한국컴퓨터정보학회 동계학술대회 논문집 제23권 제1호 (2015. 1)

사물인터넷을 위한 CoAP 및 HTTP 연동 방안

정동영°, 이병준*, 윤희용*
°*성균관대학교 정보통신대학
e-mail:{jungdy, byungjun, youn7147}@skku.edu*°

Interlocking of CoAP and HTTP for IoT

Dong Young Jung°, Byung Jun Lee*, Hee Yong Youn*
°*School of Information and Communication Engineering, Sungkyunkwan University

● 요 약 ●

CoAP은 IoT(Internet of Things, 사물인터넷)를 위해 설계된 HTTP 메카니즘을 기반의 응용 계층 프로토콜로 M2M 통신을 위해 만들어 졌다. 이로 인해 WSN과 같은 제한적인 환경에서의 전송과 프로세싱을 효율적으로 처리 할 수 있고, CoAP과 HTTP사이의 맵핑 또한 가능하다. 본 논문에서는 HTTP 클라이언트가 CoAP 서버 혹은 CoAP 클라이언트가 HTTP 서버에게 직접적인 연결을 제공하는 방안을 제시한다. 또한, 전력 수급에 제한이 있어 데이터 전송률을 최소화 할 필요가 있기 때문에 캐싱을 사용한다. 제안하는 방안의 성능을 수학적 모델링을 통해 평가, 분석한다.

키워드: CoAP, HTTP, 사물인터넷(internet of things), 무선 센서 네트워크(wireless sensor networks)

I. 서 론

사물인터넷(IoT; Internet of Things)은 유비쿼터스 센서 네트워크(USN: Ubiquitous Sensor Netwo가)나 M2M에서 발전하여 시간과 장소의 제약 없이 사물이 인터넷에 연결되는 환경을 의미한다. 또한 사물인터넷은 사람뿐만 아니라 사물들이 통신을 통해 정보를 주고받고, 이를 활용한 다양한 서비스가 가능하다. 이를 통해 동적으로 수많은 사물이 네트워크에 연결되는 확장성(scalability), 사물인터넷에 구성하는 네트워크 프로토콜, 데이터 형식, 서비스 호출 방식이 서로 다른 이질성(heterogeneity)에 대한 상호운용성(interoperability), 무선 환경 사물의 이동성과 실시간 통신 및 신뢰성 기반의 인식&적응성(awareness & adaptability)등의 특징을 지원해야 한다. 사물 인터넷을 가능하게 하는 중요한 요소 중 하나는 6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks protocol), 이 프로토콜은 저전력 무선 표준인 IEEE 802.15.4를 위해 설계되었다. 본 논문에서는 [1]의 언급을 따라 이러한 네트워크를 '제한된 네트워크'라고 정의하고, 이 네트워크에 연결된 기기들은 제한된 네트워크의 일부가 되며 '제한기기'라고 명명한다. 제한된 네트워크에서의 통신은 이더넷 네트워크와 비교하면 높은 패킷 손실률(5-10%)과 낮은 출력, 복잡한 네트워크 구조를 가지는 것이 특징이다. 예를 들어, 센서 노드들은 아주 적은 램이나 롬만을 가지고 있고 배터리로 구동되는데, 긴 배터리 구동시간을 유지하기 위하여 전파사용과 패킷 사이즈를 최소화해야 한다.

HTTP는 가장 많이 쓰이는 응용 계층 프로토콜 중 하나이다. 제한된 네트워크에서 HTTP를 대신할 프로토콜을 찾기 위해 IETF는 CoRE(Constrained RESTful Environment) 워킹 그룹을 설립했고, 이 그룹은 CoAP(Constrained Application Protocol)을 개발했다. CoAP은 HTTP의 대안책으로써, 제한된 네트워크에서의 구동을 목적으로 한다. CoAP의 한 가지 주요 이점은 처음부터 HTTP로의 맵핑이 고려됐다는 것이다. 물론 두 프로토콜을 정확하게 변환하기 위해서는 추가적인 매개체가 필요한데, 이 매개체를 '변환 프록시'라고 한다. 프록시는 CoAP와 HTTP 사이의 맵핑을 구현하고, 같은 리소스의 불필요한 전송을 피하기 위해 리소스의 캐싱을 사용한다. 이 논문에서는 CoAP-HTTP 프록시 구현을 제안한다.

II. 관련 연구

CoAP은 비교적 최신의 프로토콜이기 때문에 아직 많은 구현과 평가가 나와 있지는 않다. CoAP-HTTP 프록시 중 한 가지를 구현한 [2]에서는 CoAP 끝점(endpoint)에서부터 리소스 요청을 하기 위해 HTTP 클라이언트를 허락하는 단방향 프록시를 제안한다. 이것은 WSN에서 캐싱이 고려되지 않은 서버방향 프록시이다.

CoAP 구현의 개요와 제한된 네트워크에서의 사용을 다룬다. CoAP은 제한된 네트워크에서 아주 유망한 통신 프로토콜이기 때문이다. CoAP의 초기 버전인 드래프트 3 버전에서, 리소스 제한적인 기기에서의 사용을 위한 'libcoap'으로 명명된 [3]에서 제시한 구현이 있다. Libcoap은 TinyOS와 Contiki같은 운영체제에서 사용 가능한데, 120B에서 230B까지의 전송 범위에서 70-400ms(single hop)의 완전한 요청이 수행됨을 보여준다.

[4]에서 Contiki를 위한 좀 더 최신의 드래프트 7 버전으로 CoAP이

구현되었다. [3]과는 반대로 저자는 MAC 계층에서의 multi-hop WSN 사용률을 평가했으며, 반응 시간은 hop의 숫자와 사용률 매개변수에 크게 영향을 받았다.

리소스 제한적인 CoAP에서의 H2M(Human to Machine) 시스템을 위해서는 파이어폭스 CoAP 플러그인인 'Copper'가 [4]에서 제안되었다. Copper는 사용자가 직접 CoAP 요청을 하게 해준다. 파이어폭스 자체는 HTTP 클라이언트지만 Copper 구현은 순수한 CoAP이며 HTTP 변환이나 맵핑을 별도로 요구하지 않는다. 하지만 플러그인이라는 근본적 한계 때문에 범용적으로 사용할 수 없다는 것이 문제점이다. 정리하자면, CoAP은 헤더의 오버헤드가 기존 프로토콜과 달리 작아서 제한된 기기에서 사용하기 알맞은 새로운 통신 프로토콜이다.

III. 본 론

기존의 웹 프로토콜은 사람-사물(H2M)간 소통이 주를 이루지만, 웹의 발전으로 사물-사물(M2M) 간 소통이 점점 더 중요해지고 있다. 특히 사물웹, 사물인터넷에서는 M2M 애플리케이션이 많이 사용되므로 적절한 웹 서비스를 찾는 기술이 굉장히 중요하다. 이러한 이유로 RESTful 웹 서비스와 같은 제한된 네트워크 사용에 중점을 둔 기술들이 조명 받고 있다.

RESTful 웹 서비스에 기반하는 CoAP은 HTTP의 대안으로 개발된 프로토콜이다. TCP의 큰 오버헤드 때문에 CoAP은 UDP를 전송 프로토콜로 사용한다. TCP와 달리 UDP는 전송의 신뢰성을 보장하지 않는다. 이러한 문제를 해결하기 위해 CoAP은 추가적인 메시지 계층을 사용한다. 이 계층은 네 가지 패킷 타입 Confirmable(CON), Nonconfirmable(NON), Acknowledgement(ACK), Reset(RST)로 정의된다. 서버가 새 패킷을 받지 못하면 RST 연결로 설정하고 만약 클라이언트가 정해진 시간동안 아무런 응답도 받지 못하면 메시지 재전송 한다. 하지만 메시지의 신뢰성이 지나치게 보장되면 이로 인한 오버헤드가 발생한다. CoAP은 이러한 문제점을 보완하기 위해 신뢰도를 선택적으로 활용할 수 있게 하였다. 신뢰성이 보장되지 않아도 되는 경우에는 클라이언트는 CON 대신 NON 패킷으로 연결을 만들고 서버도 또한 같은 종류의 패킷을 보내준다.

메시지 계층의 위에 있는 CoAP 요청/전송 계층은 PUT, GET, POST, DELETE의 네 가지 동작을 제공한다. HTTP와는 달리 CoAP은 바이너리 헤더표현을 사용하기에 CoAP 헤더는 오직 4바이트만을 필요로 한다. 또한 CoAP 끝점의 주소를 사용하기 위해 URI가 사용되며 최대한 패킷사이즈를 줄이기 위해 URI도 짧게 쓰인다. URI의 모든 부분은 파싱 복잡도를 줄이기 위해 각각의 헤더 옵션을 가진다. CoAP은 M2M 애플리케이션에서의 사용을 전제하므로 HTTP에서는 지원하지 않는 비신뢰성 멀티캐스트, 비동기식 전송방식[6], 그리고 CoAP 서버의 모든 호스트 리소스 리스트 기능을 제공한다.

그림 1에서 라우터는 6LoWPAN을 사용해 IPv6에 연결된다. 프록시는 애플리케이션 데이터나 논리, 애플리케이션 자체가 바뀌어도 프록시는 독립적인 역할을 한다는 것이 장점이다. 하지만 적어도 서버-프록시 또는 클라이언트-프록시 둘 중 하나에 프록시 상황에 대한 정보가 존재해야 한다. 여기서 프록시는 두 가지 타입으로 나눌 수 있다.

포워드 프록시: 서버로의 직접적인 연결을 만드는 대신 클라이언트는 프록시로 요청을 먼저 보낸다.그 후에 프록시는 서버로 요청을 포워딩한다. 서버는 이 프록시를 클라이언트가 행동하는 것으로 인식한다. 결과적으로 클라이언트는 이 프록시를 사용하는 방법에 대한 조정이 필요하고 서버는 그럴 필요가 없다.

리버스 프록시: 리버스 프록시는 클라이언트에게 알려져 있지 않다. 클라이언트는 일반적인 서버로의 전송을 기대하는데, 이 프록시는 그 요청을 숨겨진 서버로 포워딩한다. 따라서 그 서버는 프록시와 밀접하게 연관되어 동작해야 한다. 전자와는 반대로, 클라이언트는 프록시의 존재를 몰라도 되고 프록시에 대한 어떠한 조정도 필요치 않다.

요약하자면, 프록시는 인터넷 끝점과 제한된 기기 사이의 직접적 웹 서비스 기반의 통신을 하기 때문에 사물웹, 사물인터넷에서 중요한 역할을 맡고 있다.

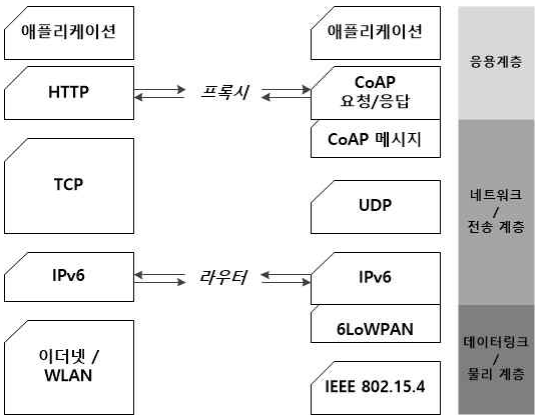

그림 1. 제한된 네트워크와 일반적인 웹 서비스 통신 스택
Fig. 1. Constrained Network and General Web Service Protocol Stack

마지막으로, 프록시의 동작을 측정하기 위해 맵핑과 캐시 기능을 테스트했다. MSP430 마이크로컨트롤러와 IEEE 802.15.4를 지원하는 TelosB를 사용하고 CoAP 클라이언트로는 파이어폭스의 플러그인인 Copper를 사용했다. HTTP 클라이언트로는 파이어폭스를 사용했으며 HTTP 서버로는 아파치 웹 서버를 사용했다. 모든 기기는 같은 이더넷 랜으로 연결되고 무선 센서는 6LoWPAN 엣지 라우터에 연결하였다.

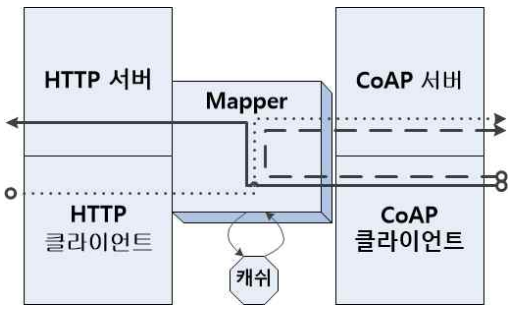

그림 2. 모듈화 된 프록시에서의 요청 흐름도
Fig. 2. Modularized Request flows at the Proxy

그림 2에서 표현한 프록시는 jCoAP 라이브러리의 일부이다. 이는 자바로 구현되어 자바 가상 머신을 사용가능한 어느 기기에서나 실행가능하다. 통신 방법은 일반적인 HTTP-HTTP와 HTTP-CoAP, CoAP-HTTP 그리고 CoAP-CoAP 4가지로 표현할 수 있다. HTTP와 CoAP과 같이 서로 다른 프로토콜 사이에도 프록시는 적절히 맵핑을 해주어 통신을 가능하게 해 준다. HTTP-CoAP 방식에서는 HTTP 클라이언트가 리소스를 요청하면 파이어폭스의 프록시 주소를 변환 프록시의 주소로 바꿔준다. 파이어폭스는 CoAP에 대해 어떠한 정보가 없어도 CoAP 서버로부터 직접 정보를 제공받게 된다. CoAP-HTTP 방식은 CoAP 클라이언트가 HTTP 서버로 요청을 보낸다. 이 요청은 프록시로 직접 보내져 수행 된다. CoAP-CoAP에서는 클라이언트에서 서버로의 요청이 보내지면 제한된 기기의 주소로 프록시가 별다른 맵핑 없이 수행이 된다. HTTP-HTTP는 일반적인 방법이므로 특별히 언급하지는 않는다.

캐싱은 리소스의 불필요한 전송을 피하며 네트워크 트래픽을 감소시킬 수 있다. 캐싱 효율의 평가를 위해 다음의 시나리오를 가정한다. x개의 클라이언트가 프록시를 사용해서 $t_r = 1/f_r$의 간격으로 빈번하게 리소스를 요청하고 리소스는 t_c의 시간만큼 프록시에 의해 캐싱된다. 가장 최악의 경우에는 프록시가 제한된 기기에게 $f_c = 1/t_c$의 빈도수로 요청을 하게 될 것이다. t라는 임의의 소요시간에 프록시는 제한된 기기에게 $t \cdot f_c$개의 요청을 하게 된다. 그러므로 프록시가 받는 요청의 수는 $x \cdot f_r \cdot t$로 구할 수 있다. 결과적으로 요청을 받았을 때, 캐시 안에 있는 그 요청의 수는 $x \cdot f_r \cdot t - t \cdot f_c$ 와 같이 두 값의 차로 유추할 수 있다. 캐시로 제공된 요청의 수에 대해 프록시 없이 이론적인 수로 요청된 캐싱율 r의 식은 다음과 같이 수식 (1)과 수식 (2)로 표현할 수 있다.

$$r = \frac{\text{캐시에 저장된 요청의 수}}{\text{총 요청의 수}} \quad (1)$$

$$r = \frac{x \cdot f_r - f_c}{x \cdot f_r} \quad (2)$$

캐싱율 r은 클라이언트에서 서버로의 요청이 얼마나 줄어들 수 있는지를 나타낸다. 캐싱율 0%는 캐시 내에 해당 요청이 없다는 것을 뜻하고, 100%는 모든 요청들이 캐시로부터 제공된다는 것을 뜻한다. 주어진 시간에서 클라이언트의 수 x가 증가할 때, 캐싱 타임 t_c가 증가할 때, f_r이 증가할 때 캐싱율이 증가한다.

표 1. 고정단위 시간동안 클라이언트(x)의 수에 따른 캐싱율
Table 1. Cashing Rate as a number of clients in a Fixed Time

	r의 이론 값	r의 실제 값
$x = 1$	50%	66%
$x = 2$	75%	79.1%
$x = 5$	90%	90.4%
$x = 10$	95%	94.8%

표 1은 일정하게 고정된 시간에 클라이언트의 수에 따른 캐싱율을 평가한 결과를 보여준다. 처음 세 가지 결과 값은 기대치보다 실제 수행한 것이 조금 더 높다. 이는 프록시가 리소스 캐싱에 충분한 시간을 가지기 때문에 추정한 값보다 실제 결과가 더 잘 나온 것이다.

IV. 결 론

HTTP 클라이언트가 CoAP 리소스에게 직접적인 연결을 제공하고 그 역도 가능한 CoAP-HTTP 프록시중 한 가지를 보았다. 이는 CoAP 프로토콜 상세에 의해 정의된 CoAP-HTTP 맵핑의 개념을 증명한 것이라고도 볼 수 있다. 프록시는 사물웹, 사물인터넷의 구현에 있어서 다른 애플리케이션 게이트웨이에 반해 독립적인 구성을 가지므로 유연성을 가진다. 프록시는 오픈소스 기술이므로 근시일내에 더 많은 발전과 평가가 있을 것이다.

앞으로는 프로토콜 통신 간 보안문제가 더욱 화두로 떠오를 것이다. 보안 헤더를 내부에 추가로 구현 한다면 제한된 기기들로 더 많은 리소스를 필요로 하게 될 것이다. 따라서 보안 프록시를 기기와 가까운 네트워크 상에 넣거나 하는 방법으로 개선이 될 것이다. 또, HTTPS 지원으로 제한된 네트워크 외 일반 네트워크에서도 암호화 등을 사용 가능하게 해야 하고 오직 승인된 사용자만이 접근 가능하게 발전해야 할 것이다.

ACKNOWLEDGEMENT

본 연구는 BK21+사업, 한국연구재단 기초연구사업 (2012R1A1A2040257), (2013R1A1A2060398), 삼성전자(S-2014-0700-000), 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신·방송 연구개발사업 (1391105003)의 일환으로 수행하였음.

참고문헌

[1] Z. Shelby, “Embedded web services,” Wireless Communications, IEEE, vol. 17, no. 6, pp. 52–57, December 2010.

[2] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota, “REST Enabled Wireless Sensor Networks for Seamless Integration with Web Applications,” in Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on, pp. 867 –872. October 2011

[3] K. Kuladinithi, O. Bergmann, T. Poetsch, M. Becker, and C. Goerg, “Implementation of CoAP and its Application in Transport Logistics,” in ”Extending the Internet to Low power and Lossy Networks (IP+SN 2011)”, Chicago, USA, 2011.

[4] M. Kovatsch, S. Duquennoy, and A. Dunkels, “A Low-power CoAP for Contiki,” in Proceedings of the IEEE Workshop on Internet of Things Technology and Architectures, Valencia, Spain, October 2011. [Online]. Available:

http://www.sics.se/adam/kovatsch11low-power.pdf

[5] C. Bormann and Z. Shelby, "Blockwise transfers in CoAP," IETF, Tech. Rep., February 2012, draft. [Online]. Available: *http://www.ietf.org/id/draft-ietf-core-block-08.txt*

[6] K. Hartke, "Observing Resources in CoAP," IETF, Tech. Rep., March 2012, draft. [Online]. Available: *http://www.ietf.org/id/draft-ietf-core-observe-05.txt*