 2018.

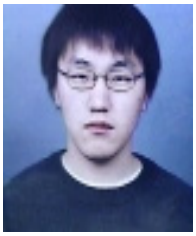
[16] J. Bauwens, et al., “Coexistence between IEEE802. 15.4 and IEEE802. 11 through cross-technology signaling,” *2017 IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS),* pp. 529-534, 2017.

[17] S. R. Pokhrel, et al., “Modeling compound TCP over WiFi for IoT,” *IEEE/ACM Trans. Networking (TON)*, vol. 26, no. 2, pp. 864-878, 2018.

[18] C. Gomez, A. Arcia-Moret, and J. Crowcroft, “TCP in the Internet of Things: from ostracism to prominence,” *IEEE Internet Computing*, vol. 22, no. 1, pp. 29-41, 2018.

[19] A. Velinov and A. Mileva, “Running and testing applications for contiki OS using cooja simulator,” in *Proc. Int. Conf. Inf. Technol. and Develop. Edu.,* pp. 279-285, 2016.

하 용 준 (Yong-jun Ha)



2012년 2월 : 동명대학교 컴퓨터공학과 졸업
2014년 2월 : 동명대학교 컴퓨터공학과 석사
2014년 3월~현재 : 동명대학교 컴퓨터공학과 박사과정

<관심분야> WSN, IoT, Routing, 저전력 통신

추 영 열 (Young-yeol Choo)

1986년 2월 : 서울대학교 제어계측 공학과 학사
1988년 8월 : 서울대학교 제어계측 공학과 석사
2002년 8월 : 포항공대 컴퓨터공학과 박사
2002년 9월~현재 : 동명대학교 컴퓨터공학과 교수
1988년 6월~2002년 8월 : posco 기술연구소 책임연구원
2005년 1월~6월 : 독일 Fraunhofer IESE visiting scientist
2015년 2월~2016년 1월 : 미국 Southern Illinois Univ. Edwardsville 방문교수.

<관심분야> 컴퓨터 네트워크, IoT, 네트워크 보안, 실시간 시스템