International Journal of Grid and Distributed Computing
Vol. 11, No. 5 (2018), pp.1-10
http://dx.doi.org/10.14257/ijgdc.2018.11.5.01

IoT Resource Management using Direct Discovery Mechanism in OCF Framework

Israr Ullah[1] and DoHyeun Kim[2,*]

[1,2]Department of Computer Engineering, Jeju National University, Republic of Korea
[1]israrullahkk@yahoo.com, [2]kimdh@jejunu.ac.kr*

Abstract

In recent past, tremendous growth have been experienced by Internet of Things (IoT) based applications in various domains. Billions of devices are expected to be connected to Internet in near future. To build IoT based application, first virtual objects are created to virtualize the physical devices by abstracting device properties in virtual objects. Later, these virtual objects are combined to compose different services which are used in diverse applications. In cyber world, resources provide an effective way to express virtual objects. Every physical device can have one or more corresponding resources representing its logical state, physical data or resources can be created even without having any physical devices associated with it i.e. soft sensors. This paper presents an IoT resource management using direct discovery mechanism based on OCF IoTivity.

Keywords: *IoT, Resource Management, IoTivity, Resource discovery, OCF*

1. Introduction

Recent developments in communication technologies have enabled the small computing devices to communicate and share information anytime, anywhere. The objective is to enable seamless usage of computing devices in daily life. This is made possible by integration of many underlying enabling technologies including Internet, tiny devices, sensors and operating system, protocols and interoperability. There are many open challenges and opportunities, inviting research attention for investigation and exploitation *e.g.*, recently coined Internet of Things (IoT) has opened a whole lot of new avenue for research and development in many different fields [1]. IoT is mainly focused of on connectivity of things (daily life objects with attached sensors or actuators) to Internet so that users can remotely monitor and control a particular activity or device.

Internet of Things (IoT) is designed to attach a small communicating device with everything that we want to monitor or control using Internet. The IoT device may be sensing device that collect some desired information or it can be an actuating device that can accept commands to perform some desired task. IoT devices have limited communication, computation and battery power and lightweight protocols are designed for efficient resource utilization over Internet *e.g.*, CoAP protocol [2]. IoT Systems have tremendous capabilities and applications [3]. In the recent past, many giant manufacturing and development organizations have made investment in this technology to realize its potential. Many projects are initiated to address different aspect of IoT based systems.

Development of IoT Composition tools is one of the key areas of research to enable mass involvement in realization of IoT vision [4]. The objective is to enable a common user to utilize IoT based system in order to achieve his/her desired task without any programming skills. These systems will also enable managers and designers to use the

---

Received (December 11, 2017), Review Result (March 8, 2018), Accepted (March 12, 2018)
* Corresponding Author

ISSN: 2005-4262 IJGDC

system on their own without bothering IoT experts to achieve some task. Do-it-Yourself (DIY) IoT Project is an attempt in the direction of IoT Composition tool development using standard Business Process Modeling (BPM) notations. The user of this system is expected to be aware of BPM notation and will only require slight training to utilize the system in his/her domain [5].

Finding a suitable service required for a particular application development over Internet is a challenging task. Normally, dynamic service discovery option does not satisfy the complex user requirements for exceeding number of requests. [6] presents an idea of on-demand service creation by combining existing services to meet complex user needs using visual and interactive interface. User desired objective is decomposed into sub-goals and attempt is made to satisfy each sub-goal in iterative fashion. In [7], the authors present a semi-automatic mechanism for service filtering using semantic ontology. Their proposed system consists of two main components, inference engine and composer. Inference engine helps in selection of best matching service by performing service filtering which are then connected by composer component to produce desired service. A similar system is reported in [8] for automatic service composition. Intelligent solutions based on AI algorithms are also explored for automatic service composition in [9]. A framework for IoT objects management and naming is proposed in [10] based on future IoT-IMS platform. However, their main focus is on addressing the issue due lack of proper rules or specification for things naming. In this paper, we have proposed an IoT resource management system using direct discovery mechanism. Objective of proposed system is to facilitate the discovery of resources in IoT based network without requiring centralized resource directory. The system also allows interaction to with discover resources to read it properties and check device operational and fault status.

Rest of the paper is organized as follows. Section 2 describes direct discovery mechanism in OCF framework. Section 3 presents detailed description of proposed system design. The implementation prototype of IoT resource management is presented in Section 4. Towards the end, we conclude this paper with an outlook to our future work.

2. Direct Discovery Mechanism in OCF Framework

Device management, diagnostics, and fault recovery capabilities are considers among the core requirements in standards developed for IoT technologies. For instance, the most popular standard developed by Open Connectivity Foundation (OCF) defines guidelines for device management, diagnostic and maintenance through two OIC core resources /oic/mon and /oic/mnt that shall be supported by all devices [11]. The monitoring resource /oic/mon is used for collection of device statistics *e.g.*, packets sent, received, last operation time *etc*. This maintenance resource /oic/mnt has two important properties (a) factory reset "fr" is used to update the device configuration to its original (default) state (factory state and equivalent to hard reboot) and (b) reboot "rb" triggers a soft reboot of a device while maintaining most of the configurations intact.

Figure 1 shows typical interaction between OIC client and server for device management operation. OCF foundation is also sponsoring an open source project IoTivity to provide reference implementation for OCF technical specifications [12]. Likewise, oneM2M is a global organization that aims at development of functional architecture, API specifications, security solution with interoperability for M2M communication and IoT technologies. oneM2M standard architecture defines three main entities *i.e.*, application entity, common service entity and network service entity [13].

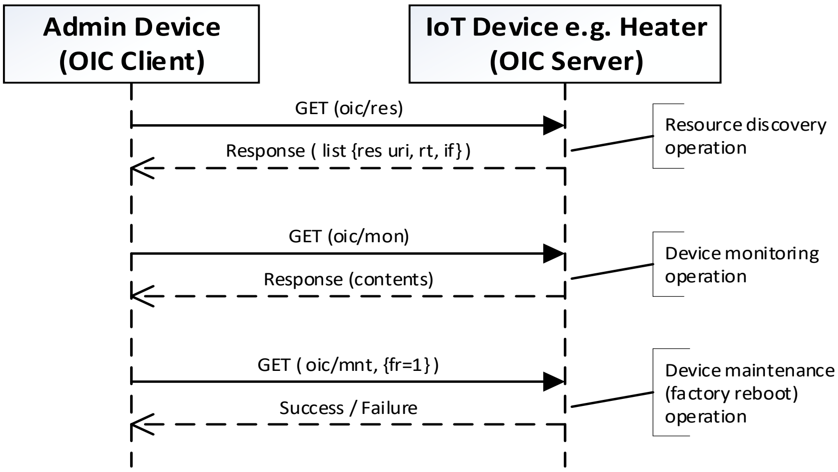

Figure 1. Typical Interaction between OIC Client and Server for Device Management Operation

Common service entity has two main components i.e. common service functions (CSF) and enabler functions (EF). CSFs includes device management among core components (as shown in Figure 2 to utilize the information for administrative purposes *i.e.*, diagnostic and troubleshooting. oneM2M specifies four key functions for device management *i.e.*, device configuration function (DCF), device diagnostic and monitoring function (DDMF), device firmware management function (DFMF), and device topology management functions (DTMF). Similarly, ITU-T also consider device management capabilities, diagnostic and fault recovery capabilities among the common service and application support functions [14].

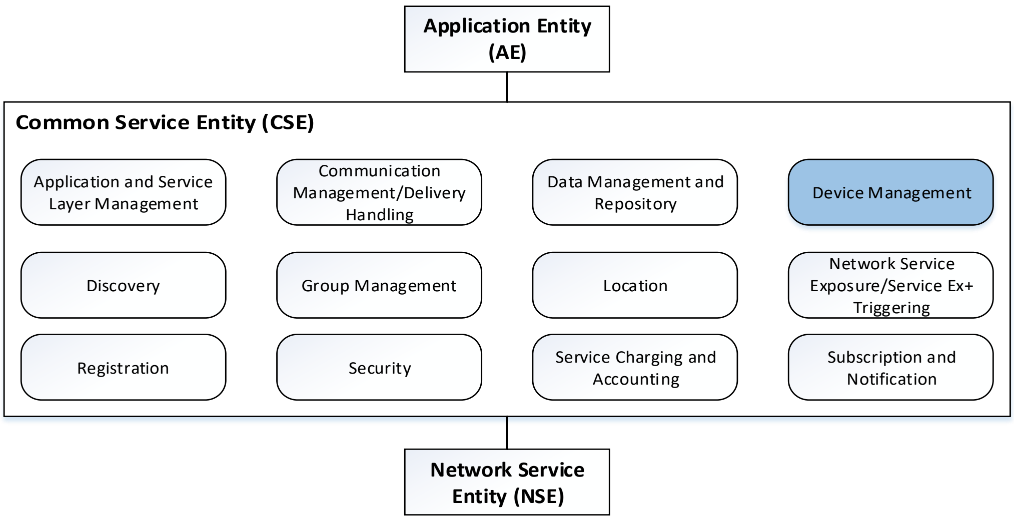

Figure 2. oneM2M Entities Relationship and Common Service Functions [13]

Open Connectivity Foundation (OCF) is a standard development organization for IoT [11]. IoTivity is an open source reference implementation for the OCF specifications [12]. The architecture enables resource based interactions among IoT artefacts, *i.e.*, physical devices or applications as shown in Figure 3. The architecture is based on the Resource Oriented Architecture design principles. In the architecture, Entities in the physical world (*e.g.*, temperature sensor, an electric light or a home appliance) are represented as resources. Interactions with an Entity are achieved through its resource representations using operations that adhere to Representational State Transfer (REST) architectural style, *i.e.*, RESTful interactions.

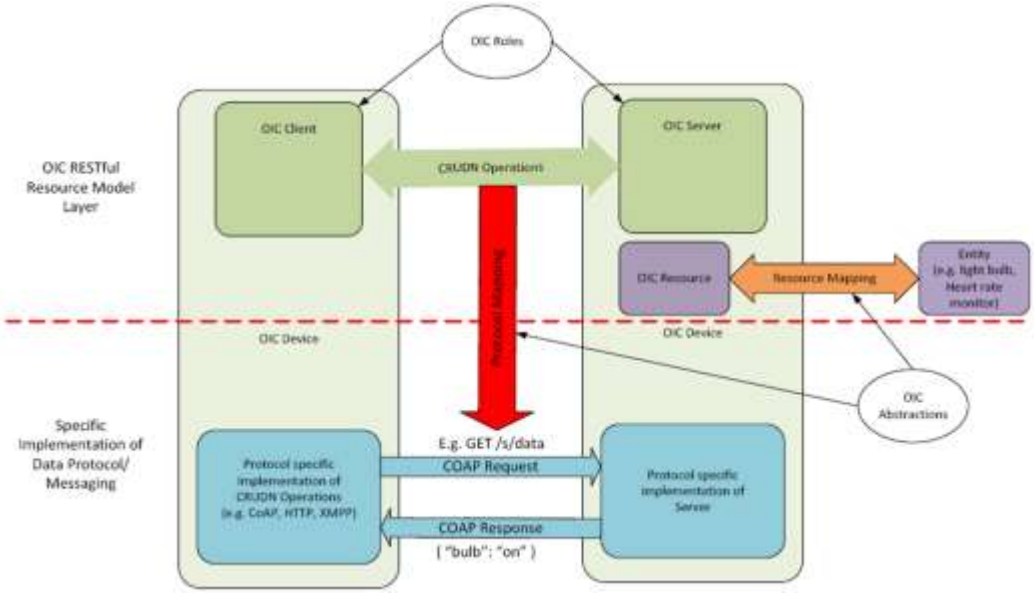

Figure 3. OCF Standard Architecture and Concepts [12]

OCF specification illustrate three standard methods for resources discovery:

(a) Direct Discovery: Client send request (unicast/multicast) directly to the devices which hold the resources. Each device, upon receiving such request, will respond with list of resources it holds.

(b) Indirect Discovery using resource directory: Each device publish its resources information to a resource directory (RD). Resource discovery requests are handled by RD on behalf of publishing devices.

(c) Advertisement Discovery: Each device host a local resource to enable resources discovery. Publishing devices shall advertise its resources information to each device local resource. Resources can be discovered by making local retrieve request.

3. Proposed IoTivity based IoT Resource Management using Direct Discovery

To illustrate working of proposed IoTivity based IoT resource management system, we have implemented a simple prototype applications to test its various component. The proposed system will consists of three essential modules.

- Boot strap server provides initial configuration information to devices.
- Devices are simple IoT devices in our network.
- Resource Manager (RM) is the client application from where we can perform various supported management operations

First, we need to start configuration server process. This process holds all the basics configuration information for IoT devices in our network. Typical configuration information includes location name, region name, currency information etc. This is just for testing purpose. In real application scenario, it may holds IoT devices initial setup information depending upon the application scenarios. Configuration information can be stored locally inside each IoT device but in that case its very difficult to make some changes. A minor changes will require contacting each IoT device and update its configuration information. However, we cannot be sure about consistency if configuration information in all IoT devices. Having a centralized configuration server make this job easy. We just to pre-configure IoT devices with information regarding configuration server and devices can get necessary configuration during its boot-up process automatically. Central configuration server also helps in maintaining consistent configuration setting across all IoT devices.

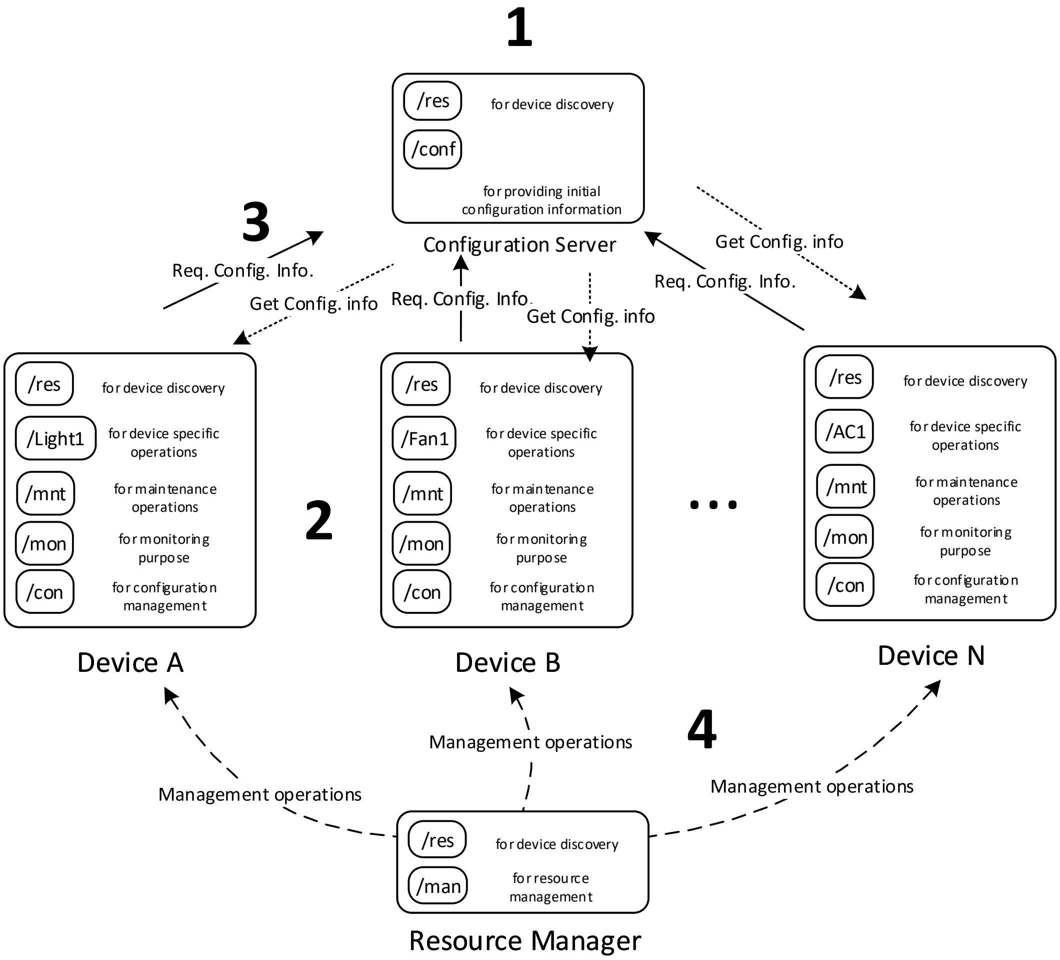

Figure 4. Proposed IoT Resource Management based on IoTivity

Once, configuration server is started successfully, afterwards, we can start our IoT devices. All IoT devices are preconfigured with address information of configuration process. During initial boot-up process, IoT device contact configuration server to get necessary information and updates its internal status accordingly. Device local variables are updated with information provided by configuration server. According to OCF specification, IoT devices needs to follow two distinct operation during initial boot-up. (a) onboarding process is used to deliver all necessary information for joining OCF network to IoT device. Upon completion on onboarding process, each IoT device has all required information and can join OCF network. (b) Configuration process is used to provide all required information for accessing OCF services to IoT devices. When this process is completed, IoT devices can access all OCF services. Configuration information are stored in core recourse /oic/con.

Finally, we need some kind of centralized monitoring and control system that can helps us checking devices operational status and make necessary changes in devices configuration as and when required. For this purpose, we have implemented a separate resource manager process that can help us performing these tasks. Resource manager process can discover all IoT devices in our network by using direct discovery mechanism. To support direct discovery mechanism, all IoT devices needs to host core resources /oic/res. This resource has links to all others resources hosted on the same device. Resource manager just make a peer inquiry RETRIEVE request on core resource /oic/res and all devices will respond by providing resources information hosted on each device. Retrieve request can be made unicast or multicast. There are other mechanisms for resource discovery in OCF specification *e.g.*, indirect discovery and advertisement discovery. Indirect discovery require a dedicated Resource Directory (RD) in the network where all devices shall publish information regarding their public resources. RD will then respond to all inquiries for resources discovery on behalf of publishing device. RD is useful when number of IoT device in the network are many. We choose direct discovery mechanism because of its simplicity and easy implementation.

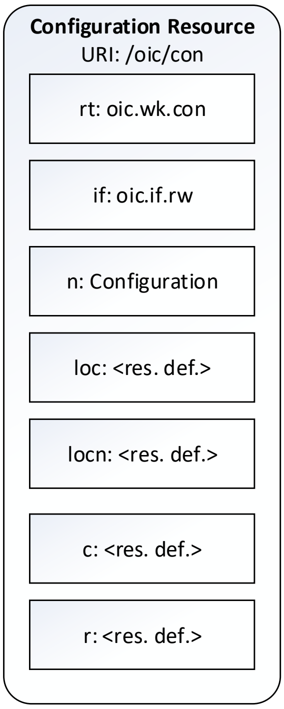

Figure 5. Structure of Configuration Resource

We have used direct discovery mechanism for resources discovery as resource manager has to query each IoT device directly to get its resources information. Every device needs to have 5 resources as shown in Figure 4. Last three resources are for device management operations *i.e.*, /mnt can be used to factory reset the device settings or reboot operations. /mon can be used to collect statistical data about the device and /con can be used to update device configuration information.

Configuration resource "/con" is used to manage information regarding the configuration service, *e.g.*, the URI of the configuration source, is pre-configured on the device. Figure 5 specify the necessary attribute and properties of configuration resource as defined in OCF standard document.

Maintenance resource "/mnt" enables execution of operations on resources like reboot, factory reset and start statistics collection etc. Figure 6 specify the necessary attribute and properties of maintenance resource as defined in OCF standard document.

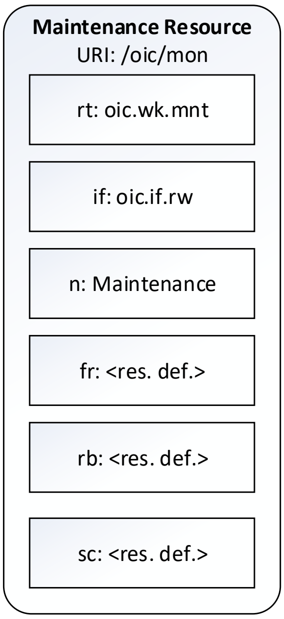

Figure 6. Structure of Maintenance Resource

Monitoring resource "/mon" is used to collect device statistics like packets sent, packets received, last acted time *etc*. Figure 7 specify the necessary attribute and properties of monitoring resource as defined in OCF standard document.

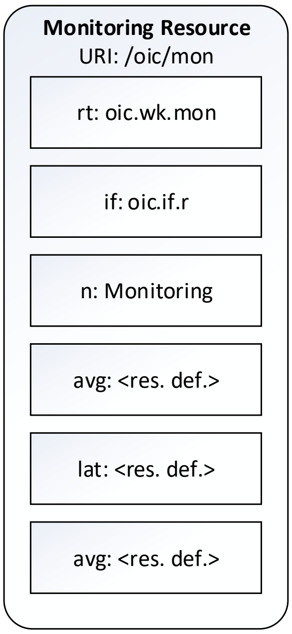

Figure 7. Structure of Monitoring Resource

We assume that there are two things (*i.e.*, resource servers) to be managed by the resource manager (*i.e.*, a resource client). Additionally, we also assume that a simple bootstrap server is installed in purpose that things can fetch essential configuration parameters to access other IoT service. This scenario shows a bootstrap procedure between things and a bootstrap server and retrieval and update procedures between things and resource manager.

4. Implementation Results and Discussion

For experimental analysis, we have implemented proposed system using Ubuntu 14.04 (64-bit) machine. IoTivity version 1.2.1 is used in this implementation setup. We have build IoTivity framework using Eclipse IDE using SCons. Among other necessary libraries required for IoTivity build process, we also have downloaded yaml and tinycbor packages. We have implemented various servers for hosting various resources. Each server is basically a separate device may be a sensing or actuating device. Each device will have the default /oic/res resource to enable resource discovery operation. Each device shall also support the core management resources *i.e.*

- For management operations **/oic/mnt**
- For monitoring purposes **/oic/mon**

4.1. Configuration Server

This server will provide configuration information to connecting devices. This server process is hosting a /conf resource. Figure 8 shows how we start the configuration server process.

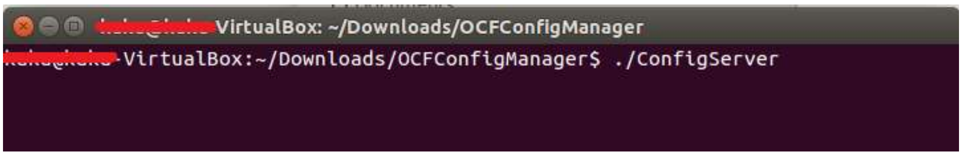

Figure 8. Configuration Server Process

4.2. IoT Devices

In this example, we only try to execute two devices and will interact with them through configuration manager in order to update its information. Figure 9 show a simple device processes and another similar process is also started in parallel.

Figure 9. IoT Device Process

Next, we first perform boot strap operation on each device by selecting appropriate menu option as shown in above figure. Boot strap process collect configuration information from running configuration manager server and device local variables is updated accordingly.

Afterwards, we choose third menu option *i.e.*, creation of configuration resource on each device. This includes creation of three resources at device level. Configuration Resource, Diagnostics Resource and Factory Reset Resource.

4.3. Resource Manager

This is basically like a client process through which user can interact and update configuration and other status information of running IoT devices. Figure 10 shows the first output window menu option when we start resource manager process.

Figure 10. Resource Manager Process

Using the resource manager process, we can perform direct resource discovery and other management operation *e.g.*, revision of configuration information and factory reset or reboot operation *etc*.

5. Conclusion

This paper presents improved IoT resource management architecture for effective organization of resources. Proposed architecture supports to facilitate management tasks for connected devices in IoT environment to *e.g.*, the improvement of configuration information, performing management operation *e.g.*, reboot and factory reset operation. Proposed system is implemented using IoTivity framework using Ubuntu operation system. Proposed system functionality is achieved using three different process (a) configuration server which provides basic configuration information to connecting IoT devices, (b) IoT devices process to mimic real IoT device with certain resource for discovery (c) resource manager process to discover IoT resources and perform management operations such as update configuration information, perform factory reset or reboot operations. In future, we look forward to include a resource directory for efficient lookup and for better visualization, we will display all resources over map at their corresponding locations.

Acknowledgement

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(2014-1-00743) supervised by the IITP(Institute for Information & communications Technology Promotion), and this work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2017-0-00756, Development of interoperability and management technology of IoT system with heterogeneous ID mechanism). Any correspondence related to this paper should be addressed to DoHyeun Kim; kimdh@jejunu.ac.kr.

References

[1] L. Coetzee and J. Eksteen, "The internet of things - promise for the future? An introduction", IST-Africa Conference Proceedings, **(2011)**, pp. 1-9.
[2] Z. Shelby, K. Hartke and C. Bormann, "The constrained application protocol (CoAP)", RFC 7252, RFC Editor, http://www.rfc-editor.org/rfc/rfc7252.txt, **(2014)** June.
[3] D. Miorandi, S. Sicari, F. D. Pellegrini and I. Chlamtac, "Internet of Things: Vision, applications and research challenges", Ad Hoc Networks, vol. 10, no. 7, **(2012)**, pp. 1497-1516.
[4] P. P. Jayaraman, "Do-it-Yourself digital agriculture applications with semantically enhanced IoT platform", Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference, **(2015)**, pp. 1-6.
[5] D. Mazzei, "Internet of Things for designing smart objects", IEEE Internet of Things (WF-IoT), **(2014)**, pp. 293-297.
[6] M. Naeem, R. Heckel and F. Orejas, "Semi-automated service composition using visual contracts", Proceedings of the 7th International Conference on Frontiers of Information Technology, FIT09, New York, NY, USA, ACM, **(2009)**, pp. 48:1-48:6.

[7] E. Sirin, J. Hendler and B. Parsia, "Semi-automatic composition of web services using semantic descriptions", In Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003, **(2002)**, pp.17-24.
[8] G. Li, S. Deng, H. Xia and C. Lin, "Automatic service composition based on process ontology", Next Generation Web Services Practices, 2007. NWeSP 2007. Third International Conference, **(2007)** October, pp. 3-6.
[9] D. E. A. Wu, "Automating daml-s web services composition using shop2", Proceedings of the Second International Conference on Semantic Web Conference, LNCS-ISWC 03, Berlin, Heidelberg, Springer-Verlag, **(2003)**, pp. 195-210.
[10] K. D. Chang, C. Y. Chang, H. M. Liao, J. L. Chen and H. C. Chao, "A framework for IOT objects management based on future internet iot-ims communication platform", 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, **(2013)** July, pp. 558-562.
[11] Open Connectivity Foundation (OCF), OIC CORE SPECIFICATION V.1.1.1.
[12] IoTivity: https://www.iotivity.org/ Accessed: 2017-04-10.
[13] oneM2M, Technical Specifications - Functional Architecture, 8 2016. version 2.10.0 Release 2.
[14] ITU-T, Requirements and reference architecture of the machine-to-machine service layer, 11 2015.

Authors

Israr Ullah, received his MCS degree from Institute of Computing and Information Technology (ICIT), Gomal University, Pakistan, in 2004. He completed his M.S. in computer science from National University of Computer and Emerging Sciences (NUCES), Islamabad, Pakistan, in 2009. Currently, he is pursuing his Ph.D. studies at Computer Engineering Department, Jeju National University, Republic of Korea. His research work is focused on application of prediction and optimization algorithms to build IoT based solutions. His research interests also include analytical modeling, network simulation and analysis of optimization algorithms.

Do-Hyeun Kim, he 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.

