Lightweight Management of Resource Constrained Sensor Devices in Internet-of-Things

Zhengguo Sheng, *Member, IEEE,* Hao Wang, Changchuan Yin, *Member, IEEE,*
Xiping Hu, *Student Member, IEEE,* Shusen Yang, *Member, IEEE,* and Victor C. M. Leung, *Fellow, IEEE*

***Abstract*—It is predicted that billions of intelligent devices and networks, such as wireless sensor networks (WSN), will not be isolated but connected and integrated with computer networks in future Internet-of-Things (IoT). In order to well maintain those sensor devices, it is often necessary to evolve devices to function correctly by allowing device management entities to remotely monitor and control devices without consuming significant resources. In this paper, we propose a lightweight RESTful web service approach to enable device management of wireless sensor devices. Specifically, motivated by the recent development of IPv6 based open standards for accessing wireless resource constrained networks, we consider to implement 6LoWPAN/RPL/CoAP protocols on sensor devices and propose a CoAP based device management solution to allow easy access and management of IPv6 sensor devices. By developing a prototype cloud system, we successfully demonstrate the proposed solution in efficient and effective management of wireless sensor devices.**

***Index Terms*—Internet-of-Things, Device management, CoAP, Wireless sensor networks, IPv6.**

I. Introduction

The concept of IoT can be traced back to the pioneering work done by Kevin Ashton in 1999 and it is initially linked to the new idea of using RFID in supply chain [1]. Since recently, this term became popular and is well known as a new communication system where the Internet is connected to the physical world via ubiquitous wireless sensor networks. Generally, sensing devices are with common features of constrained energy resources, limited processing capability, vulnerable radio conditions, real time nature of applications and no direct human interaction, etc. By inter-connecting sensor devices using low cost wireless communication technologies, which is usually named as wireless sensor networks, a new ecosystem with a large number of smart applications has been formed.

With the development of IoT technologies in the past few years, a number of major standardization alliances are gradually formed based on their interests in technology selections and commercial markets. Technically speaking, current IoT solutions can be categorized as non-IP based and IP based solutions. Most of off-the-shelf solutions belong to the former, especially for some well-known standard alliances, such as ZigBee [2] and WAVE2M [3] for office and manufacturing automation, and WirelessHart [4] and PROFIBUS [5] for real-time industrial control systems, etc. However, most of these non-IP solutions are isolated within their own verticals, which hinders the IoT development due to the incompatible nature across heterogeneous communication systems.

Motivated by the fact that the TCP/IP protocol is the de-facto standard for computer communications in today's networked world, IP based solution could be the future for IoT networks. etc. IP Smart Object Alliance (IPSO) [6] actively promotes IPv6 embedded devices for Machine-to-Machine (M2M) applications. PROFINET, a promising real-time Ethernet standard, also adapts Ethernet to the next generation of industrial automation [7]. In order to tackle the technical challenges, such as extensive protocol overheads against memory and computational limitations of sensor devices, IETF[1] takes the lead to standardize communication protocols for resource constrained devices and develop a number of Internet protocols, including IPv6 over Low power wireless personal area networks (6LoWPAN) [8], Routing Protocol for Low Power and Lossy Network (RPL) [9] and Constrained Application Protocol (CoAP) [10], etc.

Although a wide range of intelligent and tiny sensing devices have been massively deployed in a variety of application environments, many open challenges remain, which are mostly due to the complex deployment characteristics of such systems and the stringent requirements imposed by various services wishing to make use of such complex systems [11]. In order to well maintain a large scale of wireless sensor networks, for example, dynamic registration of sensor devices or monitoring sensor performance, IoT device management (DM) authorities should be able to provide a reliable and efficient way to remotely monitor and control sensor devices without consuming significant resources. This also provides a motivation to recent global IoT/M2M related standardisations. In particular, the oneM2M Global Initiative [12], [13] has been formed in order to develop one globally agreed specifications for common service layer, which can be the basis of horizontal

Z. Sheng is with School of Engineering and Informatics, University of Sussex, UK, and the Department of Electrical and Computer Engineering, University of British Columbia, Canada. (E-mail: z.sheng@sussex.ac.uk)

H. Wang is with Orange International Labs Beijing, China. (hao.wang@orange.com)

C. Yin is with Beijing University of Posts and Communications, China. (ccyin@bupt.edu.cn)

X. Hu and V. Leung are with the Department of Electrical and Computer Engineering, University of British Columbia, Canada. ({xipingh,vleung}@ece.ubc.ca)

S. Yang is with Department of Electrical Engineering and Electronics, University of Liverpool, UK. (shusen.yang@liverpool.ac.uk)

[1]The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors and researchers concerned with the developments and promotions of Internet standards of the Internet protocol suite (TCP/IP).

management platform. The IoT-A project [14], which is an European research project addressing the Reference model of IoT, is to develop IoT architectures in an interoperable manner. The project has derived entities and resources, which are subject for management functions, and provides various functions to orchestrate and manage collaboration of IoT devices.

Device management is an integration of network, system and application managements. In essence, it includes provisioning and management, configuration of network parameters, firmware upgrades and performance monitoring, etc. As a result, a number of device management standards are emerging, such as TR-069 [15] for automatic configuration of set-top box, ISO/IEC 14543-3 [16] for home and building automation (HBA), and Field Device Integration (FDI) [17] for the unified management for all field devices, etc. However, none of these standards can be directly used on IoT devices, especially on a processing and battery restricted sensor device.

We are looking at open technologies to tackle device management of WSN and the IPv6 based solution is a promising one. Motivated by the fact that the CoAP is an application layer protocol which is intended for use in resource constrained Internet devices and the simple network management protocol (SNMP), we propose a framework of CoAP based device management solution for WSN and develop device management oriented functions, resource identities and protocol, etc. Specifically, we take an approach to extend the Representation State Transfer (REST) paradigm [18], in which a lightweight web server can be embedded in resource constrained sensor devices, and map DM functions into CoAP methods. In essence, the proposed method not only integrates IoT into the network, but also manages them via the "web".

The following summarizes our contributions and key results:

- We implement the IPv6 protocol stack on sensor testbed and integrate the border router of WSN into an open-platform gateway as well as implement the HTTP-CoAP proxy implementation to the OpenWrt to realize remote access from an ordinary IP terminal to IPv6 sensor devices.
- We propose a framework of efficient device management solution for WSN based on IETF open standard CoAP and develop device management oriented functions, naming and addressing for resource identities as well as mappings of CoAP methods to management functions. The performance evaluation shows that the proposed solution can significantly reduce both packet length and loss rate by 74% and 18.75%, respectively.

The remainder of this paper is organized as follows. Related works is provided in Section II. The RESTful protocol stack is introduced in Section III. The proposed DM solution is discussed in Section IV. The prototype system is presented in Section V and performance evaluation results are shown in Section VI. Finally, concluding remarks and future work are given in Section VII.

II. Related work

Recent technology trends in the Web Services (WS) are primarily separated as Big Web Services (or WS-*) and RESTful Web Services. In the latest work of web services in WSN, Kyusakov *et al.* in [19] take an approach to deploy interoperable Simple Object Access Protocol (SOAP)[2]-based web services directly on nodes. Moreover, the web service methods are also widely applied in Automation industry. Cucinotta *et al.* in [20] propose a service-oriented architecture (SOA) with enhanced real-time capabilities by allowing for negotiation of the QoS requested by clients from web services for industrial automation. However, the above literatures prefer the WS-* architecture which may bring extensive overheads for resource constrained devices. Pautasso *et al.* in [21] compare these two architecture choices and argue that the RESTful WS can create a loosely coupled system which is better suited for simple and flexible integration scenarios, whereas WS-* can provide more advanced quality-of-service for enterprise level usages.

More recent works are dedicated for developing REST-style IoT systems to enable easy access from application servers to wireless sensor devices [22]–[25]. REST, a lightweight web service implementation, is a general design style of Internet resource access protocol. It provides a design concept that all the objects in the Internet are abstracted as resources. REST style can make applications as sharable, reusable and loose coupling services. Although its simplicity, most of existing solutions are not IP based, which means that a multiprotocol translation gateway is needed. As discussed in [26], the network protocol translation can bring more complexity than just a packet format conversion, which usually involves semantics translation between different mechanisms and logic for routing, quality-of-service and security [27], etc.

There are recent papers focusing on the implementation of IPv6 protocol stack on various system platforms. Shelby in [28] gives an overview of the web architecture and introduces the new IETF Constrained RESTful Environments (CoRE) standardization activity. Potsch *et al.* in [29] demonstrate an intelligent container testbed where the CoAP protocol is implemented on the embedded operating system TinyOS[3]. Moreover, a couple of other implementations of CoAP are also available on the Contiki[4] platform [30]–[32]. There are also papers on the protocol mappings from proprietary solutions or SOAP to CoAP, e.g. [33], [34]. However, most of these papers are either considering protocol translation to CoAP or for the purpose of connectivity evaluations on different operation platforms by assuming a virtual gateway.

Although considerable research has been done on different aspects of sensor networks, the management issue for sensor devices is still little explored. Frye *et al.* in [35] propose to

[2]SOAP is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks.

[3]TinyOS is an open source software component-based operating system and platform targeting WSN.

[4]Contiki is an open source operating system for the Internet-of-Things. Contiki allows tiny, battery-operated low-power systems communicate with the Internet.

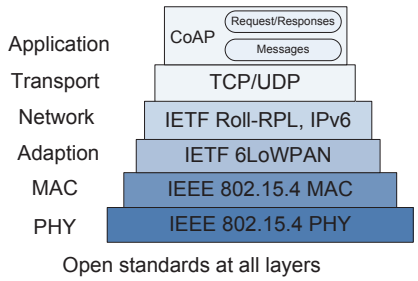


Fig. 1. An overview of IPv6 protocol stack ranging from Physical layer up to Application layer

incorporate ontology into management of Ad Hoc networks based on existing network management protocol, such as SNMP. Yang *et al.* in [36] propose a basic concept of device management scheme based on IEEE 1451 standard. Different to the previous methods, Hergenroeder *et al.* in [37] propose a hard way method by developing an extra hardware to support energy management. However, those solutions are all implemented using independent management protocols or hardware which are lack of reuse with parallel communication protocols, e.g., CoAP, and may bring extract complexity. Although there are recent works considering to introduce management into CoAP server, i.e., [38] proposes dedicated application protocol on top of CoAP to map all application functions in building automation, [39] utilizes Synchronization Markup Language (SyncML) protocol onto CoAP for DM, and [40] proposes the latest integration of CoAP with SNMP, they all either build management capabilities on top of CoAP or need to support multiple protocols simultaneously, which may bring extra overhead for resource constrained devices.

Different to the above literatures, our contribution in this paper is to develop an efficient and lightweight device management solution by extending the CoAP protocol without consuming extra resources. To the best of our knowledge, this is the first work that considers device management via CoAP protocol and realizes ease of access and management of WSN.

III. The Implemented Protocol Stack for WSN

In this section, we introduce the IETF protocol stack used in resource constrained networks. Fig. 1 illustrates the protocol stack ranging from physical layer up to application layer.

A. IEEE 802.15.4

IEEE 802.15.4 [41] is a radio technology standard for low power and low data rate applications with a radio coverage of only a few meters. It has typically a maximum data rate of 250 kbps and a maximum output power of 1 mW. The maximum packet size is 127 bytes. Besides, in order to achieve energy savings, radio power management (e.g., duty cycling) is an essential part in MAC layer mechanisms. The radio transceiver must be managed so that it can be switched off when there is no traffic but switched on when communication is engaged.

B. 6LoWPAN

The main focus of 6LoWPAN is on protocol optimization of IPv6 over networks using IEEE 802.15.4. In fact, there are two reasons to apply 6LoWPAN over IEEE 802.15.4. On the one hand, consider the maximum frame size supported by IEEE 802.15.4 is only 127 bytes and significant header space may be taken by other layered protocols (e.g., MAC layer header, IPv6 header, security header and transport layer), the payload size available for the application layer is very limited. On the other hand, since the minimum value of maximum transmission unit (MTU) specified by IPv6 is 1280 bytes which is larger than the supported size of IEEE 802.15.4, an adaptation layer right above the data link layer to segment the IPv6 packet into small pieces is required by the lower layer.

C. RPL

RPL is a distance-vector routing protocol, in which nodes construct a destination oriented Acyclic Graph (DODAG) by exchanging distance vectors and root information with a "controller". Through broadcasting routing constraints, the root node (i.e., central control point) filters out nodes that do not meet the constraints and selects the optimum path according to the metrics. In a stable state, each sensor node will have a set of "parents" and will forward packets along its parents to the "root".

D. CoAP

CoAP is a specialized web transfer protocol for resource constrained nodes and networks. CoAP conforms to the REST style. It abstracts all objects in the network as resources. Each resource corresponds to a unique Universal Resource Identifier (URI) from which the resources can be operated stateless, including GET, PUT, POST, DELETE and so on. The URI is described as a Link in the CoRE link format [42]. The CoRE link format is carried as a payload and is assigned an Internet media type.

Unlike HTTP, CoAP adopts datagram-oriented transport protocols, such as User Datagram Protocol (UDP). In order to ensure reliable transmission, CoAP introduces a two-layer structure as shown in Fig. 1: the messaging layer is used to deal with asynchronous interactions with UDP, such as Confirmable (CON), Non-confirmable (NON), Acknowledgment (ACK) and Reset (RST) messages. Whereas the Request/Response interaction layer is used to transmit resource operation requests and the request/response data.

IV. CoAP-based Device Management for WSN

In this section, we propose a framework of efficient device management solution for WSN based on IETF open standard CoAP and develop device management oriented functions, resource identities and protocol, etc.

A. Management functions

The system architecture in Fig. 2 shows the interaction relations between a IoT client (e.g., via cloud platform) and a sensor device, especially the device components interfaces to IoT client. Due to the requirements imposed to IoT services, such as no direct human interaction, reliable remote management/control and scalable features of applications, we propose

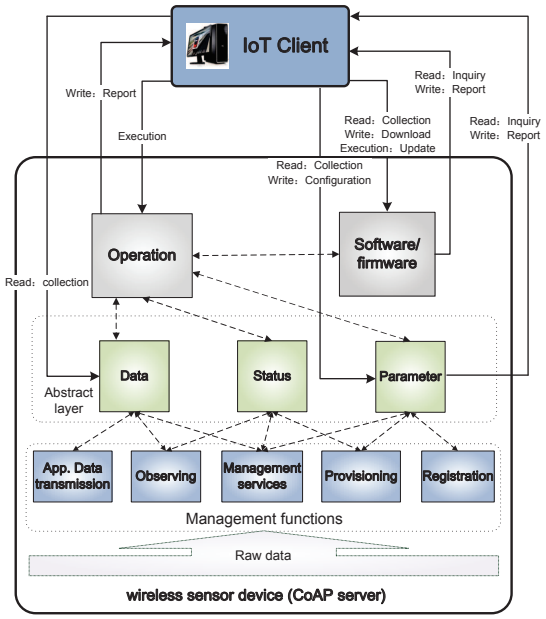


Fig. 2. A system architecture of IoT device management, including management functions, abstract layer and interconnections with client.

five major device management functions which are applicable to WSN:

1) **Registration**: It is a primary function to allow a sensor device to register/de-register with a remote management server, maintain and update registration information.
2) **Provisioning**: It is to initialize and synchronize essential information of a sensor device with a remote management server.
3) **Management services**: Once the sensor device is well connected with a remote server, a number of essential management services should take in charge to maintain IoT services, such as parameter configuration, connection diagnose, status inquiry and remote control, etc.
4) **Observing**: It is the unique feature of CoAP to allow sensor devices to "observe" resources, i.e., to retrieve a representation of a resource and keep this representation updated by the remote server over a period of time.
5) **Application data transmission**: It includes any application data that can be collected and delivered to IoT clients, or any other application protocols above CoAP.

Although the management functions can be defined in different manners, they all share common resources on one sensor device and we abstract these resources as parameters (e.g., hardware and software information), status (e.g., dynamic information for connection and faulty diagnose) and data (e.g., information collected from environment), which are defined as abstract layer in Fig. 2. The interactions with IoT client can be directly triggered with these resources via GET, PUT, POST and DELETE methods provided by CoAP.

B. Naming and addressing of resource identities

We define a simple resource model in which resources are logically organized into class. A class defines a group of resources, for example the Hardware class contains all the resources that can be used for provisioning purposes. A resource is identified by the path:

$$\sim /\{\text{Class ID}\}.\{\text{Resource ID}\}.\{\text{Sub-Resource ID}\}.\{\text{Method ID}\}$$

where the Class ID, Resource ID and Sub-Resource ID are with size of 1 byte. The Method ID[5] is to represent access methods available to a resource. The method ID is 4 bits and each bit from the Most significant bit (MSB) represents an authorized operation in a sequence of GET, PUT, POST and DELETE. The value "1" means authorized and "0" means non-authorized. Table I shows the detailed naming and addressing assignment for resources on our sensor testbed. Each class is assigned a unique identity. If a resource does not support multiple sub-resources, the sub-resource ID is set as 0. For example, for retrieving the operating system version of the testbed, the resource identity should follow CoRE link format [42]. Through the resource discovery process GET $< /.\texttt{well-known/core} >$, the IoT device should response with resources information, e.g., $<\sim /2.1.0.1000 >$. It is noted that the proposed resource model can widely support any resources in practical implementation.

TABLE I
NAMING AND ADDRESSING ASSIGNMENT

Class		Resource		Sub-Resource		Method	
Name	**ID**	**Name**	**ID**	**Name**	**ID**	**Name**	**ID**
Hardware information							(Hex→bit)
HW	1	CHIP_ID	1	Null	0	Get	8→1000
HW	1	SRAM	2	Null	0	Get	8→1000
HW	1	FLASH	3	Null	0	Get	8→1000
HW	1	MAC_addr	4	Null	0	Get	8→1000
HW	1	RADIO	5	Frequency	1	Get	8→1000
HW	1	RADIO	5	Channel	2	Get	8→1000
HW	1	USART0	6	Baudrate	0	Get	8→1000
Operating system information							
SYS	2	CONTIKI_v	1	Null	0	Get	8→1000
SYS	2	NAME	2	Null	0	Get/Put	c→1100
Network protocol information							
NET	3	PANID	1	Null	0	Get	8→1000
NET	3	RDC	2	Null	0	Get	8→1000
NET	3	MAC	3	Null	0	Get	8→1000
NET	3	NETWORK	4	Null	0	Get	8→1000
NET	3	IPv6	5	Prefix	1	Get	8→1000
NET	3	CoAP_v	6	Null	0	Get	8→1000
Onboard resources information							
RES	4	ACTUATOR	1	leds	1	Get/Put /Post	e→1110
RES	4	SCREEN	2	Null	0	Get/Put /Post	e→1110
RES	4	SENSOR	3	Humidity	1	Get	8→1000
RES	4	SENSOR	3	Illumination	2	Get	8→1000
RES	4	SENSOR	3	Temperature	3	Get	8→1000

C. CoAP-based management protocol

The proposed device management protocol is fully based on IETF defined CoAP protocol with newly defined resource

[5]The CoAP server may assign different method IDs to a same resource as long as clients' access levels are different. For example, administrator may have the full access rights of the whole IoT system, whereas some clients may only have "read" access to sensor devices.

identities to identify management resources and mappings of CoAP methods to management functions.

The CoAP is based on the exchange of short messages which, by default, are transported over UDP (i.e. each CoAP message occupies the data section of one UDP datagram). The protocol has a registered scheme of $< \texttt{coap} : // \sim >$ with a default port of 5683. Reliability over the UDP transport is provided by the built-in retransmission mechanism of CoAP, e.g., confirmable message defined by the Type field in the header. It could also be used over other transport protocols such as TCP or SMS. CoAP messages are encoded in a simple binary format. The message format starts with a fixed-size 4-Byte header. This is followed by a variable-length Token value which can be between 0 and 8 bytes long. Following the Token value, it comes a sequence of zero or more CoAP Options in Type-Length-Value (TLV) format, optionally followed by a payload which takes up the rest of the datagram.

Table II shows the detailed CoAP methods mapping to device management functions. We should note that each management function can be abstracted as a recall process to conduct with resources on sensor device, thus the RESTful approach provided by CoAP protocol can be adopted as a lightweight method to access from application servers to wireless sensor devices. Especially, the Uri-Path Option is to indicate management resource identities and the Location-Path Option is to indicate the address of remote registration server for future update and delete operations.

TABLE II
COAP METHODS MAPPING TO DEVICE MANAGEMENT FUNCTIONS

Function	Direction	Logical operation	CoAP method	Uri-Path Opt.	Location-Path Opt.
Registration	Uplink	Register	POST	√	√
	Uplink	De-register	DELETE	√	√
	Uplink	Update	PUT	√	√
Provisioning	Down/ uplink	Configuration	GET	√	
Management services	Down/ uplink	Read	GET	√	
		Write	PUT	√	
		Execute	POST	√	
		Creat	POST	√	
Observing	Downlink	Observe	GET with observe opt.	√	
	Uplink	Notify	Response to observe	√	√
App. data transmission	Downlink	Collect	GET	√	

V. PROTOTYPING SYSTEM

In this section, we present our prototyping system to implement the RESTful DM methods to IPv6 WSN. The network topology is shown in Fig. 3, where a laptop acts as a client to retrieve sensor resources via the RESTful gateway.

A. Sensor nodes

We deploy wireless sensor devices to monitor indoor environment. The sensor platform is equipped with CC2530 MCU

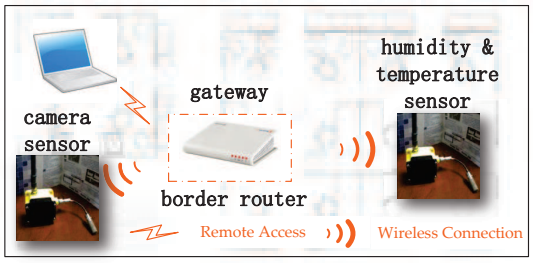


Fig. 3. Network topology of the prototype system: a laptop acts as a client to retrieve sensor resources via the RESTful gateway.

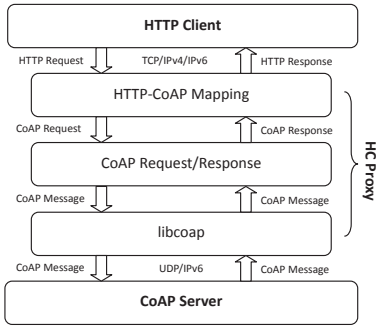


Fig. 4. Interaction process of HC proxy between a client and a server, which includes libcoap, CoAP Request/Response and HTTP-CoAP mapping.

with 8051 CPU core running at 32MHz, 8KB SRAM and 256KB flash block to support IEEE 802.15.4-compliant radio transceiver. To support IPv6 connectivity, all sensor devices are running Contiki v2.6 operating system with implementation of 6LoWPAN, IPv6 and RPL protocols based on IEEE 802.15.4. The web service running on the sensor devices relies on the application protocol CoAP.

B. RESTfull Gateway

In order to ease the access from Internet applications to sensor resources, especially for those of Internet users who cannot speak CoAP, we integrate IEEE 802.15.4 connectivity into an open-platform gateway and port the HTTP-CoAP (HC) proxy to the OpenWrt, the operation system of the gateway, to access from an ordinary IP terminal to an IPv6 sensor device.

In our prototype gateway, the HC proxy is implemented based on libcoap [43] which is an open-source C-Implementation of CoAP and conforms to GPL v2 or higher licenses. The interaction process of the HC proxy is shown in Fig. 4. Specifically, for each of the HC proxy layers, we have the following implementations:

1) libcoap layer: It defines message structure and methods to implement the CoAP messages layer based on UDP.

2) CoAP Request/Response layer: CoAP Request/Response layer encapsulates the data structure and methods relevant to CoAP Requests and Responses. It is to transmit CoAP request in the form of CoAP message through the messages layer and generate CoAP response based on received CoAP messages.

3) HTTP-CoAP mapping layer: It is to implement mapping from HTTP request to CoAP request and vice versa. When converting a HTTP request to a CoAP request, the HC proxy

needs to convert the HTTP request method, URI, header/option and payload, respectively. If a proxy encounters an error, it has to generate corresponding error response.

VI. Performance Evaluation

In this section, we provide evaluation results to illustrate the performance of the proposed CoAP based device management solution and its application in IoT system.

A. System Configuration

Our prototype system is composed of sensor devices, one HC proxy gateway and one laptop for initiating tests. We deploy the prototype system in an open office area. The HC proxy gateway and sensor devices are connected wirelessly via IEEE 802.15.4 and using channel 26. The laptop client is connected to the gateway through the Wi-Fi channel. The network topology as shown in Fig. 3 is built with a maximum 2 hops, where the camera sensor and humidity&temperature sensor are connected to the gateway with one hop distance.

B. Latency performance of CoAP protocol

In order to manage a large scale deployment of wireless sensor networks, it is necessary to keep the protocol overhead as small as possible. In this evaluation, we compare the round trip time (RTT) of CoAP ping to a sensor node with that of Internet control message protocol (ICMP)[6] ping messages to both the border router and sensor node using Linux ping6 command. The ContikiMAC is configured as no sleep mode. The IPv6 packet size is fixed as 52 bytes. Fig. 5 shows the RTT comparison between the three methods over 500 independent measurements. The average RTT to the border router and sensor node using ICMP are 49.75 ms and 79.54 ms, respectively, and the average RTT of CoAP ping to sensor node is 80.5 ms. It is noted that the RTT to the sensor node is much higher than that to the border router, this is because the processing time imposed at the gateway as well as one extra hop to transmit to the sensor node. However, the RTT of CoAP ping is very closed to the ICMP ping to the same sensor node, which only claims a 1.2% increases. The result tells that the overhead imposed by CoAP protocol is negligible and thus the CoAP based DM is a promising solution for IoT.

C. Packet length, latency and packet loss performance of the proposed DM solution

To further evaluate the performance of the proposed CoAP based device management solution, we compare it with the standard CoAP method in terms of packet length, latency and packet loss rate. Table III shows the onboard resources defined by both standard CoAP method (human-readable string) and the proposed method. The URI length is calculated from the space occupied in the RAM. It is clear that the proposed URI representation takes far less memory space than the standard URI representation in which the main space are

[6]ICMP is an Internet layer protocol to control and report message error between a host server and a gateway to the Internet. ICMP uses Internet Protocol (IP) datagrams, but the messages are processed by the IP software and are not directly apparent to the application user.

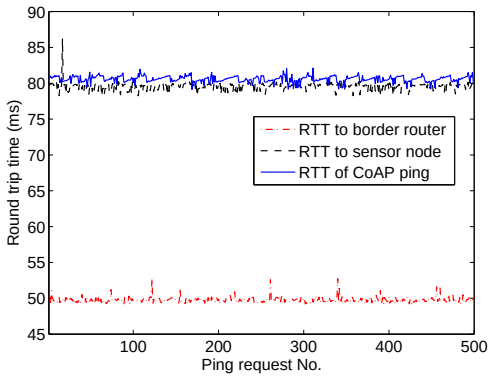


Fig. 5. RTT comparison between CoAP ping to sensor node and ICMP pings to both border router and sensor node.

consumed by "Attributes". Through the resource discovery GET < /.well-known/core >, we can receive a response with a list of available resources as shown in Table III and the total length of transmission packets for both methods are 420 bytes and 109 bytes, respectively. Since the CoAP response will be transmitted in a block-wise fashion, the standard method takes 6 blocks, whereas the proposed method only takes 2 blocks. The memory saving of 311 bytes is composed of URI savings and 4 extra CoAP block headers. The total transmitting packets can be reduced by 74%.

TABLE III
URI length comparison between standard CoAP method and the proposed DM method

Resource	Standard URI	Bytes	Proposed URI	Bytes
CoAP_version	CoAP_v	45	3.6.0.8	8
LEDs	actuators/leds	76	4.1.1.e	8
Illumination	sensor/illumination	51	4.3.2.8	8
Temperature	sensor/temperature	45	4.3.3.8	8
Humidity	sensor/humidity	42	4.3.1.8	8
Screen	Screen	57	4.2.0.e	8

To further evaluate the latency and packet loss performance of the proposed DM solution, we setup a test to send GET request (i.e., < /.well-known/core >) to retrieve onboard resources. Fig. 6 shows the histogram of the request/response delay and retransmission delay over 50 independent measurements. As can be seen from Fig. 6 (a)[7], the proposed method shows a tendency of smaller delay with very high probability that the request/response delay is below 1 s. However, the performance of the standard method is varied from 1 s to 10 s, because of the unreliable radio environment and a larger number of transmitting packets. Fig. 6 (b) shows the retransmission delay caused by packet loss, the proposed method shows smaller delay with a higher probability that the retransmission can be ensured as 0. A careful reader may notice that there are only even numbers of retransmission delay on x-axis, since once a blockwise packet or a CON message is missing, the CoAP server will postpone a retransmission (delay) in

[7]To best plot the histogram using MATLAB, we set the x-axis bin internal as 0.4 s, which means that a delay will be counted on the same bar if its value is within the same interval.

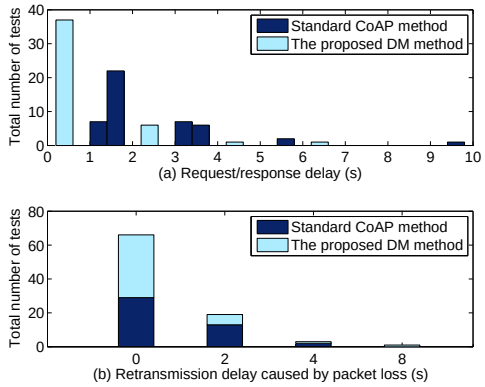


Fig. 6. Comparisons of request/response delay and retransmission delay

2^n s, where n is the number of retransmission. As a result, the average request/response delay and retransmission delay for the standard method are 3.04 s and 1.44 s, respectively, whereas the results for the proposed method are only 0.8 s and 0.36 s, respectively.

At the end, we evaluate the packet loss rate over the 50 measurements and have the following results

$$p_{\text{STD}} = 38/(38 + 50 \times 6) = 11.2\% . \quad (1)$$

where 38 is the total number of retransmission and 50×6 is the total blocks of transmission. The result for the proposed solution is

$$p_{\text{Proposed}} = 10/(10 + 50 \times 2) = 9.1\% . \quad (2)$$

where 10 is the total number of retransmission and 50×2 is the total blocks of transmission. Therefore, compared to (2) with (1), an improvement of 18.75% can be achieved by using the proposed solution.

In order to evaluate the performance of a large scale network, we set up another test to evaluate the packet loss rate in a multi-hop environment. The test is carried out in an open office area with strong Wi-Fi background noise and lowest possible WSN radio frequency output power to ensure a multi-hop fashion, which makes a sensor device can only communicate to each other within around 30 cm.

Since the sensor devices are with only 8KB RAM and 256KB flash, a maximum number of 6 hops can be obtained by optimizing the communication system. To retrieve the same onboard resources via GET request (i.e., $<$ `/.well-known/core` $>$) over the same number of measurements, Table IV shows the packet loss rate in a multi-hop scenario. We can observe that the packet loss rate increases dramatically with an increasing number of hops, because of severe environmental interference and channel congestions, etc. Moreover, additional configurations to ensure a multi-hop transmission, such as one way communication, low output power and RPL settings, also contribute to the high loss. However, the result does not affect the true that the proposed solution can always achieve better performance than the standard method.

TABLE IV
PACKET LOSS RATE IN A MULTIPLE-HOP NETWORK

		Hop 2	Hop 3	Hop 4	Hop 5	Hop 6
Proposed DM method	Received	684	602	462	439	405
	Lost	43	126	262	289	321
	Packet loss rate	5.91%	17.30%	36.18%	39.69%	44.21%
Standard CoAP method	Received	2020	1704	1173	1112	944
	Lost	161	474	1003	1068	1234
	Packet loss rate	7.38%	21.76%	46.09%	48.9%	56.65%

D. IoT application via the cloud platform

To further validate the applicability of the proposed DM method, we integrate it into our prototype IoT cloud system by connecting senor devices via the cloud platform using the proposed CoAP based management protocol. The snapshot of the management portal is shown in Fig. 7 (a). Through the pre-defined CoAP APIs, interactions with application data can be easily managed and retrieved in a unified manner without remembering all string URIs.

Consider the limited resource of sensor devices, diverse contextual data need to be uploaded to the IoT cloud platform for further processing. Such data collected from independent IoT sources often have implicit but disparate assumptions of interpretation. We use a lightweight ontology which contains a modifier using to capture additional information that affects the interpretations of generic concepts. More details about the setting of the cloud platform, e.g., the BPEL engine presented in Fig. 7, can be found from our previous work [44].

We evaluate the system performance of the IoT cloud in terms of time efficiency by setting up a simple test environment in which 5 sensor devices are used to upload computing tasks to the cloud platform with a total average rate of E $= 5/$min. The OpenGALEN ontology [45] is adopted as benchmark, and the computing tasks are to index and calculate the similarities of concepts on this ontology under the condition of four different size assertions (1000, 1500, 2000, 36000). The average results are shown in Fig. 7 (b). The time delay when performing the task via cloud consists of: 1) response and communication time between the remote IoT cloud platform and the sensor device; and 2) processing time of the task. The results show closed performance of response time with an average of 4.5s, while the process time mostly depends on the size of the data set. It is worth noting that depends on specific scenarios of IoT applications and computing capacities of IoT devices, we could choose different size of dataset for real-world deployments.

VII. CONCLUSION AND FUTURE WORK

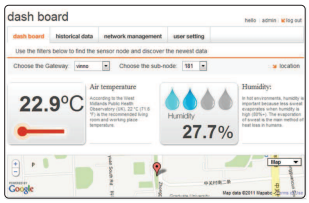
We have proposed a CoAP based DM solution and developed a prototype system to implement some basic functions on IPv6 sensor nodes. By integrating IEEE 802.15.4 connectivity and HTTP-CoAP proxy into an open-platform gateway, the remote access and management from an Internet device to an IPv6 sensor device can be realized in a unified manner. Through the performance evaluations, we have shown the

(b) Overall system performance of the IoT cloud

Response

Average time delay (ms)		Data set 1 (1000)	Data set 2 (1500)	Data set 3 (2000)	Data set 4 (36000)
Cloud platform	Response time	4683	4475	4626	4395
	Process time	40	461	702	2483
Smart phone platform	Total time	2234	4736	7445	136073

Big data uploading & analysis



(a) An illustration of management web portal

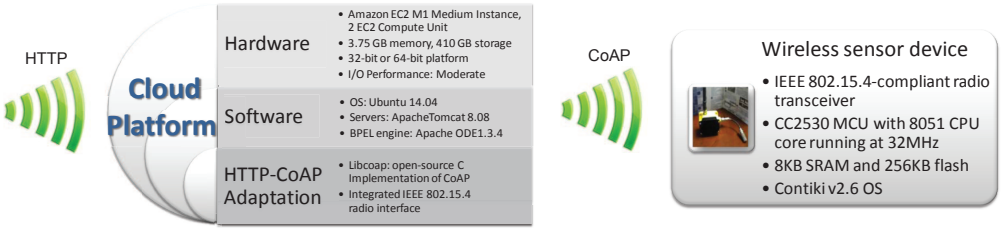


Fig. 7. User-cloud-sensor interactions in the proposed IoT system and its performance

simplicity and efficiency of the proposed solution, which is promising to drive IoT development.

In the future work, a more robust and reliable device management system for IoT needs to be built. Especially, the following research issues need to be considered with higher priorities: 1) *Real-time management* is a challenging issue for resource constrained sensor networks. The requirements of real-time communication comprises in general of response time in the required range and a small need of device resources, e.g., processor load and memory use [46]. The proposed CoAP based method can potentially provide a lightweight solution to cope with stringent industrial applications. 2) *Security, trust and privacy* [47] are also important issues to be considered in practical applications. In our case, the CoAP based management principle can utilize the transport layer bindings of UDP or SMS protocols. Thus, the security mechanisms of these channel bindings can be utilized to implement access control and policy enforcement for IoT systems. 3) *Dynamic registration, bootstrap and management* will be particularly considered for a large scale deployment with devices coming in and out and changing their characteristics and functionalities.

REFERENCES

[1] K. Ashton, "That 'Internet of Things' Thing," *RFID Journal*, July 2009.

[2] ZigBee Alliance, "ZigBee home automation public application profile," *IEEE J. Select. Areas Commun.*, Oct. 2007.

[3] A.-B. García-Hernando, J.-F. Martínez-Ortega, J.-M. López-Navarro, A. Prayati, and A. P. L. Redondo-López, "Problem solving for wireless sensor networks," *Springer*, Jul. 2008.

[4] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, and M. Nixon, "WirelessHART: Applying wireless technology in real-time industrial process control," in *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS).*, April 2008, pp. 377–386.

[5] J. Kjellsson, A. Vallestad, R. Steigmann, and D. Dzung, "Integration of a wireless I/O interface for PROFIBUS and PROFINET for factory automation," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 4279–4287, Oct 2009.

[6] I. S. O. A. (IPSO), *Available at: http://www.ipso-alliance.org*.

[7] J. Jasperneite and J. Feld, "PROFINET: an integration platform for heterogeneous industrial communication systems," in *Proc. 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, Sept 2005, pp. 8 pp.–822.

[8] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 packets over IEEE 802.15.4 networks," in *Internet Engineering Task Force, RFC 4944*, 2007.

[9] T. W. (Ed.), P. T. (Ed.), A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 routing protocol for low-power and lossy networks," in *IETF, RFC 6550*, 2012.

[10] Z. Shelby, K. Hartke, and C. Bormann, "Constrained application protocol (CoAP)," *Internect Draft, Available at: http://datatracker.ietf.org/wg/core/charter*.

[11] J. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, Feb 2014.

[12] J. Song, A. Kunz, M. Schmidt, and P. Szczytowski, "Connecting and managing M2M devices in the future internet," *Mobile Networks and Applications, Springer*, vol. 19, pp. 4–17, 2014.

[13] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common m2m service layer platform: Introduction to onem2m," *IEEE Wireless Commun. Mag.*, vol. 21, no. 3, pp. 20–26, June 2014.

[14] IoT-Architecture, *http://www.iot-a.eu/public*.

[15] H. Rachidi and A. Karmouch, "A framework for self-configuring devices using TR-069," in *Proc. Int'l Conf. on Multimedia Computing and Systems (ICMCS)*, Apr. 2011, pp. 1–6.

[16] M. Ruta, F. Scioscia, E. Di Sciascio, and G. Loseto, "Semantic-based enhancement of ISO/IEC 14543-3 EIB/KNX standard for building automation," *IEEE Trans. Ind. Informat.*, vol. 7, pp. 731–739, 2011.

[17] S. Yugang and W. Hong, "Research on field device integration model," in *Proc. International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, vol. 3, March 2010, pp. 79–82.

[18] R. T. Fielding, "Architectural styles and the design of network-based software architectures," *Doctoral dissertation, University of California, Irvine, USA*, 2000.

[19] R. Kyusakov, J. Eliasson, J. Delsing, J. van Deventer, and J. Gustafsson, "Integration of wireless sensor and actuator nodes with IT infrastructure using service-oriented architecture," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 43–51, 2013.

[20] T. Cucinotta, A. Mancina, G. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, and F. Rusina, "A real-time service-oriented architecture for industrial automation," *IEEE Trans. Ind. Informat.*, vol. 5, no. 3, pp. 267–277, 2009.

[21] C. Pautasso, O. Zimmermann, and F. Leymann, "RESTful web services vs. "big" web services: Making the right architectural decision," in *Proc. 17th Int'l conf. on World Wide Web (WWW)*, Apr. 2008.

[22] W. Qin, Q. Li, L. Sun, H. Zhu, and Y. Liu, "RestThing: A RESTful web service infrastructure for mash-up physical and web resources," in *Proc. IFIP 9th Int'l Conf. on Embedded and Ubiquitous Computing (EUC)*, Oct. 2011, pp. 197 –204.

[23] V. Stirbu, "Towards a RESTful plug and play experience in the web of things," in *Proc. IEEE Int'l Conf. on Semantic Computing*, Aug. 2008, pp. 512 –517.

[24] D. Guinard and V. Trifa, "Towards the web of things: Web mashups for embedded devices," in *Proc. of Int'l World Wide Web Conference (WWW)*, Apr. 2009.

[25] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *Proc. Internet of Things (IoT), 2010*, Dec. 2010.

[26] J.-P. Vasseur and A. Dunkels, "Interconnecting smart objects with IP: The next internet," *Morgan Kaufmann*, 2010.

[27] C. Hennebert and J. Dos Santos, "Security protocols and privacy issues into 6lowpan stack: A synthesis," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 384–398, Oct 2014.

[28] Z. Shelby, "Embedded web services," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 52–57, 2010.

[29] T. Potsch, K. Kuladinithi, M. Becker, P. Trenkamp, and C. Goerg, "Performance evaluation of CoAP using RPL and LPL in TinyOS," in *Proc. 5th Int'l Conf. on New Technologies, Mobility and Security (NTMS)*, May 2012, pp. 1 –5.

[30] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proc. 29th IEEE Int'l Conf. on Local Computer Networks*, Nov. 2004, pp. 455 – 462.

[31] M. Kovatsch, S. Duquennoy, and A. Dunkels, "A low-power CoAP for Contiki," in *Proc. IEEE 8th Int'l Conf. on Mobile Adhoc and Sensor Systems (MASS)*, Oct. 2011, pp. 855 –860.

[32] J. Schönwölder, T. Tsou, and B. Sarikaya, "Protocol profiles for constrained devices," March 2011.

[33] O. Bergmann, K. Hillmann, and S. Gerdes, "A CoAP-gateway for smart homes," in *Proc. Int'l Conf. on Computing, Networking and Communications (ICNC)*, Feb. 2012, pp. 446–450.

[34] G. Moritz, F. Golatowski, C. Lerche, and D. Timmermann, "Beyond 6LoWPAN: Web services in wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 1795–1805, Nov 2013.

[35] L. Frye and L. Cheng, "A network management system for a heterogeneous, multi-tier network," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, Dec. 2010, pp. 1–5.

[36] M. Yang, S. S. So, S. Eun, B. Kim, and J. Kim, "Sensos: A sensor node operating system with a device management scheme for sensor nodes," in *Proc. 4th Int'l Conf. on Information Technology*, April 2007, pp. 134–139.

[37] A. Hergenroeder, J. Wilke, and D. Meier, "Distributed energy measurements in WSN testbeds with a sensor node management device (SNMD)," in *Proc. 23rd Int'l Conf. on Architecture of Computing Systems (ARCS)*, Feb. 2010.

[38] M. Jung, J. Weidinger, W. Kastner, and A. Olivieri, "Building automation and smart cities: An integration approach based on a service-oriented architecture," in *Proc. 27th Int'l Conf. on Advanced Information Networking and Applications Workshops (WAINA)*, March 2013, pp. 1361–1367.

[39] N. Gligoric, S. Krco, D. Drajic, S. Jokic, and B. Jakovljevic, "M2M device management in LTE networks," in *Proc. 19th Telecommunications Forum (TELFOR)*, Nov. 2011, pp. 414–417.

[40] P. van der Stok and B. Greevenbosch, "CoAp management interfaces (draft-vanderstok-core-comi-04)," in *IETF, available at: https://datatracker.ietf.org/doc/draft-vanderstok-core-comi/*, 2014.

[41] I. C. Society, "IEEE Std. 802.15.4-2003," 2003.

[42] Z. Shelby, "Constrained RESTful environments (CoRE) link format," *RFC 6690, available at: http://tools.ietf.org/html/rfc6690*.

[43] O. Bergmann, "libcoap: C-implementation of CoAP," *Available at: http://libcoap.sourceforge.net*.

[44] X. Hu, T. Chu, H. Chan, and V. Leung, "Vita: A crowdsensing-oriented mobile cyber-physical system," *IEEE Trans. Emerging Topics in Computing*, vol. 1, no. 1, pp. 148–165, June 2013.

[45] A. Rector, J. Roger, P. Zanstor, and E. Haring, "OpenGALEN: open source medical terminology and tools," *American Medical Informatics Association*, 2003.

[46] J. Feld, "PROFINET - scalable factory communication for all applications," in *Proc. IEEE International Workshop on Factory Communication Systems*, Sept 2004, pp. 33–38.

[47] S. L. Keoh, S. Kumar, and H. Tschofenig, "Securing the internet of things: A standardization perspective," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 265–275, June 2014.