Intelligent Service Middleware Based on Sensor in IoT Environments

Jong-Hyun Park
SW Competency Enhancement Center
Hanshin University
Osan-Si, Gyeonggi-Do, 18101, Republic of Korea
Republic of korea
Jh7park@hs.ac.kr

Abstract: - In IoT(Internet of Things) environment, a number of sensors and networks exist and they have various and heterogeneous characteristics. Applications which provides a variety of services based on the sensor networks also have different service requirements. Therefor a middleware that is located between sensor networks and application systems is needed for integrating two layers. This paper proposes a general purpose middleware for providing intelligent services based on heterogeneous sensors existing in IoT environment. The proposed middleware acquires and manages sensing data in real-time. The middleware stores and manages heterogeneous sensors, node, network metadata. In addition, the middleware infers the situation based on ontologies and rules and provides intelligent services.

Key-Words: - Intelligent Middleware, Sensor-Based Middleware, ontology, Inference system

1 Introduction

In the IoT environment, the purpose of a lot of applications is to provide a variety of services to users by combining various devices including sensors. The sensors and sensor networks in the environment are heterogeneous and very diverse. There are also numerous applications for providing sensor-based services and the requirements vary. Therefore, a middleware for the integration between the two layers is essential. However, the standard for middleware is not fixed and the middleware focused on an intelligent service is under study.

This paper proposes an intelligent service middleware based on sensor(ISMS) that provides intelligent services based on various kind of sensors and network in IoT environment. The paper focuses on three systems in ISMS. The first is a sensor data management system(SDMS) which processes and manages sensor data in real-time. The SDMS processes the sensing data consecutively at a predetermined period or only when an event occurs. The second system is a metadata management system(MMS) that stores and manages metadata of sensors, nodes and networks. In order to provide various services, the information about sensors, nodes and networks should be integrated and shared. However, many applications use sensor metadata locally. Therefore, this paper proposes a method for interoperability. The last one is an intelligent service management system (ISMS) that provides intelligent serves based on sensor data and metadata. Sensor-based application systems define and provide intelligent services according to their characteristics locally. However, the paper proposes a general purpose middleware for providing various intelligent services

The remainder of this paper is organized as follows. Section 2 describes the Related Work. Section 3 shows the architecture of the middleware and three systems. Section 4 describes the inference for Intelligent service processing in ISMS. Section 5 shows an application system for sensor data monitoring and finally Section 6 provides concluding remarks

2 Related Work

There are many of the studies on IoT middleware deal with the heterogeneous sensor and device management and integration. Some of studies proposes methods to process large amounts of sensor data [1] or real-time processing [2]. [3, 4] is a middleware focused on the integration of sensor and sensor network. However, the focus of the middleware proposed in this paper is to provide intelligent services based on sensor and sensor network. [5] proposes a service-oriented middleware for integrated management of sensor data but only for managing disaster situation. In other words, their middleware deal with a specific services or domain. As in [6, 7], many of the IoT middleware platform are built for specific scenarios or services [8]. The middleware in this paper is for general intelligent services. Mobile sensor data processing engine in [9, 10] is a middleware WSN-centric middleware for

IoT middleware. Their plugin architecture improves the scalability and user friendliness of the middleware; this is because plugins for heterogeneous devices are easier to build and available in easily accessible places. [19] provides middleware services to allow creation of products and solutions for IoT. It offers developers a standards-based directory, data, and business service. Its web-based tools simplify data and control other complexities of IoT application development, and it also supports multiple data formats. [20] is a cloud-based platform for IoT environments. They focus on M2M communications, cost effective M2M application development, scalability, and ease of use. [9, 10, 11, 12] proposes WSN or cloud system-based IoT middleware which uses a centralized server to provide services. Therefore, they should consider problems such as transmission delay, reliability, and energy consumption. The middleware in this paper, however, can be used in both non-server and server environments.

3 Middleware for Intelligent Service based on Sensors

The ISMS mediates between application systems and sensor networks and guarantees the independence of the service and sensor.

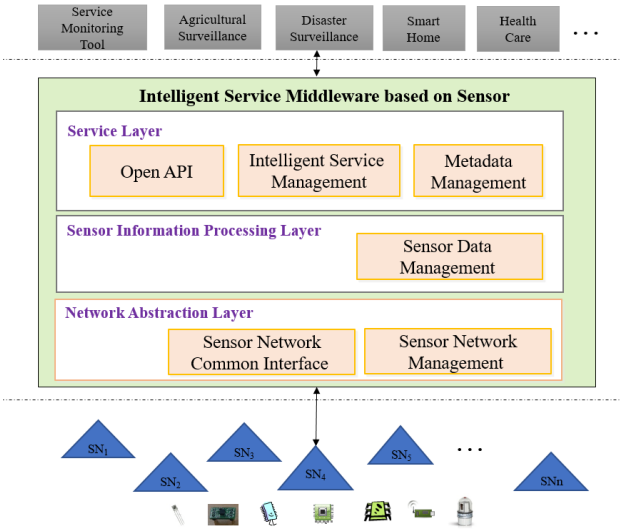

Fig. 1. Intelligent Service Middleware based on Sensor

Figure 1 shows overview of the ISMS. The middleware is composed the Service Layer, Sensor Information Processing Layer, and Network Abstract Layer. The Network Abstract Layer provides a common interface for communicating various heterogeneous sensor networks and the Sensor Network Management monitors and manages sensor networks. The Sensor Data Management in the Sensor Information Processing Layer manages sensor data and processes queries for sensor data. The Service Layer includes the Open API to support various services. The Intelligent Service Management processes intelligent services requested from applications. The Metadata Management stores and manages the metadata of sensor, node and sensor network and provides metadata services

3.1 Sensor Data Management System

The SDMS(Sensor Data Management System) gathers and manages sensor data. Also the SDMS processes queries for sensor data. Figure 2 is an architecture of the SDMS. Users can request services that uses low-level data like sensor data such as sensor data monitoring service. When a user requests service to the SDMS through the Service Layer, the SDMS processes the request using current sensor data it owns, or acquires and processes additional information through the Network Abstract Layer. The following are the functions of the modules that make up the SDMS.

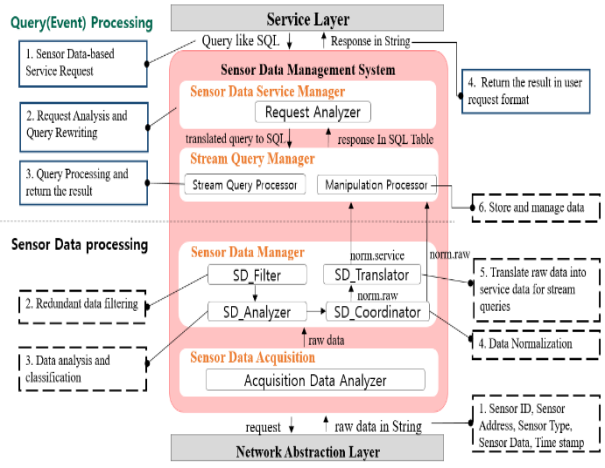

Fig. 2. Architecture of the Sensor Data Management System

- Sensor Data Service Manager: Analyzes the various types of queries requested through the Service Layer and delivers the queries to the Stream Query Manager.
- Stream Query Manager: Processes various queries requested by the Sensor Data Service Manager, and stores and manges the formatted sensor data that is pushed from the Sensor Data Manager.
- Sensor Data Manager: validates sensor data acquired from the Sensor Data Acquisition, extracts

unit field information, and generates normalized records. Generates the processed data and push it to the Stream Query Manager along with the raw data.
- Sensor Data Acquisition: Collects and analyzes various kinds of sensor data

3.2 Metadata Management System

The metadata Management System(MMS) manages and retrieves metadata of sensors, nodes, and sensor networks. This paper adopts the SensorML (Sensor Model Language) to describe sensors and nodes because the SensorML is one of the suggested standards by Open Geospatial Consortium(OGC) [14] for Sensor Web Enablemen (SWE) and provides a standard model that includes the data measured from the sensor and sensor information. The SensorML models sensors and nodes and describes the processing rules between them, but does not include network information. Therefore, this paper defines sensor network metadata as shown in Figure 3 and is written in XML for interoperability.

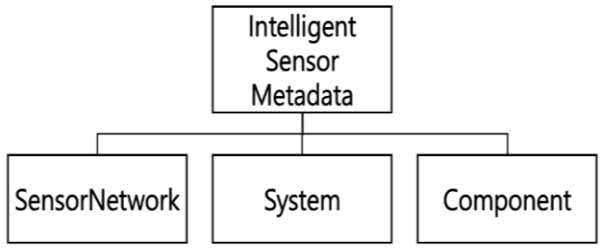

Fig.3. Intelligent Sensor Metadata

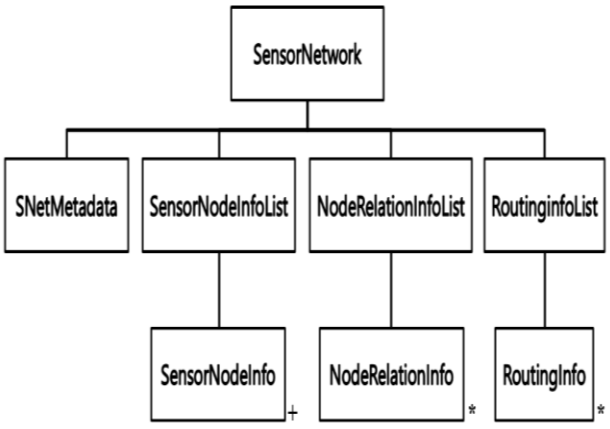

Fig.4. SensorNetwork Metadata

'SensorNetwork' describes the metadata about a sensor network, 'System' includes the metadata of the sensor node containing sensor devices, and the information about a sensor device is in ‘Component’ metadata. In order to adapt to various IoT environment, each metadata is designed to be independent but connected by reference if necessary. For example, if a node is simultaneously deployed on two different sensor networks, redundant description of the node can be avoided. In addition, if the node is in a dynamic network, it is possible to flexibly adapt to network changes from time to time. ‘SensorNetwork’ metadata can be linked to one or more ‘System’ metadata and ‘System’ metadata can refer to one or more ‘Component’ metadata. ‘SensorNtwork’ metadata in Figure 4 describes sensor network information. 'SensorNetwork' metadata consists of four parts. 'SNetMetadata' describes the meta information about the sensor network such as the location and capability of the network. 'SensorNodeInfo' contains the information about sensor nodes included in the network. Of course, since all the detailed information about a sensor node is described in the ‘System’ metadata of Figure 3. However, the 'SensorNodeInfo' describes the information of the sensor node depending on the network such as the role and position of the node in the current network. ‘NodeRelationInfo’ contains information about connections between nodes such as the connecting cost in the network, the address of neighboring node, and so on. ‘RoutingInfo’ metadata describes information for routing between nodes

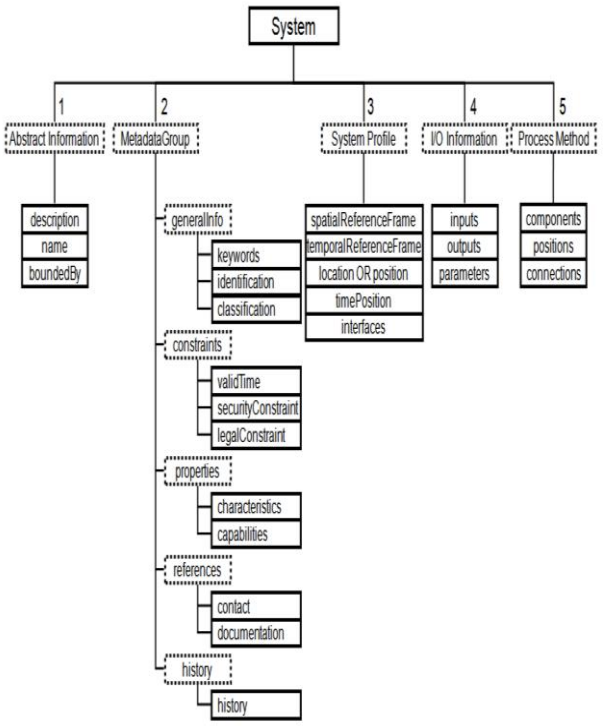

Fig.5. System Metadata

‘System’ metadata and ‘Component’ metadata for describing the sensor nodes and devices are

defined as shown in Figure 5, referring to the structure of SensorML. 'System' metadata consists of five parts. 'Abstract Information' and 'MetadataGroup' metadata describe basic information of sensor node and general meta information of sensor node, respectively. Although Figure 5 only describes the core meta information, it can be extended based on SensorML. 'SystemProfile' includes characteristics of sensor nodes. 'I/O Information' describes the input and output information of the sensor node. Finally, the 'ProcessMethod' metadata contains the information required to perform the sensor node and the information about the sensor devices included in the node optionally. The structure of 'Component' metadata for describing the sensor device is similar to 'System' metadata in Figure 5, except that the attributes of some metadata node are different.

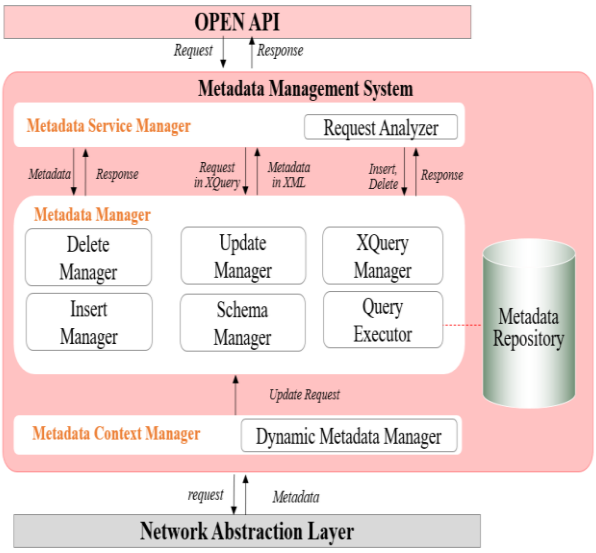

Fig.6. Architecture of Metadata Management System

Figure 6 shows the architecture of the MMS. The MMS obtains metadata from the Network Abstract Layer. It also receives some metadata from applications if necessary. The Dynamic Metadata Manager determines and controls the update interval of dynamically changing metadata. The Metadata Manager consists of six modules. The Insert Manager and Delete Manager insert and remove metadata requested by a user. The Update Manager updates the metadata by the request of a user and the Metadata Context Manger. The XQuery Manager processes the user requested XQuery queries that is written to the metadata described in XML and returns the results. The Schema Manager creates schema views to provide to users and manages the metadata schema. The Query Executor processes and manages the stored data by directly accessing the database

3.3 Intelligent Service Management System

Figure 7 is the architecture of the Intelligent Service Management System.

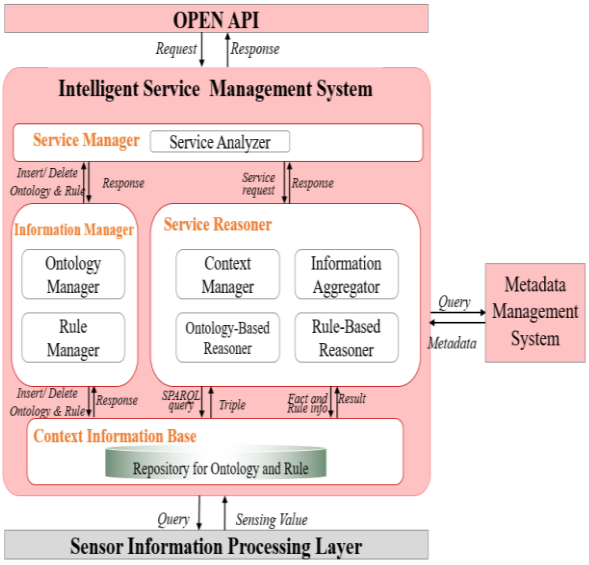

Fig.7. Architecture of Intelligent Service Management System

The Service Manager analyzes a service requested by user and transforms the request into the input for the module that can process the service. The ISMS uses an ontology and rule for providing an intelligent service. The ontology describes the concept and relationship of services and sensors and the rule includes context information for the service. The Information Manager stores and manages ontology and rule information. The Ontology Manager inserts, updates and deletes ontology individuals and, if necessary, modifies the ontology. The Service Reasoner provides the user requested service based on the ontology, rule, and sensing data. The Context Manager controls the action and data flow among modules. The Ontology-Based Reasoner infers current available sensor devices and context information needed to process user-requested service based on the ontology. To provide customized service, the Rule-Based Reasoner infers using rules that reflect the characteristics of the application and environmental context. The Information Aggregator obtains sensing data for reasoning from the Sensor Data Management Component. If the metadata is needed for reasoning, the Information Aggregator will request this

information from the MMS. The Repository for Ontology and Rule stores ontologies and rules for the service reasoning and provides them to the Information Manager and the Service Reasoner.

4. Middleware for Intelligent Service based on Sensors

Figure 8 shows the data and action flow for processing an intelligent service in the Service Reasoner

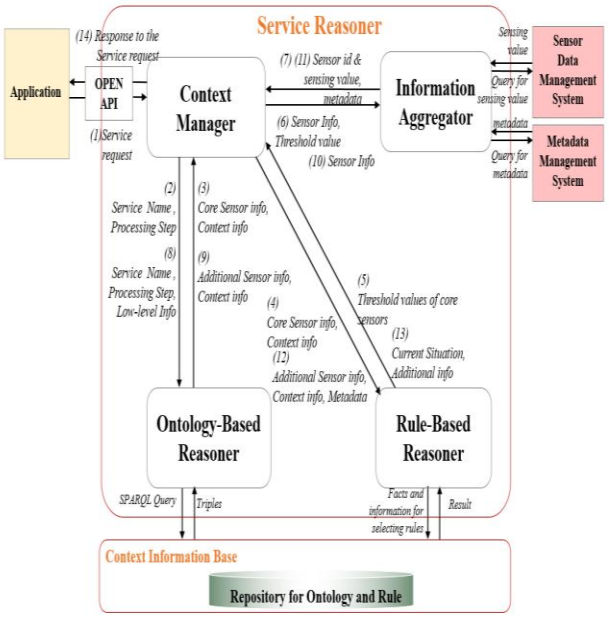

Fig.8. The data and action flow for processing an intelligent service

If a user requests a service through the OPEN API, the Context Manager obtains sensor and context information that is needed to process the requested service from the Ontology-Based Reasoner. In order to reduce the workload of the SDMS in real-time, The Ontology-Based Reasoner is inferred twice. The first reasoning is to infer core information for a user-requested service like (2) and (3) in Figure 8 and the second is to obtain additional information for the exact situation like (8) and (9). The step of the Context Manager is rule-based reasoning. This step is also executed twice. The former rule-based reasoning, (4) and (5), is to infer the threshold value for core sensor devices based on context Information extracted by the previous ontology-based reasoning. After the threshold value is determined, the Information Aggregator requests the sensing value of the core sensor devices to the SDMS when the sensing value is over the threshold value. If a sensing value is returned, the Information Aggregator obtains metadata related core sensor devices through the MMS. The Context Manager can know which sensor devices and context information are needed for the next reasoning by the extracted metadata. The second reasoning is started by obtaining the sensing value from the SDMS. As described above, the second ontology-based reasoning infers additional sensor and context information for the accurate context reasoning. After the second ontology-based reasoning, the Context Manager requests the current sensing value of the selected additional sensor devices through the Information Aggregator immediately. The latter Rule-Based Reasoner infers the current situation by using context information, metadata and current sensing value like (12) and (13). Finally, the Context Manager responds to the user-requested service.

Fig.9. The partial SS Ontology

The SS ontology consists of five sub-ontologies. The sensor ontology describes sensor information such as the sensing capability, sensor type, and unit of measurement. The capabilities of the sensor are related to the information needed to provide intelligent services. For example, if smoke information is needed for a forest fire monitoring service, it is associated with sensors that have the ability to measure smoke. If location information is needed, it is related with a sensor that measures altitude or a sensor that can acquire location information such as GPS. The Context ontology is used to describe the contextual information needed

for an intelligent service. The content of the context ontology is not a core element for providing intelligent services but is used for context-based intelligent reasoning. The node ontology describes the information of the nodes and the sensors they contain. The service ontology includes sensor-based intelligent services. The network ontology is used to represent the sensor network information and is related to the node class.

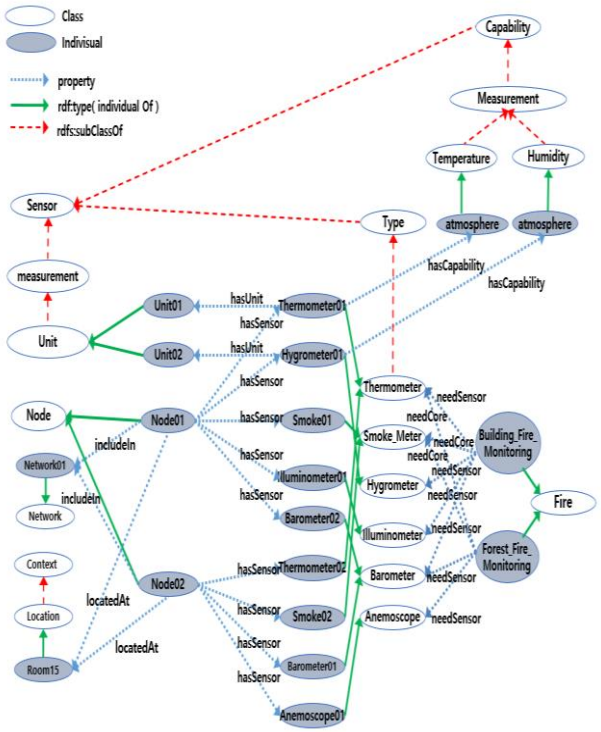

Fig.10. An ontology instance for the building fire monitoring service

Figure 10 shows an example of ontology-based reasoning with the individual of SS ontology. If a user requests a Building_Fire_Monitoring service, ISMS infers that a sensor of Smoke_Meter type is required as a core sensor for the service. In addition, ISMS infers that a thermometer, a hygrometer, an illuminometer and a barometer type sensor are required for the accurate situation-awareness. In the example, the Network01 has two thermometers, smoke and barometer sensor devices and one hygrometer and one illuminometer sensor device. Node01 and Node02 in the Nwtwork01 are located in Room15.

The rule-based reasoning infers the current situation based on the current sensing value of the inferred sensor devices and context information from the ontology-based reasoning. For example, Table 1 is a rule for accurately inferring the current fire situation in the Room15

Table 1 . A rule for the building fire monitoring service

```
(defrule BuildingFireDetectingService
   (Service Building_Fire_Mornitering)
   (Smoke_Meter (Value ?Smoke_Value))
   (Thermometer (Value ?Thermo_Value))
   (Hygrometer (Value ?Hygro_Value))
   (Location (Room15))
   (test (> ?Smoke_Value 100) (> ?Thermo_Value 30) (< ?Hygro_Value 10))
==> (assert (currentSituation ?Result)))
```

5 A Monitoring application for the Intelligent Service middleware

This paper implemented an application prototype of monitoring services to utilize the intelligent service middleware based on sensor.

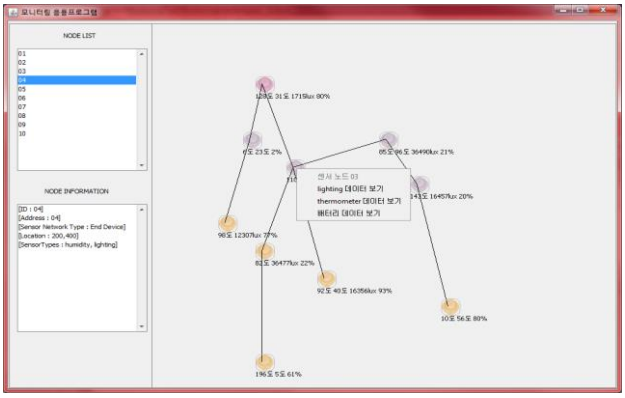

Fig.11. Monitoring Service application

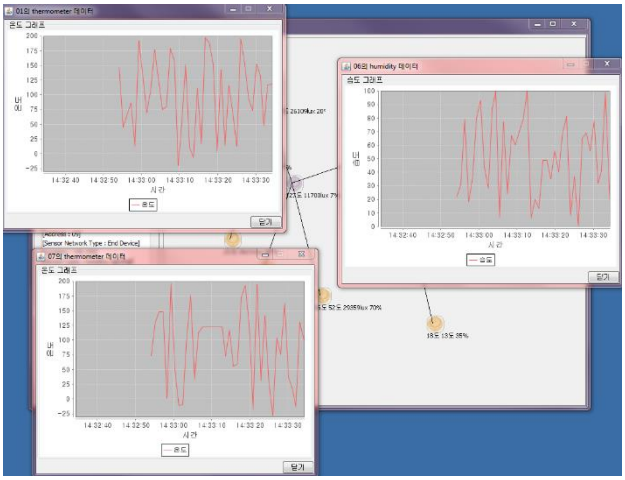

Fig.12. Sensing value monitoring in real time

Figure 11 shows the user interface of the sensor monitoring application. The left side of the interface is the list of nodes included in the network and information in real time. The right side shows the

sensor devices and metadata that make up the network. The user can select the sensor device and confirm specific information of the sensor. The user can also check the sensing value in real time by selecting the sensor device as shown in Figure 12.

4 Conclusion

This paper has proposed an intelligent service middleware based on sensor in IoT Environments that integrates heterogeneous sensors and provides intelligent services based on the sensors. The proposed middleware includes three systems. The first is the sensor data management system for collecting and managing sensing value in real time. The second is the metadata management system to store and manage metadata associated with the sensors. The paper has proposed integrated metadata to describe heterogeneous sensor, node and network information. The third is the intelligent service management system to provide sensor-based intelligent service to users. The paper has proposed the SS ontology for describing sensors and services and proposed a method for providing intelligent services using the SS ontology and rules. Of course, the three systems operate independently, so they can be used selectively as needed. This paper implemented a prototype system based on the proposed method.

We expect that the proposed middleware can be used in various applications for sensor-based intelligent service processing. In the future, we plan to expand the method to provide an intelligent service by automatically gathering semantic data like [15].

References:

[1] Bowei Liu, Ruizhang Huang, Ting Huang, Yingying Yan. MSDB: A Massive Sensor Data Processing Middleware for HBase. Proc. of 2017 IEEE Second International Conference on Data Science in Cyberspace, June 2017.

[2] Yugo Nakamura, Hirohiko Suwa, Yutaka Arakawa, Hirozumi Yamaguchi, Keiichi Yasumoto. Middleware for Proximity Distributed Real-Time Processing of IoT Data Flows. Proc. 2016 IEEE 36th International Conference on Distributed Computing Systems, June 2016.

[3] Anas A. Al-Roubaiey, Tarek R. Sheltami, Ashraf S. Hasan Mahmoud, Khaled Salah. Relia-ble Middleware for Wireless Sensor-Actuator Networks., Journal of IEEE Access, 2019, 7, pp. 14099-14111.

[4] Nawras Georgi, Aline Corvol, Regine Le Bouquin-Jeannes. Middleware Architecture for Health Sensors Interoperability. Journal of IEEE Access, 2018, 6, pp. 26283-26291.

[5] Assis, L. F. F. G., Horita, F. E. A., Freitas E. P., Ueyama, J., Albuquerque, J. P. A Service-Oriented Middleware for Integrated Management of Crowdsourced and Sensor Data Streams in Disaster Management. Journal of Sensors, 2018, 18, (6) pp. 1-27.

[6] Rita Zgheib, Emmanuel Conchon, Remi Bastide. Semantic Middleware Architectures for IoT Healthcare Applications. Enhanced Living Environments, Nanuiary 2019, pp263-294.

[7] Paolo Bellavista, Carlo Giannelli, Stefano Lanzone, Giulio Riberto, Cesare Stefanelli, Mauro Tortonesi. Middleware Solution for Wireless IoT Applications in Sparse Smart Cities. Journal of Sensors, 2017, 17, (11), pp. 1-18,

[8] Mauro A. A. da Cruz, Joel J. P. C. Rodrigues, Arun Kumar Sangaiah, Jalal Al-Muhtadi, Val-ery Korotaev. erformance evaluation of IoT middleware. ournal of Network and Computer Applications, 2018, 109, pp. 53-65.

[9] Perera. C, Jayaraman. P.P, Zaslavsky. A, Georgakopoulos.D, and Christen. P. osden: An internet of things middleware for resource constrained mobile devices. roc. of 47th Hawaii International Conference on IEEE, Jan 2014, pp. 1053-1062.

[10] Perera C, Zaslavsky. A, Christen. P, and Georgakopoulos. D. ensing as a service model for smart cities supported by internet of things. ournal of Transactions on Emerging Telecommunications Technologies, 2014, 25, (1), pp. 81-93.

[11] Xively [Online]. Available: https://xively.com/ (accessed on June 2019)

[12] Carriots [Online]. Available: https://www.carriots.com/ (accessed on June 2019)

[13] Soobin Jeon, Inbum Jung. MinT: Middleware for Cooperative Interaction of Things. Journal of Sensors, 2017, 17, (6), pp.1-25.

[14] OGC Sensor Model Language (SensorML), OpenGIS Standard (2014). Mohammad Ahmadinia, Ali Movaghar, Amir Masoud Rahmani. Semantic Data Gathering of Physical Entities in Semantic Sensor Networks Using Software Agents. Information Technology and Control, 2018, 47, (2), pp. 167-183.