한국컴퓨터정보학회 동계학술대회 논문집 제23권 제1호 (2015. 1)

CoAP 프로토콜에서 재전송 카운트를 이용한 능동적 전송 기법

이정준°, 김경태*, 윤희용*
°*성균관대학교 정보 통신대학
e-mail: jungjune86@skku.edu°, kyungtaekim76@gmail.com*, youn7147@skku.edu*

Dynamic Retransmission Scheme Using Retry Count in Constrained Application Protocol (CoAP)

Jung June Lee°, Kyung Tae Kim*, Hee Yong Youn*
°*College of Information and Communication Engineering, Sungkyunkwan University

● 요 약 ●

최근 사물 인터넷(Internet of things) 의 발달에 따라, 스마트 디바이스 간의 네트워크 및 이를 구축할 수 있는 기술에 대한 수요가 급증하고 있다. 이러한 스마트 디바이스 간의 저 전력 저 손실 네트워크(Low power and Lossy network) 환경에서 쓰이는 대표적인 프로토콜로 CoAP(Constrained Application Protocol)가 있으며, 해당 프로토콜은 다양한 네트워크 환경에 유연하게 적용할 수 있도록 패킷 재전송 주기 설정 옵션을 가진다. 하지만 하나의 디바이스에서 네트워크 환경이 패킷 손실 및 지연 여부를 구분 할 수 없기 때문에, 네트워크 상태 파악을 위해서는 수신과 응답 양측 디바이스의 패킷 흐름을 확인해야 하는 문제점이 있다. 본 논문에서는 프로토콜의 정보를 기반으로 네트워크 상태를 파악 할 수 있는 새로운 필드 값을 적용하여 CoAP 패킷 재전송 주기를 네트워크 환경의 상태에 따라 동적으로 설정해주는 알고리즘을 제안한다. 제안된 기법은 동적으로 재전송 주기를 설정함으로써, 패킷 손실에 의한 서비스 장애 극복 및 패킷 지연 상황에서의 불필요한 패킷 재전송을 방지하여 에너지 효율성을 향상시키고 서비스 안정성을 보장한다.

키워드: CoAP(Constrained Application Protocol), 저 전력 저 손실 네트워크(Low power and Lossy Network), 동적 재전송(Dynamic retransmission)

I. 서 론

최근 사물 인터넷(Internet of things)의 발달에 따라, 기존의 PC 환경 네트워크가 아닌 스마트 디바이스 간의 네트워크 및 이를 구축할 수 있는 기술에 대한 수요가 급증하고 있다. 스마트 디바이스 간의 네트워크는 제한적인 자원을 가지며 패킷 유실이 PC간의 네트워크에 비해 자주 일어나는 특성이 있는데 이를 저 전력 저 손실 네트워크라 한다[1]. 이러한, 저 전력 저 손실 네트워크의 제한을 극복하기 위해 현재 IBM에서 제안한 MQTT(MQ Telemetry Transport) 프로토콜 [2], IETF 워킹그룹 ROLL(Routing Over Low Power and Lossy network)에서 발표한 RPL(IPv6 Routing Protocol for Low-Power and Lossy Networks)[3], IETF 워킹그룹 CORE(Constrained RESTful Environments)에서 발표한 CoAP[4] 등이 사용되고 있다. 특히, CoAP 는 다양한 네트워크 환경에 적용 할 수 있도록 패킷 재전송 주기 설정 옵션을 가지고 있다. 하지만 CoAP통신이 이루어지는 네트워크 상태 파악을 위해서는 송수신 장비 모두의 패킷 전송 상태를 확인해야 하는 문제점이 있다. 이러한 문제점을 해결하고자, 본 논문에서는 CoAP에 네트워크 환경을 파악할 수 있는 새로운 필드 값을 적용하고 기존의 ACK_TIMEOUT 필드를 이용하여 패킷 손실 및 지연이 일어나는 네트워크 환경에 효과적으로 대응할 수 있는 새로운 동적 패킷 재전송 타이머 설정 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구로써 기존의 CoAP 프로토콜에서의 통신 방법에 대해 서술한다. 3장은 능동적 재전송 타이머 기법에 대해 상세히 설명하며, 마지막으로 4장에서 결론 및 향후 과제에 대해 명시 한다.

II. 관련 연구

1. CoAP

IETF CoRE 워킹그룹의 핵심 프로토콜인 CoAP 는 드래프트 18에서 RFC 표준으로 채택되었다. CoAP는 제한적인 자원과 저 전력을 요구하는 장비간의 네트워크를 고려하여 제작된 프로토콜로써, RESTful [5] 개념을 채택하여 자원 효율성을 높였으며 어플리케이션과의 연동을 고려하여 기존 HTTP 프로토콜과 유사한 포맷을 가진다.

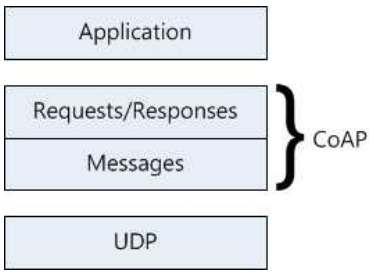

그림 1. CoAP 레이어
Fig. 1. Coap Layer

그림 1. 은 CoAP 의 논리적인 구조를 나타낸다. CoAP는 UDP와 같은 데이터 그램 방식의 전송 계층에서 비동기 통신으로 구현된다. UDP 패킷 구현 특성상 TCP 방식의 패킷 신뢰성을 전송 계층에서 보장받지 못하므로 프로토콜 자체에 신뢰성을 위한 재전송 및 타이머 관리 옵션을 가지고 있으며, RFC 규격 문서에서는 이러한 옵션의 값 을 고정하지 않고 사용자가 상황에 맞게 변경 할 수 있도록 하였다.

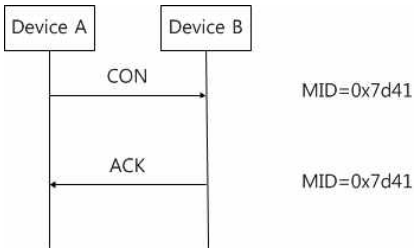

그림 2. 신뢰성 메시지 전송
Fig 2. Confirmable Message Transfer

그림 2. 는 CoAP 의 신뢰성 메시지 전달의 흐름을 나타낸다. 신뢰성 메시지 전달을 위해서 CON 필드를 사용하며, 메시지 아이디를 이용한 피기 백(piggy-back) 방식으로 패킷의 송수신이 정상적으로 이뤄졌음을 확인 한다.

네트워크에서 통신이 원활하게 이뤄지지 않는 케이스는 크게 패킷 손실과 패킷 지연이 있다. CoAP 는 이러한 네트워크 상황에 대응하기 위해 요청 패킷전송 후 프로토콜의 ACK_TIMEOUT 값으로 설정된 시간 이내에 응답 패킷을 수신하지 못하면, 재송신을 한다. 그림 3, 그림 4 와 그림 5는 각 상태별 CoAP 통신 흐름도를 나타낸다.

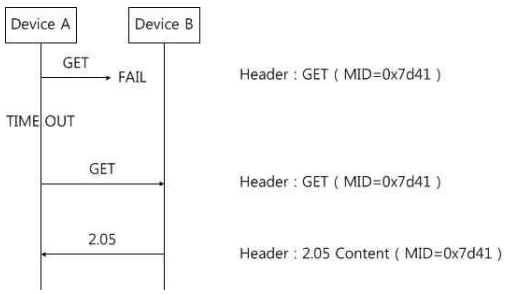

그림 3. 요청 패킷 손실 시 CoAP 통신
Fig. 3. Request Packet Loss Case

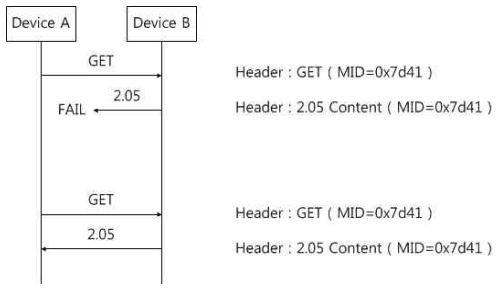

그림 4. 응답 패킷 손실 시 CoAP 통신
Fig. 4. Response Packet Loss Case

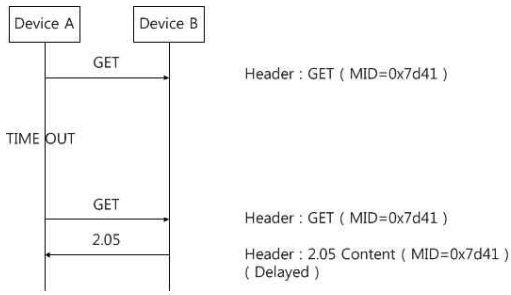

그림 5. 패킷 지연 시 CoAP 통신
Fig. 5. Packet Delay Case

그림 3,4,5 의 통신 과정은 패킷 손실과 패킷 지연 상태 모두 패킷 재전송으로 대처함을 나타내는데, 이 때문에 타이머를 길게 설정 할 경우 패킷 손실 시 재전송 시간이 느려서 통신이 원활하지 못하고, 타이머를 짧게 설정 할 경우 패킷 지연 시 불필요한 재전송 횟수가 늘어나는 단점이 있다.

III. 본 론

CoAP는 통신의 신뢰성을 위해 요청 패킷에 피기백 방식으로 응답 하고, 패킷 손실과 지연에 대처하기 위해 타임아웃 발생 시 동일한 패킷을 재전송 한다. 하지만 하나의 타이머 값을 사용하여, 타이머를 길게 설정 할 경우 패킷 손실 시 재전송 시간이 느려서 통신이 원활하지 못하고, 타이머를 짧게 설정 할 경우 패킷 지연 시 불필요한 재전송 횟수가 늘어나는 단점이 있다. 본 논문에서는 효율적인 패킷 재전송을 위해 재전송 원인을 파악할 수 있도록 CoAP 필드에 ReqRetry 옵션을 추가하고, 이를 이용한 동적 패킷 재전송 타이머 설정 알고리즘을 제안한다. 재전송의 원인이 되는 손실과 지연 네트워크는 패킷의 손실 여부를 기준으로 구분된다. 패킷의 송수신이 최초의 요청에 대한 응답으로 이루어 질 경우 해당 패킷은 손실되지 않았다고 판단 할 수 있다. 이러한 송수신의 확인을 위해 CoAP 피기백 방식의 특성을 활용 하였다. 피기백 방식은 응답 패킷을 생성할 때 요청 패킷의 정보를 복사하여 사용한다. 따라서 요청패킷에 재전송 정보를 포함시킬 경우, 응답 패킷이 최초의 요청에 의한 응답인 지 확인 할 수 있다. 추가하고자 하는 재전송 카운트 옵션과 기존 패킷의 호환성을 위해, 표1과 같이 기존 RFC문서에 정의한 옵션들과

동일한 포맷을 이용한다.

표 1. ReqRetry 옵션 포맷
Table 1. ReqRetry Option Format

옵션 번호	이름	포맷	길이	기본 값
40	ReqRetry	uint	0-2 byte	0

옵션 번호는 프로토콜에서 특정 옵션 사용 시 어떠한 옵션인지를 구분하기 위한 구분자로, 기존에 존재하는 옵션 번호와 충돌하지 않는 40번을 선택한다. 옵션 이름은 ReqRetry 로 정하며, 현재 패킷이 몇 번째 재전송 시도에 의해 발송된 패킷인지를 의미한다. 포맷은 옵션의 값 필드에 기록되는 데이터의 타입인 unsigned int 형을 택하였으며, 길이는 0-2 바이트로 제한한다. 기본 값은 0이며 이는 최초 패킷 전송 시 재전송 시도가 없기 때문에 0임을 의미 하며, 타임아웃에 의한 재전송 발생 시 1씩 값을 늘린다. 이 옵션을 사용한 요청 패킷을 수신한 장비는, 응답 패킷에 피기백 방식으로 자신이 받은 ReqRetry값을 복사하여 응답 한다. ReqRetry가 적용된 CoAP 통신 흐름도는 그림 6,7,8 와 같다.

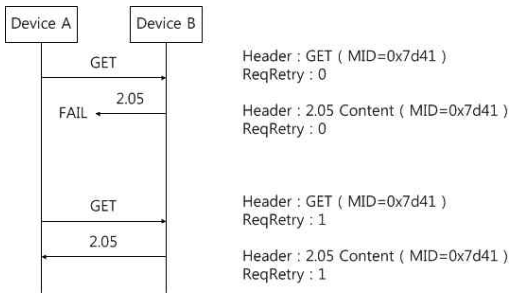

그림 6. 응답 패킷 손실 시 ReqRetry 가 추가된 CoAP 통신
Fig. 6. Request Packet Loss with ReqRetry Case

그림 6은 응답 패킷 손실로 인해 타임아웃이 발생하고, 타임아웃에 의한 재전송을 반영한 ReqRetry 필드 값 을 나타낸다. 요청 측은 응답 패킷의 ReqRetry 값이 0 이 아닌 경우 패킷이 손실되었다고 판단 할 수 있다.

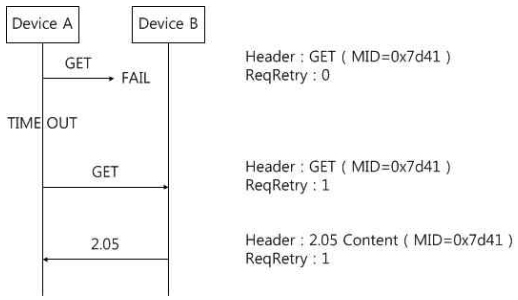

그림 7. 요청 패킷 손실 시 ReqRetry 가 적용된 CoAP 통신
Fig. 7. Response Packet Loss with ReqRetry Case

그림 7은 요청 패킷 손실로 인해 타임아웃이 발생한 상황이며 그림 6과 마찬가지로 응답 패킷의 ReqRetry 값이 0 이 아닌 경우 패킷이 손실되었다고 판단 할 수 있다.

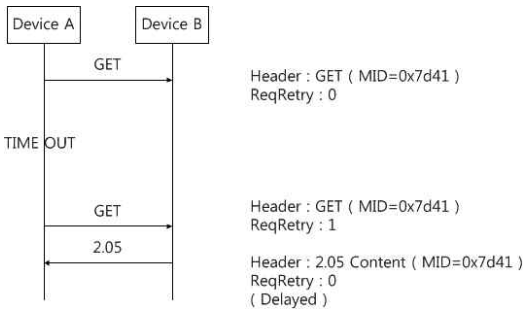

그림 8. 패킷 지연 시 ReqRetry 가 적용된 CoAP 통신
Fig. 8. Packet Delay with ReqRetry Case

그림 8의 패킷 지연 상황에 의해 ReqRetry 값이 0 인 요청의 응답이 늦게 도착하는 상황이다. 요청 측은 응답 패킷의 ReqRetry 값이 0 임을 확인하여 해당 패킷이 최초 요청에 의한 응답임을 알 수 있으며, 이를 근거로 네트워크 상태에 의한 패킷 손실은 없다고 판단 할 수 있다.

ReqRetry 값에 의해 프로토콜의 정보만으로 네트워크 상태 파악이 가능해지므로, ReqRetry 값을 이용한 능동적인 재전송 타이머 설정 알고리즘을 작성 할 수 있다. 패킷 손실이 잦은 네트워크의 경우 패킷 재전송 횟수를 늘려서 서비스 안정성을 확보해야 하므로 ACK_TIMEOUT 값을 작게 설정하여야 하고, 패킷 지연이 있는 네트워크 환경에서는 패킷 응답을 기다리기만 하면 재전송을 할 필요는 없으므로 ACK_TIMEOUT 값을 크게 설정해야 한다. 또한 ACK_TIMEOUT의 값이 너무 작으면 서비스에 지장이 생기므로, 최솟값 은 IETF 의 재전송 타임아웃 가이드라인[6] 에 따라 1초로 제한한다. 네트워크 상황 별 ReqRetry 값과 이에 따른 효율적인 타이머 설정 알고리즘은 알고리즘 1 과 같다.

알고리즘 1. ReqRetry를 이용한 능동적 ACK_TIMEOUT 설정
Algorithm 1. Dynamic ACK_TIMEOUT Setting with ReqRetry

```
repeat
  패킷 요청 및 응답
  If(타임아웃 발생)
    If(ReqRetry == 0)
      ACK_TIMEOUT = ACK_TIMEOUT +1;
    If(ReqRetry != 0)
      If(ACK_TIMEOUT 〉= 2)
        ACK_TIMEOUT = ACK_TIMEOUT -1;
until;
```

위와 같은 알고리즘을 사용하여 타이머 값을 결정할 경우 패킷 손실 상황에서는 재전송 주기가 짧아져서 패킷 손실에 의한 서비스 장애를 극복 할 수 있고, 패킷 지연 상황에서는 응답 패킷 대기 시간이 길어져서 불필요한 패킷 재전송을 막음으로써 에너지 효율을 늘릴 수 있으므로, 이를 통해 서비스 안정성 및 장비의 에너지 효율 향상을 기대한다.

IV. 결 론

저 전력 저 손실 네트워크의 제한을 극복하기 위해 기존에 제안된 CoAP의 패킷 재전송 타이머는 네트워크 상황을 파악 할 수 없어서 패킷 손실 상황이나 패킷 지연 상황에 유동적으로 대처 못하는 단점이 있다. 본 논문에서는 이러한 문제점을 해결하기 위해 CoAP에 새로운 필드를 추가하여 응답 패킷 정보를 통해 네트워크 상황을 파악 할 수 있도록 하였고, 이를 이용한 효율적인 재전송 타이머 알고리즘을 제안함으로써 패킷 손실과 지연에 능동적으로 대처 할 수 있는 새로운 알고리즘을 제안하였고, 이를 통해 서비스 안정성 및 장비의 에너지 효율을 향상하였다.

향후 연구 목표로는 알고리즘에 의한 정책 결정을 보다 효율적으로 하기 위해 Markov Decision Process를 이용한 수학적 모델링을 진행 할 예정이다.

ACKNOWLEDGMENT

본 연구는 BK21+사업, 한국연구재단 기초연구사업 (2012 R1A1A2040257), (2013R1A1A2060398), 삼성전자(S-2014-07 00-000), 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신·방송 연구개발사업 (1391105003)의 일환으로 수행하였음.

참고문헌

[1] V. Cackovic and Ž. Popovic “Management in M2M networks”, Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention, pp. 501-506 (2014)

[2] Andy Stanford-Clark and Hong Linh Truong "MQTT For Sensor Networks (MQTT-SN) Protocol Specification Version 1.2" International Business Machines Corporation (2014)

[3] T. Winter (Ed.), P. Thubert (Ed.), and the ROLL Team. "RPL:IPv6 Routing Protocol for Low power and Lossy Networks." Internet Engineering Task Force RFC 6550. (2012)

[4] Z. Shelby, B. Frank, and D. Sturek "The Constrained Application Protocol (CoAP)" Internet Engineering Task Force RFC 7252 (2014)

[5] Fielding, R.T and Taylor, R.N. "Principled design of the modern Web architecture" Software Engineering, 2000. Proceedings of the 2000 International Conference pp.407-416 (2000)

[6] M. Allman. “Retransmission Timeout Considerations” Internet Engineering Task Force draft-allman-tcpm-rto-consider-01 (2012)