International Journal of Grid and Distributed Computing
Vol. 11, No. 3 (2018), pp.1-10
http://dx.doi.org/10.14257/ijgdc.2018.11.3.01

A Hierarchical IoT Federation Architecture Based on Local Cloud, Platform and Middleware for Massive Context Data Acquisition in Multiple Sensor Networks

Songai Xuan[1] and Do-Hyeun Kim[2*]

[1, 2]*Department of Computer Engineering, Jeju National University, South Korea*
[1]*xuansongai@foxmail.com*, [2]*kimdh@jejunu.ac.kr*

Abstract

Recently, we have been constructed sensor networks for smart space much more. It can provide users a smart environment and make users living or working conveniently and easily. In this paper, we present a hierarchical IoT federation architecture based on local Cloud, platform and middleware for massive context data acquisition in multiple sensor networks. Proposed IoT federation architecture has the interworking between sensor platform and local IoT cloud in multiple sensor networks, which can collect massive environment data for analysis of situation. And we design and implement a HDFS agent for Interworking mechanism between sensor platform and local IoT Cloud. We consider Hadoop for local IoT Cloud.

Keywords*: Wireless Sensor Network; Sensor platform; Hadoop; Local Cloud, Middleware, Interworking*

1. Introduction

The Internet-of-Things (IoT), or more prosaically Machine-to-Machine (M2M), has received significant attention lately from both industry and academia as an emerging paradigm that manages billions of devices, gateways, sensors, and actuators connected to the Internet [1]. There is an overall view of interworking architectures, which enables exposure of various underlying network services for M2M applications running on top of the service layer, such as device triggering, device location, device management, *etc* [2]. And another paper presents an introduction of standardized interworking interfaces and procedures based on oneM2M global standards, and tests them through use cases involving multiple IoT service platforms [3]. The interworking involves smart city applications/services running on multiple IoT service layer platforms interoperating with each other.

In this paper we have proposed a hierarchical IoT federation architecture based on local cloud, platform and middleware for massive context data acquisition in multiple sensor networks. It aims to collect massive environment data for situation analysis and provide users a convenient smart environment. We design and implement a HDFS agent for the interworking mechanism between sensor platform and local IoT Cloud.

The remainder of this paper is divided as follows, Section 2 presents the related work. Section 3 proposes the hierarchical IoT federation architecture based on local cloud, platform and middleware. Section 4 proposes the design of HDFS agent for interworking sensor platform and Hadoop. Section 5 presents the implementation and results of the proposed architecture. And section 6 concludes the paper.

Received (December 26, 2017), Review Result (February 16, 2018), Accepted (February 18, 2018)
* Corresponding Author: Do-Hyeun Kim

ISSN: 2005-4262 IJGDC

2. Related Work

REST (Representational State Transfer) is an architectural style, which is often used in the development of web services. REST is a popular building style for cloud-based APIs. A RESTful API means web services used REST architecture. REST architecture involves reading a designated web page that contains an XML file, which describes and includes the needed content. REST typically runs over HTTP (Hypertext Transfer Protocol) and is often used in mobile applications, social networking web sites, mashup tools and automated business processes. REST use a limited number of operations (GET, POST, PUT and DELETE) to enhance the interactions between clients and services. And it is flexible because of assigning resources their own URIs (Universal Resource Identifiers) [4].

Hadoop as an open source project of the Apache foundation is the most representative product for the cloud computing research and application. The Hadoop's distributed framework provides developers with a base architecture for distributed systems. The Hadoop users can develop distributed applications without understanding the underlying details of the distributed system and make full use of the cluster storage resources, network resources and computing resources. The core design of Hadoop is MapReduce and Hadoop Distributed File System (HDFS) [5]. Figure 1 shows the architecture of Hadoop. Hadoop frame includes four modules: MapReduce, HDFS, YARN and Common Utilities. Hadoop MapReduce is YARN-based system for parallel processing of large data sets. Hadoop Distributed File System (HDFS) is a distributed file system that provides high-throughput access to application data. Hadoop YARN is a framework for job scheduling and cluster resource management. Hadoop Common Utilities are Java libraries and utilities required by other Hadoop modules. These libraries provide filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop [6].

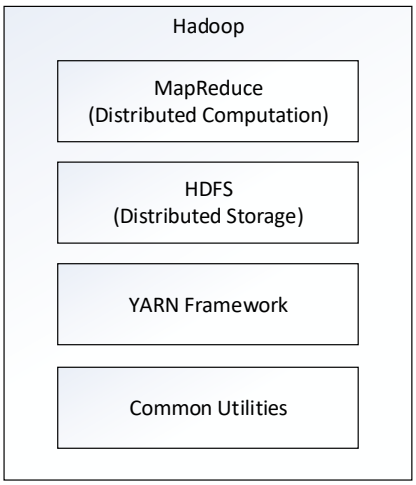

Figure 1. Hadoop Architecture

HDFS is a highly fault-tolerant system, suitable for deployment in cheap machines. HDFS can provide high throughput access data and it is very suitable for large-scale data sets. HDFS has a high fault-tolerance characteristic, and is designed for deployment on low-cost hardware [7]. It provides high throughput to access the application data, suitable for those with large data set applications. HDFS is a distributed file management system for massive data storage. In this system, we use HDFS to store the sensor data files, and we manage the files by calling Hadoop commands in Java Application

3. A Hierarchical IoT Federation Architecture Based on Local Cloud, Platform and Middleware

Figure 2 shows the hierarchical IoT federation architecture based on local Cloud, platform and middleware in sensor networks. As shown in the figure, each sensor platform connects local IoT cloud. And each sensor platform can connect many sensor middleware, each sensor middleware can connect many sensors. Sensor platform provide sensor information and sensing data storage service. Sensor middleware get sensing data from sensors and save the data into database via the service provided by sensor platform. Then sensor platform will request sensing data and sensor information and upload the data to local IoT cloud in multiple sensor networks.

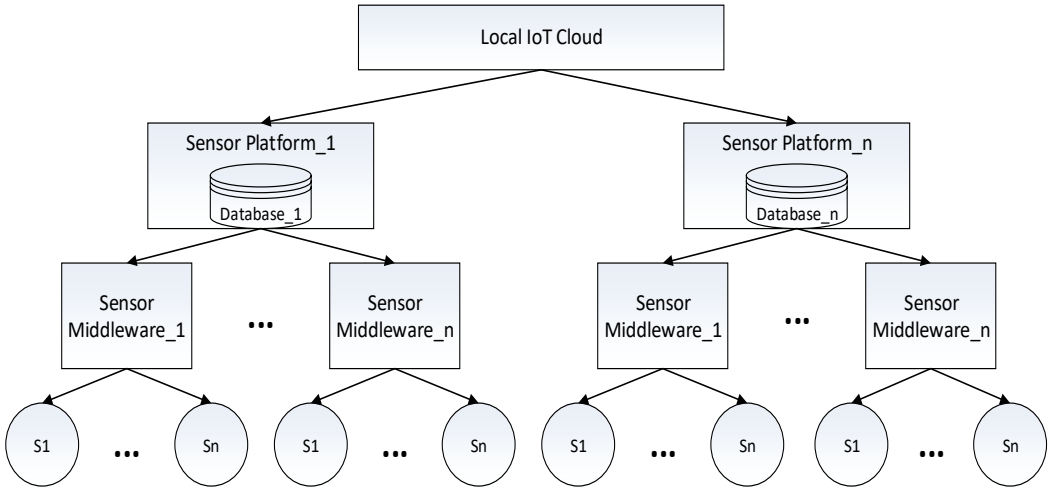

Figure 2. Hierarchical IoT Federation Architecture based on Local Cloud, Platform and Middleware

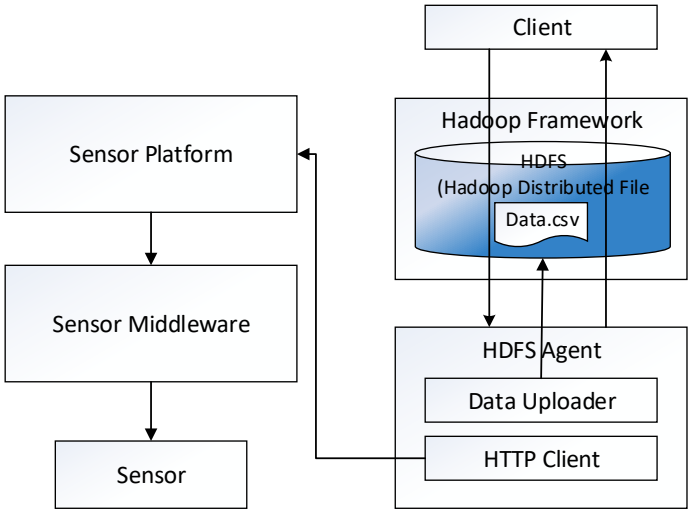

Figure 3. Interworking Mechanism between Sensor Platform and IoT Cloud

Figure 3 shows the conceptual model of this system. We will develop a HDFS Agent to connect Sensor middleware and platform and Hadoop Distributed File System (HDFS). Sensor Middleware will get sensing data from Sensor and save sensing data into database via the service provided by Sensor Platform. There is a RESTful API in Sensor Platform and a HTTP Client in HDFS Agent. HTTP Client will get sensing data and sensor information from Sensor Platform via the RESTful API. And Data Uploader in HDFS

Agent is able to receive the sensing data from HTTP Client, convert the sensing data to files (txt, csv), and upload the files to Hadoop Distributed File System (HDFS) in Hadoop framework. Client will control the process start or stop and show users the sensing results.

4. Design of HDFS agent for Interworking Sensor Platform and Hadoop

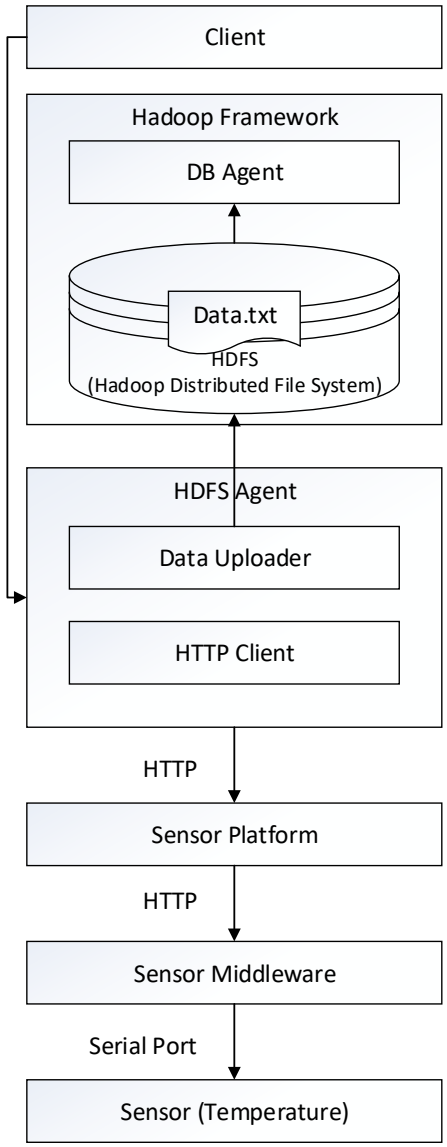

Figure 4. Configuration for Interworking Sensor Platform and Hadoop

Figure 4 shows the implement architecture. Sensor Middleware will get sensing data from Sensor and save sensing data into database via the service provided by Sensor Platform. There is a RESTful API in Sensor Platform, the communication between Sensor Platform and Sensor Middleware based on this API and used HTTP protocol. Sensor connect to Sensor Middleware by serial port. The communication between HDFS Agent and Sensor Platform also based on the RESTful API. There is a HTTP Client in HDFS Agent. HTTP Client will get sensing data and sensor information from Sensor Platform via the RESTful API. And Data Uploader in HDFS Agent is able to receive the sensing data from HTTP Client, convert the sensing data to files (txt, csv), and upload the files to

Hadoop Distributed File System (HDFS) in Hadoop framework. Client will control the process start or stop and show users the sensing results

Figure 5 shows the detail design for HDFS Agent. In HDFS Agent. There is a HTTP Client and a Data Uploader. In HTTP Client, there are Data Transmitter, Data Parser and Data Receiver. When HDFS Agent get sensing data from Sensor Platform, Data Receiver in HTTP Client will receive data first and send to Data Parser, then Data Parser will send to Data Transmitter, and Data Transmitter will send data to Data Uploader and Client.

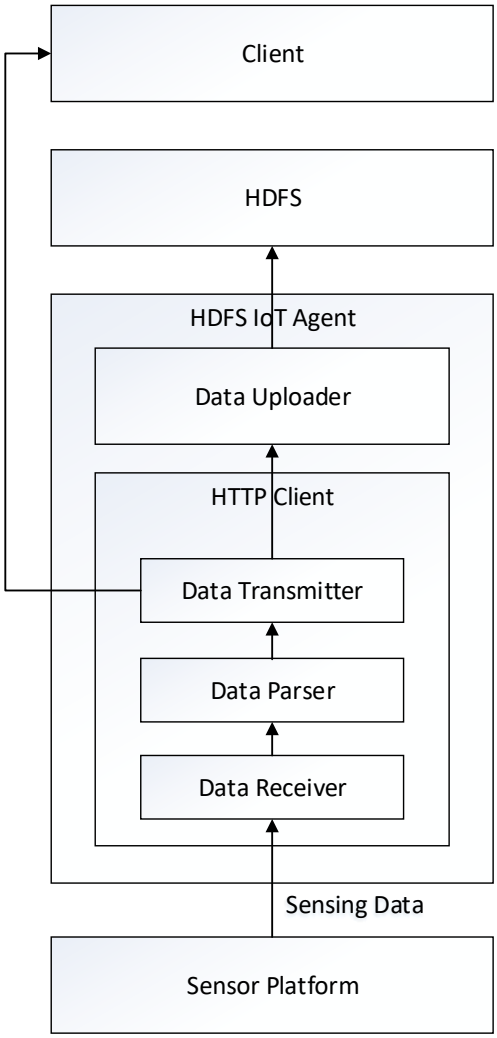

Figure 5. Detail Design of HDFS Agent

Sensor Platform support to collect massive context data. Sensor middleware take a role in collecting sensing data sent by sensor then sending and saving sensor platform. Sensor middleware request sensor information (ID, Type) from sensor platform, and also verify pertinent IP address and Platform access privileges. Sensor middleware takes various sensors' sensing data format information and parse processing through received sensing data. Sensor middleware take a role in monitoring state of the port connected, and accesses sensing data sent from sensor node and saves at memory through sensing data Parser.

Figure 6 shows the sequence diagram of this system. Client send start request to Data Uploader and request sensing data from Data Uploader. Data Uploader sends request message to HTTP Client, HTTP Client sends request message to sensor Platform, Sensor Platform sends request message to Sensor Middleware, Sensor Middleware sends request

to Sensor and Sensor will return sensing data to Sensor Middleware. Sensor Middleware will return sensing data to Sensor Platform, Sensor Platform will save sensing data into database and return sensing data to HTTP Client. HTTP Client will return sensing data to Data Uploader. Data Uploader will return sensing data to Client, make a data file (txt, csv) and uploads file to HDFS. Finally, Client will send stop request to Data Uploader, all process will stop.

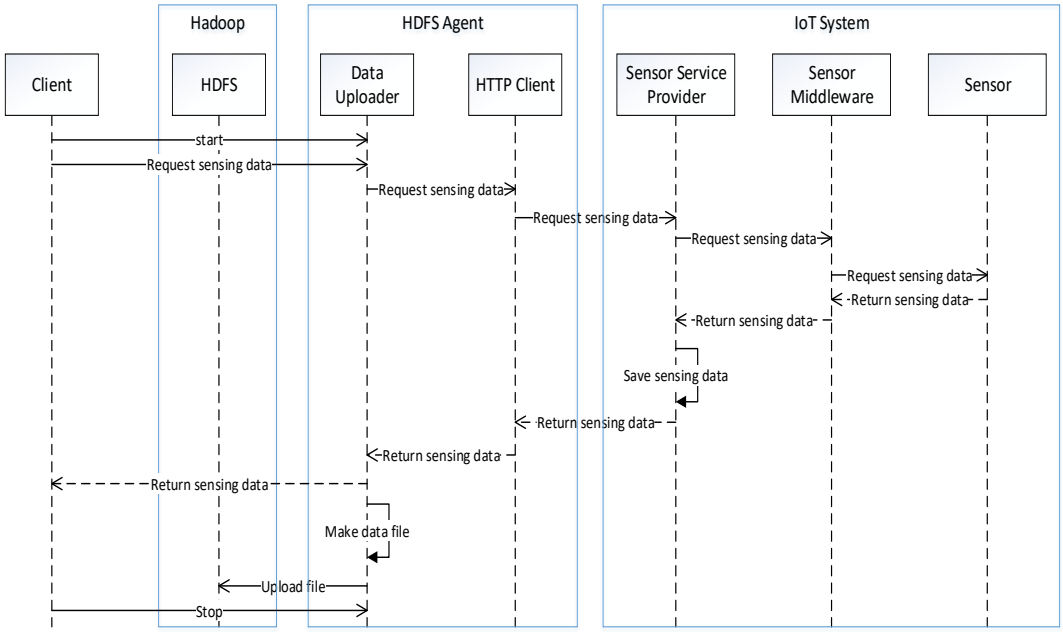

Figure 6. Sequence Diagram of this System

5. Implementation and Results

Table 1 shows the development environment of sensor middleware and platform. All of the design implemented on Windows 10 operating system, we used Microsoft Visual Studio 2015 as development tool and build the database in Microsoft SQL Server 2016.

Table 1. Development Environment of Sensor Middleware and Platform

Components	Version
Operating System	Windows 10
Microsoft Visual Studio	2015
Microsoft SQL Server	2016

Table 2 shows the development environment for HDFS Agent. All of the design also implemented on Windows 10 operating system, we used Spring Tool Suite 3.8.4 as development tool and the implementation is based on java 1.8.

Table 2. Development Environment of HDFS agent

Components	Version
Operating System	Windows 10
Java	JRE 1.8
Spring Tool Suite	3.8.4

Table 3 shows the configuration environment for Hadoop. We installed Hadoop 2.7.3 on Windows 10 operating system based on java 1.8. And we used Spring Tool Suite 3.8.4 as development tool.

Table 3. Installation Environment of Hadoop

Components	Version
Operating System	Windows 10
Java	JRE 1.8
Hadoop	2.7.3
Spring Tool Suite	3.8.4

Figure 7 shows the sensor which connect to Sensor Middleware via Serial Port.

Figure 7. Temperature Sensor

Finally, we need to run Client, the results of the client are shown in figure 8. We need to choose a sensor in the combo box, if the sensor is not in using, we cannot click the "start" button to start working. After we choose a sensor, the sensor information text area will show the information of this sensor, this process is shown in figure (a). Click "start" button will get sensing data through Sensor Platform, make data files and upload files to HDFS, this process is shown in figure (b). And as shown in figure (c), click "Stop" button will stop all the process.

Figure 8. Client Results

After finish running the whole system, we can check the files storage situation in our local file system, like the Figure 9 shows.

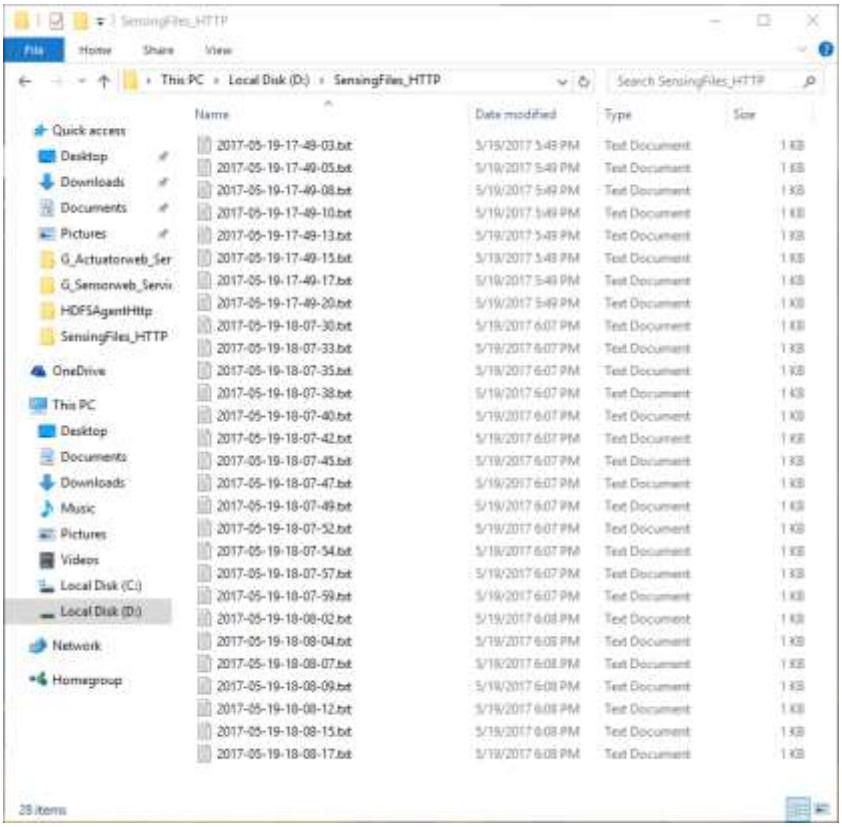

Figure 9. Sensing File List in Local File System

And we can also check the files storage situation in Hadoop Distributed File System, like the Figure 10 shows.

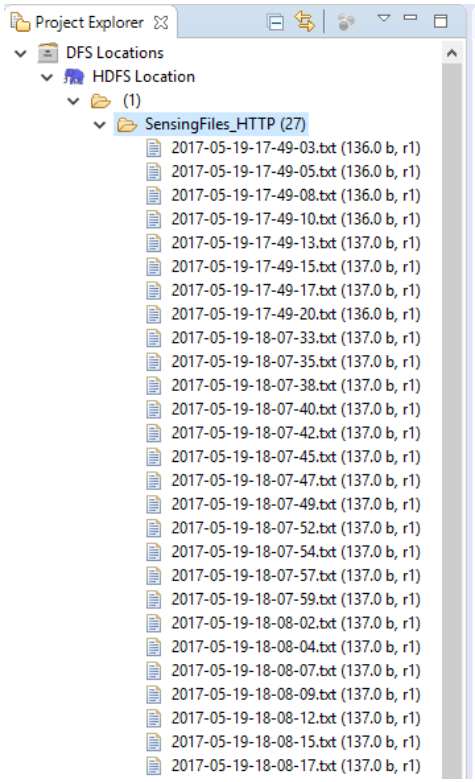

Figure 10. Sensing File list in Hadoop Distributed File System

In file list, each file's name is current time of sensing data. And the content is a string split by commas including sensing time and sensing data. Figure 11 shows the sensing data on 17:49:03, May 19, 2017.

Figure 11. A File of the Sensing List in HDFS

6. Conclusion

In this paper, we have present a hierarchical IoT federation architecture based on local Cloud, platform and middleware for massive context data acquisition in multiple sensor networks. During the study, we have a deep understanding about the using of multiple sensor networks and local IoT cloud. In future, we will try to extend this system by the implementation of a more complete IoT system, which can analysis the environment data directly and make this system more useful.

Acknowledgments

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), and this research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2017-2014-0-00743) supervised by the IITP(Institute for

Information & communications Technology Promotion). Any correspondence related to this paper should be addressed to DoHyeun Kim; kimdh@jejunu.ac.kr.

References

[1] J. Song, A. Kunz, M. Schmidt and P. Szczytowski, "Connecting and managing m2m devices in the future internet", Mobile Networks and Applications, **(2013)**, pp. 1–14.
[2] S. Husain, "Interworking architecture between oneM2M service layer and underlying networks", Globecom Workshops (GC Wkshps), 2014. IEEE, **(2014)**.
[3] J. Kim, J. Yun and S. C. Choi, "Standard-based IoT platforms interworking: implementation, experiences, and lessons learned", IEEE Communications Magazine, vol. 54, no. 7, **(2016)**, pp. 48-54.
[4] J. Flanders, "RESTful. NET: Build and Consume RESTful Web Services with", NET 3.5. "O'Reilly Media, Inc.", **(2008)**.
[5] T. White, "Hadoop: The Definitive Guide", O'Reilly Media, Inc., **(2009)**.
[6] https://www.tutorialspoint.com/hadoop/hadoop_introduction.htm
[7] C. Lam, "Hadoop in Action", 2010-12-22.
[8] H. Lu, C. Hai-Shan and H. Ting-Ting, "Research on hadoop cloud computing model and its applications", Networking and Distributed Computing (ICNDC), 2012 Third International Conference on. IEEE, **(2012)**.
[9] L. Jiang, "An IoT-oriented data storage framework in cloud computing platform", IEEE Transactions on Industrial Informatics, vol. 10.2, **(2014)**, pp. 1443-1451.
[10] H. Derhamy, "A survey of commercial frameworks for the Internet of Things", Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on. IEEE, **(2015)**.

Authors

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

Songai Xuan, she is currently perusing M.S. in Department of Computer Engineering, Jeju National University, Republic of Korea. She received her B.S. degree in Computer Science from YanBian University, China in 2016.

