Zero Configuration HTTP-CoAP Proxy Implementation based on CGI

Jongsoo Jeong[1], Jeehoon Lee[12], Haeyong Kim[1], Gyusang Shin[1], and Seon-Tae Kim[1]

[1] Embedded Software Research Department, Electronics Telecommunications Research Institute, Daejeon, Korea
[2] Department of Computer Software and Engineering, University of Science and Technology, Daejeon, Korea
{jsjeong,zardu22,haekim,gsshin,stkim10}@etri.re.kr

Abstract. CoAP is a web protocol for constrained environments, and also designed in consideration of interoperability with HTTP. However, lack of best practices of implementing proxies that provide HTTP-CoAP translation obstructs verification and spread of new Web of Things services based on interactions between HTTP and CoAP devices. In this paper, we introduce a HTTP-CoAP proxy implementation. It does not require any configurations on clients. It also can be easily added on top of conventional HTTP servers by utilizing CGI.

Keywords: Web of Things, HTTP, CoAP, proxy, CGI, REST

1 Introduction

The RESTful architecture is preferable to the Web of Things on constrained environments due to its low complexity, and loose-coupling stateless interactions [1]. However, HTTP over TCP that is a well known REST protocols set is not suitable to resource constrained devices that is based on 8, 16bit MCUs with only 10 KB RAM and 200 KB ROM, and lossy links such as IEEE 802.15.4 wireless. Because they were designed for PC-class computers which have relatively more resources, and reliable wired communication links such as Ethernet. In order to solve this problem, IETF CoRE Working Group suggested Constrained Application Protocol (CoAP) [2], a lightweight web protocol for constrained environment.

CoAP is designed in consideration of interoperability with other clients based on HTTP that is a de facto RESTful protocol. According to the current CoAP draft, HTTP devices can communicate with CoAP end points via an intermediary proxy (HC Proxy) that can translate from HTTP to CoAP and vice versa. To realize interoperations, however, there is still an obstacle: the clients should be aware of the existence of the proxy. If the current clients' environment is required to be modified to access CoAP resources, spreading various Web of Things pilot services might be hindered.

Pervasive 2012 Demos
Newcastle, UK, 18-22 June, 2012

In this paper, we propose a zero configuration HTTP-CoAP proxy implementation methodology to communicate with CoAP servers for HTTP clients without any modifications on each side. Our method is extending the existing HTTP server by adding a CGI that can role as a HC Proxy, so that it makes implementing HC Proxy simple. The reminder of the paper is organized as follows. In Section 2, we briefly introduce proxy models for HTTP-CoAP translation. In Section 3, our implementation is described. Section 4 concludes this work.

2 Proxy for HTTP-CoAP Translation

As we mentioned previously, the purpose of this paper is to make a proxy that can support communications with CoAP servers without any modifications on end points. In order to achieve it, followings are required.

a) There should be no configuration at the client side. The clients should be able to access CoAP resources without any knowledge about '`coap://`' scheme.
b) Implementing the HC Proxy should be simple.

Both servers and clients require different configurations according to how it provides proxy services. In the CoRE working group, a document that covers them in the view point of the proxy for HTTP-CoAP mapping, and suggests best practices was submitted [3]. Table 1 shows considerations to implement the HC Proxy according to the proxy service types.

Table 1. Comparison of Proxies for HTTP-CoAP Translation

	Interception	Forward	Reverse
Destination	Actual CoAP resource	Proxy	Proxy
Request target	Actual CoAP resource	Actual CoAP Resource	Sub resource of the proxy
Configuration	No	Client	Proxy
URI mapping	Homogeneous mapping	Not necessary	Embedded mapping or additional complex mapping mechanism

In interception proxy deployments, both clients and servers do not require any configurations. However, homogeneous URI mapping should be guaranteed. Thus, all internal resources should be serviced by only CoAP, and accessing to the HTTP resources in the network might be blocked by the proxy unexpectedly. In forward proxy deployments, clients should be aware of the existence of the proxy explicitly, so that they need some configurations. On the other hand, reverse proxy deployments do not require any configurations on the clients and servers. The proxy acts as a single server behalf of the network, and all CoAP resources are mapped to the proxy's sub resources.

3 HTTP-CoAP Proxy based on CGI

Since the reverse proxy is seen as a single HTTP server to the external clients, we implemented the HC Proxy on top of the Apache2, one of the well-known HTTP server programs. In order to implement the HC proxy, the most important part, we used CGI. CGI is originally used to generate dynamic web pages on the HTTP server. When a CGI program is executed, it can perform various operations as far as the system permits without any interactions with the originator. If the proxy should be implemented for the actual embedded devices such as the border router, CGI will be an efficient solution because the CGI program can be implemented lightly using C language.

```
http://{Proxy address}/coap/{CoAP URI path}
```

Our HC proxy represents the CoAP resources as above URI by using the embedded URI mapping method. The URL part makes that requests destine to the proxy explicitly. Following delimiter '`coap`' indicates that the request should be forwarded to the HC proxy. The HC Proxy sends a CoAP request to an actual target CoAP resource that is represented as '`coap://{CoAP URI path}`'. When the proxy receives a response from the resource, it generates a web page to be replied to the originator. Figure 1. shows the overall operations of the HC Proxy.

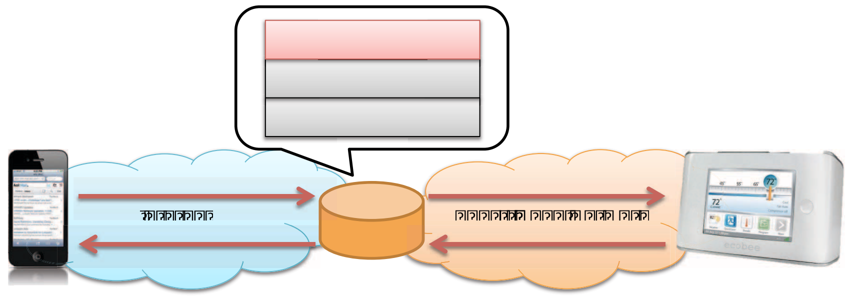

Fig. 1. HTTP-CoAP Proxy based on CGI

Since the CoAP is still not implemented on the OS (Linux), we implemented the CoAP functionalities in the HC Proxy together. The HC Proxy performs retransmissions based on exponential backoffs described in the CoAP. If there is no response until timeout, the proxy replies to the originator with the response code '500 Internal Server Error'.

Figure 2. is a demonstration of our HC Proxy implementation. We used a Linux and Apache2 HTTP server to execute the CGI. This linux machine is connected to the Internet via WiFi, and uses the Click based Border Router for LLNs [4] to organize an IEEE 802.15.4 based LoWPAN. We also implemented a CoAP server on the wireless plug device that can measure power consumption

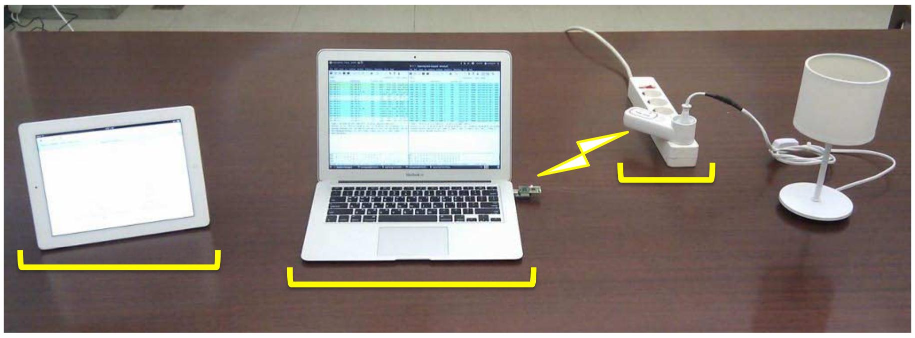

Fig. 2. HTTP-CoAP Proxy Demonstration

of a plugged appliance, and control a switch by using NanoQplus [5] operating system. In this demonstration, users can interact with the light connected to the CoAP device by using the tablet PC and its web browser that supports only HTTP.

4 Conclusion

In this paper, we implemented the reverse proxy that translates HTTP-CoAP using CGI. We expect that our implementation will be a good solution to implement and evaluate Web of Things services without any modifications on clients until the CoAP becomes an another de facto web standard.

Acknowledgments. This work was supported by the Industrial Strategic technology development program, 10035570, Development of self-powered smart sensor node platform for smart & green buildings, funded by the Ministry of Knowledge Economy (MKE), Korea.

References

1. Zeng, D., Guo, S., Cheng, Z.: The Web of Things: A Survey. In: Journal of Communications, Vol 6, No 6, 424-438 (2011)
2. Shelby, Z., Hartke, K., Bormann, C., Frank, B.: Constrained Application Protocol (CoAP). draft-ietf-core-coap-09, IETF (2012)
3. Castellani, A., Loreto, S., Rahman, A., Fossati, T., Dijk, E.: Best practices for HTTP-CoAP mapping implementation. draft-castellani-core-http-mapping-03, IETF (2012)
4. Jeong, J., Kim, H., Shin, G., Kim, S.: Poster: Click based IP border router for low-power and lossy networks. In: Proc. of the 9th ACM Conference on Embedded Networked Sensor Systems, Seattle US (2011)
5. Kim, S., Kim, H., Song, J., Yu, M., Mah, P.: NanoQplus-a multi-threaded operating system with memory protection mechanism for wireless sensor networks. In: Proc. of the 1st China-Korea WSN Workshop (CKWSN), Chongqing China (2008)