International Journal of Smart Home
Vol. 10, No. 11, (2016), pp. 69-78
http://dx.doi.org/10.14257/ijsh.2016.10.11.07

A Resource-Efficient System Architecture for Processing Various Sensor Data in Smart home Environment

Changmin Lee[1], Changhyun Byun[2] and Hyeonjeong Shin[3*]

[1]Center of Research & Development, Fasoo.com, Inc.,396 World Cup Buk-ro Mapo-gu, Seoul 121-795, Republic of Korea
[2]Center of Research & Development, Fasoo.com, Inc.,396 World Cup Buk-ro Mapo-gu, Seoul 121-795, Republic of Korea
[3]Department of Computer Science, Shinhan University,30,beolmadeul-ro 40beon-gil, Dongducheon-si, Gyeonggi-do 483-777,Republic of Korea
[1]cmlee@fasoo.com,[2]cbyun@fasoo.com,[3]hjshin@shinhan.ac.kr

Abstract

The Internet of Things (IoT) technology is gaining vast popularity and it is expected to become ubiquitous in the near future. It envisions the idea of a fully connected network of smart objects, enabling cooperative and intelligent distributed functionalities. Especially, smart homes provide innovative, automated and interactive services for residential customers through distributed and collaborative operations. In this paper the challenge of processing large amounts of sensor data at network gateways with limited memory and computing power is described. A resource-efficient system to process various heterogeneous data is introduced.

Keywords: *internet of things, smart home, data processing, resource efficiency*

1. Introduction

Internet of Things (IoT) technology is receiving growing attention from research, industry and government organizations [1-2]. The IoT paradigm consists in fully connected smart objects (such as sensors, smartphones, embedded systems), providing integrated services. The number of connected smart objects is growing impressively, expecting to exceed 26 billion by 2020. Thus, smart objects will soon generate large amounts of data traffic requiring appropriate management [3].

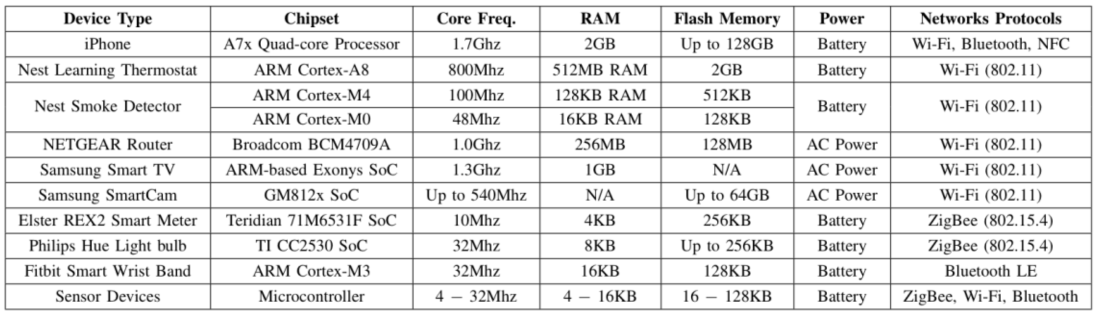
Two main approaches exist to handle potential large amount of sensor data generated by the different smart objects. In current IoT approaches, time-series sensor data generated from smart objects are transmitted to cloud servers which process the received data and make decisions for appropriate actions. However, as more and smarter objects are developed and possibly generate data for each and every event, transmitting the entire data to outside servers may not be practical. Furthermore, some of the actions may need to be confined within the corresponding IoT environment, such as a smart home. As an alternative, various sensor data can be locally processed at the network gateways and only the relevant information for the target application is transmitted to the outside server. The latter approach mandates a fast and limited-memory processing algorithm.

Additionally, the processing at gateway level of sensor data coming from multiple sources is further challenged by the following issues: (1) resource constraints of gateways: As can be seen in Figure 1, due to the self-organizing nature of IoT networks, gateway functionalities could be assigned to devices with very limited memory and computing power capabilities, therefore incapable of performing standard data

[*]Corresponding Author

processing; (2) heterogeneity of data types: the smart home environment relies on a vast and heterogeneous set of objects, each possibly generating different data types; (3) variability of sensor data inter-arrival times: due to the different functionalities implemented, some smart objects may generate data with regular intervals, as opposite to devices generating data on event basis.

With the objective of overcoming the above identified challenges, in this paper we propose a fast and memory-efficient system for processing various heterogeneous data, indispensable for resource-constrained devices.



Device Type	Chipset	Core Freq.	RAM	Flash Memory	Power	Networks Protocols
iPhone	A7x Quad-core Processor	1.7Ghz	2GB	Up to 128GB	Battery	Wi-Fi, Bluetooth, NFC
Nest Learning Thermostat	ARM Cortex-A8	800Mhz	512MB RAM	2GB	Battery	Wi-Fi (802.11)
Nest Smoke Detector	ARM Cortex-M4	100Mhz	128KB RAM	512KB	Battery	Wi-Fi (802.11)
	ARM Cortex-M0	48Mhz	16KB RAM	128KB		
NETGEAR Router	Broadcom BCM4709A	1.0Ghz	256MB	128MB	AC Power	Wi-Fi (802.11)
Samsung Smart TV	ARM-based Exonys SoC	1.3Ghz	1GB	N/A	AC Power	Wi-Fi (802.11)
Samsung SmartCam	GM812x SoC	Up to 540Mhz	N/A	Up to 64GB	AC Power	Wi-Fi (802.11)
Elster REX2 Smart Meter	Teridian 71M6531F SoC	10Mhz	4KB	256KB	Battery	ZigBee (802.15.4)
Philips Hue Light bulb	TI CC2530 SoC	32Mhz	8KB	Up to 256KB	Battery	ZigBee (802.15.4)
Fitbit Smart Wrist Band	ARM Cortex-M3	32Mhz	16KB	128KB	Battery	Bluetooth LE
Sensor Devices	Microcontroller	4 − 32Mhz	4 − 16KB	16 − 128KB	Battery	ZigBee, Wi-Fi, Bluetooth

Figure 1. Smart Home Device Capabilities [3]

2. Related Work

2.1. Smart Home

A smart home comprises of a multitude of connected devices belonging to different application areas. These devices are characterized by heterogeneous hardware and software resources, and they support different communication technologies. By coexisting, interacting, and cooperating among each others, these devices form a distributed heterogeneous network. As an example, Figure 1 shows a typical smart home scenario, consisting of many connected devices belonging to different applications, communicating among each other using different technologies and connected to few gateways/routers which provide connectivity to outside networks such as Internet.

2.1.1. Smart Home Applications: The smart home concept encloses multiple applications belonging to the different areas, interacting with each other.

- **Lighting control**: Intelligent home lighting systems provide automated lighting control through LED lights and controllers detecting ambient conditions such as the presence of users or sunlight. The system automates the actions of turning on and off the lights and controlling their brightness according to the user's preferences and activities or energy savings rules.
- **Appliance control**: Home appliances such as refrigerators, ovens, and washing machines contain embedded devices for the control of their functioning. In the past, these embedded devices constituted stand-alone systems, each providing dedicated control only for a single appliance. The significant dropping in prices of the transceiver chips, with the consequent inclusion of communication functionalities into these embedded devices, have created unlimited possibilities for cooperative and automated appliance control. In example, high-energy consuming appliances such as washing machines and ovens could schedule operations during off-peak energy rates. Also, washing and drying machines can share laundry setting information.
- **Entertainment**: A typical house has one or more entertainment center(s) consisting of different devices (*e.g.*, Digital TV, DVD player, satellite

decoder, digital multimedia receiver) together with multiple media players such as tablet PCs, smartphones, MP3 players. Entertainment systems in smart homes provide connectivity, access to shared resources and distribution of content according to users' preferences.

- **Safety system**: A safety system of a smart home consists of smoke detectors, zone intrusion detectors, burglar alarms, surveillance cameras, interconnected and integrated with the communication infrastructure and the information systems. Alarms are properly and timely reported to the intended receiver(s) which could be the residents or a police station. Climate control: Heating, ventilation and air conditioning (HVAC) can be integrated and controlled jointly in an automated way, with the ultimate goal of providing customized climate control while saving energy. Room temperatures can be regulated based on human presence, in example providing at nighttime higher temperatures in the bedrooms, or lowering the climate control when the house is empty and activating it just before the residents come back home.
- **Assisted living**: In a smart home assisted leaving and telecare can be provided for elder people to assist and monitor them, with the ultimate goal of permitting them to live longer in their houses. Sensors can recognize the activity of a person and assistance can be provided accordingly. In example, detecting a person waking up should automatically turn on the lights also unexpected behaviors such as a person falling on the floor, shall be detected and reported to the emergency units promptly.

2.1.2. Devices: Several types of devices can coexist in a smart home. Possibilities for new smart devices are endless, since any residential device with the addition of intelligent computational capabilities can become part of a smart home. Here the main devices currently available are listed, grouped by target application.

- **Lighting control**: Light bulbs, light strips.
- **Appliance control**: laundry machines, refrigerators, ovens. Safety system: smoke detectors, intrusion detection devices, security cameras, smart door-locks.
- **Entertainment**: Smart-TVs, set top-boxes, media players, laptops, wireless speakers.
- **Health and assisted living**: smart wristbands, portable ECGs, pulse oximeters.
- **Network devices**: gateways, routers, network storage devices, mobile phones, printers.

2.1.3. Operating Systems: Due to the size, energy, computation and storage limitations typical of the embedded systems implemented in the majority of the devices in a smart home, their operating systems (OSs) must be extremely lightweight, while supporting the rich set of application, communication and security features needed. There exist a few operating systems for IoT devices that can currently be adopted for research and development purposes. Many researchers and developers are working for applying their innovative solutions into to these OSs, with the ultimate goal of testing them for production. Following, the main OSs applicable for smart home devices are presented.

- **Contiki** [5]: this open source OS for IoT applications, which provide connectivity and applications' support for low-cost, low-power micro-controllers. Contiki represents the most adopted solutions for developing IoT solutions, It is written in C language and it has been ported to a number of microcontroller architectures, including the Texas Instruments MSP430, Atmel AVR, and the ESB platform [5]. Probably influenced by its popularity, there have been numerous studies of the protocol implementation vulnerabilities of Contiki [6].

- **Tiny OS** [7]: this free and open-source tool consists of a component-based OS and a development platform targeting wireless sensor network (WSN) applications. TinyOS is implemented using nesC programming language as a set of cooperating tasks and processes. Started as a collaboration between the University of California, Berkeley in co-operation with Intel Research and Crossbow Technology, it has since grown to be an international consortium, the TinyOS Alliance.
- **RIOT OS** [8]: it is a microkernel-based OS which match the specific software requirements of typical IoT devices. Thanks to its modular implementation, RIOT OS guarantees minimum memory usage and permits ad-hoc customizations and configurations to meet the specific application requirements. Because of the minimized kernel size, RIOT OS requires only few hundreds bytes of RAM and storage. Although RIOT OS is still in development, it has the potential to become the preferred OS for IoT devices, due to its lightness and customization ability.

In addition to the solutions listed above, the IoT devices currently on the market implement proprietary OSs. Although developed mainly for research purposes and supported by voluntary contributors, the open source OSs listed above represent mature and valid software solutions possibly ready to be customized to match the specific security requirements of smart devices to be developed.

2.1.4. Communication Protocols: The heterogeneity of the hardware and software components in a smart home also reflects in the communication protocols available. Different solutions are used to transport information between devices, depending on traffic characteristics, device capabilities, and surrounding environment. The main communication protocols used in a smart home are briefly described next, grouped according to the OSI model classification.

2.1.4.1. Physical and Data Link Layers

- **IEEE 802.15.1 Standard [9]**: it defines the wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPANs) targeting high- speed data transfers and multimedia distribution for home entertainment over short distances around 10 meters. It operates using frequency hopping spread spectrum (FHSS) to combat interference and jamming, achieving a maximum data rate of 1Mb/s over 1MHz channels in the unlicensed industrial, scientific and medical (ISM) band at 2.4 GHz. The Bluetooth and Bluetooth LE (Low-Energy) protocol architectures build on top of the IEEE 802.15.1 PHY and MAC layers. The IEEE 802.15.1 Standard addresses the network security aspects of authentication (through a challenge-response 128-bit private key scheme) and encryption (through variable size up to 128 bits private key), without addressing message integrity issues. IEEE 802.15.4 Standard: it specifies MAC and PHY protocols for low-rate WPANs, targeting at networks characterized by devices such as sensors and embedded devices with limited traffic, low energy available and constrained memory and processing capabilities. The standard achieves a maximum data rate of 250 Kb/s transmitting on 5MHz channels in the 2.4 GHz ISM band using Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). The ZigBee high level communication protocol suite is based on the underneath IEEE 802.15.4 physical and data link layers. Eight different suites provide security guarantees for the applications, ranging from adopting encryption only (AES-

CTR), authentication only (AES-CBC-MAC), or encryption and authentication (AES- CCM).

- **IEEE 802.15.4 Standard [10]**: IEEE 802.15.4 specifies MAC and PHY protocols for low-rate WPANs, targeting at networks characterized by devices such as sensors and embedded devices with limited traffic, low energy availability, and constrained memory and processing capabilities. The standard achieves a maximum data rate of 250 Kb/s transmitting on 5MHz channels in the 2.4 GHz ISM band using Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA).
- **IEEE 802.11 Standard [11]**: this technology provides MAC and PHY specifications for high-rate communications in Wireless Local Area Networks (WLANs) with ranges from 20 to 250 meters. In its 802.11a and 802.11g specifications it achieves 54 Mb/s data transmissions using CSMA/CA on 20 MHz channels, through the use of Orthogonal Frequency Division Multiplex (OFDM) modulation. The latest developments of 802.11n and 802.11ac significantly increase the data rates, setting the theoretical upper bounds to 150 and 866.7 Mb/s respectively. These improvements are reached through the use of larger bandwidths through channel aggregation up to 40 MHz for 802.11n and 160 MHz for 802.11ac in the 2.4, 3.6, 5 GHz frequency bands and by adding multiple-input multiple-output (MIMO) antenna functionalities. In its first inception, IEEE 802.11 Standard used the Wired Equivalent Privacy (WEP) method to encrypt data, which was shown to be severely weak. In later implementations, robust solutions such as Wired Equivalent Privacy (WPA) and 801.11i emerged.

2.1.4.2. Network and Transport Layers: Network layer protocols for smart home applications can either belong to custom designed solutions to address the specific application requirements (such as Zigbee or Bluetooth network layer protocols) or implement IP-based networking functionalities using the underneath physical and data layer solutions described above. In this context, the latter case has gained increasing interest due to the advantage of supporting IPv6 functionalities such as large address space, stateless and stateful address configuration, needed by the IoT applications. Protocols such as IPv6 over Low power Wireless Personal Area Networks (6loWPAN) [12], Routing Protocol for Low power and Lossy Networks (RPL) [13], and Multicast Protocol for Low power and Lossy Networks (MPL) [14] are examples of future adaptation and networking protocols tailored for IoT applications, including smart homes.

Regarding the transport layer, UDP is preferred for resource-constrained devices, since it saves power by going to sleep after transmitting a packet, oppositely to TCP, which enforces to stay awake to process acknowledgments. At the same time, UDP lacks in reliability, which may be required at different levels, depending on the specific application considered. The reliability requirements of the transport protocol depend on the target application. For example, different mechanisms could be implemented for packet-based applications (which require all packets be received reliably at the destination) versus event-based applications (which require events, and not necessarily all individual packets, be reliably reported to the destination). An event-based application might call for less complex transport mechanisms.

2.1.4.3. Application Layer: The main application protocols for IoT and smart home environments are summarized below.

- **eXtensible Messaging and Presence Protocol (XMPP) [15]**: this open-standard protocol implements a message-oriented middleware based on XML and it is widely tested and adopted for IoT applications such as smart grids and remote monitoring.

- **Constrained Application Protocol (CoAP) [16]**: this software protocol is targeted for resource-constrained electronics devices that need to be controlled or supervised remotely, through Internet-based networks. CoAP is designed to minimize the complexity of mapping with HTTP, while also meeting specialized requirements such as multicast support, low overhead, and implementation simplicity.
- **MQ Telemetry Transport (MQTT) [17]**: this connectivity protocol represents an extremely lightweight publish/subscribe messaging transport designed for low complexity, low power and low footprint implementations. It runs on connection-oriented transport layer protocols such as TCP or on non-TCP/IP networks through its MQTT-S variant.

2.2. Rule-Engine

There are many kinds of rule engines available. Among the free rule-based and java-based inference engines, Drools and Jess thoroughly studied and described in the following table. The table gives an exhaustive comparative chart for the selected inference engines measured on different performance metrics (See "Table 1.").

Table 1. Comparative Chart

Rule Engine	Drools	Jess
Algorithm	RETE Algorithm	RETE Algorithm
OWL-DL Entailment	No	Yes
Java Support	Yes	Yes
Rule Support	DRL (Own Rule Format)	SWRL
Version	6	7
Licensing	Free / Open source	Academic use only

2.2.1. Drools: Drools [18] is a java-based object-oriented rule engine, which is open-source, so we can freely use and modify the code in java. It uses an optimized version of the RETE algorithm [4], called RETE-Object-Oriented algorithm to support high performance. It has its own writing rules, which is called DRL (Drools Resource Language) and is flexible enough to match the semantics of problem domain with DSLs (Domain Specific Languages), graphical editing tools, web based tools and developer productivity tools. It has useful features which include rule debugging and rule authoring tools like IDE plug-in.

2.2.2. Jess: Jess [19] is also a java-based rule engine. It uses an enhanced version of the RETE algorithm to process. It can also directly manipulate and reason about Java objects. It is also a powerful Java scripting environment, from which you can create Java objects, call Java methods, and implement Java interfaces without compiling any Java code. It supports SWRL (Semantic Web Rule Language) [20] and the rules can be expressed in XML or Lisp languages. Although Jess is not open-source, it is available with no cost for academic use but we cannot get and modify the source code.

3. A Resource-Efficient System Architecture

3.1. Data Format in Smart Home Environment

There exist many commercial smart home products such as NEST smoke detector, NEST smart thermostat, Bluetooth wrist bands, and *etc*. Each product has different data types; XML, JSON, and their own data structure to transfer information.

3.2. Data Processing Gateway

Our resource-efficient system utilizes the known RETE algorithm [4], a pattern-matching solution developed for implementing production rule systems, constituting the core engine of most of the rule engine systems. Our system generates a set of memory nodes, called beta nodes, interconnected as an acyclic directed graph representing higher level rule sets for handling the data from the smart objects (See “Figure 2.”). Each memory node is set with a fixed size, based on the maximum number of stored data calculated based on the effective alive time of each sensor data.

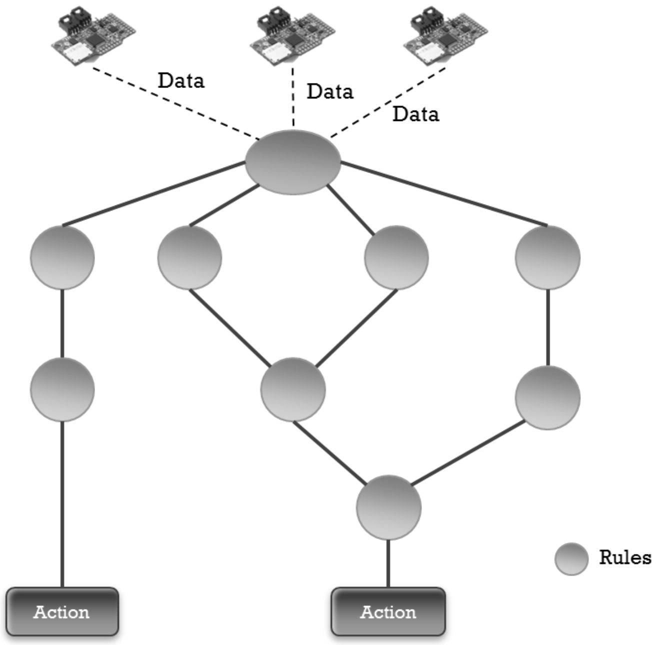


Figure 2. Example of RETE-Network

3.3. Architecture of Data Processing in Gateway

Gateway is performed by interaction of four sub modules. Figure 3, shows data flow, its directions, and an architecture of data processing. The system consists of four sub modules, Crawling module, Rule-based module, Storage module, and Visualization module. The first module is called Crawling Module, which receives data from various smart home objects and sends to main data controller called Backbone running on the system. Secondary, Rule-based Module, an inference engine also known as rule engine, has two main functionalities, which are data filtering and data analyzing. Filtering data is performed by filtering rule. Once the Rule-based Module receives data from smart devices through the Backbone, the module performs filtering process to leach unnecessary data by checking filtering rules defined and sends back results to the Backbone to save the filtered data into database. Another main functionality of the Rule-based Module is data analyzing. If there is already enough filtered and stored data in the database, the Rule-based module starts to discover new knowledge by following rules. When any new knowledge is generated during analyzing process, the module sends the data to the Backbone to save the knowledge into database. The main role of the Storage module is receiving data from the Backbone and storing the data into the database. Finally, the Visualization module offers users to adjust rules for data processing and observe the statistics of data processing rates.

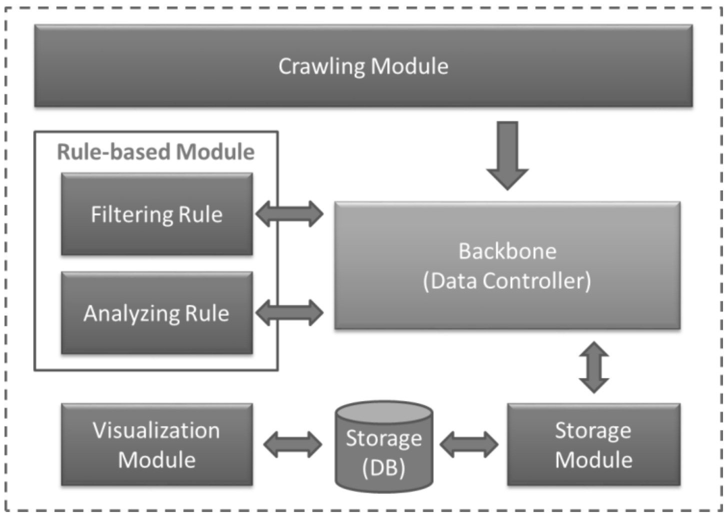


Figure 3. Architecture of Data Processing in Gateway

4. Testbed

To test proposed system, Zolertia Z1 Sensor node and gateway devices are chosen as test devices. Figure 4, depicts our experimental environment containing multiple sensor motes and a gateway. Each of motes will send their sensing data using UDP communication. If any motes are too far from the gateway then they will send the data to nearby mote to forward to the gateway so that data can be guaranteed to be delivered to the gateway. There are two scenarios to test performance of our algorithm, uniform interval settings for all devices and different interval settings for each device.

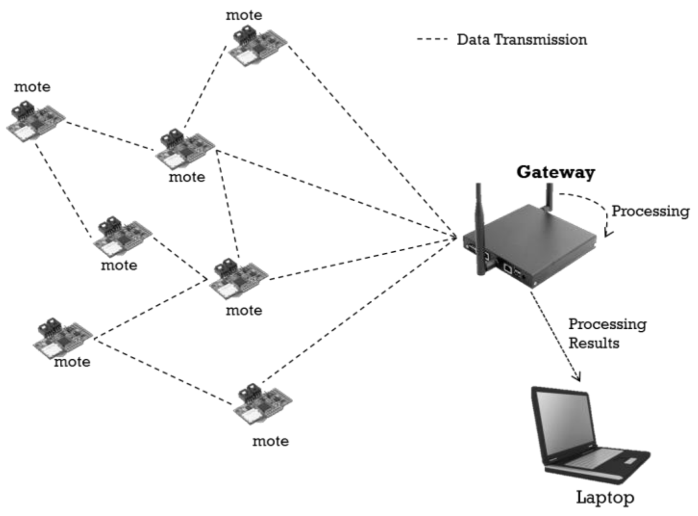


Figure 4. Testbed for Resource-Efficient Data Processing System

5. Conclusion

In this paper, we introduced architecture of data processing in gateway for smart home environment. We adopted rule based processing module to process heterogeneous data type from various smart home products. For this paper, we chose Zolertia Z1 as test sensor motes instead of actual commercial products. Zolertia products provides various features to test our system in smart home environment, and are developed in C language thus it`s easy to modify motes to be adopted to our resource-efficient system.

6. Future Work

There exist few challenges to improve our system; 1) operating with existing commercial products (such as NEST products), and utilizing different gateway devices other than Zolertia product. 2) It would be excellent to work with IEEE 802.11 network and IEEE 802.15.4 together since we only utilize our system to work with IEEE 802.15.4 network only. 3) A quantitative performance analysis of information processing for our system need to be carried out for devices with different computational capabilities and using various synthesized and real data types.

Another improvement to this research is to adapt a built-in data-mining module. Applying data-mining techniques to the IoT data can yield interesting perspectives to understanding individual and human behaviour, detecting group and community, or discovering new pattern. Therefore, a system that supports collecting and mining the IoT data in efficient ways is needed for researchers to be able to use this untapped resource.

References

[1] L. Atzori, A. Iera and G. Morabito, “The internet of things: A survey,” Computer networks, vol. 54, no. 15, **(2010)**, pp. 2787–2805.

[2] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions”, Future Generation Computer Systems, vol. 29, no. 7, **(2013)**, pp. 1645-1660.

[3] C. Lee, L. Zappaterra, K. Choi and H. A. Choi, “Securing smart home: Technologies, security challenges, and security requirements”, Proceedings of the Communications and Network Security, San Francisco, CA, USA, **(2014)** October 29-31, pp. 67-72.

[4] Forgy, Charles L, “Rete: A fast algorithm for the many pattern/many object pattern match problem”, Artificial intelligence, vol 19, no. 1, **(1982)**, pp. 17-37.

[5] A. Dunkels, B. Gronvall and T. Voigt, “Contiki-a lightweight and flexible operating system for tiny networked sensors”, Proceedings of the 29th Annual IEEE International Conference in Local Computer Networks, **(2004)**, pp. 455-462.

[6] K. Chugh, A. Lasebae and J. Loo, “Case study of a blackhole attack on 6lowpan-rpl”, Proceedings of 6th International Conference on Emerging Security Information in SECURWARE, **(2012)**, pp. 157-162.

[7] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer and D. Culler, “Tinyos: An operating system for sensor networks”, In Ambient intelligence, Springer, **(2005)**, pp. 115-148.

[8] E. Baccelli, O. Hahm, M. W¨ahlisch, M. Gunes and T. Schmidt, “Riot: One os to rule them all in the iot”, **(2012)**.

[9] J. S. Lee, Y. W. Su and C. C. Shen, “A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi”, Proceeding of 33rd Annual Conference of the IEEE in Industrial Electronics Society, IECON 2007, **(2007)**.

[10] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta and Y. F. Hu. “Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards”, Computer communications, vol. 30, no. 7, **(2007)**, pp.1655-1695.

[11] B. P. Crow, I. Widjaja, L. G. Kim and P. T. Sakai, “IEEE 802.11 wireless local area networks”, IEEE Communications magazine, vol. 35, no. 9, **(1997)**, pp. 116-126.

[12] J. Hui, D. Culler and S. Chakrabarti, “6lowpan: Incorporating ieee 802.15. 4 into the ip architecture”, IPSO Alliance White Paper, **(2009)**.

[13] T. Winter, “Rpl: Ipv6 routing protocol for low-power and lossy networks”, **(2012)**.

[14] J. Hui and R. Kelsey, “Multicast protocol for low power and lossy networks (mpl)”, **(2013)**.

[15] P. Saint-Andre, “Extensible messaging and presence protocol (xmpp): Core”, **(2011)**.

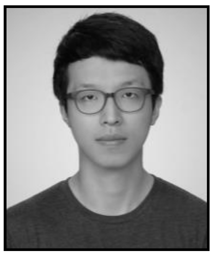
[16] Z. Shelby, K. Hartke, C. Bormann and B. Frank, “Constrained appli- cation protocol (coap), draft-ietf-core-coap-13”, Orlando: The Internet Engineering Task Force–IETF, **(2012)** December.

[17] U. Hunkeler, H. L. Truong and A. S. Clark, “MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks”, Proceeding of 3rd international conference in Communication systems software and middleware and workshops, COMSWARE 2008, IEEE, **(2008)**. pp. 791-798.

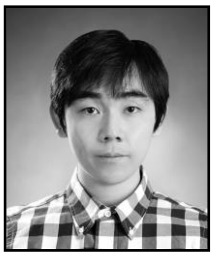
[18] P. Browne, “JBoss Drools business rules”. Packt Publishing Ltd, **(2009)**.

[19] E. Friedman, "Jess in action: rule-based systems in java", Manning Publications Co., **(2003)**.
[20] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B.Grosof and M. Dean, "SWRL: A semantic web rule language combining OWL and RuleML", W3C Member submission, vol. 21, **(2004)**, pp. 79.

Authors



Changmin Lee, He received B.S. in computer science from National Institute for Lifelong Education in Seoul, South Korea, M.S. degree in computer science from Towson University, U.S.A. Ph.D. degree in computer science from George Washington University, U.S.A. Currently, he is a research engineer in development division at Fasoo.com, Inc. His research interests include internet of things security, network security, processing algorithm, machine learning, data mining, and digital rights management.



Changhyun Byun, He received B.S. in computer science from National Institute for Lifelong Education in Seoul, South Korea, M.S. and D.Sc. degrees in computer science from Towson University, U.S.A. Currently, he is a research engineer in development division at Fasoo.com, Inc. His research interests include internet of things security, processing algorithm, machine learning, data mining, and digital rights management.

Hyeonjeong Shin, He is currently a professor in School of Convergence Engineering/Computer Science & Engineering at ShinHan University. His research interests include internet of things security, machine learning, and data mining.