A Scalable IoT Framework to Design Logical Data Flow using Virtual Sensor

Le Kim-Hung, Soumya Kanti Datta, Christian Bonnet
EURECOM, Biot, France
Emails: {lek, dattas, bonnet}@eurecom.fr

François Hamon, Alexandre Boudonne
GREENCITYZEN, Marseille, France
Emails: {francois.hamon, alexandre.boudonne}@greencityzen.fr

***Abstract*— During recent years, we have witnessed an explosion in the Internet of Thing (IoT) in terms of the number and types of physical devices. However, there are many limitations of these devices regarding their computing power, storage, and connection capabilities. They affect on-device processing of sensed data significantly. Centralized treatment of IoT data has proven challenging for many use cases demanding real time response. This paper aims at augmenting sensor data processing using the concept of virtual sensors. We propose a scalable virtual sensor framework that supports building a logical data-flow (LDF) by visualizing either physical sensors or custom virtual sensors. The process produces high-level information from the sensed data that can be easily perceived by machines and humans. A web-based virtual sensor editor (VSE) is also implemented on the top of the framework to simplify creation and configuration of the LDF. The VSE supports cross-platform and real-time verification for composed LDF. The paper also presents a catalog of supported virtual sensor type along with preliminary performance study.**

***Index Terms*—Cloud Computing; Industrial Internet of Things; Logical Data Flow; Scalability; Virtual Sensor.**

I. INTRODUCTION

The Internet of Things (IoT) is expected to bring connectivity to every object in the physical world. From connected cars [20], buildings [21] and smart cities [22], the IoT creates many opportunities in various domains [1]. According to Wikibon report [2], by the end of 2020, 212 billion IoT smart objects are expected to be deployed worldwide. Despite the rapid growth, the IoT is still facing some major challenges relating interoperation, performance, data reliability. On the contrary, the cloud computing environment offers a massive ability in term of computing and storage. Thus, integrating the IoT with Cloud technology is expected to provide scalable storage and sensing services enabling unlimited IoT device connecting to the Internet, which is called the CloudIoT paradigm [3].

However, there are critical challenges in deriving high-level information from raw data of the physical devices. With the exponential growth of IoT smart objects in term of power and functionality, certain situations cannot be handled at the device level. For instance - (i) a comprehensive query for checking the average of humidity in a region. This query may request all data of humidity sensor deployed in the corresponding area. (ii) Predicting missing data based on the historical data. To handle these situations, virtualization of the physical sensor in cloud environment, namely Virtual Sensor (VS), is considered as a practical approach. VS is a logical reflection of one or a set of physical sensors on the Cloud platform and is able to handle complex tasks which cannot be performed on physical sensors [4] [5]. The key aspect of VS is to facilitate and enrich the functionalities of physical sensors at the software level to adapt to different purposes and scenarios [6].

In this paper, we present a scalable virtual sensor framework (sVSF). It simplifies creating and configuring VSs with the programmable operators (rule, formula or function). These VSs are linked together to be a logical data-flow that enables producing the high-level information from collected data. On the top of the framework, a Virtual Sensor Editor (VSE) is also implemented to facilitate building and configuring the LDF by offering the drag-drop actions on HTML5 web interface. In order to achieve scalability and performance, sVSF is implemented based on clustering architecture along with various strategies such as executing LDF following asynchronous model, using No-SQL database to store and query data.

Our framework supports various types of virtual sensors at Infrastructures as a Service (IaaS) and Platform as a Service (PaaS) such as singular, accumulator, aggregator, selector, qualifier, context-qualifier, and simple predictor [7]. In the sVSF, VSs are treated as physical sensors. The output data of VS is stored in the database in the same way as physical. Thus, each VS carries the historical data which is valuable for further data analyzing. Furthermore, the proposed framework remarkably fits for Low Power Wide Area Networking (LPWAN) scenario where band width (12 bytes in SIGFOX and up to 250 bytes in LoRa), and data rate (typically 10 kilobits per seconds) are limited [17]. These restrictions reduce the quantity of collected data that affects directly on further data analysis and context-awareness in term of accuracy and trustworthiness.

On the top of the framework, sVSF offers a cross-platform development environment for the virtual sensor by enabling HTML5 technology. The lower layers take principal responsibility to process and generate high-level information based on the composed LDF. In order to enhance scalability and performance, the core framework is implemented following clustering architecture using Node.js language. Our paper has four highlighted contributions:

- Identifying the limitations of current virtual sensor framework in term of VS functionality and usability.
- Reviewing the concept of virtual sensor and its taxonomy.
- Presenting a scalable virtual sensor framework to increase information quality from sensed data, and implementing

clustering model along with various strategies to archive high scalability and performance.

- A cross-platform development tool for virtual sensor, namely virtual sensor editor, is also offered to speed up designing logical data-flow.

The remainder of this paper is constructed as follows: Section II reviews the related works and highlights their limitations. Our proposed framework is introduced in detail in Section III. Section IV focuses on experiment and performance evaluation. Finally, we summarize the paper and discuss future works in Section V.

II. STATE-OF-THE-ART

In this section, we review virtual sensor and current virtual sensor framework in the IoT. Their limitations are also identified. At the end of the section, we present our motivations to bridge the gaps by delivering a scalable IoT virtual sensor framework.

As a definition in [15], a virtual sensor reflects a physical sensor that is able to obtain and represent data on cloud. Following [14], a virtual sensor is an emulation of a physical sensor which collects its data from underlying physical sensors. There are many ways to define and categorize virtual sensor. In [15], the authors categorize virtual sensors into two types: (i) **Task level**: represents the physical sensor as a virtual object that could be processed, calculated. (ii) **Node level**: represents a subset of physical sensor as a virtual topology. In [14], virtual sensor is classified into four typical types: (i) **One-to-Many**: One physical sensor is represented by many virtual sensors. (ii) **Many-to-One**: One and more physical sensor is presented by one virtual sensor. (iii) **Many-to-Many**: This is the combination of two under types. (iv) **Derived**: One virtual sensor can represent different physical sensor types. While in other types, the virtual sensor only represents the same physical device type. At the IaaS level, [16] categorizes virtual sensor based on their offering services. Therefore, in the IoT, the definition and taxonomy of the virtual sensor are chaos and heterogeneous.

The authors of [8] propose a web-based virtual sensor editor tool to facilitate designing virtual sensor process. This tool visually aggregates either the physical sensors or customized virtual sensors. It also supports calculating and visualizing real-time sensor values on graphic charts. VSs are created by aggregating physical sensors. The graphic interface supports native HTML5 drag-drop and real-time virtual sensor evaluation. As a result of HTM5 characteristics and call-by-need strategy, this tool enables cross-platform and scalability. Similarly, the authors of [9] present a web-based interactive framework to visualize and authorize sensors as well as actuators for indoor scenario. Each IoT Thing serves as a node, which is visualized within a 3D indoor scene. Thus, the end-user can monitor, link and program sensors and actuators respectively. This framework works based on event handling model which treats incoming data as an event. In order to handle complex events, that must be processed on multiple sensors; the author proposes a hierarchical graph for visual summarizing sensors, actuators and their relations.

In the IoT environment, physical sensors are distributed and affected by many adverse factors. Thus, sensed data need to be processed, filtered and transformed for precise measurement and providing high-level information. Many middleware platforms are designed to process the IoT data on either multi-sensors or multi-stream [10] [11]. These works aim to improve the information quality of data coming from heterogeneous data sources. In the same scope, the authors of [12] present a virtual sensor environment that can handle real-time sensing data processing. This approach uses Complex Event Processing (CEP) as a virtual sensor engine. Their main contributions are to take the benefit from CEP and allow the user to define the custom analytic algorithm along with data analysis block on the incoming data. The authors of [13] address solving major challenges about implementing virtual sensor at Software as a service (SaaS) and Platform as a service (PaaS) level. They propose explicit sensor-cloud architecture with four separate modules to handle specific tasks such as sensing, processing, storing and communicating. Each module is equipped an API supporting the end-user building applications and sharing sensed data to either the IoT users or services.

The limitations of state-of-the-art are given below.

- In [8], we notice limited functionalities for the virtual sensor. These functions are only able to perform on incoming data. There is no discussion regarding virtual sensor types and which types are supported by their tool.
- The authors of [9] have not shown how to configure the algorithm of CEP. They also do not offer a graphic interface to facilitate the configuration process of data analysis block for end-user. Likewise, the works of [10] [11] more focus on services and implementation than simplifying configuration process at the user level.
- The work of [12] just focuses on the indoor scenario. There is no mention on the mechanism to create and configure a custom virtual sensor as well as an actuator.

From all points above, there is not a comprehensive virtual sensor framework proposing an effective web-based interface along with a robust backend. Utilization of virtual sensor to present historical data is also not mentioned. Our framework is designed and implemented to mitigate their limitations.

III. VIRTUAL SENSOR FRAMEWORK

In this section, we present the definition and taxonomy of the virtual sensor as well as our virtual sensor framework architecture, which is designed as a modularized layered application. Such framework operates over clustering architecture and asynchronous model to maximize scalability and performance. At the end of the section, we describe the workflow of the proposed framework in specific deployment scenarios to emphasize its benefit.

A. Virtual Sensor

We define the virtual sensor as a virtual object. Such object is equiped an operator to perform specific functionalites. In our framework, we support three type of operator including rule,

formula and function. We also proposed a new taxonomy of the virtual sensor based on its operator. For instance, a virtual sensor is labelled an "Accumulator" type if its operator contains accumulated functions. The following is the list of supported virtual sensor type :

- **Singular**: This type allows performing one-to-one mapping between the physical sensor and its reflected interface in cloud side. Through this virtual interface, the end-user can configure sensor configuration to obtain the data from a physical sensor. At the first stage, the sensor configuration is stored as a sensor driver that is selected by the user via VSE.
- **Accumulator**: A virtual sensor could perform accumulation function on its sensing data within a particular duration. For example, a rainfall physical sensor uses the counter value to identify the rainwater volume. An accumulator VS is useful to present rainwater volume within 24 hours by accumulating on this counter value.
- **Selector**: A virtual sensor enables to acquire sensing data from one or many physical sensors replied on defined criterions. For instance, a selector virtual sensor represents all temperature data that is higher than 10.
- **Aggregator**: A virtual sensor can perform basic statistics (averaging, maximum, minimum, etc.) on physical sensors. The functions of aggregator can mitigate the limitations of physical sensor regarding memory and computing. For example, in the case of humidity sensor deploying in various regions, an aggregator sensor can be implemented to calculate the average of humility for a particular area.
- **Qualifier**: The same with the singular type but virtual sensor only is activated if sensing value satisfies qualifier function. This VS type is configured by using "IF ELSE" statement. For example, one qualifier virtual sensor monitoring temperature can generate an alert when sensing value higher a defined threshold.
- **Context-qualifier**: The same with qualifier but the qualifier function performs on a bundle of sensor.
- **Predictor**: This virtual sensor performs prediction next sensing value base on analyzing previous data. Such virtual sensor is necessary in case of occurring error of physical sensor.
- **Compute**: A virtual sensor is equipped a complex function, that analyzes sensing data from a set of the sensor to propose a higher-level information. For example, a Compute virtual sensor could offer the car state based on observed data (engine temperature, oil level, gas level) from car's sensors.

We also present a novel component named "Logical Data-Flow" which represents a chain of virtual sensors to perform a specific task. For example, a logical data flow can be used to determine remains of liquid in a tank from ultrasonic sensor data. This LDF is probably a chain of one singular virtual sensor, one selector virtual sensor and one aggregator virtual sensor.

B. Virtual Sensor Framework Architecture Overview

The primary goal of our framework is to produce high-level of information from sensing data using logical data flow. This framework also simplifies creation and configuration logical data flow by offering an interactive virtual sensor editor and many types of productive operators such as rule, formula, and function. Fig. 1 depicts the framework architecture composing of three horizontal layers which are described below.

- **Connection Layer** – This layer takes responsibility to maintain the connection of sVSF and Sensor Data Service Platform (SDSP) where aggregates and pre-process collected data from physical sensors. There are three core components: (i) Sensor Data Connector: This connector is used to interact with SDSP through RESTful web services and MQTT. (ii) Sensor Configuration Synchronization: This component oversees synchronizing sensor configuration between sVSF and SDSP. In addition, after successful testing, the configuration of a singular VS will be applied to corresponding physical sensor managed by SDSP through calling a RESTful web service. (iii) Sensor Tracking: This component is used to track the new sensor, which is recently registered to SDSP. The sensor profile will be saved in the database and reused in VSE.
- **Processing Layer** – This layer contains the database and a primary engine to execute the logical data-flow and virtual sensor functionalities. There are two databases: (i) Sensor Data Storage database is a permanent database to store sensor information, LDF. (ii) Temporary Data Storage database is used to store temporary values of virtual sensor as well as intermediate results of LDF. Such data will be removed after a certain time configured by the administrator. Processing layer also manages a "Sensor Composition" (CP) component to retain a record of configuration and relationship among virtual sensors at the presentation layer. When a new virtual sensor is created by dragging and dropping onto VSE, such changes will be caught and stored by CP. In addition, such CP ensures logical data-flow is executed following asynchronous model in the engine. The processing layer carries a user-defined function library that contains the custom functions declared by end-user; this function can be called directly from CP.
- **Presentation layer** – This layer oversees rendering an interactive HTML5 web interface, namely virtual sensor editor. The editor supports the end-user to effectively interact with virtual sensors to create a logical data-flow. Virtual sensors are visualized as linkable boxes which can be configured either functionality or appearance via setting panel. In addition, these boxes can link together to create logical data flow. All such configurations are handled by "Sensor Composition" in under layer. Presentation layer also carries a "Data-flow Profile Selector" component supporting the end-user to select and reuse virtual sensor template as well as logical data flow from the database.
- **Administration layer** – This layer takes the role in authorizing and supervising user access right on virtual sensors and logical data-flows. The end-users are only

allowed to perform certain actions based on their roles. For instance, a standard user cannot delete a logical data flow. Administration layer is also used to manage general settings of the framework such as the time life of temporary data, sensor configuration synchronization interval.

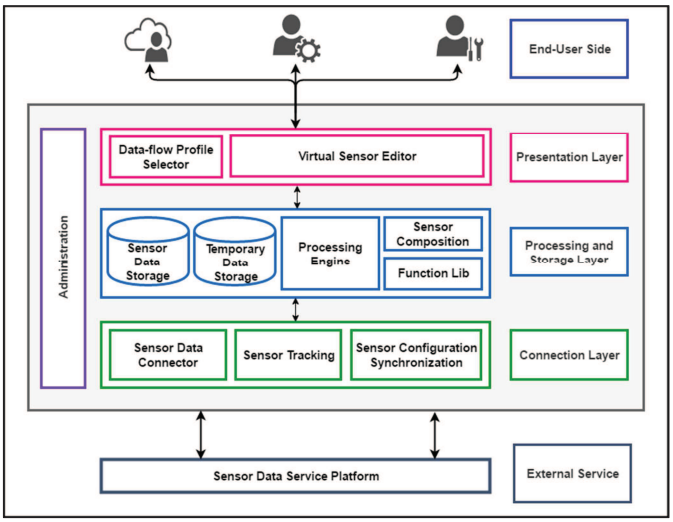

Fig. 1. The proposed architecture and its building blocks.

C. Virtual Sensor Editor

One of a key element of sVSF is an HTML5 web-based virtual sensor editor, which enables cross-platform development environment. The editor is a WYSIWYG (what you see is what you get) system. This allows the user to simply build a LDF by creating, configuring and connecting VSs together. VSE is implemented using native HTML5 and JointJS[1] library to maximize portability and availability across various end-user platforms. There are four highlighted attributes of this editor: (i) **Drag-drop interface**: The end-user is able to simply create and link the virtual sensors by drag-drop action, (ii) **Real-time Evaluation**: After creating, user-defined LDF could be evaluated and receive result immediately, (iii) **Reusability**: VS configuration and LDF is stored and shared between the end-users.

As shown in Fig. 2, a virtual sensor is represented as a box consisting of input, output ports and a VS operator. The number of these ports and sensor configuration depend on the type of virtual sensor. For example, a singular virtual sensor which serves as a physical sensor has one output and no input port. In Configuration panel on the left side, a drop box is added to allow the user to configure sensor driver. By default, we use the green and red color to identify virtual sensor type. But the end-user can change this attribute. Each output port can be assigned to one or more input of different boxes via data links. After data link is established, the later sensor enables to select the output of former sensor as an input parameter for its operator. The auto-complete feature is also equipped to speed up this selecting process.

Our sVSF offers recursive composing for the virtual sensor. A defined virtual sensor can be used as an input to construct other virtual sensors. After logical data flow is completely established, the evaluation feature enables the user to execute the data flow on self-generated data and receive the result immediately. Thus, the user can evaluate or correct the configuration in case of error. At the final state, the complete logical data flow is saved in the database and reused for next time.

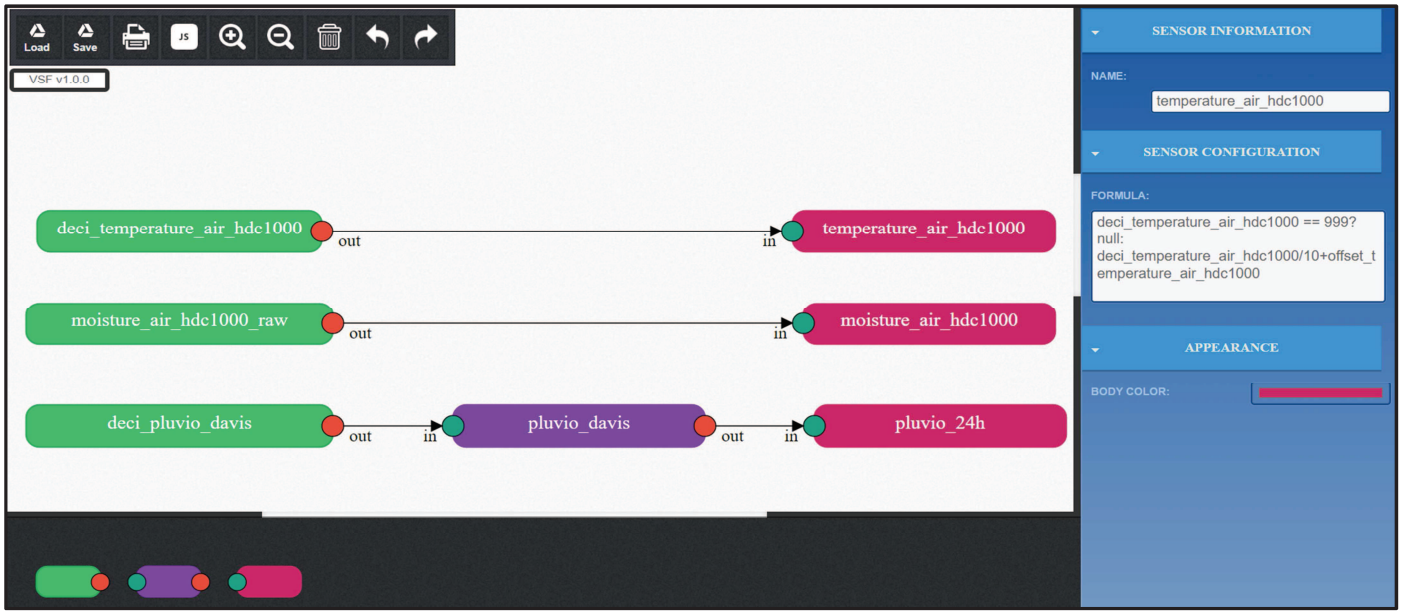

Fig. 2. Virtual Sensor Editor Interface.

[1] http://resources.jointjs.com

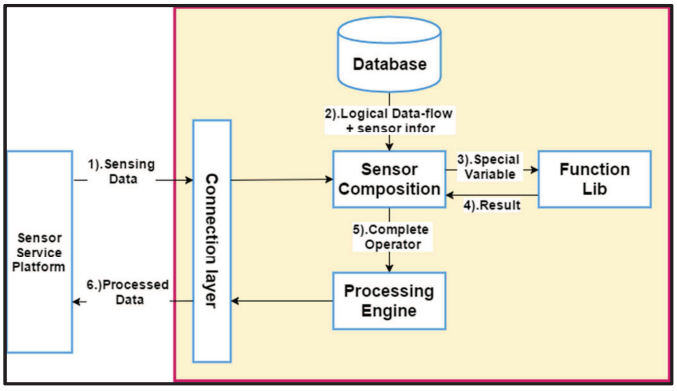

Fig. 3. The framework operation diagram.

D. Virtual Sensor Framework Workflow

The overall blueprint of our sVSF workflow is shown in Fig. 3. Physical sensors register their resource descriptions under CoRE Link Format [18] with our backend. Once registered successfully, such sensors and resources are discovered through simple search queries. For instance, a query to search all the temperature sensor can be *GET /rd-lookup/ep?rt=temperature*. All sensing data from registered sensors will be forwarded to sVSF via SDSP.

In sVSF, VS function supports three types of the operator: rule, formula, and function. At the first state, the incoming data is received by connection layer. After that, this data is conveyed to Sensor Composition which handles and processes the VS operators and relationships in the logical data flow. This component also takes responsibility to convert VS operators to the proper mathematical operations and ensures it is executed in correct order in the processing engine. The conversions of VS operator into mathematical operation goes through two phases: First, the logical data flow content and sensor information are loaded into CP. More detail, as shown in Fig. 4, a logical data flow comprises four VS operators, and sensor metadata are inserted in CP. Second, the particular variables of the operator are extracted and calculated by calling the corresponding functions in Function Lib Component (FL). These variables are reserved to simplify particular operations. For example, as shown in Fig. 4, *$device.tank_level_change.data[24]* represents all sensing data of "tank_level_change" sensor within 24 hours. After calculating, the values of special variables are added into virtual sensor operator before storing in temporary data storage in order to be reused in another stage. The lifetime of this temporary data can be set up by the sVSF administrators In the case of the virtual sensor has an input from another virtual sensor. This input value is also considered as a special variable and directly access via sensor name. For instance, sensor named "tank_volume" can use the output value of "tank_level" sensor via declaring a speacial variable named "tank_level". Furthermore, to maximize performance, Sensor Composition take responsibility for organizing the working schedule based on the asynchronous model. The un-relational virtual sensors, which are not linked together, will be arranged into the same thread and executed in parallel in processing engine. For example, in Fig. 2, all green virtual sensors will be performed in parallel.

The most important component in processing layer is processing engine, where VS operators are executed. These executions are performed in parallel by a JavaScript library, namely MathJS[2]. Finally, the output of processing engine will be stored in the database to be reused before responding final result to SDSP.

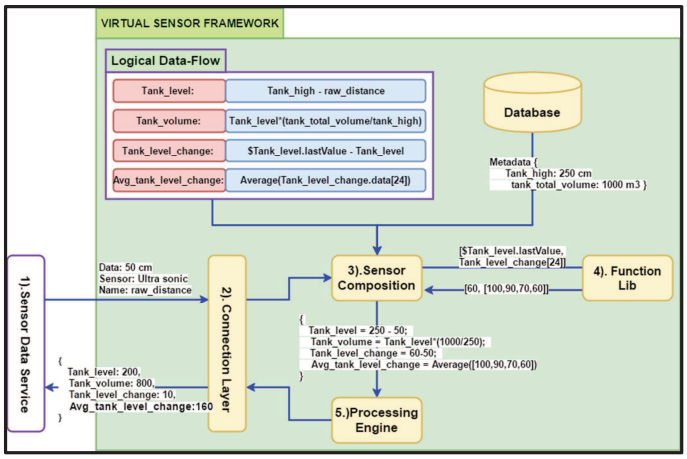

Fig. 4. The generating high-level information process.

IV. PROTOTYPE AND EVALUATION

In this section, we describe the utilization of our framework in a practical use-case. We also discuss how to archive high performance, scalability in our framework. Finally, an evaluation of our strategies is proposed.

Our first work has been applied to an industrial project for tank monitoring. The primary goal of this project is to manage the chemical volume via a level sensor which is plugged at the top of a tank. The raw data of this sensor is the distance from the top of the tank to the chemical surface. Fig. 4 illustrates the whole process of generating high-level information such as remaining of the chemical level (Tank_level), remains of chemical volume (Tank_volume), change of chemical level (Tank_level_change), average on this change within 24h (Avg_tank_level_change). Firstly, the raw data contains sensor information and sensed data is sent to sVSF. This data is handled by Sensor Composition where corresponding logical data-flow and sensor metadata is loaded from the database. In this case, the metadata is the tank information such as the height of tank (Tank_high) and the total volume of tank (Tank_total_volume). At this component, special variables such as last value of such sensor (Tank_leve.lastValue) or historical sensing data within 24h (Tank_leve_change.data[24]) are calculated by calling the corresponding functions in Function Lib component. All obtained information (special variable value, raw data value, sensor metadata) is injected into logical data low. Before transferring and executing at Processing Engine, logical data flow is converted to the

[2] http://mathjs.org

mathematical operations. The final result is responded to SDSP via Connection Layer

SVSF is developed by integrating MathJS library into NodeJS Express framework[3] which uses event-driven architecture. Nodejs also leverages a non-blocking I/O model that allows request being processed asynchronously. In order to enhance the framework scalability, we use clustering architecture. A cluster comprises a set of servers running simultaneously. Each server is called node. The cluster is elastic to adapt to the unexpected change in term of the number of concurrent user by dynamically add or remove a node to the cluster. There are two types of nodes: master node and worker node. The master node is used to manage the worker node in the cluster. It plays a role to distribute requests among different nodes in the cluster. Other strategies are proposed to increase the performance:

- The first strategy is to store the output of the virtual sensor in a temporary database. Such value could be re-used as the input of other VS sensor instead of re-calculation.
- The second strategy is to use an in-memory database[4] to speed up data querying process. Our framework uses a NoSQL database named Apache CouchDB[5]. Comparing with a relational database, CouchDB stores the data in an independent document and its self-contained schema. As the result, it provides a massive scalability and powerful full-text search
- The final strategy is to apply the asynchronous model to execute logical data-flow, meaning that all independent virtual sensor or virtual sensor in the same stage is executed in parallel.

To evaluate the effectiveness of proposed strategies, we have to consider two scenarios: (i) Significantly increasing the number of simultaneous physical sensor in SDSP. (ii) Increasing the complexity of logical data flow regarding the number of VS. All evaluations are performed on a computer with following configuration: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, 2401 MHz, 2 Core(s), 4 Logical Processor(s), 8GB of RAM and the operating system is 64-bit Windows 10. The clustering model is set up and deployed using native Clustering Module[6] provided by NodeJS .The data rate of the physical sensor is one message per second. For the first scenario, we have conducted an experiment with different scale of sensor network, which increases from 100 to 450 concurrent physical sensors. The logical data flow comprises 50 virtual sensors. Fig. 5 illustrates the performance changes after adopting our enhancements. As shown in the figure, our enhancement is remarkably effective. Without clustering model and enhancement strategies, the response time is significant increase when expanding the scale of sensor network. After applying clustering model using 4 or 8 clusters, the response time is highly stable under 1 second regardless the size of sensor network. With 450 concurrent physical sensors, the normal response time is over 6 seconds comparing with 875 milliseconds and 650 milliseconds of the model using 4 and 8 clusters respectively.

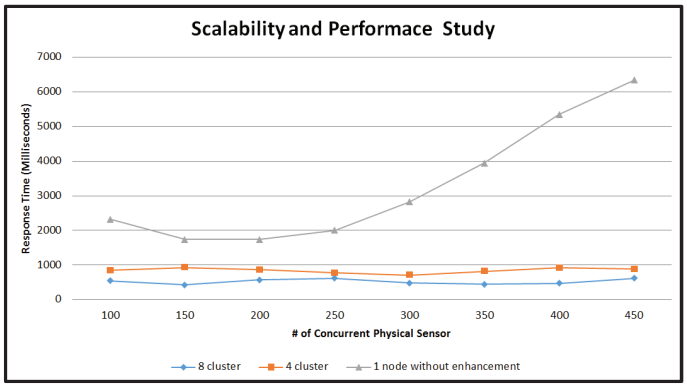

Fig. 5. The effect of our enhancement in scalability and performance

In the second scenario, the simulations are performed with different logical data flows size, which contains from 1 to 100 virtual sensors. Each logical data-flow is evaluated by various sensor network scale in SDSP. As shown in Fig. 6, when increasing the number of concurrent physical sensor, the response time lightly increases regardless logical data flow size. In case the scale of the logical data flow is moderate (comprising fewer than 50 virtual sensors), our system is able to serve a data message under 800ms even when 50 concurrent physical sensors are running. In the case of scaling up to 100 concurrent physical sensors, the response time is still under 1 second.

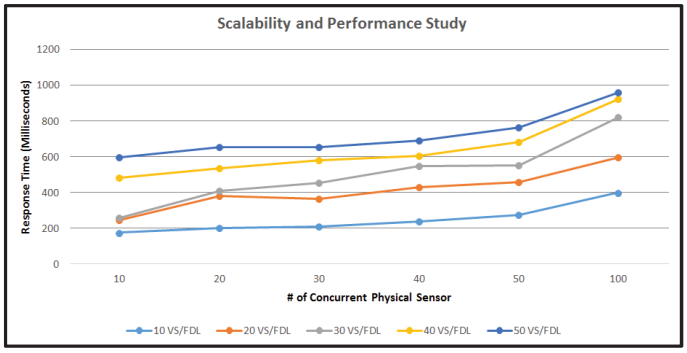

Fig. 6. The framework's performance.

V. CONCLUSION AND FUTURE WORK

In this paper, we review a concept of the virtual sensor and propose a new virtual sensor taxonomy based on its functionality. The limitations of the existing virtual sensor frameworks are also considered in term of virtual sensor functionality and usability. Motivating to bridge the gaps, we proposed a scalable virtual sensor framework that allows producing high-level information from sensed data, by creating a logical data flow over a collection of the virtual sensor. A web-based virtual sensor editor is also offered to accelerate creating and configuring logical data-flow. In evaluation section, a serial of strategies to enhance the performance and

[3] http://expressjs.com/fr
[4] https://en.wikipedia.org/wiki/In-memory_database
[5] http://couchdb.apache.org
[6] http://nodejs.org/api/cluster.html

scalability are discussed and evaluated. Regarding future works, we are currently working on integrating our platform into the oneM2M based framework [19] and FIWARE[7] architecture for ensuring interoperability.

ACKNOWLEDGMENT

The work is supported by GreenCityZen Company.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347-2376, Fourthquarter 2015.

[2] D. Floyer, "Defining and sizing the industrial Internet," Wikibon, Marlborough, MA, USA, 2013.

[3] A. Botta, W. de Donato, V. Persico and A. Pescapé, "On the Integration of Cloud Computing and Internet of Things," 2014 International Conference on Future Internet of Things and Cloud, Barcelona, 2014, pp. 23-30.

[4] S. Kabadayi A. Pridgen C. Julien "Virtual sensors: Abstracting data from physical sensors", Proceedings of the 2006 International Symposium on on World of Wireless Mobile and Multimedia Networks, pp. 587-592 2006.

[5] I. Leontiadis C. Efstratiou C. Mascolo J. Crowcroft "Senshare: Transforming sensor networks into multi-application sensing infrastructures", Proceedings of the 9th European Conference on Wireless Sensor Networks, pp. 65-81 2012.

[6] A. Gupta and N. Mukherjee, "Implementation of virtual sensors for building a sensor-cloud environment," 2016 8th International Conference on Communication Systems and Networks (COMSNETS), Bangalore, 2016, pp. 1-8.

[7] S. Bose A. Gupta S. Adhikary N. Mukherjee "Towards a sensor cloud infrastructure with sensor virtualization", Proceedings of the Second Workshop on Mobile Sensing Computing and Communication ser. MSCC '15, pp. 25-30 2015.

[8] J. Zhang et al., "Supporting Personizable Virtual Internet of Things," 2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing, Vietri sul Mere, 2013, pp. 329-336.

[9] Y. Jeong, H. Joo, G. Hong, D. Shin and S. Lee, "AVIoT: web-based interactive authoring and visualization of indoor internet of things," IEEE Transactions on Consumer Electronics, vol. 61, no. 3, pp. 295-301, Aug. 2015.

[10] X. Zheng D. E. Perry C. Julien "BraceForce: a middleware to enable sensing integration in mobile applications for novice programmers", Proceedings ACM International Conference on Mobile Software Engineering and Systems, pp. 8-17 Jun. 2014.

[11] L. Hu F. Wang J. Zhou K. Zhao "A Data Processing Middleware Based on SOA for the Internet of Things", Journal of Sensors, Jan. 2015

[12] Brunelli D., Gallo G., Benini L., "Sensormind: Virtual Sensing and Complex Event Detection for Internet of Things", De Gloria A. Applications in Electronics Pervading Industry, Environment and Society. ApplePies 2016. Lecture Notes in Electrical Engineering, vol 409. Springer, Cham.

[13] A. Gupta and N. Mukherjee, "Implementation of virtual sensors for building a sensor-cloud environment," 2016 8th International Conference on Communication Systems and Networks (COMSNETS), Bangalore, 2016, pp. 1-8.

[14] Sanjay Madria, Vimal Kumar, Rashmi Dalvi, "Sensor Cloud: A Cloud of Virtual Sensors", IEEE Software, vol. 31, no. , pp. 70-77, Mar.-Apr. 2014, doi:10.1109/MS.2013.141.

[15] A. Gupta and N. Mukherjee, "Rationale behind the virtual sensors and their applications," 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, 2016, pp. 1608-1614.

[16] Sunanda Bose, Atrayee Gupta, Sriyanjana Adhikary, and Nandini Mukherjee, "Towards a Sensor-Cloud Infrastructure with Sensor Virtualization", Proceedings of the Second Workshop on Mobile Sensing, Computing and Communication 2015 (MSCC '15), New York, NewYork, USA, pp. 25-30.

[17] Ferran Adelantado, Xavier Vilajosana, Pere Tuset, Borja Martinez, Joan Melia-Segui, et al., Understanding the Limits of LoRaWAN. IEEE Communications Magazine, Institute of Electrical and Electronics Engineers, 2017

[18] Z. Shelby, IETF RFC 6690: Constrained RESTful Environments (CoRE) Link Format. IETF, 2012.

[19] S. K. Datta, A. Gyrard, C. Bonnet and K. Boudaoud, "oneM2M Architecture Based User Centric IoT Application Development," 2015 3rd International Conference on Future Internet of Things and Cloud, Rome, 2015, pp. 100-107.

[20] S. K. Datta, R. P. F. Da Costa, J. Härri and C. Bonnet, "Integrating connected vehicles in Internet of Things ecosystems: Challenges and solutions," 2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), Coimbra, 2016, pp. 1-6.

[21] S. K. Datta, C. Bonnet and N. Nikaein, "An IoT gateway centric architecture to provide novel M2M services," 2014 IEEE World Forum on Internet of Things (WF-IoT), Seoul, 2014, pp. 514-519.

[22] S. K. Datta, R. P. Ferreira da Costa, C. Bonnet and J. Härri, "oneM2M architecture based IoT framework for mobile crowd sensing in smart cities," 2016 European Conference on Networks and Communications (EuCNC), Athens, 2016, pp. 168-173.

[7] https://www.fiware.org