DNS Naming Services for Service Discovery and Remote Control for Internet-of-Things Devices

Seokhwa Kim*, Keuntae Lee*, and Jaehoon (Paul) Jeong†
*Department of Computer Science & Engineering, Sungkyunkwan University, Republic of Korea
†Department of Interaction Science, Sungkyunkwan University, Republic of Korea
Email: {seokhwakim, keuntaelee, pauljeong}@skku.edu

***Abstract*—This paper proposes the service discovery and the remote control for Internet-of-Things (IoT) devices based on the Constrained Application Protocol (CoAP) and Domain Name System (DNS). As the number of devices increases, the manual configuration of domain names and services might be unmanageable for users. A legacy DNS Name Autoconfiguration (DNSNA) for IPv6 networks can be used to register the DNS names of IPv6 devices automatically into a DNS server. However, DNSNA is not applicable for the service discovery and remote control because of providing only for the DNS names of IoT devices. Therefore, we provide a strategy, which is suitable for IoT devices based on the CoAP. Our proposed remote control architecture is based on a standard protocol called CoAP for IoT devices, which have limited computing power. It uses using DNSNA to provide appropriate services to our remote control system. In this scheme, an IoT device can register its DNS name and service list into a DNS server. Through our remote control system, users can autoconfigure devices and their services. Thus, we can easily monitor and remote-control any IoT device using a smartphone or tablet PC.**

***Index Terms*—Internet of Things (IoT), Constrained Application Protocol (CoAP), Domain Name System (DNS), Service Discovery.**

I. Introduction

The Internet-of-Things (IoT) includes many things such as physical devices, building system, and various embedded devices. Also, the IoT devices can provides to a user with many services for convenient. According to Gartner, the number of IoT devices will be 11 billion units by 2018 [1]. However, it is very inefficient for users who have to manually identify many devices and their service list. Therefore, previously research has performed for naming services that called DNS name autoconfiguration for IoT devices in IPv6 (DNSNA) and IPv4 (DNSNAv4) [2], [3] which are proposed a naming system for IoT devices using the Domain Name System (DNS) to be matched the Internet Protocol (IP) address to identify the devices. DNSNA has been designed to provide all kinds of IoT devices with an efficient DNS naming service. However, it used just simple socket programming for monitoring or control. That is not suitable strategy to the IoT devices which is not provided sufficient computing power. Also, this is not appropriate for IoT devices with many services, because it focuses only on naming services. It is very inconvenient for users to manually identify and configure many services.

To solve this problem, we use DNS-based service discovery (DNSS-SD) [4] and a Constrained Application Protocol (CoAP) [5]. DNS-SD can search services of IoT devices. That is standard protocol proposed by IETF for various function of many devices service using DNS system (e.g, IoT devices and web server). It registered detailed information in for DNS-SD [4]. That is, we create a query to register DNS name and services in the DNS server based on [4] using a DNS naming scheme called DNSNA. In this way, users easily identify service of IoT devices.

To provide efficiently control scheme for there devices, we use the CoAP that is implemented as a binary header for devices, which have limited computing power. This protocol is proposed by Internet Engineering Task Force (IETF) standard at 2014. The CoAP can translate directly to Hypertext Transfer Protocol (HTTP) [6], which is used by most Web sites. CoAP is based on UDP and supports the HTTP methods (e.g, GET, PUT, and POST) used by website. In other words, this CoAP can control the IoT devices through the web and application environment with using users smart devices such as smartphones, smart pads or through some websites. CoAP is possible to reduce network overhead because CoAP is constructed using binary format. The CoAP client requests the message, including method. The node corresponding CoAP server, responds with some data that included temperature, status, and another one. So, users can remote control to devices using this procedure.

Consequently, we propose remote control based on CoAP as well as service discovery. The contributions of this paper are as follows:

- We propose how to identify and effectively manage IoT device services using DNS naming service in the IoT environment.
- This paper proposes the architecture for IoT devices based on CoAP in the IoT network environment.
- We implemented using the standard protocol to extend the Dynamic Host Configuration Protocol (DHCP) [7] in IPv4 networks.

The rest of this paper is organized as follows. Section II summarizes related work for DNS based IoT system. Section III describes the design and protocol of our remote control system. Section IV explains and experiment in our testbed. In Section V, we conclude this paper along with future work.

II. Related Work

In this section, we introduce related works for our research. Nowadays, the IoT is widely used for various internet services. The number of IoT devices in the network is steadily increasing. A naming system is an essential element to identify each IoT device, even though each IoT device in the network has the same make and device type. For these reasons, the IoT research has paid much attention to provide identifying each device.

Bonjour is a zero-configuration implementation that supports DNS naming services and service discovery for Apple products [8]. Also, it can support device search and service search for easy use of Apple devices. Bonjour is based on Multicast DNS (mDNS) [9], which is used by every device as both a DNS resolver and a DNS server. mDNS uses multicast to send packets to all nodes located on the local network. However, it is not suitable for multi-links with several hops. Then, the device creates a new service, multicasting to other devices. At received the message for new service, device is multicast the same scheme in a previous way. So, Bonjour can not supports situation in many device environment.

Lee et al. proposed the DNSNA scheme for IoT devices. Based on this, DNSNA has been designed to provide all kinds of IoT devices with an efficient DNS naming service. DNSNA [2] was designed to operate in a variety of IoT environment having various sensors ranging from low capacity to high capacity. It consists of the following four steps. The first step is DNS name generation. The second is DNS name collection. The third is DNS name registration. The fourth is IoT device list retrieval. DNSNA uses either the router advertisement (RA) DNS option [10] over IPv6 neighbor discovery (ND) [11] or the DNS option over Dynamic Host Configure Protocol version 6 (DHCPv6) [13] . Once it obtains a DNS Search List (DNSSL) [10], the target network device autonomously constructs its DNS name using the DNSSL and its device information, and then registers its DNS name into a DNS server via a router in the same subnet. An IoT device generates a DNS name from the DNSSL in the DNS option(s). The IoT device checks the uniqueness of the generated DNS name by performing duplicate address detection (DAD) for its IPv6 address corresponding to the DNS name in the subnet where it exists. If there was no reaction in the DAD, the IoT device regards the DNS name as unique such that the other nodes in the subnet do not use the DNS name, so it configures the DNS name as its own. Otherwise, it repeats the generation of another DNS name and the execution of the DAD until the request for the DAD is not responded by any other node. A router sends node information (NI) query to collect the pair of the DNS name and the corresponding IPv6 address of each IoT device. The router verifies that the DNS name of the collected IoT device is registered with the DNS server. However, as mentioned in the previous section, DNSNA is not suitable for services of IoT device because this system focused only DNS naming system. To apply our service discovery, we need to a new scheme for services.

Lee et al. proposed a new scheme called DNSNAv4 for currently networks environment [3].DNSNAv4 has been designed to provide efficient the naming system in currently IPv4 networks environment for IoT devices. DNSNAv4 was designed using to extend Dynamic Host Configuration Protocol (DHCP) with DNSSL option. It consists of following four steps similar to DNSNA [2]. First is IoT devices generate the DNS names using DNSSL. Second is DNS name collection. Third is DNS name registration. Forth is IoT device list retrieval. DNSNAv4 was implemented using to expand DHCP. The DNSSL of the DHCP options is sent to the IoT device. IoT devices that have received DNSSL will generate their DNS names. The rest of the process of DNSNAv4 is the same as the existing DNSNA. However, DNSNA also focused on only DNS naming system not services of IoT device. So, we need some to add functionality for IoT device services for our research.

III. Proposed Our Remote Control System

In this section, we introduce the architecture of our system.

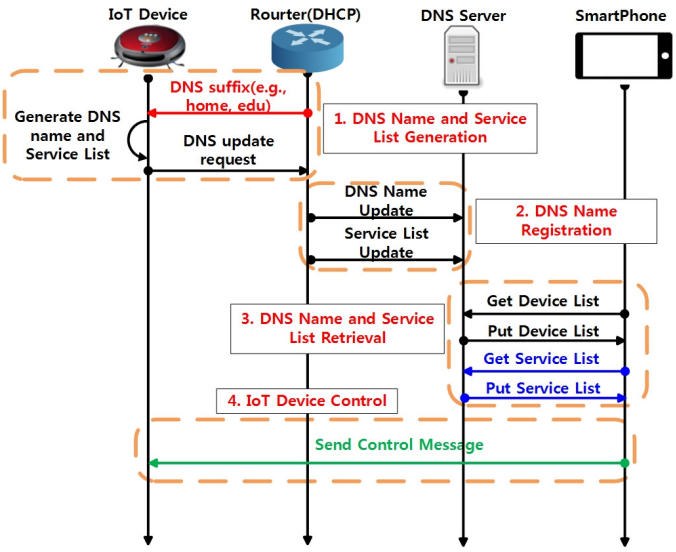

Fig. 1. Sequence Diagram of Our Remote Control System

Fig. 1 is a sequence diagram in service discovery and remote control for IoT devices. This consists of four steps for configuration. The first is DNS names and service list generation using DNS suffix (e.g., home, edu, com). In this step, IoT device will create the DNS name and service list of device. The second step is DNS name and service list registration, which is a step that included duplicate name detection. If the DNS name conflicts, a router sends a notification message to the IoT device. If it dose not conflicts the DNS name, the router requests a query of DNS dynamic update to a DNS server. The third is the DNS name and service retrieval. In this step, the smart device get list of devices from the DNS server. If the user selects a device, the DNS server sends service lists of it IoT device from the ZONE files. The fourth step

is IoT devices control. Following this step, the smart device constructs the CoAP message, which is included method (i.e., GET, POST, PUT, DELETE). Next, the smart device sends the message to the IoT devices such as robot cleaner, air-conditioner, smart TV, and so on. This allows the user to control the IoT device. The first and second steps are identical to DNSNA and DNSNAv4 [2], [3]. However, we have added the process of creating and registering service records. Also, we consist control scheme which, use the CoAP.

A. Service Record Format

<Instance Name>.<Service Type>.<Domain>

Fig. 2. DNS based Service Record Format

Fig. 2 shows the format of service records used in DNS-SD [4]. The DNS-SD use service record (SRV), pointer record (PTR), and text record (TXT). The detailed information about SRV record :

- The instance names represent user-friendly names (e.g., audio, printer, and tv).
- The service type represents which services and communication protocols the instance name is based on (e.g., ._printer._TCP, ._power._UDP).
- The domain shows the group to which the service record belongs. In addition, the domain type used in DNSNA and DNSNAv4 is selected.

The PTR type uses when users find the nodes to provide any services. For instance, when users finding the nodes with audio service possible to using the PTR format. It constructed using the service type and the domain to except instance name. TXT type use for additional information of service nodes. For instance, It uses to notify the status of users in chat program.

<unique_id>.<object_identifier>.<location>.<domain_name>

Fig. 3. DNS Name Format Using Object Identifier

Especially, we focused on the domain name format. As mentioned previously, our system is a different version of DNSNA. Therefore, we use the domain format to be proposed by DNSNA and DNSNAv4. The proposed the DNS name format uses a more hierarchical structure using OIDs as an IoT naming service, which is defined in DNSNA and DNSNAv4 [2], [3]. We can apply an oneM2M object identifier (OID) [12] as part of the DNS name format for an IoT device on the internet. ITU-T and ISO/IEC developed OIDs to assign a globally unique ID to an M2M node. The information contained in the DNS name format in Fig. 3 is described below:

- The unique id is an identifier that ensures uniqueness. The DNS name in ASCII characters. The identifier can be a sequence number or alphanumeric with readability, for example, the TV name can be TV1, such as the product name.
- The object identifier is a compound ID, such as a management organization, administration, country code, M2M, manufacturer ID, model ID, serial number. For example, in the case of 0.2.481.1.100.3030.10011, 0 is the management organization ITUT, 2 is management, 481 is country code Korea, 1 is M2M node, 100 is node manufacturer, 3030 is node model, and 10011 is node sequence number.
- The location indicates the physical location of the device, such as a living room or room. It includes micro locations such as the corners of the center, left floor, wall, and living room.
- The domain name is the DNS domain name. (e,g,. skku, edu or home) This is specified differently depending on the user, company.

B. Service and DNS Name Registration

Our goal is to provide users an efficient management for IoT devices using the DNS name and the service list of devices for monitoring and remote control in our environment. Accordingly, we introduce how to register service list and DNS name. A scheme for DNS name is used to be mentioned previously by Section III-A.

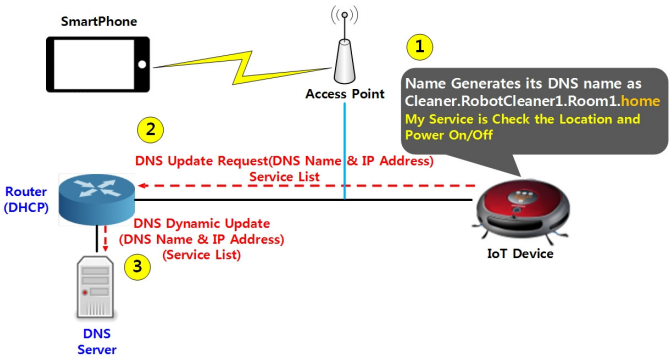

Fig. 4. Service List and DNS Name Registration

Fig. 4 shows the procedure of sequence how to create and register information of devices. The sequence consists of following three steps using DNS naming format and service record format.

- First, the IoT device creates the DNS name and the service records. The DNS name contains the IoT devices information such as unique ID, location, and device model. IoT device service list included providing some actions such as to control Audio, Printer, and Light.
- Second, IoT device requests to DNS update query including Internet Protocol IP address, DNS name, and service list.
- Third step is the router's procedure to request DNS dynamic update to DNS server using previously information of IoT devices. If the same DNS name is registered before, the router sends a message. If it is not a previously

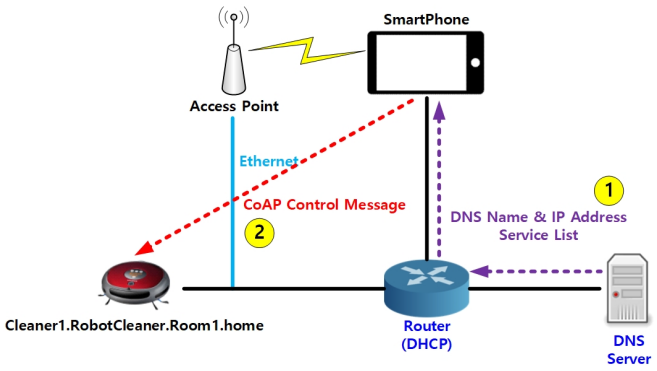

Fig. 5. Remote control for IoT Device

```
Timestamp : Sun Jul  9 11:14:43 2017
DHCP Server is running.........
============================================================
* Server Port           : 1234
* Dns server Ip address : 115.145.178.183
* Dns server rndc key   : +u/OLk+CG9invibwZ612rg==
============================================================

Timestamp : Sun Jul  9 11:15:44 2017
============================================================
* Recv Message       LEN : 305
------------------------------------------------------------
* Type       : 111,    DNS Update Request
* DNS Name   : IoTDev1.raspberry.IoTDev.Room1.cpslab.
* IP Address : 115.145.178.11
* Length     : 305
* Recv SRV Record List
FireSensor._led._udp.IoTDev1.raspberry.IoTDev.Room1.cpslab.
LightLvingRoom._led._udp.IoTDev1.raspberry.IoTDev.Room1.cpslab.
LightToilet._led._udp.IoTDev1.raspberry.IoTDev.Room1.cpslab.
LightKitchen._led._udp.IoTDev1.raspberry.IoTDev.Room1.cpslab.
============================================================
```

Fig. 6. Server Program Log

registered DNS name, the router will forward the DNS dynamic update query to the DNS server. The information of IoT devices is inserted on zone files in DNS server.

C. Remote Control for IoT Device

Recently, research on IoT has been performed about devices with low computing capacity. In addition, the IETF adopted a standard the CoAP [5]. CoAP has established in the CoRE working groups in IETF. The CoAP can search resource events (e.g., temperature and humidity) on M2M nodes in the upper application layer, including the transport layer with TCP and UDP. Also, It sends the resource events to a node based on Representational State Transfer (REST). The CoAP can support resource discovery using the Uniform Resource Indicator (URI). The users can search using method, which is provided by CoAP resource in their devices. Also, the users can be modified or create the resources using methods such as PUT and POST.

A service is defined as a set of information, such as protocol and host. IoT devices are the CoAP server with resources from CoAP point of view. So, the service is defined such as coap://[domain information]:5683. In terms of DNS-SD, we define services using the subtypes or the protocol types.

Fig. 5 express overall of our remote control system for IoT devices control. Follow the Fig. 5, user smart devices get DNS names and service list from DNS server. Next, users can make the control message using CoAP, which message is contained the method of HTTP such as PUT, GET, POST, etc. Also, The CoAP message is used same the method types to HTTP.

The smart devices will send control message by to be constructed format of the CoAP forward to IoT devices. Also, A packet has the URI that is a unique address of resources in Internet.

IV. Experiment in Testbed

We implement our remote control system in a real world to prove the goal of the system for IoT devices services and remote control in a smart-home environment. We used a Raspberry Pi as an IoT device in our testbed, a desktop PC as an authoritative the DNS server, and another Raspberry Pi for a Dynamic Host Configuration Protocol (DHCP) server, the smartphone is used as a CoAP client. The CoAP implementation uses C based libcoap and JAVA based Callifonium coap. Because smartphones support application implementation using JAVA, we run CoAP on smartphone. IoT devices also use libcoap to act as a CoAP server. A detailed description of IoT devices can be found in Section IV-B. And, a detailed description of smartphone can be found in Section IV-C.

A. Server Program

Fig. 6 shows a log of the server program. Starting the server program, it shows the using port number, DNS server address and rndc key, which is a key for DNS dynamic update. The server program receives the message included DNS name, IP address, and service records. Then, the server program validates the DNS name. The server will ignore the packet if the DNS name it received is a previously registered . However, if it is valid, the server registers the DNS name in the list of local DNS names. The server then uses the already acknowledged packets to request a DNS dynamic update to the DNS server. Fig. 6, The red mark is a message that the server program receives from the IoT device and registers the DNS name of the IoT device.

B. Internet-of-Things Device

Fig. 7 shows the captured image of the IoT device that included CoAP client program. The IoT devices received IP address and DNSSL option from DHCP, which is already configured to send DNSSL options (cpslab). The device displays a list of services if it can provide. Then the device creates DNS name for management and service records corresponding each service. The device sends both of DNS name and service records to the server. Originally, if it is not finished the registration procedure, the CoAP controller and IoT controller will be not set. In Fig. 7, client program normally works. It checks the CoAP based remote control in Fig. 7. User can control using smartphone or other smart devices to the IoT devices. Also, we can reduce overhead on the network using CoAP. The first red box shows the DNSSL option, DNS name,

```
DNSNA Client Program Starting....

Timestamp : Sun Jul  9 11:15:44 2017

- param 1 : Interface        = eth0
- param 2 : IP Address       = 115.145.178.11
- param 3 : DNSNA-Id Default = model.category.location
- param 4 : DNSNA-Name File  = local_name.txt
- param 5 : DHCP Server IP   = 115.145.178.183

DNS Search List    : cpslab.
Local Machine Name : IoTDev1.raspberry.IoTDev.Room1
Create Domain Name : IoTDev1.raspberry.IoTDev.Room1.cpslab.

* Device Service List
- FireSensor._led._udp
- LightLvingRoom._led._udp
- LightToilet._led._udp
- LightKitchen._led._udp

Timestamp : Sun Jul  9 11:15:44 2017

=====================================================
 * Send Message               LEN = 305
                To    115.
=====================================================
 * Type       : 111, DNS Update Request
 * Code       : 0, Send Message
 * DNS Name   : IoTDev1.raspberry.IoTDev.Room1.cpslab.
 * IP Address : 115.145.178.11
 * Create SVR Record List
FireSensor._led._udp.IoTDev1.raspberry.IoTDev.Room1.cpslab.
LightLvingRoom._led._udp.IoTDev1.raspberry.IoTDev.Room1.cpslab.
LightToilet._led._udp.IoTDev1.raspberry.IoTDev.Room1.cpslab.
LightKitchen._led._udp.IoTDev1.raspberry.IoTDev.Room1.cpslab.
=====================================================
 Start the CoAP controller
 IoT cotroller setting is completed
 Received CoAP Message for LightLivingRoom : ON
 Received CoaP Message Payload for LightToilet : ON
 Received CoAP Message Payload Firesensor : ON
 Received CoAP Message Payload Firesensor : OFF
 Received CoAP Message for LightLivingRoom : OFF
 Received CoaP Message Payload for LightToilet : OFF
```

Fig. 7. Program Log in the IoT Devices

and services of IoT device. The second red box is a message, including SRV, IP address, message types and DNS name, for registration to the server. The last red box shows the received control message using CoAP from smart device to IoT device. A detailed information in application describes at section IV-C.

C. Smartphone Application

We use the smartphone as a CoAP client to receive DNS names and services of the IoT device from the DNS server. For smartphone application development, we use the Android platform. In addition, we implement it using Callifonium coap to use CoAP function in smartphone. Callifonium coap is an open source implementation of CoAP based on Java and can operate on any PC with JVM installed. Fig. 8 shows the smartphone in our test bed. Users can easily check the service, IP address and DNS name supported by IoT device in smartphone. It also uses the GET, POST, PUT, and DELETE methods for CoAP. The red box shows that the smartphone got the IoT device's resources through the GET method.

V. Conclusion

We propose the new scheme for remote control and service discovery for IoT devices with minimize humans interaction

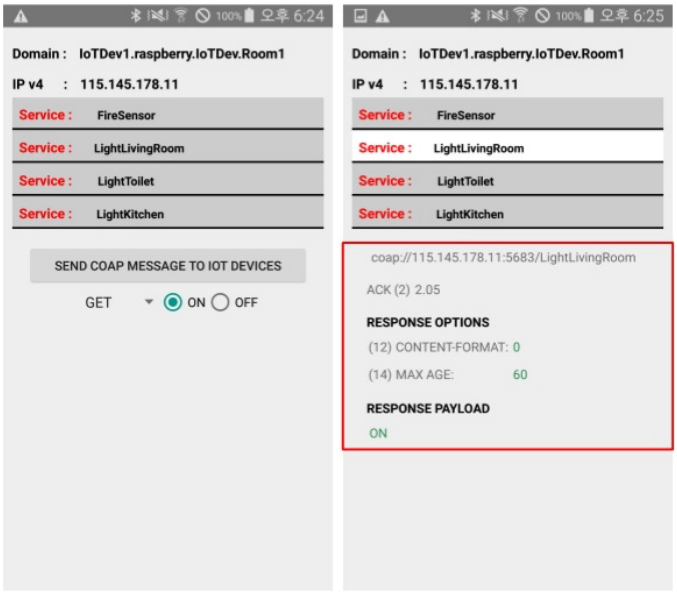

Fig. 8. Application in Smartphone

in networks. Through our remote control system, IoT devices provide DNS-SD. Most IoT devices have insufficient computing power for their services. For this our remote control system uses the CoAP. The CoAP includes the method of HTTP. For instance, CoAP client possible to obtain the resources of IoT devices using the method of GET. Since CoAP supports resource discovery, it works complementary to the proposed DNS naming scheme proposed by DNSNA. Also, CoAP and DNS-SD are standard protocols established by the IETF. Therefore, we can easily apply it to the other software platforms using standard protocols. As a result, our architecture provides effective management techniques for IoT devices. Users easily can identify IoT devices and the service list of devices. We believe our architecture is a promising approach for the convenient services in networks. As future work, we will research on to provide safety communication each device security issues for our remote control system. We will also consider cloud based remote control and management for IoT devices.

Acknowledgments

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2017R1D1A1B03035885).

Note that Jaehoon (Paul) Jeong is the corresponding author.

References

[1] Gartner's 2014 Hype Cycle [Online]. Available: http://www.gartner.com/newsroom/id/2819918

[2] Sejun Lee and Jaehoon (Paul) Jeong and Jung-Soo Park, "DNSNA: DNS Name Autoconfiguration for Internet of Things Devices", in Proceedings of the 18th International Conference on Advanced Communication Technology (ICACT), Feb. 2016.

[3] Keuntae Lee, Seokhwa Kim, and Jaehoon (Paul) Jeong, “DNSNAv4: DNS Name Autoconfiguration for Internet-of-Things Devices in IPv4 Networks”, 31th International Conference on Advanced Information Networking and Applications Workshops - Device Centric Cloud (DC2), Taipei, Taiwan, Mar. 2017.
[4] S. Cheshire and M. Krochmal, “DNS-Based Service Discovery”, IETF RFC 6763, Feb. 2013.
[5] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP)”, IETF RFC 7252, Jun. 2014.
[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, “Hypertext Transfer Protocol – HTTP/1.1”, IETF RFC 2616, Jun. 1999.
[7] R. Droms, “Dynamic Host Configuration Protocol”, IETF RFC 2132 Mar. 1997.
[8] Apple, “Bonjour, https://developer.apple.com/.
[9] S. Cheshire and M. Krochmal, “Multicast DNS”, IETF RFC 6762, Feb. 2013.
[10] J. Jeong, S. Park, L. Beloeil, and S. Madanapalli, “IPv6 Router Advertisement Options for DNS Configuration”, IETF RFC 6106, Nov. 2010.
[11] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, “Neighbor Discovery for IP version 6 (IPv6)”, IETF RFC 4861, Sep. 2007.
[12] M2M, “Object Identifier based M2M Device Identification Scheme”, http://www.onem2m.org.
[13] R. Droms, J. Bound, B. Volz, C. Perkins, and M. Carney, “Dynamic Host Configuration Protocol for IPv6 (DHCPv6)”, IETF RFC 3315, Jun. 2003.