International Journal of Control and Automation
Vol. 9, No. 5 (2016), pp. 151-160
http://dx.doi.org/10.14257/ijca.2016.9.5.15

Design and Implementation of Node Discovery and Registration Based on RD Using IETF CoAP Protocol in IoT Environment

Wen-Quan JIN[1], Yong-Geun Hong[2], And Dohyeun Kim*[1]

[1]*Department of Computer Engineering Jeju National University, Jeju-Si, Republic of Korea*
[2] *ETRI, 218 Gajeong-ro Yuseung-Gu, Daejeon, Republic of Korea*
E-mail : pluskmk12@live.com, yghong@etri.re.kr and kimdh@jejunu.ac.kr
** Corresponding author: DoHyeun Kim, kimdh@jejunu.ac.kr*

Abstract

Recently, IETF (Internet Engineering Task Force) presented CoAP (Constrained Application Protocol) for the communication between sensor and actuator nodes in a constrained environment such as small amount of memory and low power. IETF CoAP protocol can convert easily, and can use to monitor or control infrastructure utilities through low-power sensor and actuator networks in IoT (Internet of Things) and M2M (Machine-to-Machine) environment. IETF CoAP protocol provides the special requirements of this constrained environment, especially considering energy, automation, and other IoT applications. In this paper, we design and implement a node registration and discovering based CoAP in IoT environment. The node is a CoAP node which is designed for working in constrained environment. For node registration in IoT, we used CoRE RD (Resource Directory) functionalities to register CoAP node's information which works in IoT environment.

***Keywords**: IoT, CoRE, Resource Directory, CoAP, M2M*

1. Introduction

IETF CoRE (Constrained RESTful environments) Working Group started global standardization for CoAP (Constrained Application Protocol) protocol in 2010, and recently it announces RFC (Request for Comments) 7252 [1]. Therefore, researchers studying and developing to realize CoAP in limited sensor or actuator network. CoAP is a specialized web transfer protocol to use with constrained nodes and constrained networks. The protocol is used for M2M applications such as smart energy and building automation. M2M interactions typically result in a CoAP implementation acting in both client and server roles. It is designed as a framework for resource-oriented applications intended to run on constrained IP networks which support RESTful architectural style, and POST, GET, PUT and DELETE. It is stateless and exposes directory structure-like URIs and defined mappings to compact binary forms and transport over UDP. We host a CoAP endpoint to a node. The node is a device that may be a board with sensors or actuators. The node can be implemented by acting in both client and server roles. The node must send a request to a server to register information as a client. As the sever role, the node receives requests from clients and process it. It may response a result for the request.

IETF CoRE aims at realizing group has studied CoAP protocol based on the REST architecture for the constrained nodes and networks. IETF CoAP supports the devices constrained in terms of memory, processing and power *i.e.* small low power sensors, switches and valves etc. CoAP uses a request/response interaction model between

*Corresponding Author: DoHyeun Kim, *kimdh@jejunu.ac.kr* application nodes, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and internet media types. CoAP is designed to easily interface with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead, and simplicity for constrained environments. CoAP allows these devices to interact and communicate over the Internet. There are two entities required for CoAP communication *i.e.* CoAP client and CoAP server. A CoAP server may also act as a client and vice versa, if both of these entities have resources to share and request certain resources from each other [1].

The CoAP server discovers RD (Resource Directory). RD hosts descriptions of resources held on other servers, allowing lookups to be performed for hose resources. The discovery of RD means finding location of the register function set in the RD using a CoAP server which may register the resources whatever it wants to share. Once a complete path is obtained for a registration function set in the RD, the CoAP server may then register resources to the RD. The CoAP clients then requests the RD to look up for registered resources. The RD then returns the access paths for the registered resources according to the request of the client. The returned resources may include simple or composite resources and the client can communicate with these resources [2]. Until now there is no design to implement the node registration based on RD using CoAP protocol in detail.

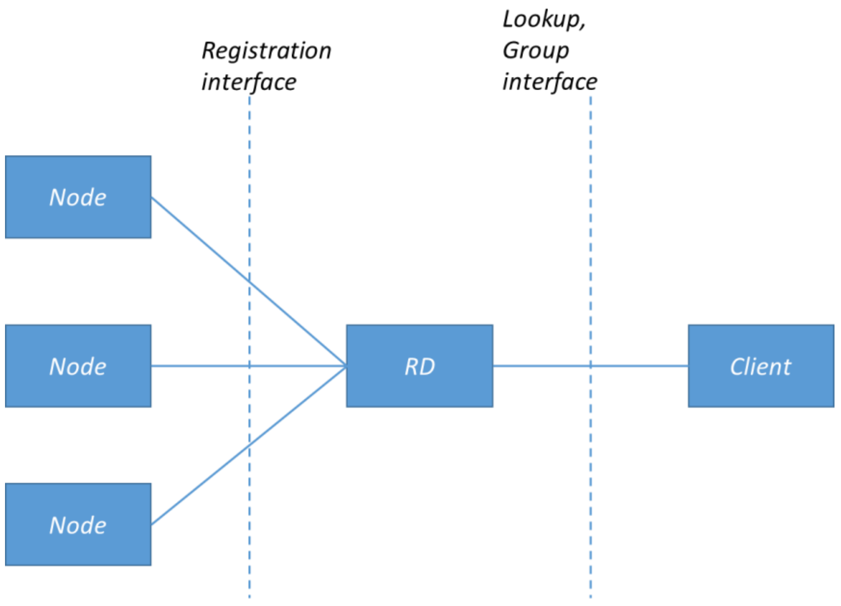


Figure 1. The Resource Directory Architecture

A node registration mechanism is important for the efficiency connectivity of IoT systems. A node in IoT, may not act always. There should be an entity to record the state of nodes to make clients know which node is active. These problems can be solved by employing an entity called a RD, which hosts descriptions of resources held on other servers, allowing lookups to be performed for those resources [2]. In this paper, we present design and implement the node registration and discovery mechanism based on IETF CoAP protocol using RD in IoT networks. We present software designs and implementation result for a node registration and discovery using RD. This mechanism provides a connectivity between a node and a client on Internet. And CoAP nodes communicate with RD using CoAP. First, in Section 2, we present a design of node registration based on CoAP. The node is designed to register based on RD. Section 3 explains how we implement the designed registration based on RD using CoAP protocol.

2. Node Registration and Discover Based on RD using IETF CoAP

A RD is used as a repository of resources for nodes for Web Links on other web servers. Figure 1 illustrates the conceptual RD architecture for registration and lookup of those resources. A node is a web server associated with a scheme, IP address and port. Nodes are assumed to proactively register and maintain resource directory entries on the RD, which are soft state and need to be periodically refreshed. An endpoint is provided with interfaces to register, update and remove a resource directory entry. Furthermore, a mechanism to discover an RD using the CoRE Link Format is defined. It is also possible for an RD to proactively discover Web Links from nodes and add them as resource directory entries. A lookup interface for discovering any of the Web Links held in the RD is provided using the CoRE Link Format. The CoAP node communicates with RD through CoAP. RD and client can communicate through CoAP or HTTP. If HTTP is selected in the communication, the RD has to implements CoAP and HTTP [2].

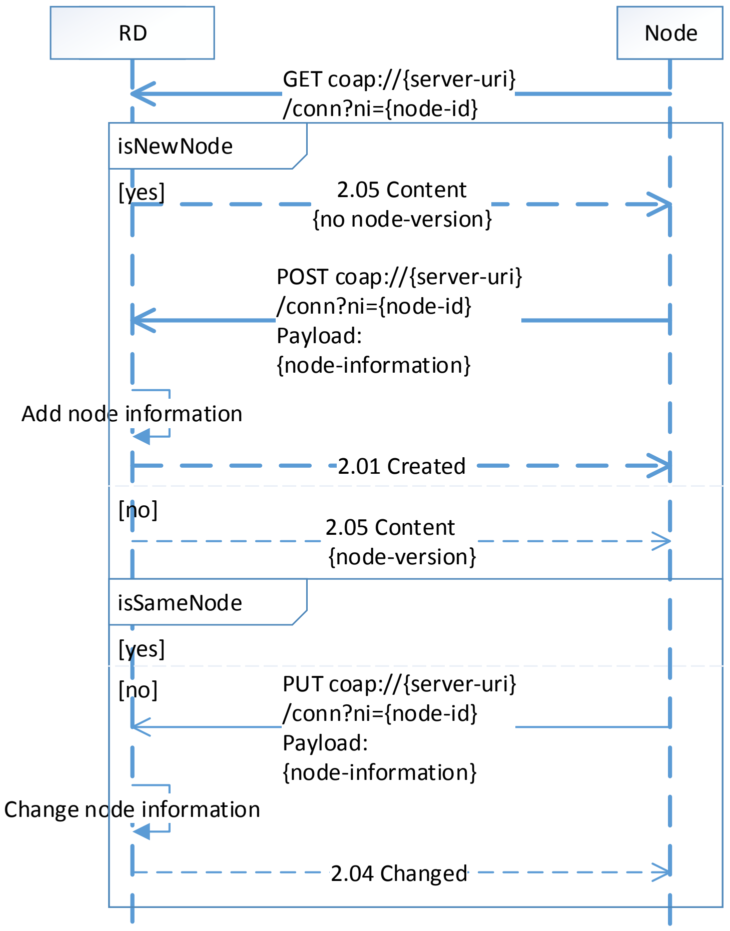

RD
Node
GET coap://{server-uri}/conn?ni={node-id}
isNewNode
[yes]
2.05 Content {no node-version}
POST coap://{server-uri}/conn?ni={node-id} Payload: {node-information}
Add node information
2.01 Created
[no]
2.05 Content {node-version}
isSameNode
[yes]
[no]
PUT coap://{server-uri}/conn?ni={node-id} Payload: {node-information}
Change node information
2.04 Changed

Figure 2. Sequence Diagram for Discover and Registration of a Node

We assume that a node know the RD's IP address, port and the path of its RD at first. Then it can make use of an RD for discovering the RD. A node sends either a multicast or unicast to GET request message for discovery.

Next, a node perform to register its resources using the registration interface after discovering the location of an RD using POST request message. The RD accepts a POST request message from a node containing the list of resources to be added to the directory as the message payload. The list of resources of a node stores in the database of RD.

Figure 2 shows the sequence diagram for process of a node registration and discovery. First, a node sends GET request with “ni” parameter to “conn” resource of a RD. The parameter is a node ID which is used for retrieving a node in the database of a RD. If it is confirmed through a node ID, then RD responds to a node version, otherwise RD responds a string to notify the node that there is no node information. After the node receives the response, it sends POST request to the RD with “ni” and a payload which includes a node’s information to register as a new node.

When a node registers itself to the RD server, the registration request should contain its node identifier. A node ID can be used to identify the node [3]. This node identifier may be included in the node ID option in the registration request, or may be included in the URI-Query option in the CoAP message. The value must be unique for each node within a RD server. The value can be in the form of binary bits, IMEI (International Mobile Equipment Identity number), IEEE 802 MAC Address, or other identifiers which can uniquely identify its identity.

A single node integrates multiple units to enable their own functions such as sensors or actuators. The unit ID in the CoAP node enables the usage of composite nodes consisting of multiple sensors and actuators while having a single IP address for communication [4]. The integrated resources can be individually or collectively communicated with and/or controlled using CoAP messages.

```
{
    "id" : "{node-id}",
    "node_version" : "{node-version}",
    "node_uri" : "{node-uri}",
    "mw_uri" : "{middleware-uri}",
    "sleep_state" : "{0|1}",
    "sleep_duration" : "{0|1}",
    "notify_enable" : "{0|1}",
    "notify_interval" : "{0|1}",
    "units" : [{
            "id" : "{unit-id}",
            "resource_type" : "{resource-type}",
            "unit_interface" : "{unit-interface}",
            "unit_state" : "{0|1}",
            "recording_interval" : "{0|1}"
            "changed_notify_enable" : "{0|1}"
        }
    ]
}
```

Figure 3. Example of a Node’s Properties

Figure 3 shows an example of a profile for a node’s properties that we present. The information of the profile is formatted in JSON. It includes node’s information and unit’s information which are included in a node’s application. Node version is used for synchronization the information between node and RD. Other attributes used for functionality of the node are sleep state, sleep duration, notify enable and notify interval. There can be multiple units in a node. The information of a unit can be appropriately different without id of the unit. Information of a unit includes a resource type and interface which are proposed in the RFC 6690 [5]. The resource type attribute is an opaque string used to assign an application-specific semantic type to a resource. The interface description attribute is an opaque string used to provide a name or URI indicating a specific interface definition used to interact with the target resource.

Each node has its own URI with an IP address, but it is quite common for a node to change its IP address due to rebooting. We design the node as seamless to synchronize its information with RD. A client can retrieve a node URI via a node ID to find a node in a network. For example, a client need to send a command to actuate a unit of a node, then the client will get a node ID which is related to its requirement. The RD is registered by the node. The client requests a command to the RD with node ID and unit ID which are the parameters of the query. The RD will forward the command to the node with unit ID via node URI, the node URI is retrieved using node ID from database in RD server. When an RD retrieves information of a node through the node ID, it also checks the state of the node through the node ID. In the case, if the node's state is sleep or power off, the command will not be sent by RD to the node.

3. Implementation of Node Registration and Discovery

In the CoAP communication based IoT environment, nodes have automatic registration process in the software. A node connects to RD through configured RD server's URI to register its information or update it. Information of a node in the database of RD server may be inserted by an administrator in advance. So, for this case, the processes are used for updating node's information or do nothing. A node synchronizes information of the node with RD via the version attribute of the node's profile.

We via a Java framework implemented RD server, and via a C library implemented the node software. The node is implemented in Linux C compile environment because of the node working in a constrained environment. RD server is implemented in a java compile environment. The RD is a web application which supports a web link to be accessed by users such as web clients and HTTP clients. We implement the CoAP network in different environments, and have verified the interoperability. Table 3.1 shows the IDEs for implementing the CoAP network. The RD is made with Java Runtime Environment (JRE) in the Window OS. We used eclipse as a development tool. The node is made with C language in the Ubuntu Linux environment and executed in GCC. The CoAP is an application layer protocol and it can be included in a program. A CoAP client sends message to a CoAP server through the CoAP and the server parses the received message to execute the requested function.

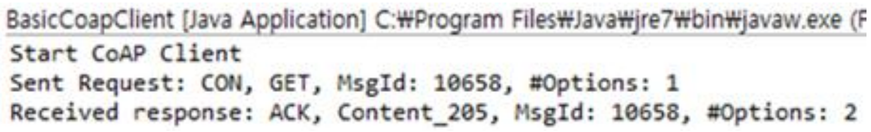
Table 1. Implementation Environment for Coap Network

Development environment		
Environment	RD	node
OS	Windows 8.1	Ubuntu 12.4 (Linux)
Run environment	Java 7	gcc
Tool	Eclipse	gedit
Language	Java	c

To implement the CoAP in C compiler GCC based Linux environment and JRE we used two libraries. In JRE, RD is implemented by Californium (Cf) CoAP framework which is based on the Java language, and the node software which runs on C environment, that is implemented by Libcoap C implementation library. It is based on the C language. The Cf was successfully tested at the ETSI IoT CoAP Plugtests [6]. The license is "3-clause BSD" [7]. The Libcoap is potting in TinyOS and the license is "BSD".

The CoAP protocol is UDP upper protocol, and when a response message is not reached to a server in the pre-defined time it sends message again. Repeat this again 5 times and if it is not received any response message then it means that the communication is failed.

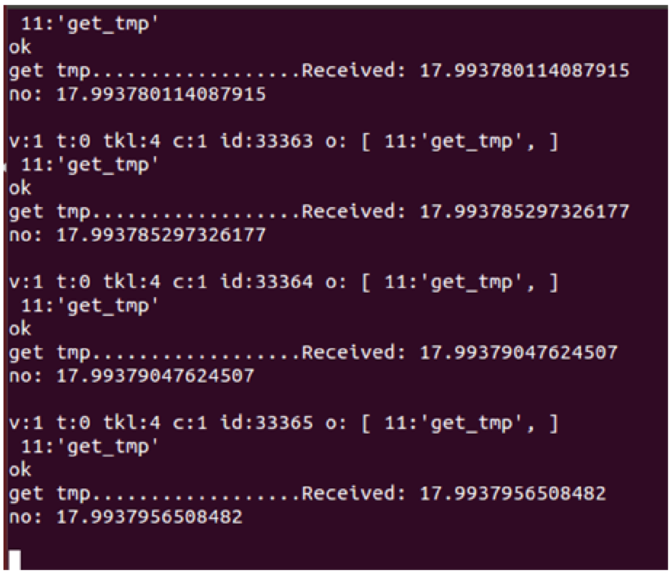
Figure 4 shows the result of a CoAP communication. When a CoAP client sends requirement message to a CoAP server via the CoAP, the CoAP server processes it and sends a message to the CoAP client. This figure shows the message data of a CoAP communication. The value “CON” is the T (Type) value of the CoAP message format, and the “GET” is the method type. IETF CoRE WG defines 4 method types for CoAP. In the “MsgId: 10658”, the “10658” means ID of message, and this value can be created automatically or developer can define it. The “#Options: 1” means the number of message attributes included in CoAP message, and basically it is “MsgId” attribute. The contents shown are presented by RD.



```
BasicCoapClient [Java Application] C:₩Program Files₩Java₩jre7₩bin₩javaw.exe (F
Start CoAP Client
Sent Request: CON, GET, MsgId: 10658, #Options: 1
Received response: ACK, Content_205, MsgId: 10658, #Options: 2
```

Figure 4. Experiment Results of CoAP.

Figure 5 shows experiment screen of node emulator. This program is executed with a console in Linux environment, and receives a message from CoAP based entity such as RD.

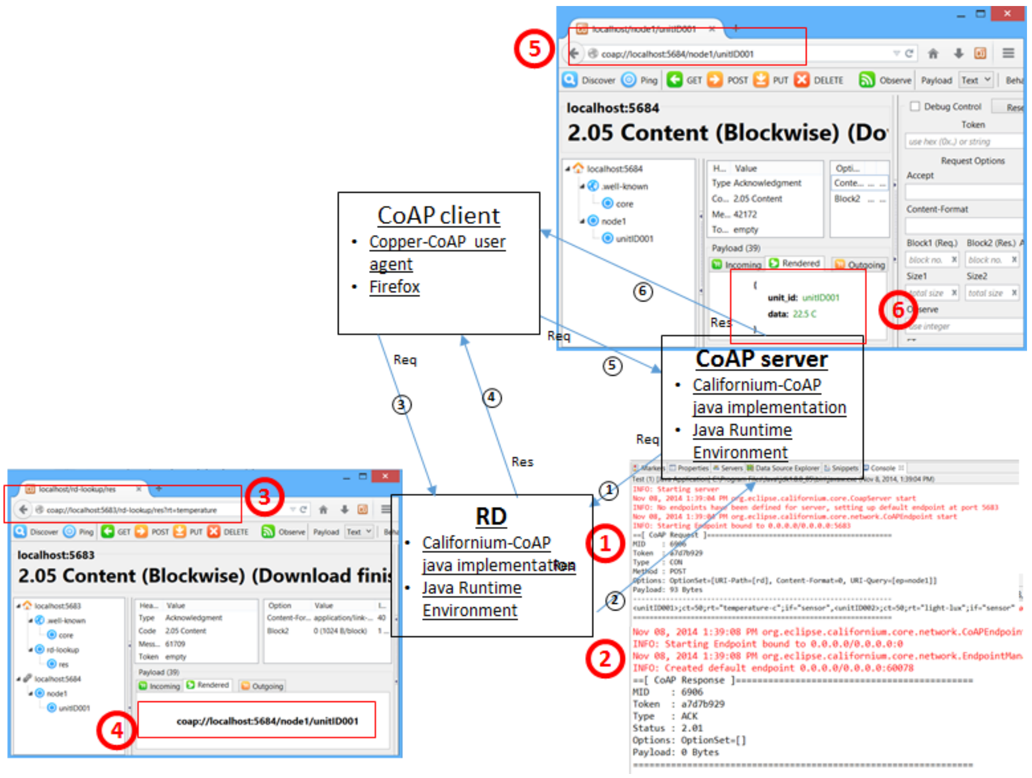


```
 11:'get_tmp'
ok
get tmp.................Received: 17.993780114087915
no: 17.993780114087915

v:1 t:0 tkl:4 c:1 id:33363 o: [ 11:'get_tmp', ]
 11:'get_tmp'
ok
get tmp.................Received: 17.993785297326177
no: 17.993785297326177

v:1 t:0 tkl:4 c:1 id:33364 o: [ 11:'get_tmp', ]
 11:'get_tmp'
ok
get tmp.................Received: 17.99379047624507
no: 17.99379047624507

v:1 t:0 tkl:4 c:1 id:33365 o: [ 11:'get_tmp', ]
 11:'get_tmp'
ok
get tmp.................Received: 17.9937956508482
no: 17.9937956508482
```

Figure 5. Experiment Results of Coap Server.

The CoAP based node includes units; these units are identified by unit id. A node also has node ID for identification. The implementation for the RD functionalities is registered node which we proposed and discover. For this mechanism, we presented a prototype to show the CoAP interaction between each element in the IoT environment. In the interaction, there are RD, CoAP client and CoAP server which enable registration and discovery of CoAP node. As shown in Figure 6, the RD and CoAP server used Cf framework to implements and run in Java Runtime Environment. CoAP client uses Copper-CoAP user agent to interact with CoAP server and RD via Firefox (Web browser) [8]. According to the mechanism, RD server includes CoAP server and RD, and the node includes CoAP client.

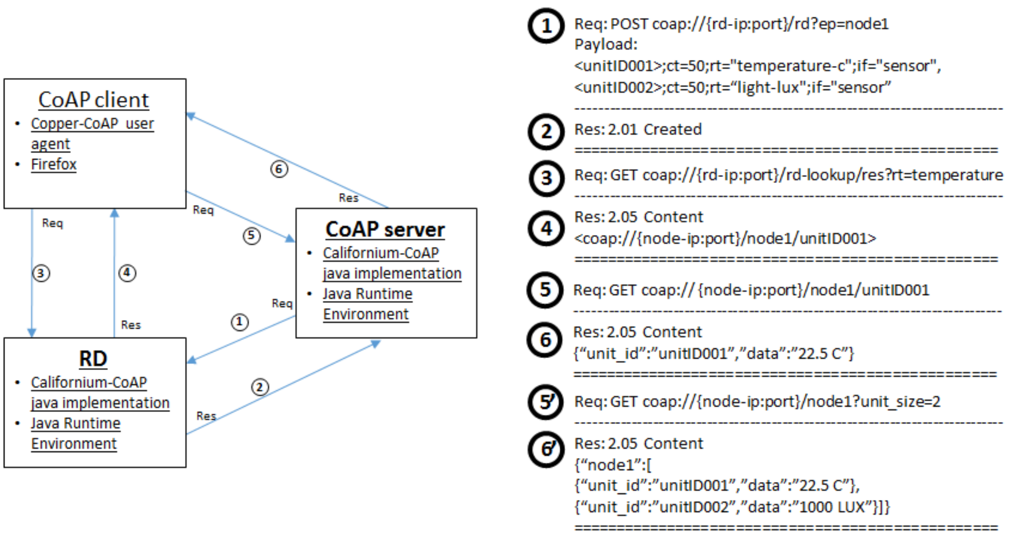
Figure 7 shows requests and responses for the interaction between RD, CoAP client and CoAP server. The process shows from a registration process to acquired contextual data process sequentially. The presented data are results of each action in a CoAP based IoT environment.


5
coap://localhost:5684/node1/unitID001
localhost:5684
2.05 Content (Blockwise) (Do
unit_id: unitID001
data: 22.5 C
6
CoAP client
Copper-CoAP user agent
Firefox
Req
Res
CoAP server
Californium-CoAP java implementation
Java Runtime Environment
RD
Californium-CoAP java implementation
Java Runtime Environment
3
coap://localhost:5683/rd-lookup/res?rt=temperature
localhost:5683
2.05 Content (Blockwise) (Download finis
4
coap://localhost:5684/node1/unitID001
1
2

```
INFO: Starting server
Nov 08, 2014 1:39:04 PM org.eclipse.californium.core.CoapServer start
INFO: No endpoints have been defined for server, setting up default endpoint at port 5683
Nov 08, 2014 1:39:04 PM org.eclipse.californium.core.network.CoAPEndpoint start
INFO: Starting Endpoint bound to 0.0.0.0/0.0.0.0:5683
==[ CoAP Request ]============================================
MID    : 6906
Token  : a7d7b929
Type   : CON
Method : POST
Options: OptionSet=[URI-Path=[rd], Content-Format=0, URI-Query=[ep=node1]]
Payload: 93 Bytes
---------------------------------------------------------------
<unitID001>;ct=50;rt="temperature-c";if="sensor",<unitID002>;ct=50;rt="light-lux";if="sensor"
===============================================================
Nov 08, 2014 1:39:08 PM org.eclipse.californium.core.network.CoAPEndpoin
INFO: Starting Endpoint bound to 0.0.0.0/0.0.0.0:0
Nov 08, 2014 1:39:08 PM org.eclipse.californium.core.network.EndpointMan
INFO: Created default endpoint 0.0.0.0/0.0.0.0:60078
==[ CoAP Response ]============================================
MID    : 6906
Token  : a7d7b929
Type   : ACK
Status : 2.01
Options: OptionSet=[]
Payload: 0 Bytes
================================================================
```

Figure 6. Prototype Development for Coap Entities

First, it shows request and response to a node uses an RD's registration function set and sends a CoAP POST message to the RD. The RD receives a valid request from the node, the source IP address and Port number from the CoAP request parameters or the message source address portion (default). The RD then extracts the Unit IDs from the message payload and save the information and returns a response message to the node. It shows a client requesting for a specific type of resource (Temperature) registered with the RD. For this purpose, it sends GET request to the RD with the Resource Type (rt). The RD receives the message, checks if the message is a valid CoAP request or not and then gets the IDs for all the registered resources with the resource type value equivalent to the one requested by the CoAP client (Temperature). The RD then creates a response message with the list of node IP address and resource IDs and sends it to the client. The client may then choose a specific resource from this list and communicate with it directly using the CoAP. Finally, CoAP client sends request message to CoAP server to acquire contextual data through the URI which is selected in the resource list.



```
①  Req: POST coap://{rd-ip:port}/rd?ep=node1
    Payload:
    <unitID001>;ct=50;rt="temperature-c";if="sensor",
    <unitID002>;ct=50;rt="light-lux";if="sensor"
    --------------------------------------------------------------
②  Res: 2.01 Created
    ==============================================
③  Req: GET coap://{rd-ip:port}/rd-lookup/res?rt=temperature
    --------------------------------------------------------------
④  Res: 2.05 Content
    <coap://{node-ip:port}/node1/unitID001>
    ==============================================
⑤  Req: GET coap:// {node-ip:port}/node1/unitID001
    --------------------------------------------------------------
⑥  Res: 2.05 Content
    {"unit_id":"unitID001","data":"22.5 C"}
    ==============================================
5' Req: GET coap://{node-ip:port}/node1?unit_size=2
    --------------------------------------------------------------
6' Res: 2.05 Content
    {"node1":[
    {"unit_id":"unitID001","data":"22.5 C"},
    {"unit_id":"unitID002","data":"1000 LUX"}]}
    ==============================================
```

Figure 7. Development Of Coap Entities in an Iot Environment

4. Conclusion

IETF presented CoAP for constrained environment in an IoT environment and there are several extensions for CoAP. For CoAP based node, there is an identification of node and identification of multiple units in a node. In this paper, we presented a design and implement for registration and discovering of a node based CoAP using RD functionalities which considered the identification mechanism in IoT environment. The registration and discovering is an implemented system which includes node, RD and client based on CoAP. The registration and discovering of a node introduce are implemented using Cf framework and Libcoap library.

Acknowledgments

This work was supported by ICT R&D program of MSIP/IITP. [R0166-15-1008, Internet based IoT Interworking Technology Standards Development]. And this work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.B0126-15-1078, Creation of PEP based on automatic protocol behavior analysis and Resource management for hyper connected for IoT Services). Corresponding author; DoHyeun Kim (e-mail: kimdh@jejunu.ac.kr).

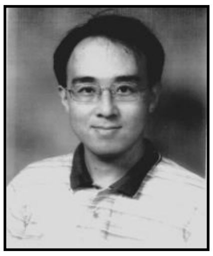
References

[1] A. Rahman, "Enhanced Sleepy Node Support for CoAP", Internet-Draft, draft-rahman-core-sleepy-05, February **(2014)**.
[2] Z. Shelby and C. Bormann, "CoRE Resource Directory", Internet-Draft, draft-ietf-core-resource-directory-01, December **(2013)**.
[3] K. Li and G. Wei, "CoAP Option Extension: NodeId", Internet-Draft, draft-li-core-coap-node-id-option-01, June **(2014)**.
[4] Y. Hong, Y. Choi, D. Kim, M. Khan and W. Jin, "CoAP Endpoint Unit Identification for Multiple Sensor and Actuator in a Node", Internet-Draft, draft-hong-core-coap-endpoint-unit-id-00, July **(2014)**.
[5] Z. Shelby, "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, August **(2012)**.
[6] libcoap: C-Implementation of CoAP, <http://libcoap.sourceforge.net/>.
[7] Californium (Cf) CoAP framework in Java, <http://people.inf.ethz.ch/mkovatsc /californium.php>.
[8] Copper (Cu), <https://addons.mozilla.org/en-US/firefox/addon/copper-270430/>.

Authors

Wen-Quan JIN, he received the B.S degree in computer science and technology from Yanbian university of science and technology, China, in 2013, and the M.S degree in computer engineering in Jeju national university in 2015.

Yong-Geun Hong, he is a special fellow at Protocol Engineering Center in ETRI (Electronics and Telecommunications Research Institute). He is a technical leader of Internet of Things (IoT) and Machine-to-Machine (M2M) standardization projects in ETRI. He received his B.S., M.S. and Ph.D in computer engineering from the Kyoungpook National University, Daegu, Korea. Previously, he was an engineer works for MJL Technology. At there, he researched and implemented the UMS (Unified Messaging System). He joined ETRI in 2001 and since then, he has been working on Internet protocol. Now, he is working for the IoT related standardization at IETF (Internet Engineering Task Force). His research interests include IPv6, Internet Mobility, Internet QoS, M2M, and IoT.



Do-Hyeun Kim, received the B.S. degree in electronics engineering from the Kyungpook National University, Korea, in 1988, and the M.S. and Ph.D. degrees in information telecommunication the Kyungpook National University, Korea, in 1990 and 2000, respectively. He joined the Agency of Defense Development (ADD), from Match 1990 to April 1995. Since 2004, he has been with the Jeju National University, Korea, where he is currently a Professor of Department of Computer Engineering. From 2008 to 2009, he has been at the Queensland University of Technology, Australia, as a visiting researcher. His research interests include sensor networks, M2M/IOT, energy optimization and prediction, intelligent service, and mobile computing.