An Improved CoAP Scheme for 6LoWPAN Networks

Heng Wang
Key Laboratory of Industrial Internet of Things and Networked Control
Chongqing University of Posts and Telecommunications
Chongqing, China
wangheng@cqupt.edu.cn

Ping Wang
Key Laboratory of Industrial Internet of Things and Networked Control
Chongqing University of Posts and Telecommunications
Chongqing, China
wangping@cqupt.edu.cn

Na Bao
Key Laboratory of Industrial Internet of Things and Networked Control
Chongqing University of Posts and Telecommunications
Chongqing, China
georgina0000@126.com

Abstract—CoAP (Constrained Application Protocol) is a lightweight protocol proposed and standardized by IETF (Internet Engineering Task Force). It is designed for constrained environments, such as sensor nodes and sensor networks. In this paper, we present an improved CoAP scheme based on 6LoWPAN (IPv6 over Low power Wireless Personal Area Network) networks. The existing CoAP proxy mechanism requires a large cache space in proxy endpoints and is not optimized for large sensor networks. An enhanced proxy and cache mechanism of CoAP is designed to address this issue. The proposed mechanism uses query mode of discovery and significantly reduces the energy consumption and cache space usage. The cache is a two-dimensional array that stores resource information and identifiers with same structure at each line. Moreover, considering the alarm service widely used in the sensor networks, we also add a simple alarm method in CoAP to support the transmission of alarm information. The improved proxy and alarm mechanisms are implemented and verified in a 6LoWPAN platform.

Keywords-CoAP; IPv6; wireless sensor networks; proxy; alarm

I. INTRODUCTION

The use of web services for WSN (wireless sensor networks) is an important part in M2M applications, such as smart energy and lighting, building automation. The CoAP (Constrained Application Protocol) is proposed by IETF to suit of the REST web services architecture in constrained environment, for example wireless sensor networks. CoAP provides a request/response interaction model between CoAP client and constrained servers. Similar to HTTP, CoAP supports several key items of the Web services, such as URIs, method definitions, code description for responses, Internet media types, and so on. Thus, CoAP is designed to easily interface to HTTP with the Web. CoAP is based on UDP, which is quite different with HTTP that based on TCP/IP. However, CoAP applications in this transport pattern can easily cause errors in the process of message interaction. Thus, CoAP characterizes a reliable transport mechanism by supporting a simple stop and wait mechanism and defining message ID field in frame header for message reliable test and request/response match. Constrained networks such as WSN often have features of battery power shortage and cache space limitation. In this case, the sensors in the network often support a periodic dormancy mechanism. When a sensor is asleep, requests are disregarded. These limitations bring lots of inconvenience to the application developer. To deal with the endpoints sleepy problem, CoAP characterizes a proxy mechanism. A proxy is a CoAP endpoint in WSN that can be tasked by CoAP client or CoAP servers specified in advance to complete on their duty. These proxy devices may need more battery power to support being awake all the time. CoAP supports four methods to perform operations to resources in servers. Each method is defined along with its behavior, and cutting from HTTP to satisfy special needs of WSN. CoAP supports a safe and idempotent method GET, a neither safe nor idempotent method POST, two unsafe but idempotent methods PUT and DELETE.

II. 6LOWSN STACK

The CoAP scheme is designed and implemented in our protocol stack named 6LoWSN (IPv6 over Low power Wireless Sensor Networks). The hierarchical model of the stack is made up of 6 layers: the application layer, the transport layer, the network layer, the adaptation layer, the

media access control layer, the physical layer. The CoAP scheme is designed and implemented in the application layer of the 6LoWSN. Programming in C language, the 6LoWSN follows the FSM (finite-state machine) model. Judging by the environment and situation of program, the behavior of the device is different from time to time. The FSM model of application layer is described in Fig. 1.

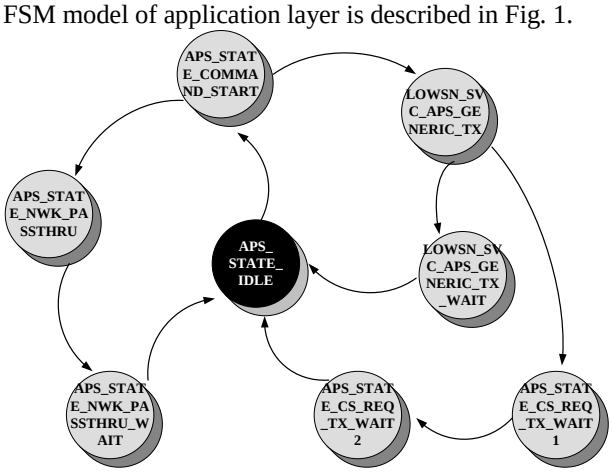

Figure 1. The FSM model of application layer.

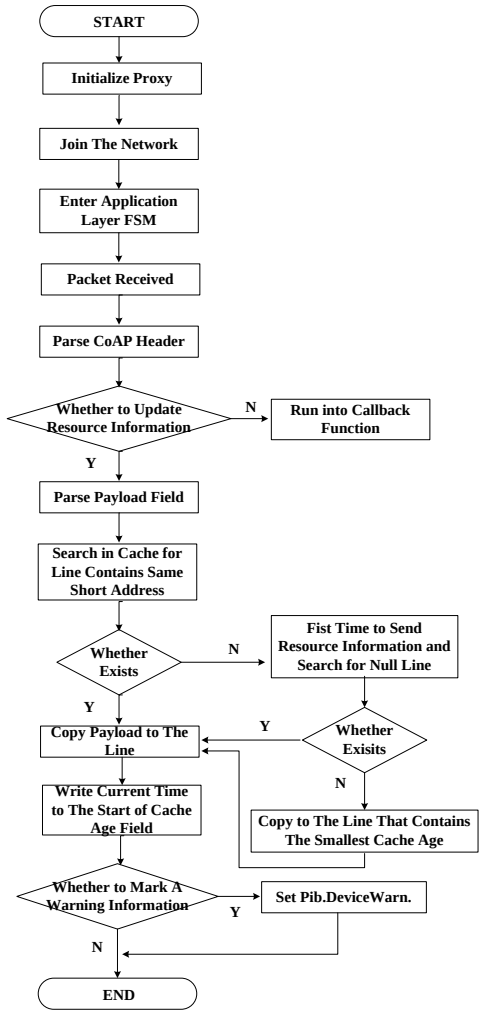

Figure 2. The flow chart of resource updating progress.

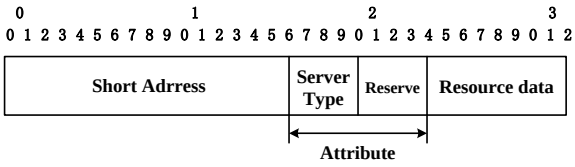

Figure 3. The frame structure of payload field.

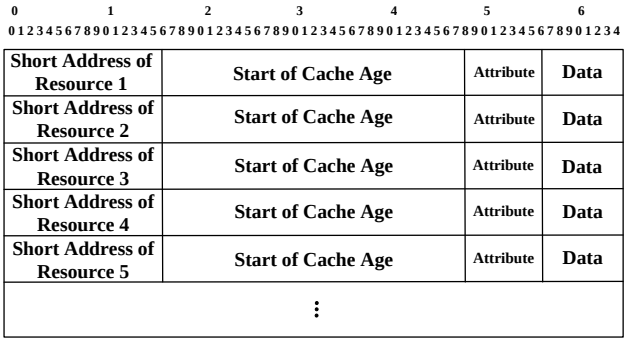

Figure 4. The structure of proxy cache.

III. COAP SCHEME

A. *Proxy and Cache Mechanism*

In an architecture for a constrained RESTful environment, proxies are divided into "forward-proxy" and "reverse-proxy", according to different functions. The CoAP client usually selects forward-proxies itself and entitles them the authority of clients. The reverse-proxy is set in WSN and stands in for servers.

A lightweight proxy and cache mechanism is optimized based on the original proxy function in CoAP. The proxy acts as a "reverse-proxy", and periodic receives and caches resource information from servers in the network. The resource information is encapsulated in the payload sent by servers periodically while awake. The payload structure is described in Fig. 3. The CoAP client sends operation requests to proxy rather than the servers when the servers are asleep. Then the proxy receives and starts to cache the data in payload field of the update message. The cache is defined a two-dimensional array, which contains resource information and resource identifiers with same structure at each line. The cache structure is described in Fig. 4. We use short address and the type of servers to identify resources in cache space. Servers periodically send packet that contains resource information to specified proxy. The proxy parses each packet and prepares to copy to the right line in cache. The first and the second byte of each cache line records the server short address that act as identifiers in cache of different servers. The preparation before caching describes as follows. The proxy parses the packets to see if it is a resource updating packet. If positive, search in cache for line that marked with the same short address. If negative, discard the packet or call the user call back function. If a resource is first to send update message to the proxy, a null line is searched for data cache. If a resource is first to send information to proxy without null line available, a line with the smallest start value of cache age is searched for data cache. This situation means that the cache space has been filled up. So it is important to adjust the number of proxies with the network scale. After right location found in cache,

the proxy copies the payload to the cache line in the order describes in Fig. 3 and Fig. 4. At last, to complete the information data update progress, the MAC timer is recorded to the cache age field as the start value of cache age in cache line. The information data update progress is describes in flow chart in Fig. 2.

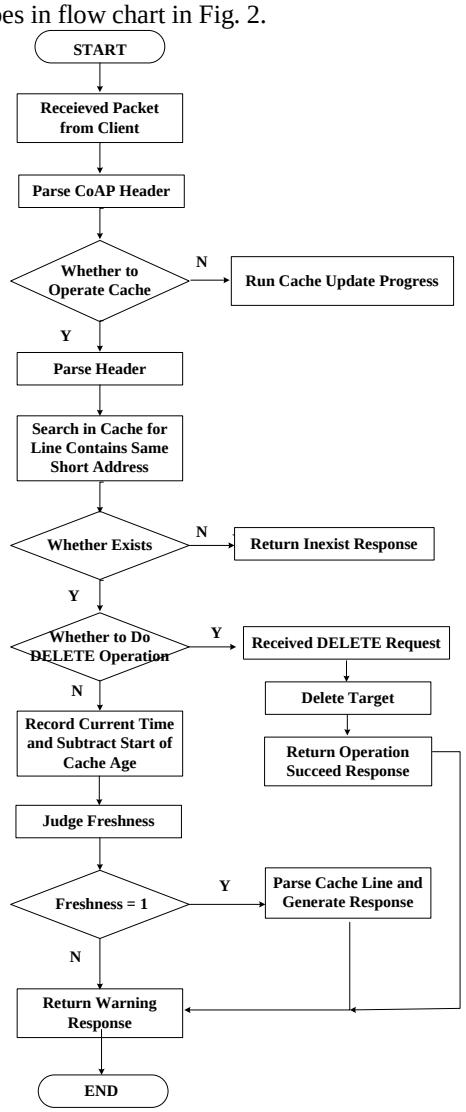

Figure 5. The flow chart of resource operating progress.

Further, the reliability of resource information is measured by a freshness judging model, optimized and implemented in CoAP proxy and cache mechanism. This model involves four data: the start value of cache age, the final value of cache age, the max age defines in progress, the freshness that judges the reliability of data in cache. When receives a request from client, the proxy records the MAC timer as the final value of cache age. Then minus the final value of cache age with the start value of cache age and record the difference. This difference is the value of cache age of the resource information. Then compare the max age with the cache age, if the max age equals to the cache age or larger than the cache age, the freshness is set to be 1. Otherwise, it is set to be 0. The reliability should be judged by the value of freshness. If the value of freshness is 1, the data in cache line is regarded as reliable data. If the value of freshness is 0, the data in cache line is regarded as unreliable data, which should be deleted in cache. Then return response to the client according to operations and freshness judge results. The cache operating progress is described in flow chart in Fig. 5.

B. Alarm Mechanism

An alarm mechanism is designed and implemented in CoAP scheme. On the server side, by marking out resource that over passes threshold range as soon as the updating packet is sent. On the proxy side, the resource information is received as warning information being cached and marked in proxy. This mechanism involves two parts: the client realizes warning information from proxies, the client realizes warning information and deals with warning situation of servers. The progress described as follows.

On the proxy side, regardless of operation methods, the marked out resource information is first sent back to CoAP client. Then the CoAP client is able to realize the warning information as soon as possible. After the alarm response being sent back, the proxy would delete the marked out information and clear warning status identifier. The alarm mechanism in proxy is described in flow chart in Fig. 6.

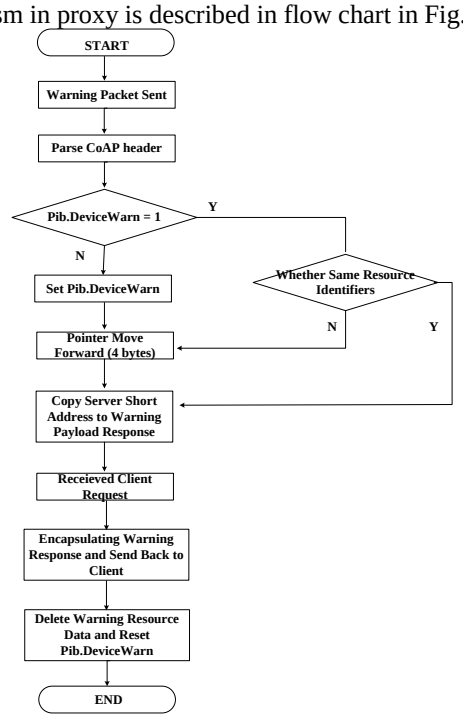

Figure 6. The flow chart of proxy warning progress.

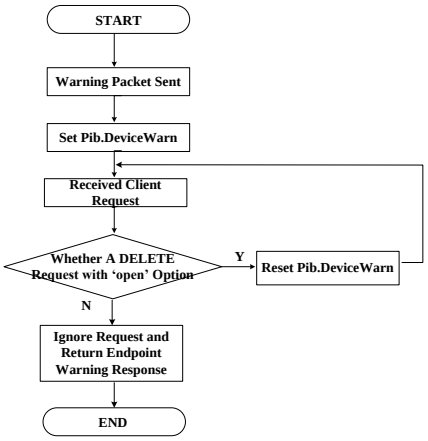

Figure 7. The flow chart of server warning progress.

Further measures are developed to this alarm mechanism on the server side. If a server is marked out for warning resources, the server is in status of alarm. Regardless of operation methods, the server under warning status should first returns a state describe response to client. If it is not true, via sending a DELETE request with a 'open' option value to the server, the client can clear the alarm. If it is true, via sending a DELETE request that contains a 'close' option value to the server, the client can dissociated the server with the proxy and the client. Only by sending a DELETE request that contains an 'open' option value to the server could associate the server to the proxy and client again. The alarm mechanism in server is described in flow chart in Fig. 7.

TABLE I. DESCRIPTION OF FUNCTIONS VERIFICATION IN DETAILS.

	Operation Object	Example URIs	Response code	Response Payload	Time Delay (ms)
GET	proxy	coap://[aaaa::1147:00ff:fe00:1699]:61619/?type=temperature	2.05 Content	<169a>;n="shortaddr", <169b>;n="shortaddr", <169c>;n="shortaddr"	217
	server	coap://[aaaa::1147:00ff:fe00:169a]:61619/.well-known/core		<temp>;n="temperature", <upper>;n="tempupper bound",<lower>;n="temp lower bound"	172
		coap://[aaaa::1147:00ff:fe00:169a]:61619/temperature		The current temperature is 17.4C.	182
POST	proxy	coap://[aaaa::1147:00ff:fe00:1699]:61619/169atemperatre	2.05 Content	169a: Fresh temperature data is 17.4C.	199
	server	coap://[aaaa::1147:00ff:fe00:169a]:61619/upperbound	2.01 Created 2.02 Deleted 2.04 Changed	Judged result: 05C~95C.	195
		coap://[aaaa::1147:00ff:fe00:169a]:61619/lowerbound	2.01 Created 2.02 Deleted 2.04 Changed	Judged result: 05C~95C.	187
PUT	proxy	coap://[aaaa::1147:00ff:fe00:1699]:61619/169atemperatre	2.05 Content	169a: Fresh temperature data is 17.4C.	210
	server	coap://[aaaa::1147:00ff:fe00:169a]:61619/upperbound	2.01 Created 2.02 Deleted 2.04 Changed	Changed to: 05C~95C.	187
		coap://[aaaa::1147:00ff:fe00:169a]:61619/lowerbound	2.01 Created 2.02 Deleted 2.04Changed	Changed to: 05C~95C.	192
DELETE	proxy	coap://[aaaa::1147:00ff:fe00:1699]:61619/169atemperatre	2.02 Deleted	169a: Temperature resource deleted.	199
	server	coap://[aaaa::1147:00ff:fe00:169a]:61619/upperbound	2.02 Deleted	The upper bound deleted.	190
		coap://[aaaa::1147:00ff:fe00:169a]:61619/lowerbound	2.02 Deleted	The lower bound deleted.	186
		coap://[aaaa::1147:00ff:fe00:169a]:61619/open	2.05 Content	Operate succeed.	202
		coap://[aaaa::1147:00ff:fe00:169a]:61619/close		Operate succeed.	209

IV. EXPERIMENTAL SYSTEM DEVELOPMENT

A realistic wireless sensor networks based on the 6LoWSN stack testbed have been implemented for our CoAP scheme. The 6LoWSN is able to simulate sensing data from the sensors interface, and transfer the sensing information over 2.4 GHz IEEE 802.15.4 communications upon the CC2530 hardware platform. The 6LoWSN software is written in C language, and followes the FSM model. The Copper (Cu) CoAP user-agent for Firefox supports a handler for the CoAP URI scheme. It provides users to browse and interact with IoT (Internet of Things) devices. We chose Firefox browser equipped with Copper (Cu) to be the CoAP client. Every sleepy sensor in the network act as a server endpoint, that contains resources and supports services. The non-sleepy node in the network serves as a CoAP proxy endpoint to receive and cache resources information and servers attribute. The testbed for CoAP scheme is shown in Fig. 8. The validation is divided into two parts: function verification and network performance verification.

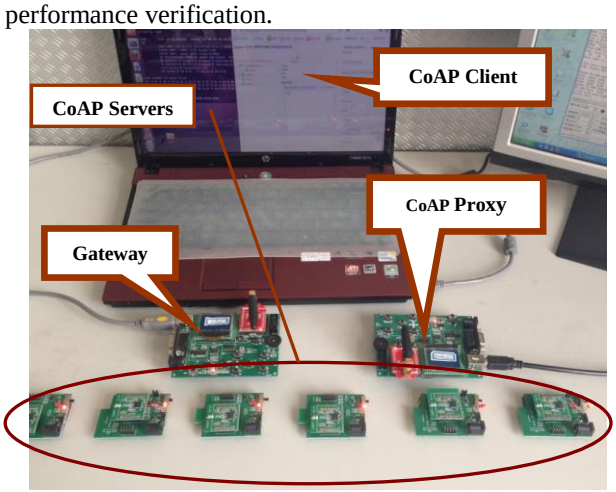

Figure 8. A part of the testbed for CoAP scheme.

A. *Function Verification*

The lightweight proxy and cache mechanism, supports operation methods of GET, POST, PUT, and DELETE. The function verification of cache operating details is shown in Table. I. An example of function verification of cache data update is shown in Fig. 9.

When proxy received warning information, CoAP client first returns the warning information including the short address of the warning server. The function verification of warning mechanism in proxy is shown in Fig. 10. The function verification of warning mechanism in servers is shown in Table. I.

B. *Network Performance*

The network time delay is an important data to measure the network interface. Time delay is caused by transporting messages from one endpoint of the network to another .It includes sending time delay, transporting delay, processing delay, and queuing delay. Among them, sending time delay and processing delay is our main considerations.

In Table. I, we listed the time delay from an operation request being sent to a match operation response being displayed. Judge from the time delay details in Table. I,

the CoAP scheme we designed and implemented works well in WSN environment with short time delay. The 6LoWSN has a good network performance.

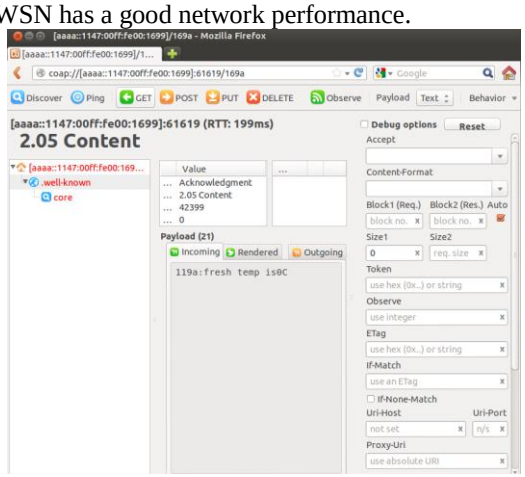

Figure 9. An example of the lightweight proxy and cache mechanism.

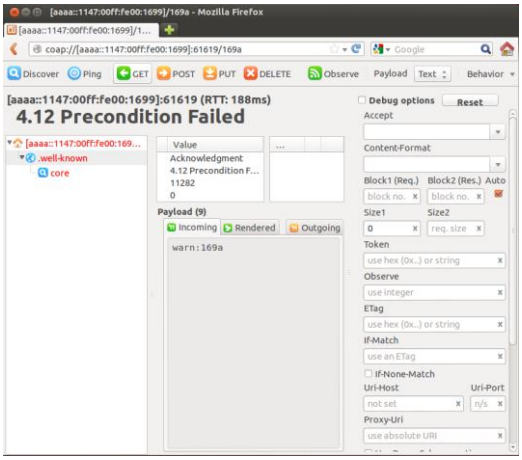

Figure 10. An example of the alarm mechanism in proxy.

V. CONCLUSION

In this paper, we present an improved CoAP scheme with enhanced proxy and alarm mechanisms. The scheme is implemented and tested in 6LoWPAN networks. The proposed scheme has the advantage of low energy consumption and proxy cache space.

ACKNOWLEDGMENT

This work was supported by the Fundamental and Advanced Research Program of Chongqing (Grant No. cstc2013jcyjA40008), the National Natural Science Foundation of China (Grant No. 61301125), and the Youth Top-notch Talent Support Program of Chongqing.

REFERENCES

[1] A. Rahman, Enhanced Sleepy Node Support for CoAP: InterDigital Communications, LLC, draft-rahman-core-sleepy-04 [P], 2013-08-08[2014-04-11].

[2] E. Dijk, Ed., A. Rahman, Ed.. Miscellaneous CoAP Group Communication Topics: Philips Research, InterDigital Communications, LLC, draft-dijk-core-groupcomm-misc-05 [P], 2013-09-09[2014-09-12].

[3] E. Dijk, Ed., Sleepy Devices using CoAP - Possible Solutions: Philips Research, draft-dijk-core-sleepy-solutions-02 [P], 2013-11-08[2014-05-12].

[4] Girum Ketema Teklemariam , Jeroen Hoebeke , Ingrid Moerman , Piet Demeester. Facilitating the creation of IoT applications through conditional observations in CoAP [J]. EURASIP Journal on Wireless Communications and Networking, 2013, Vol.2013 (1): 1-19.

[5] Jouni Mänpää, Jaime Jiménez Bolonio, Salvatore Loreto. Using RELOAD and CoAP for wide area sensor and actuator networking [J]. EURASIP journal on wireless communications and Networking, 2012, Vol.2012 (1): 1-22.

[6] K. Li, CoAP Payload-Length Option Extension: Huawei Technologies, draft-li-core-coap-payload-length-option-02 [P], 2013-08-16[2014-02-17].

[7] T. Fossati. A Link-Format Attribute for Locating Things: KoanLogic, draft-fossati-core-geo-link-format-attribute-01 [P], 2013-08-11[2014-04-14].

[8] T. Fossati. Multipart Content-Format Encoding for CoAP: KoanLogic, draft-fossati-core-multipart-ct-03 [P], 2013-08-14[2014-04-17].

[9] T. Fossati, P. Giacomin, S. Loreto. Publish Option for CoAP: KoanLogic, Freelance, Ericsson, draft-fossati-core-publish-option-02 [P], 2013-08-20[2014-04-23].

[10] Y. Doi, K. Lynn. CoAP Content-Type Parameter Option: TOSHIBA Corporation, Consultant, draft-doi-core-parameter-option-03 [P], 2013-08-08[2014-02-09].

[11] Z. Shelby, S. Krco, C. Bormann. CoRE Resource Directory: Sensinode, Ericsson, Universitaet Bremen TZI, draft-ietf-core-resource-directory-00 [P], 2013-06-03[2013-12-05].